IBM

An Introduction to IBM WebSphere Everyplace Suite Version 1.1

Accessing Web and Enterprise Applications

Build business solutions to reach anyone, anywhere, anytime

Support wireless networks, WAP, dial-up and Internet connections

Single sign-on to multiple applications and services

Juan R. Rodriguez
Richard Appleby
Bernt Bisgaard
Hao Wang
Adrienne McGrory
Abdulamir Mryhij
Amy Patton
Muhammed Omarjee

Redbooks

ibm.com/redbooks

International Technical Support Organization

SG24-5995-00

An Introduction to IBM WebSphere Everyplace Suite Version 1.1 Accessing Web and Enterprise Applications

October 2000

Take Note!

Before using this information and the product it supports, be sure to read the general information in Appendix B, “Special notices” on page 225.

First Edition (October 2000)

This edition applies to Version 1.1 of IBM WebSphere Everyplace Suite for use with the AIX Operating System.

Comments may be addressed to:
IBM Corporation, International Technical Support Organization
Dept. HZ8 Building 678
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

This redbook is about building business solutions using the IBM WebSphere Everyplace Suite Version 1.1 product to enable Web and enterprise application access from pervasive computing devices. It helps you to understand this product and focuses on implemented architectures and technologies included in this release such as wireless communications, transcoding, security, caching proxy, load balancing, messaging, and single sign-on, among others.

IBM WebSphere Everyplace Suite is an integrated end-to-end software solution for mobile e-business. In this redbook, you will find information that will help you plan to successfully implement solutions that businesses must address to be able to access Web and enterprise applications from desktop browsers and the new class of client devices such as WAP phones, Palm Pilots, WorkPads and others.

A basic knowledge of HTTP and WAP protocols as well as some understanding of Web and Java technologies (XML, HTML, WML, servlets, and JSPs) and the terminology used in Web and enterprise applications is assumed.

The team that wrote this redbook

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

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

Richard Appleby is a Solutions Architect working in IBM UK, based at Hursley Park development laboratory near Winchester, England. He has 15 years of experience in development and consulting on transactional systems, the last two years of which have been in the e-business and pervasive computing arenas. He has previously held test, development and design roles in both the CICS and MQSeries development organizations, and provided end-to-end architectural consultancy to some of the largest corporate customers of IBM. He has previously written about both the MQSeries and Tivoli TME/10 products.

Bernt Bisgaard is an advisory IT specialist in Denmark. He has five years of experience in architecting e-business solutions, and now works in the Pervasive Computing division of IBM initiating projects for customers and advising on pervasive solutions. He holds a Master’s degree in computer systems engineering from the Technical University of Denmark. Bernt joined IBM in 1992.

Hao Wang is a senior executive of Deuk Company, a technology and management consulting firm based in Cambridge, Massachusetts. He has a Ph.D. degree in the area of optoelectronics from Massachusetts Institute of Technology (MIT) and an MPA degree from Harvard University. Hao is the founder of Deuk Company and Noah.Net, Inc., a wireless application service provider (ASP) enabling business-to-business and business-to-consumer pervasive computing. Hao has published many technical journal articles.

Adrienne McGrory is an IT Specialist in the Finance Services Sector of IBM, based in Edinburgh, UK. As a technical architect, she has experience working with the WebSphere software products, implementing strategic e-business projects in the UK and The Netherlands.

Abdulamir Mryhij is a senior IT consultant with more than 16 years of experience in the IT industry. He has participated as a Software Engineer, consultant, project manager, and chief architect in many projects around the world and specializes in software engineering and application architecture. Amir holds a Master's degree in Information Technology from the University of Western Sydney. He is also an IBM-certified e-Business Designer and is Microsoft-certified for SQL server, Windows architecture, and Windows NT. Amir is country consulting services manager at GBM-Qatar.

Amy Patton is a technical developer at Immersant in Cambridge, Massachusetts. Amy specializes in Web applications development, from network and database design to Java user interfaces. Amy graduated from Boston University with a Bachelor of Arts in Computer Science.

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

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

Anthony Wrobel, CD Choi, Barbara Wetmore, George Hall, Henry Welborn
IBM Research Triangle Park, North Carolina, USA

Samuel Camut, Pinwu Xu, Ronnie Jones, Salim Zeitouni, David Chuang
IBM Research Triangle Park, North Carolina, USA

Thomas Seelbach
IBM Hawthorne, New York, USA

Gill Spencer
Extended e-Business Solution Centre, IBM Hursley, UK

Leonard Hand
IBM Global e-business Solutions Integration Center, Dallas, Texas, USA

Comments welcome

Your comments are important to us!

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

- Fax the evaluation form found in "IBM Redbooks review" on page 239 to the fax number shown on the form.
- Use the online evaluation form found at **ibm.com**/redbooks
- Send your comments in an Internet note to redbook@us.ibm.com

Part 1. Introduction

In this part of the redbook we provide useful information for understanding the evolution of e-business and pervasive computing. You will also find high-level architecture information showing how the components in IBM WebSphere Everyplace Suite are deployed to provide complete solutions in different scenarios. Hints and tips are inserted to provide you with a better understanding of the IBM WebSphere Everyplace Suite product.

Chapter 1. Evolution of e-business

When the Internet was first created no one could have imagined the impact it would eventually have on our personal and business lives. Over the years it has grown, changing from a small network with limited abilities into a vast system that spans the globe. And as it has grown it has evolved; its users are constantly designing and implementing new features. Once it was a simple text-based system with poor usability; now the Internet also offers a wide variety of rich multimedia facilities that are ideally suited to novice computer users.

As a consequence, the users of the Internet have also changed. Once the preserve of highly skilled academics and technicians, the Internet can increasingly be used by anyone. This in turn has led to the Internet being adopted by businesses, which were quick to realize its potential for both saving and making money.

This redbook is about building business solutions that access the Internet from what IBM terms *pervasive computing devices*[1]. To successfully implement these types of solutions there are some specific challenges that businesses must address. In this book we talk about these solutions, and in particular discuss the challenges that are involved in creating the larger, more complex solutions that are typically deployed by enterprises and service providers.

1.1 The Internet

Since this redbook concentrates on the business use of the Internet (often simply referred to as e-business), let's initially consider how and why businesses use the Internet. Viewed as simplistically as possible, they use it for one of two main reasons:

- To save money, typically by carrying out an existing function more efficiently, and therefore more cheaply, or
- To make more money, by increasing their number of customers, retaining their existing customers, or increasing the frequency with which their customers use or purchase their products.

Over the years the degree to which companies can use the Internet has grown. Initially there was little opportunity to use the Internet for a variety of reasons. The major reasons were that the number of potential customers (both individuals and other businesses) who were using the Internet was low, the Internet was intrinsically unreliable, and there were few technologies that could be brought to bear.

The turning point was the introduction of the standard for the HTTP protocol, which brought into being the World Wide Web. This introduced a usable technology to the Internet, and as a result attracted more users to the Internet. As the number of individuals and businesses rapidly increased, the incentive for businesses to make use of the Internet increased as well. In parallel, other technologies (such as CGI, Java, Servlets, JSPs, etc.) that are now used to create e-business solutions were being invented and refined. The result of this has been that e-business capability has evolved through a series of phases.

[1] Pervasive computing is the group of enabling technologies that will allow the Internet to become truly accessible to anyone, anywhere, and at anytime. See 1.2, "Pervasive computing" on page 8, or `http://www.ibm.com/pvc` for more information.

Typically each phase is more complex than the last, but provides greater potential benefits to the business.

As businesses tend to be conservative in nature, early business users of the Internet moved from one phase to another over time as the capability became available and trusted. This is still the case, and most adopters still typically pass through one or more of these phases, though there is a class of business (the so-called dot-coms) that start by making use of the most advanced technologies available to gain a business advantage over their rivals.

1.1.1 Limited business integration

The simplest e-business solution is to use the Internet only to advertise a business's existence, usually providing a simple information service. This may comprise product information (the equivalent of online brochures), contact information, basic product pricing, etc. This type of offering is usually described as *static publishing* (or simple Web presence). This is illustrated on the left of Figure 1.

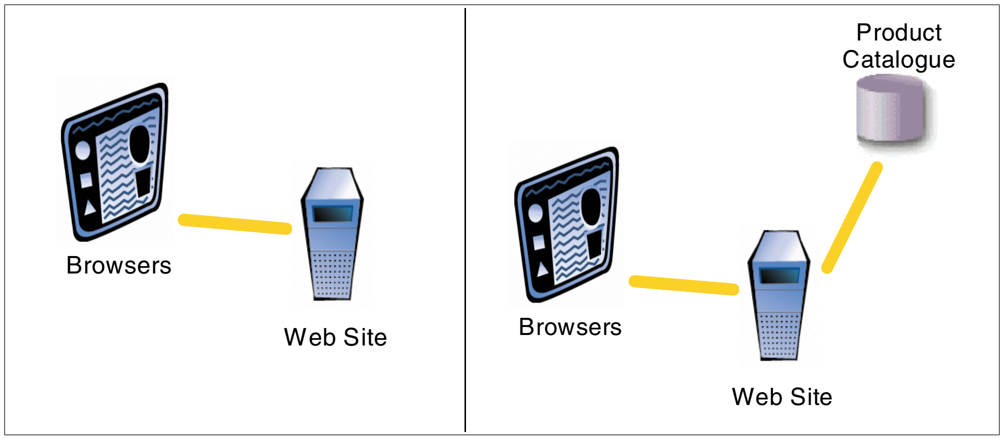


Figure 1. Simple e-business solutions

Slightly more advanced than this is where a business may actually provide some limited dynamic information to its customers (or potential customers) through its Web site. This will often be in the form of more detailed access to its product information, perhaps generating the prices dynamically based on some formula that takes into account the number of items to be ordered, taxation, and delivery costs. This is illustrated on the right of Figure 1.

1.1.2 Business integration

The next level of complexity is when businesses start to integrate their core business processes into their e-business solution. Usually we can divide this phase into two main parts: allowing customers to view the data within the core business processes, and allowing them to actually make updates via those core processes.

A good example of this might be where a business allows its customers to see if there are items in stock before he calls the sales desk. This can be simply achieved by providing an interface to the corporate inventory system.

Somewhat more complex again is where the business allows its customers to update its core business processes via the e-business system. This is typically the point at which a business begins to actually transact business over the Internet, allowing its customers to place orders through its e-business system. One example of this is integrated e-commerce, and a typical solution of this type is shown in Figure 2. Interfaces to both the inventory and order processing systems are required, as might be an interface to some type of billing system (which for clarity is not shown).

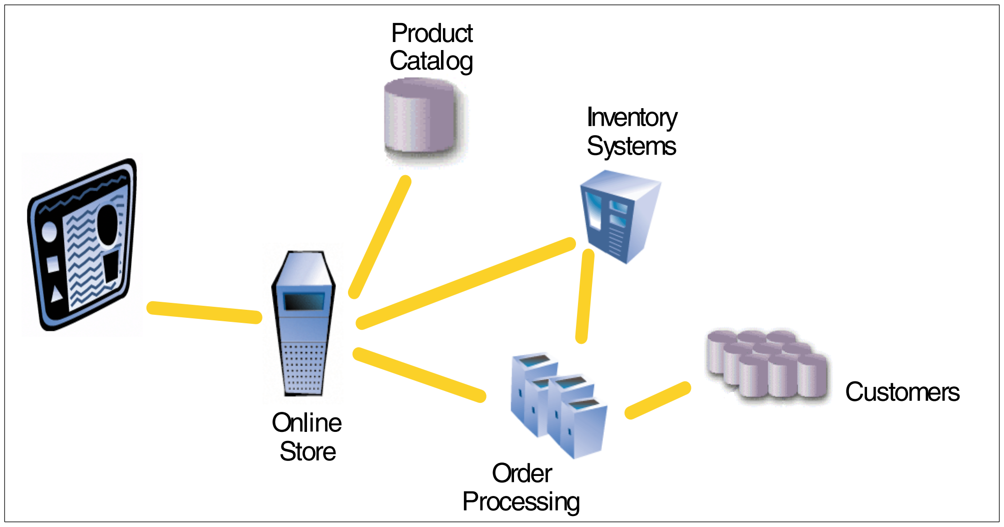


Figure 2. e-business solution integrated with core business systems

The final level of complexity at this stage is where a business also starts to integrate its e-business solution with other businesses' systems, typically using extranets. A good example of this is *supply chain management* (SCM). In the previous example, if an order cannot be fulfilled because the business has insufficient stock, there is a risk that the customer will buy from a competitor. To reduce that risk, the business can provide a date when it will be able to fulfill the order; it will still get the business if that date is acceptable to the customer. This benefits both the customer, who can place his order without additional effort, and the business, which receives the order even though it doesn't actually have stock at the time of the order.

To do this requires that the business integrate its core systems (particularly the inventory systems) with its suppliers' systems, and be able to calculate the time to create stock, given the outstanding orders. This is illustrated in Figure 3.

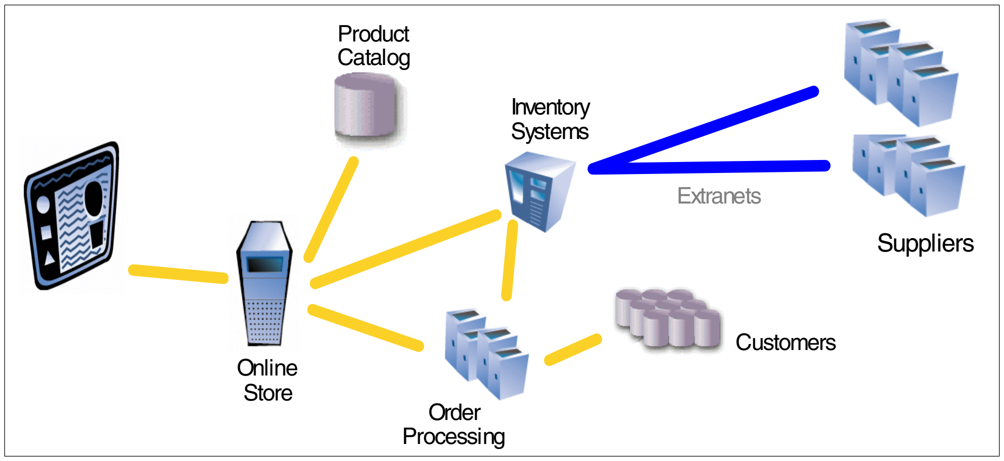


Figure 3. Integrating e-business solutions over extranets

1.1.3 Personalization

A feature often added to many types of solutions is personalization. This is where content or information is presented in a different way to one user of the solution than it is to another. This personalized content may be delivered by either the base solution or in the form of additional paths (such as e-mail, postal mail or telephony), as is appropriate to the solution provider's business.

The main advantages of personalization are that it allows a business to better target its customers, builds a better individual relationship with each customer, improves the customer's loyalty, and so increases the business done with each customer. Typical uses are up-selling and cross-selling strategies, provision of specific pricing policies or discount schemes, etc. As can be seen, this is a key technology for *customer relationship management* (CRM).

A sophisticated implementation normally requires a database of customer interactions and some way to analyze those interactions, plus mechanisms to interface to the customer more directly. This could be implemented as illustrated in Figure 4 on page 7.

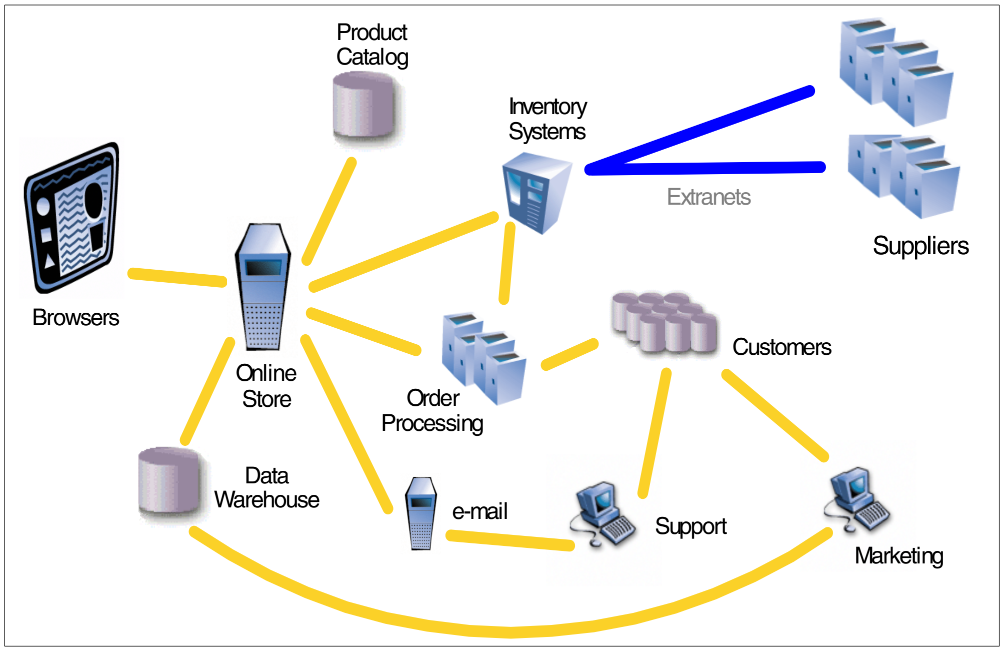


Figure 4. A fully integrated e-business system including CRM and targeted marketing

1.1.4 Aggregation

So far the types of e-business solutions have been particularly relevant to businesses whose business models are based on selling products. In this context the next development, known as *aggregation*, would be building a mall of aggregated shops, rather than a single shop.

However, this technique is even more appropriate to non-retail businesses whose business models are based on the provision of information. In this case aggregation is where many sources of data and services are pulled together into a form and location where they can be more readily accessed. This adds extra value to a solution simply because it allows the business to provide more information, hopefully in a more readily accessible form.

The originators of this idea were the original Internet search engine sites such as Yahoo, Lycos, and the like. They were already providing a vitally important service to users of the ever-growing PDAs, and as a consequence were “well-known locations” on the Internet. Their business models were (and are) largely driven by advertising revenue. To make money they needed to attract increasing numbers of visitors to their sites, which in turn required that the information and services that they offered were attractive to as broad a range of people as possible.

To achieve this the search engine sites started to add new and aggregated content to their sites, including news, weather, PDA-mail, travel services, and company directories (Yellow Pages) etc. Refer to Figure 5 on page 8 for an illustration of this type of solution.

The result of adding aggregation to their sites was that these businesses became the first *Internet Portals*. They have since been joined by both *Internet Service Providers* (ISPs) and *Application Service Providers* (ASPs) in aggregating content into competing portals.

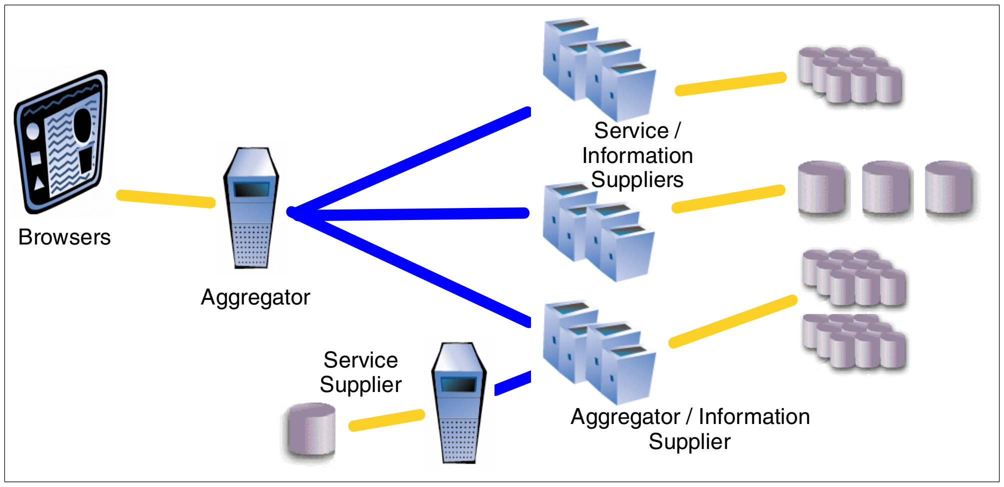

Browsers
Aggregator
Service / Information Suppliers
Service Supplier
Aggregator / Information Supplier

Figure 5. Aggregation in an e-business solution

1.1.5 Aggregation with personalization

As described in the previous section, aggregation allows a business to create a portal. However, when used on its own it only allows the creation of a rather simple portal. By combining the features of aggregation with significant levels of personalization it is possible to provide dramatically improved usability to the end users.

They can ensure that they are only presented with information that is of specific interest and relevance to them. This means that a personalized portal solution can add significantly more value to its data, and so its customers, than a non-personalized portal that simply aggregates data. This in turn means that more users will use the portal.

Additionally, by monitoring and understanding the personalization decisions that the portal users are making, it becomes possible to gather more information about the users, and further personalize the content, especially targeted dynamic content such as advertisements.

It is also worth mentioning *marketplaces* at this point. Marketplaces are a specific example of an e-business solution that makes use of both content aggregation and personalization with the specific intention of bringing together customers and businesses that wish to interact with one another. Although at this level there are functional similarities between portals and marketplaces, the detailed implementation will be radically different, and this is beyond the scope of this redbook.

1.2 Pervasive computing

Pervasive computing is a series of technologies that enable people to accomplish personal and professional tasks typically using a new class of portable, intelligent device. It gives people convenient access to information that is relevant to them at the time, whenever and wherever they are, with appropriate levels of security. The key feature of these devices is that they provide usability that is comparable to everyday devices such as telephones or domestic appliances, while still allowing users access to the information that they need, typically provided by today's e-business solutions over increasingly intelligent networks.

1.2.1 A brief history

Over the last decade, there has been a dramatic increase in the use of embedded computers within apparently common-place devices. This has led to a change from a world where computers are seen and used as distinct machines, to a world of sophisticated, computerized devices that are often neither perceived nor used as computers. By most estimates, the vast majority of computers are now of this type. Computers of varying capabilities can be found in telephones, microwave ovens, cash registers, cars, and a huge range of other devices and systems that we all use on an everyday basis.

In parallel with this growth in embedded computing there has been a massive trend towards the commonplace use of ubiquitous personal digital communications. This has been centered in Europe where the early decision to adopt a common digital cellular infrastructure based on *Global System for Mobile Communications (GSM)* technology was taken in the early 1990s. This system spread rapidly to most areas of the world apart from the USA and Japan, where local (and sometimes fragmented) standards were adopted instead.

The almost universal nature of the GSM infrastructure, in combination with an aggressive pricing model, ensured that the penetration of this technology was both rapid and extremely deep in Europe. As of mid-2000, adoption approaches 90% of the total population in some regions, and exceeds 50% in many. Exactly comparable figures for USA are not readily available, but while approximately 35% of the population use wireless communications, only about 15% have access to personal digital wireless communications.

In the mid-1990s a new class of small, user-centric computing devices began to be appear. Known as the *Personal Digital Assistant* (or PDA), these provided functionality that would normally be provided by paper diaries, address books and notebooks, and were initially popular only with highly computer-literate adopters of new technology. Largely these were viewed as expensive toys.

However, consumer acceptance of these PDAs (such as the Palm Pilot, IBM WorkPad, Psion 5, Sharp Zarus, etc.) has been very favorable, fueled to a considerable degree by the ability of users to create their own applications, which are often shared freely over the Internet. This has been a key strategy for the Palm Pilot in particular, which now commands well over 70% of the US PDA market[2].

In the last few years the newer PDAs have been designed around open architectures with embedded TCP/IP networking support and infrared communications ports allowing simple connection to suitably data-enabled digital cellular telephones. The combination of small devices, increased processing power, readily available networking support (usually mobile networking based on the use of the cellular network) resulted in the arrival of the first true pervasive computing devices, capable of adding real value to their users, and quickly losing the image of expensive executive toys.

Development now continues apace to continue to integrate networking support into many of the new and existing devices that contain embedded computers,

[2] Source: NPD Intelect, June 2000

using new technologies and open standards such as the *Open Systems Gateway initiative* (OSGi), *Bluetooth*, *fast IR, Wireless Ethernet* and more. This is resulting in new classes of devices such as the Internet appliance,- an Internet-capable device that has all the usability features of a normal household appliance (such as a television or washing machine) and yet communicates via open standards with e-business solutions to provide added value to its users.

1.2.2 The future

The chart in Figure 6 illustrates the expected growth in the use of the Internet from these new classes of device, along with the matching decrease in the use of the Internet from existing classes of devices, especially the “standard” desktop system. The key feature from this chart is that the largest growth is in the systems that are the most mobile - handheld systems and smart telephones - and convenient, such as the Internet appliances.

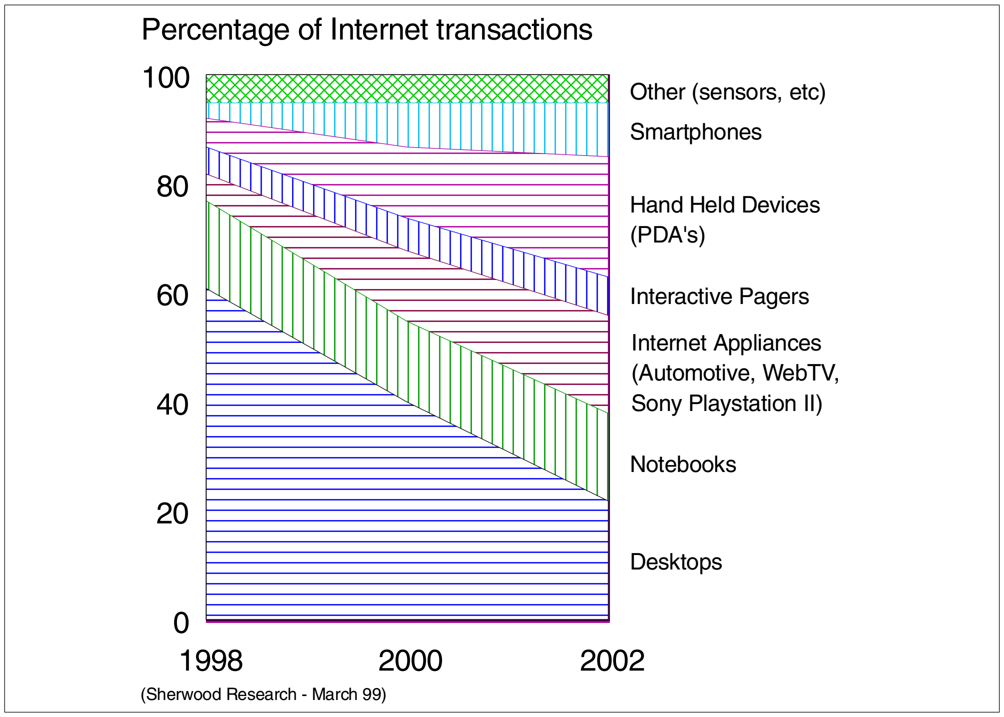


Figure 6. Changes in devices that are initiating Internet transactions

This increase in the popularity of mobile devices accessing the Internet is in tune with the view that work is no longer a place, but rather a state of being. The promise of pervasive computing is that it will simplify our lives by combining open standards-based applications with our everyday activities, allowing us to seamlessly mix our work and leisure activities in a way that traditional computing systems cannot achieve. In short, computing will no longer be a discrete activity bound to a desktop, but instead will be part of our everyday lives, wherever we may be.

Interestingly, the next developments are already occurring in this area, with major manufacturers integrating PDA functionality into cellular telephones. Indeed, these new devices bear little resemblance to cellular telephones, as the following picture (Figure 7 on page 11) illustrates. These new types of hybrid devices will further blur the distinction between smart telephones, that is, that are equipped with a Wireless Application Protocol (WAP) browser, and PDAs over the next few years.

Figure 7. New generations of PDA / telephones

Devices such as this provide obvious advantages in terms of usability and the timeliness of access to information. Together these provide businesses with the opportunity to provide services and information in a way that is more acceptable to the customer. This alone can result in significant improvements in customer satisfaction, and a commensurate improvement in brand loyalty and increased business, which has tangible benefits to any organization.

As these devices are built around open Internet-based standards, providing services and information over them will typically be implemented using self-care techniques that will also result in significant cost savings when compared to similar manual systems. The devices can also incorporate new features that will also allow new business opportunities to be realized.

Perhaps the most obvious example is provided by location-based services. Currently it is possible to understand which cellular telephony “cell” is providing service to a device, and through the use of Geographic Information Systems (GIS) it is possible to provide some coarse-grained location-based services, such as basic directions to the nearest branch of a restaurant chain. However, there are various technologies currently available for locating a device to an accuracy of a few meters, Global Positioning System (GPS) and base station triangulation being the most accurate. By using this information it will be possible to provide a wide variety of value-added services to the user, such as personalized satellite navigation, or calling a taxi to your location (even when you don’t know where you are!).

1.3 The challenge

The challenge is to implement solutions that provide the types (particularly the more complex types) of e-business solutions that are required by today's businesses, while providing access through the various types of pervasive devices that we have been discussing.

Although most of these pervasive devices communicate using Internet-based standard protocols, the extremely diverse nature of the devices means that there are still significant differences in the capabilities of both the devices and their network connectivity. For example, screen size, input mechanisms, information processing capability, and storage capacity all vary greatly, as can the network bandwidth.

As mobile communications in particular is still a relatively expensive, error-prone and low-capacity medium, some devices will be equipped with non-browser based communications facilities to support occasionally connected models of operation. These devices typically rely on synchronization techniques to exchange data with e-business solutions, and are based on either messaging or database synchronization.

Merging more complex e-business solutions with these new (and emerging) pervasive computing technologies to create new and innovative solutions requires a significant integration effort.

As an example, consider the creation of a portal that can be accessed via a variety of devices ranging from traditional Internet browsers, through interactive digital TV systems and Internet appliances to WAP-based telephones. This type of solution is sometimes referred to as a *multi-modal portal.* Functionally such a system can be viewed as illustrated in Figure 8.

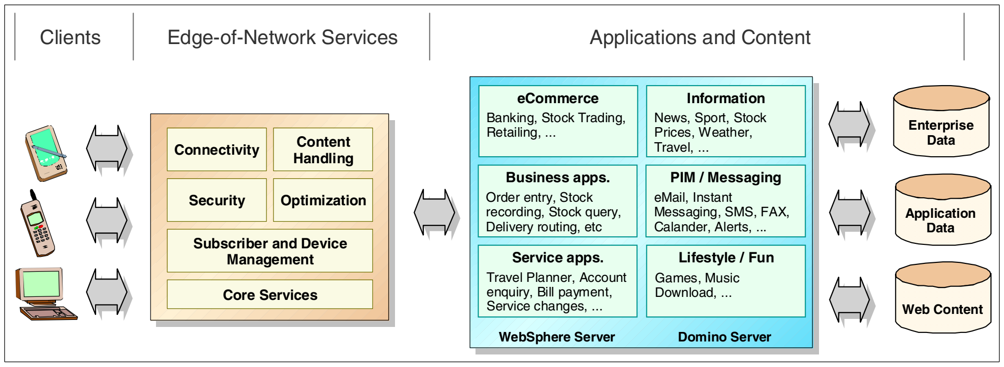

Clients
Edge-of-Network Services
Applications and Content
Connectivity
Content Handling
Security
Optimization
Subscriber and Device Management
Core Services
eCommerce
Banking, Stock Trading, Retailing, ...
Information
News, Sport, Stock Prices, Weather, Travel, ...
Business apps.
Order entry, Stock recording, Stock query, Delivery routing, etc
PIM / Messaging
eMail, Instant Messaging, SMS, FAX, Calander, Alerts, ...
Service apps.
Travel Planner, Account enquiry, Bill payment, Service changes, ...
Lifestyle / Fun
Games, Music Download, ...
WebSphere Server
Domino Server
Enterprise Data
Application Data
Web Content

Figure 8. Multi-modal Portal infrastructure, from a functional perspective

Each of the major functional building blocks contains many smaller functional or business requirements that will need to be addressed to create a workable solution. Each of these may need to be implemented by one or more products or applications all integrated together into a working whole, with assurance of performance, availability and scalability.

In order to provide the infrastructure for creating portals, a variety of servers are required that must be reliable, flexible, and scalable. These include some of the following core services:

- *Connectivity*, consisting of the core functions of the solution, including products such as HTTP servers, Web application servers, collaboration servers, database services, messaging services, load balancing and caching services and network connectivity services capable of supporting a wide range of network access technologies.
- *Portal Infrastructure*, consisting of systems to assist with consistent style (look and feel), branding initiatives, personalization, localization, and knowledge management systems.
- *Subscriber and Device Management*, to allow the provision (or self-provision) of portal users, access control, billing and self-care functions.
- *Customer Data Warehouse*, consisting of a repository of user information and preferences that can be data-mined. This allows information to be extracted, providing one-on-one marketing or the automated supply of appropriately targeted content to users.
- *Business Services*, which supply the underlying information and services to attract the users to the portal infrastructure.

Chapter 2. Product overview

As we mentioned in the first chapter, the major challenge that faces us is to implement the complex e-business solutions that are required by today's businesses, to enable access from the various types of new or emerging pervasive devices that are becoming increasingly common. At the same time we have to ensure that our solutions have high performance, high levels of availability and excellent scalability.

IBM WebSphere Everyplace Suite is the IBM solution for addressing the issues raised in Chapter 1, "Evolution of e-business" on page 3. As the name suggests, this is not a single product, but is rather an integrated suite of existing products, combined with additional software to provide close integration of all the product administration and configuration, and to implement a coherent security model for the entire e-business solution.

Everyplace Suite is ideal for organizations that wish to build solutions that are built around:

- Network access
- Adaptive network access
- Adaptive (multi-modal) portal

IBM WebSphere Everyplace Suite provides the functionality necessary to enable both network access, and application and content serving to multiple device types. It also provides the functionality to extend e-business applications to the new classes of pervasive computing devices discussed previously, including WAP phones, PDAs, Internet appliances and screenphones in addition to the large installed base of Internet browsers.

Everyplace Suite logically fits between your clients and the business applications and data that those clients wish to access. Because it is located on the edge of the network where your solution is implemented, we say that Everyplace Suite provides "edge-of-network" functionality to the solution.

Everyplace Suite is not part of the infrastructure that comprises those business applications and data, but rather it provides the infrastructure to enable many different devices to access those applications and data via networks. This is illustrated in Figure 9, which shows how Everyplace Suite logically fits into a deployment of a multi-channel, end-to-end e-business solution.

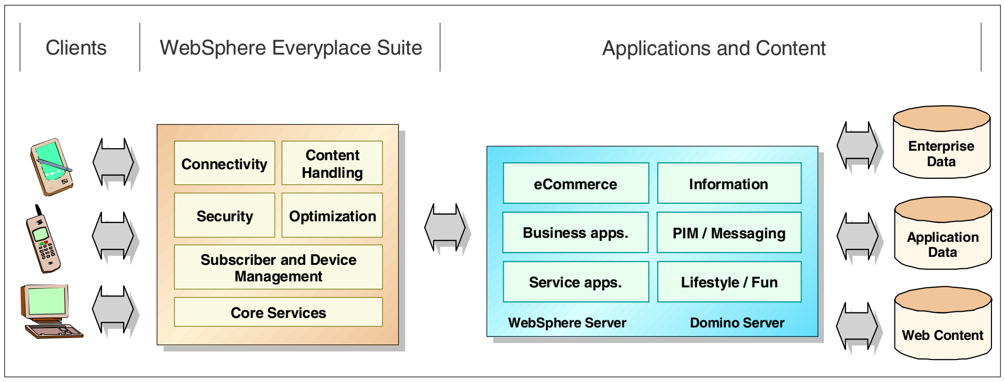


Figure 9. WebSphere everyplace suite, end-to-end solution

2.1 Functions in Everyplace Suite

If you refer back to Figure 9 on page 16, you can see that we broke the Everyplace Suite system into six functional areas; connectivity, content handling, security, optimization, subscriber and device management, and finally, core services.

If we examine these abstract functional areas in more detail, we can see more detailed functionality contained within each of them. Refer to Figure 10 for a description of the technologies that are contained within each major functional area.

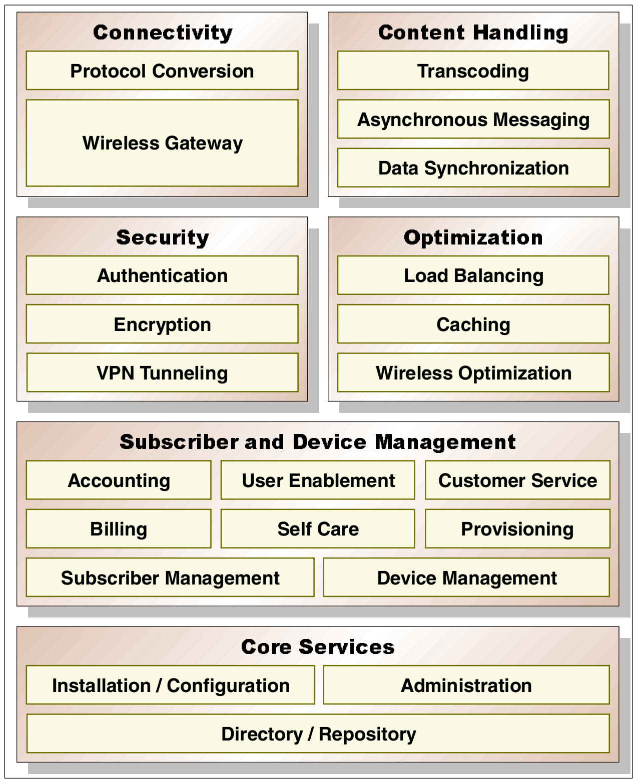


Figure 10. IBM WebSphere Everyplace Suite functions

Connectivity

Connectivity is the function that allows various types of device to connect to Everyplace Suite, and from Everyplace Suite into your business applications and data. Functionally there are two aspects to this; one is the provision of gateways to physically connect various types of networks to our end-to-end solution, the other is 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. The protocol conversion allows us to provide (say) a TCP/IP programming interface on a remote system, and then to flow the resulting TCP/IP packets over (say) the physical X.25 network that that remote device is connected to, and then out into our business applications.

Content handling

Content handling is the issue of moving the content from your e-business solutions (either applications or data) to the point where it is required. This typically has two main aspects; the moving and then ensuring that when the data is at the point where it is needed, it is in a form that can be used.

To that end, this can be broken down into three main areas:

- Transcoding

  Since pervasive computing devices come in all sizes and have widely varying display capabilities, transmitted content must be customized to fit the capabilities of the requesting device.

- 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 Suite 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 perform work offline, and connect to the network whenever it's convenient.

Security

IBM WebSphere Everyplace Suite provides an integrated security model that provides the user with a significantly enhanced end-user experience. The key feature is the ability to provide single sign-on across all the components of an end-to-end e-business solution, which fully integrates with standard security techniques such as virtual private network (VPN) technology, firewall protection, etc.

Optimization

As with all e-business solutions, performance is a critical issue. If the performance is poor it is easy for a customer to go elsewhere. Customer expectations are constantly increasing, and as the popularity of your solution increases it must be capable of scaling to cope with the increased demand. This is provided through the inclusion of caching and load balancing support to ensure high performance and scalability.

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 Suite includes Tivoli technology to manage subscribers and their devices, easing the burden of administration and system maintenance.

Base services
The base services provide the underlying framework on which the rest of Everyplace Suite is built. This includes the directory and repository services where all the data is stored, the installation, configuration and administration, plus all the other services that underpin them, including the HTTP servers, databases, etc.

2.2 Products in Everyplace Suite

In this section we briefly highlight how each of the previously discussed functional areas has been implemented in Everyplace Suite, showing which products implement which function. Clearly at the level that we are discussing the products and functions, this is not going to be exact; there will be some grey areas where products provide more than one function, or more than one function is implemented by the same product. None the less, this will provide you with a useful feel for how the products fit together.

The products that form the components of Everyplace Suite are:

- Everyplace Wireless Gateway 1.1
- Everyplace Authentication Server
- WebSphere Transcoding Publisher 1.1.2
- Tivoli Personalized Services Manager 1.1
- MQSeries Everyplace for Multiplatforms 1.0
- Mobile Data Synchronization Server
- WebSphere Edge Server - Caching Proxy (Web Traffic Express) 1.0
- WebSphere Edge Server - Load Balancer (Network Dispatcher) 1.0
- IBM SecureWay Directory 3.2
- IBM WebSphere Application Server 3.5
- IBM HTTP Server 1.3.12
- IBM DB2 Universal Database 7.1
- IBM WebSphere Everyplace Suite Install
- IBM WebSphere Everyplace Suite Administration Console

Refer to Figure 11 on page 19 for a graphical representation of how the various products have been used to implement the functionality that we discussed in “Functions in Everyplace Suite” on page 16.

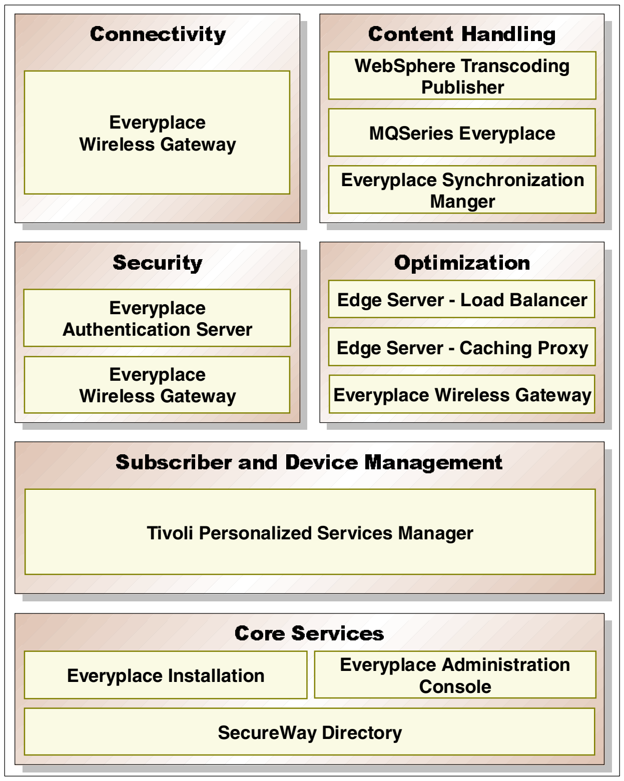

Connectivity
Everyplace Wireless Gateway
Content Handling
WebSphere Transcoding Publisher
MQSeries Everyplace
Everyplace Synchronization Manger
Security
Everyplace Authentication Server
Everyplace Wireless Gateway
Optimization
Edge Server - Load Balancer
Edge Server - Caching Proxy
Everyplace Wireless Gateway
Subscriber and Device Management
Tivoli Personalized Services Manager
Core Services
Everyplace Installation
Everyplace Administration Console
SecureWay Directory

Figure 11. IBM WebSphere Everyplace Suite products

2.2.1 Connectivity

IBM Everyplace Wireless Gateway provides secure wired and wireless connectivity between your enterprise network and the whatever external communications networks that you need to support, for example, GSM, CDMA, TDMA, X.25, etc. It also implements protocol translation, and provides support for interfacing to short messaging centers via a series of APIs.

Refer to Chapter 7, “Supporting wireless devices” on page 89 for more information.

2.2.2 Content handling

IBM WebSphere Transcoding Publisher transforms one form of content into another so that it can be presented on a device that is different from the originally intended target. IBM WebSphere Transcoding Publisher performs this transformation automatically and on-the-fly, reducing or eliminating the need to maintain multiple versions of content. A good example of this is changing HTML content that is intended for desktop PCs into WML content that is suitable for displaying on a WAP-enabled mobile phone.

Refer to Chapter 8, “Transcoding Web application content” on page 119 for more information.

IBM MQSeries Everyplace enables pervasive devices to participate in commercial messaging, sending messages between applications, and assuring their delivery (once and only once), in a secure and highly efficient manner, operating in both connected and disconnected scenarios.

Refer to Chapter 10, “Pervasive messaging and queuing” on page 171 for more information.

IBM Everyplace Synchronization Manager is another complementary technology to allow pervasive devices to work in semi-connected modes. It enables pervasive devices to operate applications “offline”, and synchronize the results of their activities with a server database when connectivity is re-established.

IBM Everyplace Synchronization Manager is not covered in detail in this redbook, because at the time we were writing it, this component was not available to us.

2.2.3 Security

IBM Everyplace Authentication Server forms the core of the Everyplace Suite security functionality. It provides user and device authentication capabilities that together enable a single, device-independent user sign-on. It also provides the pass-through of authentication information to business application and data servers.

Refer to Chapter 6, “Authentication” on page 69 for more information.

IBM Everyplace Wireless Gateway provides Virtual Private Network support, which enables an enterprise to extend its private intranet across a public network, such as the Internet, creating a secure private connection by way of a private IP tunnel.

Refer to Chapter 7, “Supporting wireless devices” on page 89 for more information.

2.2.4 Optimization

Edge Server - Load Balancer and *Caching Proxy* together provide:

- Highly scalable caching functions that reduce network bandwidth costs and dramatically improve response times when fetching data from application and data servers.
- Dynamic monitoring and balancing of load across components of the end-to-end solution.

Refer to Chapter 11, “Caching Proxy” on page 183, and Chapter 12, “Load Balancer” on page 203 for more information.

*IBM Everyplace Wireless Gateway*contributes to the optimization of the solution by compressing and optimizing the data flows across the networks that it manages, reducing the network bandwidth requirements, and speeding the response times.

Refer to Chapter 7, “Supporting wireless devices” on page 89 for more information.

2.2.5 Subscriber and device management

Tivoli Personalized Services Manager (TPSM) provides a comprehensive set of management services to the solution, including content personalization, subscriber enrollment, customer care and customer self-care, report generation, and interfaces to external billing systems. It also includes the ability to distribute and update software and data to devices, and provide status on the system availability.

Refer to Chapter 9, “Subscriber and device management” on page 135 for more information.

2.2.6 Base services

IBM SecureWay Directory is the central LDAP directory that contains information related to users, devices, and networks, in addition to the system configuration information. This directory makes it easy for the various components of WebSphere Everyplace Suite (and indeed, any other server that is added to the configuration) to access the information, without having to replicate the data in other repositories.

Everyplace Administration Console provides a single console for system administrators to perform installation and diagnostic procedures, administrative procedures, and system maintenance procedures.

Everyplace Suite Installation and Everyplace Suite Configuration provide a single graphical interface through which coordinated install and configuration of all the various Everyplace Suite components can be managed. It maintains common configuration information within an LDAP directory, allowing installation and configuration of subsequent components to automatically take into account the installation options chosen for earlier components.

Everyplace Suite Installation and Everyplace Suite Configuration are not covered in this redbook, which is aimed primarily at architects and solutions designers.

Chapter 3. Architecture

This chapter introduces a high-level architecture, showing how the components of IBM WebSphere Everyplace Suite 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 *suggest* here. 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 prerequisite components.

Note

All figures in this chapter illustrate suggested *logical* architectures. A *physical* implementation is likely to deploy multiple components on one machine and to duplicate components on multiple machines.

As the complete architecture template is quite complex, we have divided it into four “layers”:

- Core components
- Central repositories
- Performance - caches
- Availability - dispatchers and clusters

We discuss these in more detail in the following sections. Scalability issues in particular are addressed Chapter 4, “Performance and scalability” on page 37.

3.1 Overview

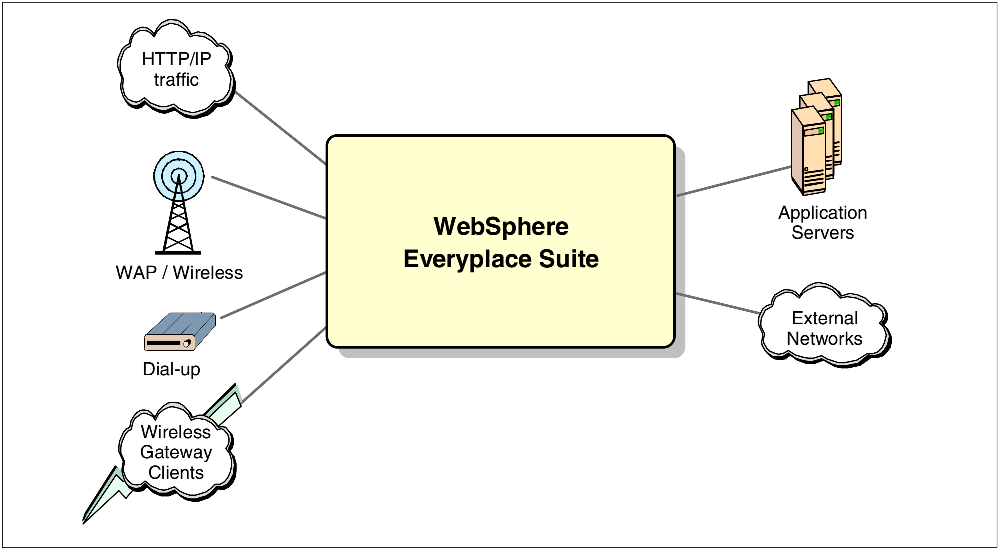


Figure 12. IBM WebSphere Everyplace Suite(WES) architecture - input and output paths

Clients can connect to an IBM WebSphere Everyplace Suite solution in two ways:

- **Via HTTP / IP**

  This is typically used by traffic from the Internet, intranet, and third-party gateways. Physically the connection can be a directly attached LAN or it can be a network attached through a router.

- **Via other protocols**

  This includes several types of wireless and wireline networks, including Wireless Application Protocol (WAP), dial-up connections and PCs or other devices using IBM Everyplace Wireless Gateway client.

Application servers and content can either reside locally inside the Everyplace Suite domain, or be located on an external network. Everyplace Suite can be used to provide access to both:

- **Application servers inside Everyplace Suite domain**

  A solution built around application servers and services that reside within the same domain, supporting single sign-on. These can serve content residing locally (such as your own product catalog) or externally (such as external data feeds that are invisible to the user).

- **External networks**

  This is directing traffic to other sites, and is similar to the operation of an Internet Service Provider. However, it could also be used in other business models (for example an enterprise offering intranet access to employees using a variety of different devices).

3.2 Core components

IBM WebSphere Everyplace Suite is an “edge of network” product. By this, we mean that it forms an interface between network(s) and core business applications. Clients can 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 Suite 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.

Based on our experience of building e-business solutions for IBM customers, we foresee that Everyplace Suite will principally be used in three main 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 Suite 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.

3.2.1 Network access

Providing large-scale network access is a common business requirement, typically implemented by Internet Service Providers. 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 IBM Everyplace Wireless Gateway clients.

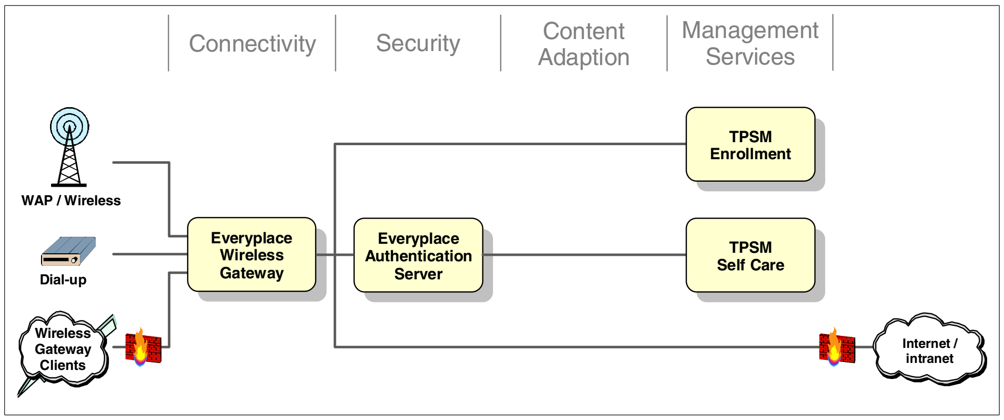


Figure 13. WES core architecture - Network access

The core components of an Everyplace Suite architecture for this model are shown in Figure 13. The central component is the IBM Everyplace 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 Suite servers 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, as by definition, self-enrollment is meaningful only when carried out by a user who is unknown to the system.

Other user services offered would be handled by the TPSM 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 the IBM Everyplace Authentication Server.

3.2.2 Adaptive network access

Either Internet Service Providers or enterprises may wish to extend their services from simply providing network access to providing adaptation of the content that is retrieved using their network access.

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 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 to the images, and the removal of unnecessary information, such as comments or spaces in the markup language.

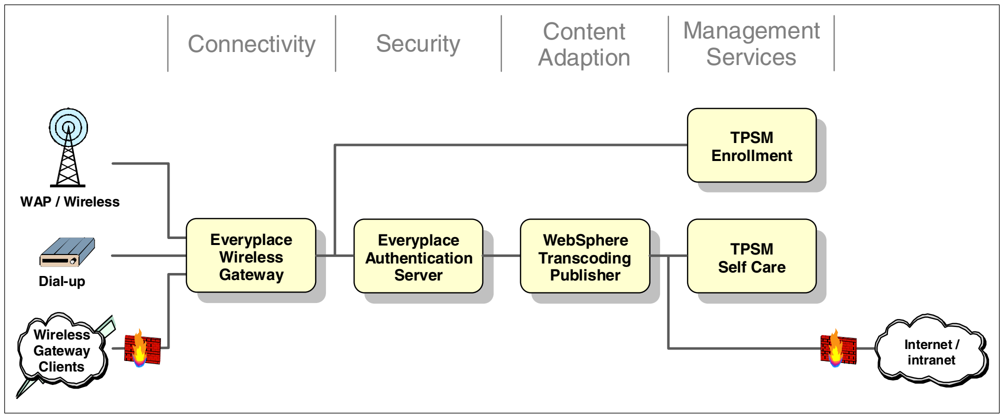


Figure 14. WES core architecture - adaptive network access

As with the basic network access model, the access to the network itself is still provided by the IBM Everyplace Wireless Gateway, and the services for enrollment and self-care are carried out by components of Tivoli Personalized Services Manager.

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

The main use of the IBM Everyplace Authentication Server is to provide security service to the solution. However, the Authentication Server 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. Further information about adaptation can be found in Chapter 8, “Transcoding Web application content” on page 119.

3.2.3 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 Suite is *not* a platform for building core business applications; it is a framework for building secure, integrated, adaptive access to a portal, while also allowing the adaptation of the content that is to be aggregated into that portal. See Chapter 1, “Evolution of e-business” on page 3 for a discussion of the definition of and requirements for portals.

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

This model is also equally applicable either 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.

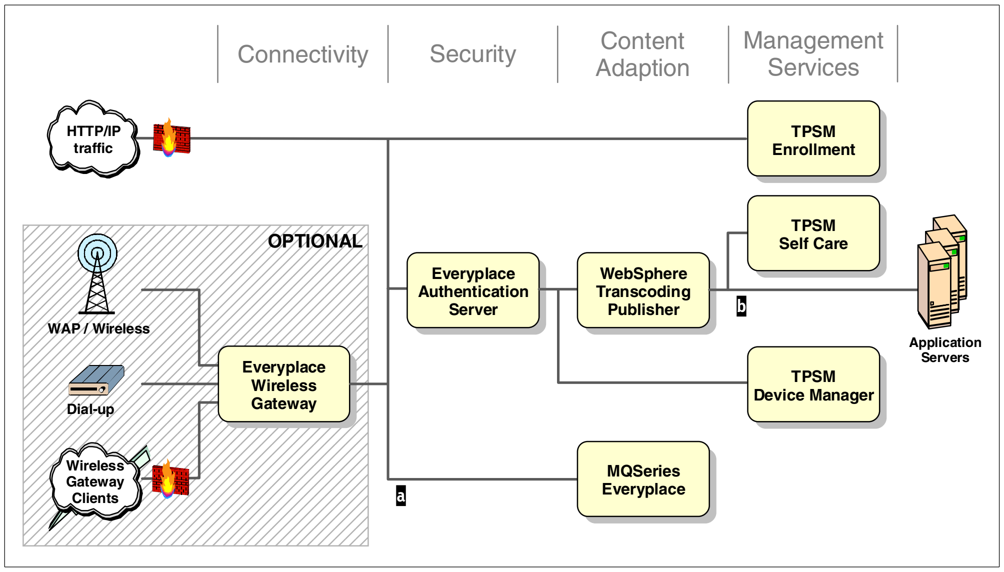


Figure 15. WES core architecture - Adaptive (multi-modal) portal

Here the Everyplace Suite components form an infrastructure that is the front-end to one or more application servers, as illustrated in Figure 15. 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 IBM Everyplace Wireless Gateway.

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

There are, however, two notable exceptions:

- TPSM enrollment

  As with the earlier models, the enrollment component of TPSM 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 the Authentication Server.

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. However, as always, 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 services.

Next we find the core business applications, information services, and the TPSM self-care component. These will all rely on Authentication Server 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 15. Clearly, the choice of attachment points determines if content adaptation is available or not. Firewalls may also be added as appropriate.

3.3 Unauthenticated access

Sometimes it is necessary to provide unauthenticated users with access to public resources, either application servers or Everyplace Suite components such as TPSM 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.

3.3.1 Dual security zones

The simplest way to resolve this is to duplicate the infrastructure components as necessary. If you consider that TPSM enrollment is a public service, then you can see that all of our previous models took this approach. We placed the TPSM 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 16.

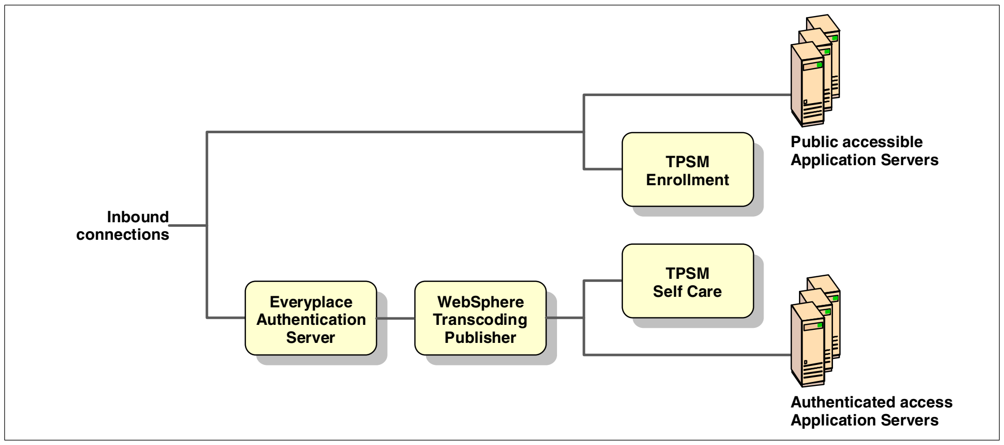

Public accessible Application Servers
TPSM Enrollment
Inbound connections
TPSM Self Care
Everyplace Authentication Server
WebSphere Transcoding Publisher
Authenticated access Application Servers

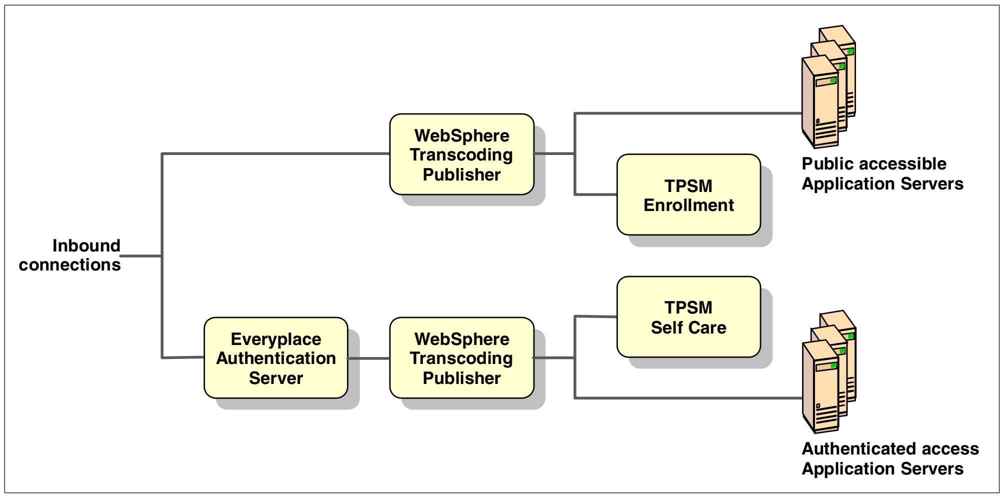
Figure 16. Unauthenticated access - dual security zones

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.

3.3.2 Dual security zones with adaptation

The dual security zone approach works well, but without enhancement it cannot easily support multiple device types, as 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.



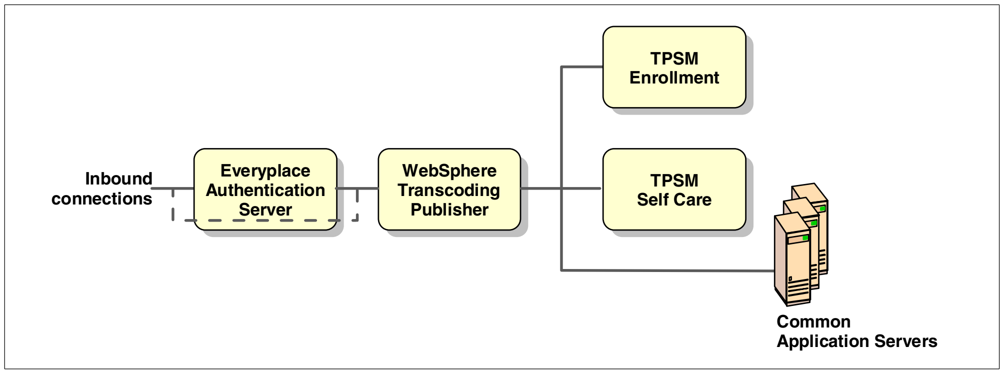
Inbound connections
WebSphere Transcoding Publisher
TPSM Enrollment
Public accessible Application Servers
TPSM Self Care
Everyplace Authentication Server
WebSphere Transcoding Publisher
Authenticated access Application Servers

Figure 17. 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.

3.3.3 Shared security zone with adaptation

An alternative to the previous options is to loosen the EAS security so that users who are unauthenticated by Everyplace Suite may access the same infrastructure as authenticated users. This requires changes to the default configuration for Authentication Server.



TPSM Enrollment
Inbound connections
Everyplace Authentication Server
WebSphere Transcoding Publisher
TPSM Self Care
Common Application Servers

Figure 18. 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 Authentication Server and setting access control lists (ACLs) in the application servers.

The downside to this approach is that security configuration must be maintained in two places: the ACLs as usual, but now also in the Authentication Server.

For details on how to achieve this, refer to Chapter 6, “Authentication” on page 69.

3.4 Central repositories

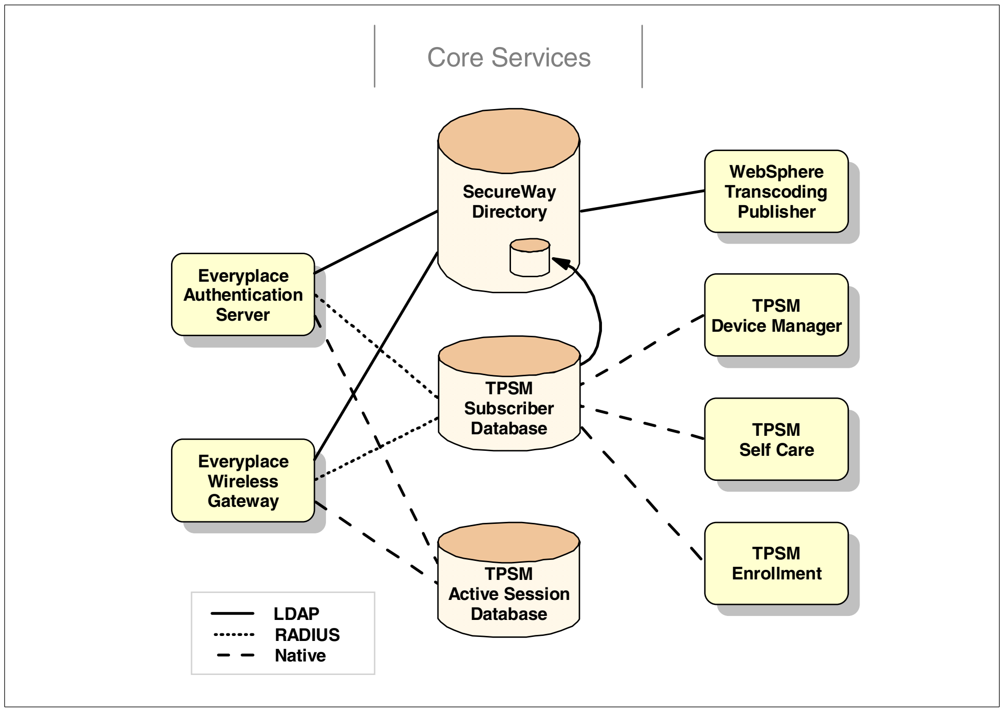


Figure 19. WES architecture - Central repositories

The Lightweight Directory Access Protocol (LDAP) services provided by IBM SecureWay Directory is the central repository for data, around which all of Everyplace Suite is built. Information describing the infrastructure and component configuration, in addition to operational data, is kept there.

User profiles are stored in LDAP and are accessible both to Everyplace Suite components and to other applications. However, the TPSM Subscriber Database is the master copy of all the user profiling and configuration data, and replicates changes to the LDAP repository as they occur. This approach allows all the components access to the sophisticated data model that has already been implemented in TPSM.

Access and authentication are achieved using the user name and password from the TPSM Subscriber Database, accessed via the RADIUS protocol. The Active Session Table (AST), maintains the affinity between an authenticated session, the logged-in user and his client device. Specialized access to the AST has been implemented and optimized using native database protocols. Together these databases provide a single sign-on facility.

IBM DB2 Universal Database is recommended as the underlying database for both the TPSM databases and IBM SecureWay Directory. Some components in Everyplace Suite either require or recommend that they be provided with direct access to a DB2 system. In a typical solution we would expect to implement multiple database instances on multiple machines.

Refer to the individual component chapters for further information on their use of the Everyplace Suite databases.

3.5 Performance - caches

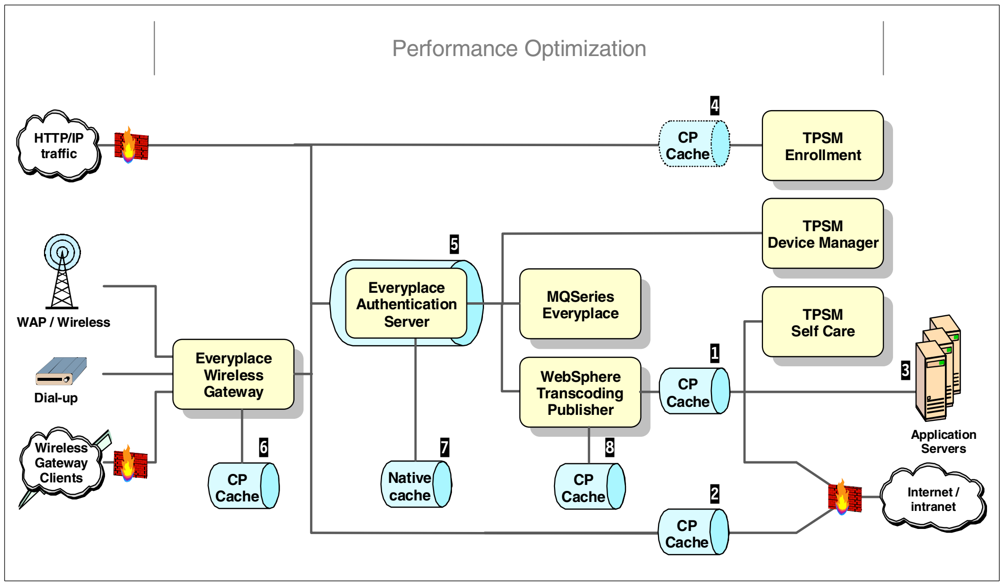


Figure 20. WES architecture - performance

Although performance is affected by all the components of an implementation, this section will provide a high-level overview of using the Edge Server - Caching Proxy within a Everyplace Suite context.

Figure 20 shows where the caching function can most beneficially be used. You will rarely configure Edge Server - Caching Proxy at all the points shown, even though a single instance can in some circumstances be shared. Notice three fundamentally different uses of Web Traffic Express in the figure:

In-stream cache	Caching Proxy is a part of the main traffic route doing caching of content passing by.
Component host	Caching Proxy is used as an application server doing rule-based launching of other components.
Application value-add	A component uses Caching Proxy as a plug-in for specialized caching.

The different uses of Edge Server - Caching Proxy all conform to sophisticated rule-based configuration, letting an administrator tune and adjust the function of Caching Proxy at several levels to achieve the best performance.

3.5.1 In-stream cache

A cache provides benefits where the *same content* is repeatedly retrieved from a relatively slow source. The cache can be positioned in front of that slow source (perhaps a Web server inside the domain), or the cache can be used in front of a whole network (such as the Internet) to generally optimize access to any site. As illustrated in Figure 20 this could be the case in front of:

1 Content to be transcoded[1]:

- Application servers inside the Everyplace Suite domain
- TPSM Self Care component
- Internet / intranet content with adaptation

2 Internet / intranet accessed with no content adaptation

3 Back-end data source providing content for hosted services or applications (not shown in the figure - refer to 8.3, "Building with transcoders" on page 122)

4 Possibly in front of TPSM enrollment, if this component is placed as shown

3.5.2 Component host

Currently only one component in the Suite is written to be hosted and launched by Web Traffic Express:

5 IBM Everyplace Authentication Server

This component will be deployed centrally in the main HTTP traffic route.

3.5.3 Application value-add

Three WES components are able to use caching in a more controlled, tailored manner. Specific requests and replies are generated by the exploiting application:

6 IBM Everyplace Wireless Gateway uses Caching Proxy for caching encoded WAP content.

7 IBM Everyplace Authentication Server uses a built-in cache for caching active sessions.

8 IBM WebSphere Transcoding Publisher uses Caching Proxy for caching transcoded content.

See the individual chapters on these three components for details.

3.6 Availability - dispatchers and clusters

The following states our subjective conclusion and provide a *general* recommendation for the individual Everyplace Suite as illustrated in Figure 21. Availability should be seen as an end-to-end issue, and this section is to be seen only as a guideline, not a definite answer. Most WES components support both clustering with the Edge Server - Load Balancer and hardware failover using HACMP on IBM RS/6000; some even provide a built-in solution to increase availability. We will not discuss in detail the alternatives for each component.

[1] If transcoding is not utilized, this content caching function could be set up on the same instance providing hosting for Authentication Server (WTE 5 in the figure).

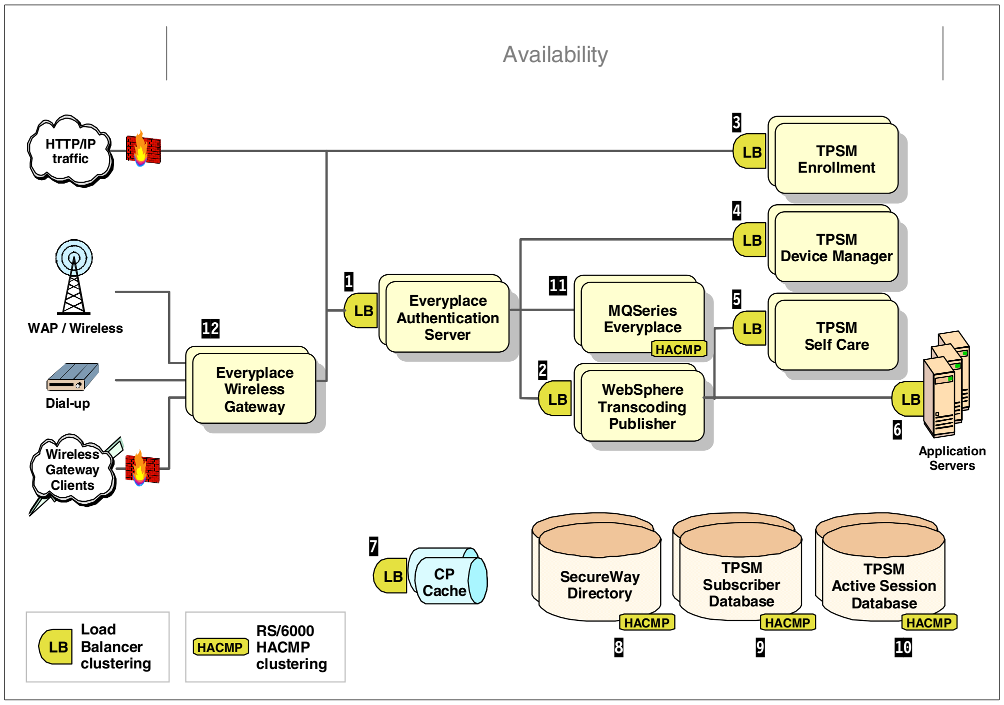


Figure 21. WES architecture - Availability

IBM WebSphere Everyplace Suite includes the Edge Server - Load Balancer as the key component to improve availability and scalability. Implementing a solution on IBM RS/6000 hardware opens up another choice for availability: High Availability Cluster Multi-Processing (HACMP).

Availability, when seen in the context of the entire architecture, requires more than simply introducing dispatchers in front of every component. Sometimes hardware clustering may provide a better result, and sometimes failover support is built into the software, eliminating the extra cost and network latency introduced by a dispatcher.

Load Balancer

In general, components that are facing incoming HTTP traffic over TCP/IP are well suited for use with the Load Balancer. This is true for:

1 IBM Everyplace Authentication Server

2 IBM WebSphere Transcoding Publisher

3 Tivoli Personalized Services Manager, Enrollment

4 Tivoli Personalized Services Manager, Self Care

5 Tivoli Device Manager Server

6 Other application servers inside the Everyplace Suite domain

7 Edge Server - Caching Proxy (depending on its use and configuration)

Often these Web servers and intermediaries use significant processing resources, which could raise scalability issues. Using Load Balancer also gives the benefit of providing “horizontal” scalability, that is the ability to balance the load between a number of servers.

The Load Balancer product and the use of it will be discussed in Chapter 12, “Load Balancer” on page 203.

HACMP for databases

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
- 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 big) 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:

8 IBM SecureWay Directory

9 TPSM Subscriber database

10 TPSM Active Session Table database

Alternatives for specialized servers

IBM Everyplace Wireless Gateway and IBM Everyplace Authentication Server both connect to the two TPSM databases using a RADIUS server and an AST server, both of which are a part of TPSM (not shown in Figure 21). For these servers there are three availability options (this list is not prioritized):

- Let the RADIUS and AST servers use HACMP as their underlying TPSM databases.
- Use the failover configuration option built into both the Wireless Gateway and the Authentication Server. This allows you to define both a primary and a secondary RADIUS and AST server. The secondary server will be used if the primary is not responding.
- Pair together every Authentication Server with a specific instance of both a RADIUS and an AST server. The Load Balancer in front of Authentication Server should then make its dispatching based on the status of all three components. For example, if an AST server becomes unavailable, no requests should be dispatched to the Authentication Server using it. To achieve this, you will have to develop and engage a Customized Advisor providing Load Balancer with the required information.

HACMP for IBM MQSeries Everyplace

For one component in the Everyplace Suite, the recommendation is less obvious:

11 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.

Built-in clustering for IBM Everyplace Wireless Gateway

12 The IBM Everyplace Wireless Gateway has built-in clustering and fai-over functions supporting all protocols. This specialized support is preferable to Load Balancer, since that supports IP only.

HACMP could be used as a supplement or alternative to the built-in clustering, but is less specialized and does not improve scalability. Consequently the general recommendation is to use the built-in clustering alone. This is discussed further in Chapter 7, “Supporting wireless devices” on page 89.

Chapter 4. Performance and scalability

In this chapter we discuss architecture, design, development and deployment issues that affect the performance and scalability of large-scale e-business solutions. Consequently, everything that we discuss in this chapter is directly applicable to a WES-based solution, even though nothing is specific to WES. However, since WES is likely to be utilized in some of the world's largest e-business opportunities, we feel it is important to repeat some general advice on architecting, designing and deploying solutions that will both perform and scale well.

For decades, IBM has been building solutions, including the hardware, software, and networks needed to run them, with performance in mind. Value-add networks and the Web have been around for decades and years respectively, but the explosive growth in the areas of e-business and e-commerce is a very recent trend. These e-business solutions represent a new dimension simply because the computing model in this environment is based on many new technologies.

The technology and tools may be too new for reliable performance and scaling estimates and projections to be made. How do you predict performance of Web-based applications? How do you understand the potential performance and scalability issues of building your solution in an environment where everything associated with a project is measured in "Web years[1]"? And finally, how do you measure end-to-end performance anyway? New metrics, new bottlenecks, and new demands for performance abound in this arena.

Many of us have encountered e-business solutions that were architected without well-understood requirements for performance. Without clear performance objectives during the initial architecture definition and the high-level design, the performance of the solution becomes an afterthought.

We now have enough experience to know that these projects often fail when they are deployed,- not because they cannot deliver the required business function, but because they cannot service the demand generated by their customer community in a satisfactory way. Subsequent attempts to solve the resulting performance problems are then made after hardware and software products have been selected and the application code developed. Room for maneuvering at this stage of a project is extremely limited, and resolving the issues can be an unpleasant experience.

A key message is that to create successful solutions we must include performance considerations at the architecture and design phases, and use appropriate methodologies throughout the entire development process to ensure the overall success of our solutions.

4.1 Definitions

In e-business systems, as with any other type of system, performance and scalability are closely linked with and vitally related to capacity. Talking about one without reference to the others makes little or no sense. However, be aware that

[1] Currently Web-based solutions are being architected, deployed, and maintained on an accelerated schedule when compared to "normal" solutions. The accepted wisdom is that schedules for Web-based solutions are typically four times more aggressive than normal. Hence a "Web-year" is actually comparable to 3 months of "normal product" time.

scalability, performance, and capacity are terms that are frequently used synonymously.

It is vital to understand that although they are related to one another, they each provide a different perspective, or view, of the overall picture. So, before we go too much further, here are some definitions.

4.1.1 Capacity

Capacity is the maximum output of a given system or component. In an e-business context capacity describes how many “operations” (transactions, “hits”, etc.) can be completed in a given unit of time by a given system or component while maintaining its business objectives.

4.1.2 Performance

At its simplest level, performance is a measure of how quickly work is completed. In an e-business context, performance is typically defined as the response time for the end user, simply because that is the measurement that has the most effect on the end users’ experiences and hence their satisfaction with the solution.

Important

A major advantage of measuring the overall response time is that it is a simple measurement to make. However, in e-business solutions that use the Internet, a medium over which you have no control, it is important that:

1. Performance measurements should be taken at the point where the e-business solution is attached to the Internet, and,
2. Additional allowances should be made for some average level of network delay and latency that will be introduced by the Internet connections between the measuring point and the end user.

4.1.3 Scalability

scalability is the capability of a system (or component) to readily adapt to a greater or lesser intensity of use, volume, or demand while still meeting business objectives. Typically the major business objective in this context will be providing an acceptable level of performance, but other requirements such as maintaining appropriate levels of availability, manageability, and security are also likely to be necessary.

4.2 Designing for performance and scalability

As a direct consequence of performance being measured as the systems end to end response time, a “performance problem” is usually perceived as a “slow” response time when a user takes one or more actions using the system. These actions may be to request a page of simple static information or carry out a complex series of interlocking processes that store or analyze information on the user’s behalf.

The user, quite rightly, has no concept of how much work or how many systems are actually involved in processing his request; simply that the response time is inadequate. An apparently “simple” transaction from the user’s perspective may turn out to be a very complex series of data flows involving multiple systems in multiple locations. It may cause processing to cascade through many systems before a summarized result is presented to the user.

Unfortunately sometimes not even the application developers and systems support teams understand exactly what work and systems are involved in each of these transactions. Today's highly integrated e-business solutions are increasingly complex, and we need to adopt an approach that helps us to understand the performance and scalability issues that surround them.

A key concept present in almost any formal methodology is that of breaking a complex problem down into smaller, simpler, more readily understandable pieces. We can apply this approach to our e-business solutions, breaking them down into a series of components or "building blocks". For simplicity we can break almost any e-business solution down into the following high-level components:

- Client
- Internet / network
- Web servers
- Application and integration servers
- Connectors
- Core business systems (Enterprise systems)

Refer to Figure 22 for a view of these components.

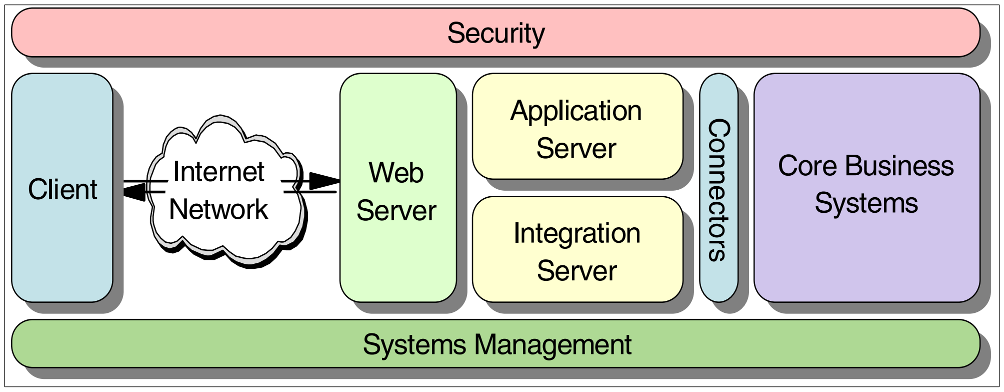


Figure 22. Components in a typical e-business solution

Adopting this approach allows us to move away from the "long response time" description of performance to a meaningful decomposition and analysis of each component and its performance and scalability requirements. Clearly there are still some system-wide considerations such as security, systems management and load balancing that span components, but this approach often simplifies consideration of some of these issues, too.

Consider the diagram in Figure 23 on page 40, which shows a single flow of data through the components of a (relatively) simple e-business solution with only two back-end servers.

Each line on the diagram represents something that will require time, and so will affect the performance of the solution.

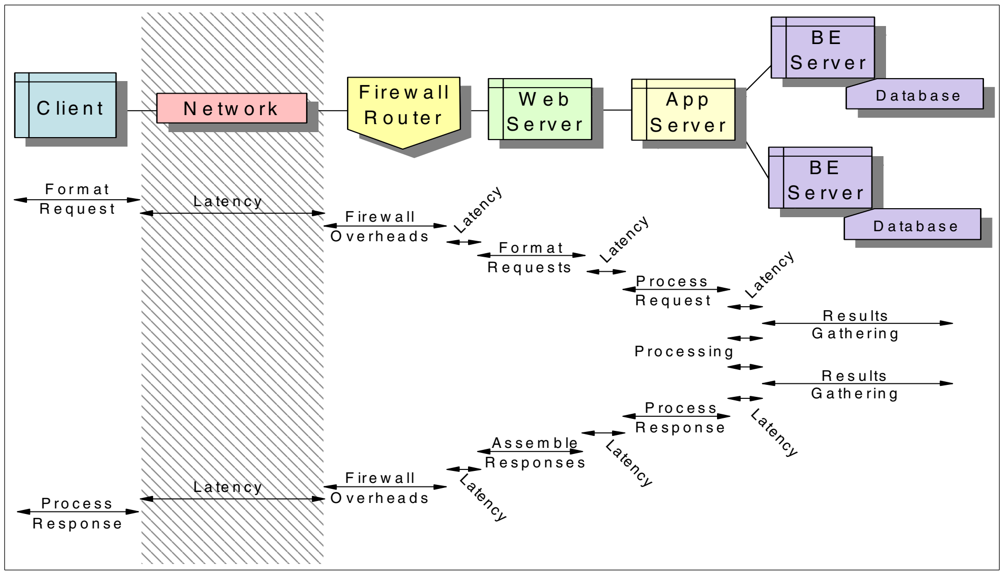


Figure 23. Latency in an e-business solution

Clearly this could be a solution that is to be deployed using the Internet. In that case the Internet portion of the solution is essentially out of your control from the perspective of affecting the performance and scalability of the solution. Alternatively, if this were an intranet solution then you will typically expect to either have control over the whole network infrastructure, or be able to make more precise predictions of its performance.

We represented this in our diagram by shading the portion that may or may not be under your control. This shaded portion of the solution is referred to in point 2, in the box labelled “Important” on page 38, and is the section of the network that you must make an allowance for, but typically will not be able to measure in an Internet solution.

So, how does this help you to improve the performance and scalability of your solution? The answer lies in the use of an *end-to-end performance budget.*

4.2.1 End-to-end performance budget

The end-to-end performance budget is a metric that you should establish before you start your design. It is derived from the business requirements for the end-to-end performance of the overall solution, and knowledge of the specific scenario and products that exist within the solution.

Simplistically, you take your end-to-end response time, and if appropriate remove any allowance for using the Internet. You then divide your solution into its main components and carry out some “educated guesswork”.

Assign a nominal percentage of the total time to each component, based on your knowledge of the work to be done in each component, your network characteristics etc. For an example, refer to Figure 24 on page 41.

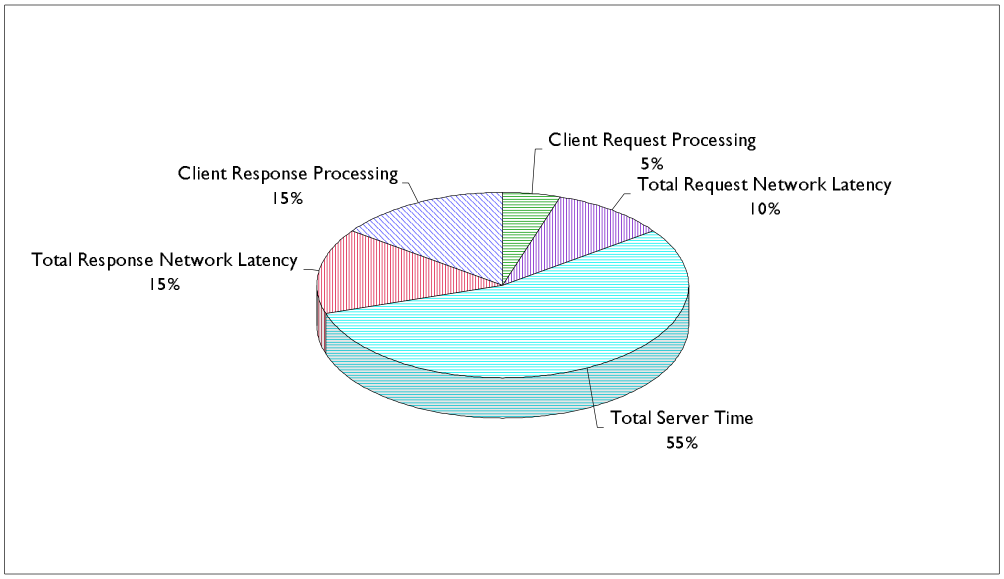


Figure 24. End-to-end performance budget.

If your goal is to achieve a 2-second interactive response time (after possibly making some allowance if you intend to use the Internet), then this example will provide you with the following performance targets for each component:

- Client request processing - 5% of the budget, or 100 msec
- Total request network latency - 10% of the budget, or 200 msec
- Total server time - 55% of the budget, or 1100 msec
- Total response network latency - 15% of the budget, or 300 msec
- Client response processing - 15% of the budget, or 300 msec

Note

Client response processing (typically the rendering of HTML) can be surprisingly time consuming. Solutions that are using between 20% and 40% of their entire performance budget in this way are common. Most development teams are unaware that the “look and feel” of their solution can have such a major effect on performance.

You can then apply the target number for each component (for example,1100 msec total for the servers) to the detailed data flows you defined during your initial design phase. Based on the total work that has to be done by each server, the network latency of your enterprise network, and the total number of trips between server systems, you can begin to evaluate how close to your target you are, or even if a given component is unlikely to meet the target.

For very complex solutions it may be appropriate to carry out some simple prototyping and benchmarking at this point to help improve the accuracy of your predictions.

However, in general you will find that you need to iteratively revisit your estimates, balancing them as best you can. Where a component does not meet its performance target, you may find that you have “spare” budgets within the other components that you can reallocate. If not, then you need to find ways to improve the performance of both the individual components and the solution as a whole, to allow you to meet your targets. We describe some classic techniques that you can use to achieve this later in this chapter.

The key feature of a performance budget is not to spend a lot of time getting all the numbers exactly right the first time, but rather to start thinking about the flows of data in the system, and where work is being done. This should help you to identify bottlenecks in your solution, allowing you to take the appropriate actions to remove or reduce them as early as possible.

4.3 Scalability

We have defined what capacity, performance, and scalability are, and stated that they are related to one another, but not how. Figure 25 helps to illustrate the relationships between performance, capacity, and scalability.

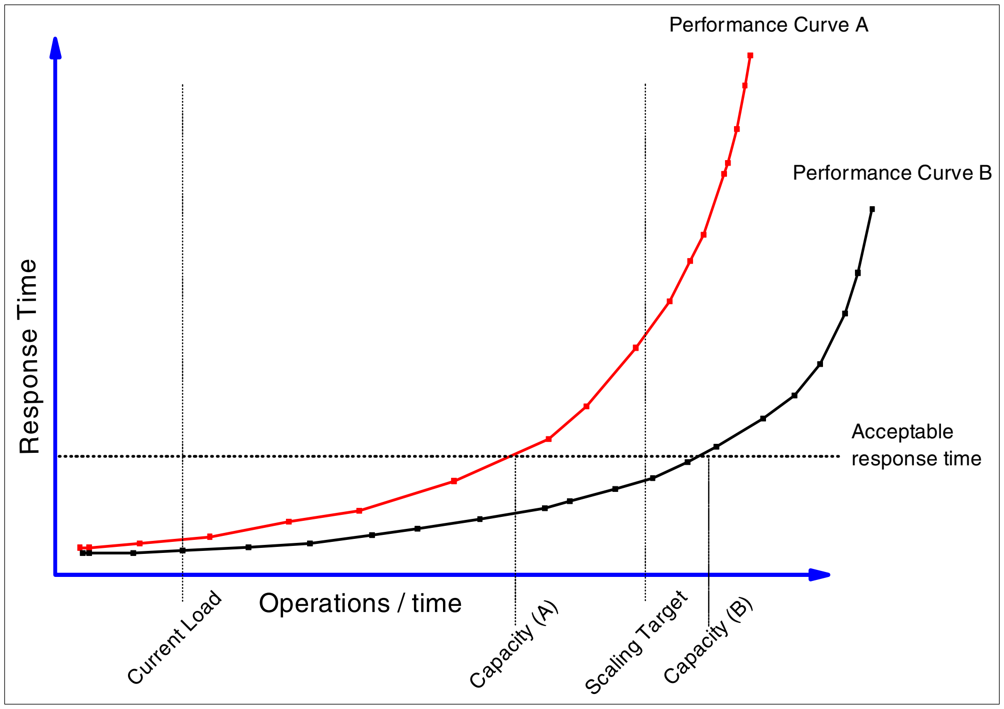


Figure 25. Scaling a system

Response time is plotted on the Y-axis, and is a direct indicator of performance. Operations/time is plotted on the X-axis, and is often referred to as the system load. The acceptable level of response time is one of our business objectives, and this is shown as a horizontal line across the chart.

The performance curves of two e-business systems are also shown on the chart. Determining these curves in a complex system can currently only be achieved by real performance testing and measurement. Mathematical calculations can, however, be invaluable in determining and managing the risk of selecting one component rather than another.

As you would expect, if we increase the load on the systems, the response times increase (that is, the performance decreases). The maximum capacity of each system can then be easily derived by examining the point at which the performance curve intersects with the line representing acceptable response time: *Capacity (A)* and *Capacity (B)* respectively.

Now, let's assume that we have a solution that is supporting our business, operating at *Current Load*, but that our marketing department (which is very creative) is about to launch an advertising campaign that will generate a lot more business. To support the customers generated by that campaign, our solution will need to process a new load, labeled *Scaling Target*.

If the current system that we have supporting this solution follows *Performance Curve A,* then we can see that we have a problem. System A will not support the increased load while maintaining our business objectives (the acceptable response time). System A cannot scale to support the required load.

If instead we have System B supporting our solution, then we have no problem; by examining *Performance Curve B* we can see that we can support the load that the campaign will generate. The response time for each transaction will rise (that is, performance will suffer), but it is within a level that is acceptable to our business.

And this is the key to why performance and scalability are so interrelated. To enable System A to scale to support the required load we need to make its performance curve look more like that of System B. Essentially this means that for any given load we have to improve its performance. So, as we change the performance of a solution (or component), we are altering its scalability, too. With this in mind there are only four things that we can do to scale a system or component:

1. Increase the speed or capacity of the component (add more "power")
2. Improve the efficiency of the component (do more with the same "power")
3. Decrease the load on the component (move work elsewhere)
4. Reduce the response time expectations of the component's users

The first two items directly affect either the capacity and/or performance of a component, and given the relationships between capacity, performance, and scalability, their effect is self-evident.

The third item is a little more complex. In an end-to-end system built from many components, some of those components are more scalable than others. By "protecting" the components that are not as scalable with components that are, the scalability of the overall system can be maximized. For example, data can be accessed much faster if it is held in cache than if the same data has to be retrieved from a database. The cache is intrinsically more scalable than the database. If we can ensure that a large number of requests for data are handled by the cache rather than the database, we reduce the load on the database (by transferring it to the cache) and make the overall system more scalable as a result.

Finally, it's fair to say that for practical purposes only the first three items are credible alternatives; changing end-user expectations is not usually considered a viable approach to scaling!

4.3.1 Scaling techniques

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 usually 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 talk very briefly about the main techniques that can be applied to components of an e-business solution to increase their scalability. As you will see, each falls into one of the categories that we outined in points 1 to 4 on page 43, and each is more or less applicable to particular situations.

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.

4.3.1.1 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 (CPU cycles, I/O bandwidth, etc.) in a single physical unit.

A faster machine scales because it can do more work in the same time. Making a machine faster can be achieved in many ways; the simplest is upgrading the hardware. Faster CPUs, more CPUs, more memory, or more disks may all help. Upgrading system software may also make an existing machine faster.

The key requirement in being able to use a faster machine is that the components that run on it must be capable of migration as appropriate. Particular care must be taken to ensure an application can make use of the enhanced facilities that are being supplied; for example, migrating from a single processor to multiple processors will only help if the component can make use of the additional processors.

It is also possible to replace an existing server with a different, faster, physical machine. Examples would be replacing a uniprocessor Intel machine running Windows NT with a four-way RS/6000 running AIX, or replacing such an AIX machine with a S/390 sysplex. In these cases the software components must be capable of migration to the new platform; designing your components around open standards is key to enabling this.

4.3.1.2 Replicated machines (horizontal scaling)

Using replicated machines 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 tends to lead 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 general an implementation with multiple machines requires that there be both a workload-balancing mechanism to distribute the work among the replicated machines, and a number of replicated components running on those replicated machines, capable of processing the workload.

There are many schemes for balancing workload. These vary from simple "round-robin" approaches, where the workload distribution mechanism simply gives the next item of work to the next system, looping around them forever, to more sophisticated systems that take into account availability or even load on the replicas, including CPU, I/O and disk usage, before apportioning work to the system best able to deal with it.

The key feature of replicas is that they each provide an identical service. Typically each replica is exactly the same as all the others, in terms of both hardware and software, and the number of replicas is varied according to the workload that is to be supported. As replicated machines generally scale very nearly linearly (at least until the capacity of the load-balancing mechanism is reached), this can be a very effective technique.

There are, however, drawbacks to this approach too. If the service running in a replica has a state to preserve across client requests, then that state must be shared between all the replicas, as it is (ideally) not possible to determine which replica will process the next request from that client. The sharing of such state information can significantly complicate the deployment by requiring the introduction of technologies such as shared file systems (like NFS, AFS, DFS etc.), shared databases or even new technologies such as Storage Area Networks (SANs). Each has its advantages and disadvantages, but in general, introducing any of these into the solution alters the scaling characteristics of the overall solution yet again.

Approaches that can be adopted to avoid these issues include:

- Removing the need to store any state. Often this is impractical since the application may not be suitable for conversion to a stateless design.
- Reducing the state information to a size that can be passed on every request, either in the form of a cookie or within a URL. Again, the application's state requirements may preclude this option.
- Various techniques that have evolved to cope with a short-term state (usually a session-based state) where the solution ensures that as long as a session is in existence the requests from a given client will always be directed to the same replica. However, the use of such session affinity tends to counteract the benefits of the replicated machine's technique to some degree.

4.3.1.3 Specialized machines

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.

A good example is the use of cryptographic coprocessors, which are extremely effective at carrying out bulk encryption and decryption, often being orders of magnitude faster than software solutions.

However, there are general issues associated with the introduction of special-purpose machines. Since the system is by its very nature, specialized, does it perform all the functions that you require of it, or is it too specialized for the uses that you are considering? As a result of this, does the benefit that it provides warrant the additional cost and complexity of deploying it? Does it fit into your systems management infrastructure? Will the technology be quickly superseded, rendering the expensive specialized machine useless?

4.3.1.4 Segmenting Workload

Segmenting the 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.

If we wished to segment a Web server we could decide that a Web server carries out the following work:

- Serving HTML pages, images (GIFs, JPEGs)
- Executing CGI programs
- Executing Servlets
- Handling redirection

We could then decide to provide specialized Web servers to carry out each of those functions in an optimal fashion. Segmenting the workload in this way usually requires that the application be changed to redirect the requests to various servers, although techniques such as Content Based Routing can mitigate this, at the price of introducing other potential scalability and performance issues.

4.3.1.5 Request batching

Splitting your system into multiple layers has many advantages (reuse of existing systems, separation of function, physical partitioning etc.) but it also places an additional processing load on both the client and server components, and introduces network latency into the solution.

Measurements of distributed applications frequently show that the largest factor affecting response time and system load is the number of messages that flow between the distributed elements of the application. In extreme cases the costs of processing the work are swamped by the communications overhead. The primary reason for this overhead is the desire to build robust, reusable interfaces between all the components of an application. This drive for reuse tends to create fine-grained, detailed interfaces that require several interactions to accomplish any given task.

Since the overheads associated with issuing a request from one component to another are virtually identical no matter what the request is, it makes much better sense to make fewer but larger and more complex requests, reducing the total amount of overhead. This reduces the load on both components by eliminating the overhead costs associated with multiple requests, reduces the latency, and possibly improving end-user response times by reducing time spent by the client carrying processing of overhead tasks.

The drawback is that although the ideal communications interface between two elements would be a single flow of data, this would create highly specialized interfaces, with little opportunity for reuse. It would also provide little flexibility for future enhancement of the function provided by the application elements.

In short, there is never a perfect answer to the compromise between good interface design principles and reduced communications overhead. However, an excellent balance can be obtained by combining some of the best principles of each. The client element should define and implement the cross-process protocols, but use the fine-grained, well-defined facilities provided by the server to implement them. In practice this may mean that the client defines a single message-based flow, and provides code that resides on the server platform to decode the message before locally calling the server-defined (fine-grained) interfaces.

4.3.1.6 Data aggregation

If your solution has to make frequent access to data that is spread across many systems and applications, it may overload those applications and will probably involve far too much latency to allow your solution to provide acceptable response times to your users.

This is particularly true when your solution is attempting to support high levels of personalization based on customer-specific data where there are very large numbers of customers - a classic situation for many large enterprise solutions.

The idea of user data aggregation is (as the name implies) to aggregate user data into a more readily accessible form. It is designed to provide rapid access to the user data for a very large number of concurrent users, and will be implemented around appropriate technologies (often replicated relational database systems) to achieve those aims. The key feature that enables this technique to work is that, provided 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.

Reworking all the applications involved in a solution to move all their data to the aggregation is generally impractical due to budgetary and schedule constraints. However, if the legacy applications can be modified to publish updates to the aggregated data store, then significant benefits can still be obtained. Once enough information is being published, new e-business applications can refer to the aggregated data to get high-speed access to the user data, supporting personalization, cross-selling, etc. Over time the legacy applications can be modified to also use the customer information service as a source of user data.

In short, the aggregation can be seen as a cache for the user data. Initially updates are “trickled” into the aggregation as they happen to the master copies of the data, but over time the master copies of the data will migrate to the aggregated data store.

4.3.1.7 Connection pooling

When an application is distributed, a connection between the distributed layers must exist. For example, to use a database, you establish a connection to the database so that queries can be performed.

So why is connection pooling important? Each client system needs a connection to allow it to perform its work. As we increase the scale of a solution, we notice there is the potential for lots and lots of connections. It transpires that there is a certain amount of overhead required to establish and terminate a connection. This requires resources and consumes part of the end-to-end performance budget.

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.

Connection management is all about pre-establishing connections to a resource, managing pools of those connections, and sharing those connections among the systems that need to use them. By doing this effectively we eliminate the need for each system to establish and terminate individual connections. We also reduce the total number of connections required by the whole system. This allows the systems to run more efficiently and reduces the load of supporting large numbers of client systems. This in turn increases the scalability of the overall solution significantly.

Unfortunately to achieve this, we must also introduce another layer (the connection manager) into our solution, which will affect the end-to-end performance budget. However, the good news for us is that the overhead associated with a connection manager is almost always less than the cost of establishing connections for each client.

4.3.1.8 Caching

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.

Caches may take many forms, including browser caches, proxy caches, file caches, Web server caches, application server caches, database caches, and device-based caches. In addition to these specific forms of caches, there are two main categories of caches available, typically referred to as static cache and active (or sometimes predictive) cache. Each type of cache has particular advantages, but in all cases, an effectively managed cache can significantly increase the scalability of a Web site.

Web servers provide two types of data: static data from files stored at a server, and dynamic data that is constructed by programs that execute at the time a request is made. Dynamically generated pages generally require significantly more resources to generate, thereby reducing Web server performance. High-performance Web servers can typically deliver several hundred static files per second. By contrast, the rate at which dynamic pages are delivered is often significantly slower. It is not uncommon for a program to consume over a second of CPU time in order to generate a single dynamic page. For Web sites with a high proportion of dynamic pages, the performance bottleneck is often the CPU overhead associated with generating dynamic pages.

Static caches are currently the most common type of cache, and are particularly appropriate to static content. Simplistically, when the cache is asked for content it will retrieve it from the source server, and then save a local copy that can be served very rapidly when future requests for the same content are received.

It is also possible to use static caches for dynamic content, but gaining significant advantage can be difficult. Advantage is only gained where multiple copies of the dynamic content can be served from the cache before the content becomes outdated (or “stale”). Given the trend for the Internet to make more use of dynamic content, improving the caching of dynamic content is becoming a very important issue.

Active caches have been designed specifically to help with this problem of caching dynamic content, and so improve the performance and scalability of solutions that use dynamic content. They essentially allow us to serve “dynamic content” with an apparent performance that is similar to that of static content. Active caches achieve this by pre-forming and retrieving the content of dynamic pages before a user request arrives, and storing the resulting pages (or partial pages) in cache in readiness for any requests. Various techniques are used to achieve this, but since there are no established standards in this area, all are more or less proprietary. Examples include:

- Statistical analysis of previous-use patterns by the cache.

  For example, if the cache knows that there are many accesses to the CNN Web site around 8:00 am (perhaps when people arrive at work) then it can load those pages into cache at (say) 7:30 am.

- Linking the cache to the underlying data that the dynamic content is built from.

  In this case, if the cache knows the relationship between objects (or pages) that may be cached, and the underlying data that periodically changes and affects the values of those objects, then as the data changes the cache can request new copies of the objects to load into its cache. Ideally the method couples the pages to discrete elements of data within the database, as this prevents objects whose values have not changed from being mistakenly invalidated after a database change.

- Sophisticated subscription-based systems.

  These provide centralized, trend analysis of network usage to determine one-use “hot” pages that are then cached automatically by all those caches that subscribe to the system. A good example where this can work well is a piece of news (such as the Starr Report probing into the Clinton administration), which generated huge network traffic when it was released, and could not have been easily predicted by other means.

Typically with any type of caching the key issue is determining when a cached page has become obsolete (or “stale”). Caches use a variety of methods to achieve this. The simplest is to check the expiration interval that is in the information itself, and to keep a copy in the cache until the expiration period ends. The next request for the object causes the cache to be reloaded with a new copy. More sophisticated mechanisms are typically used, but discussing them is beyond the scope of this book.

For more information on caching techniques in general, and the caching product supplied with WES in particular, please refer to Chapter 11, “Caching Proxy” on page 183.

4.3.2 What techniques to use where

Determining which of the techniques outlined in 4.3.1, “Scaling techniques” on page 44 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 26 provides an excellent summary of where you *may* derive benefit from each of the techniques.

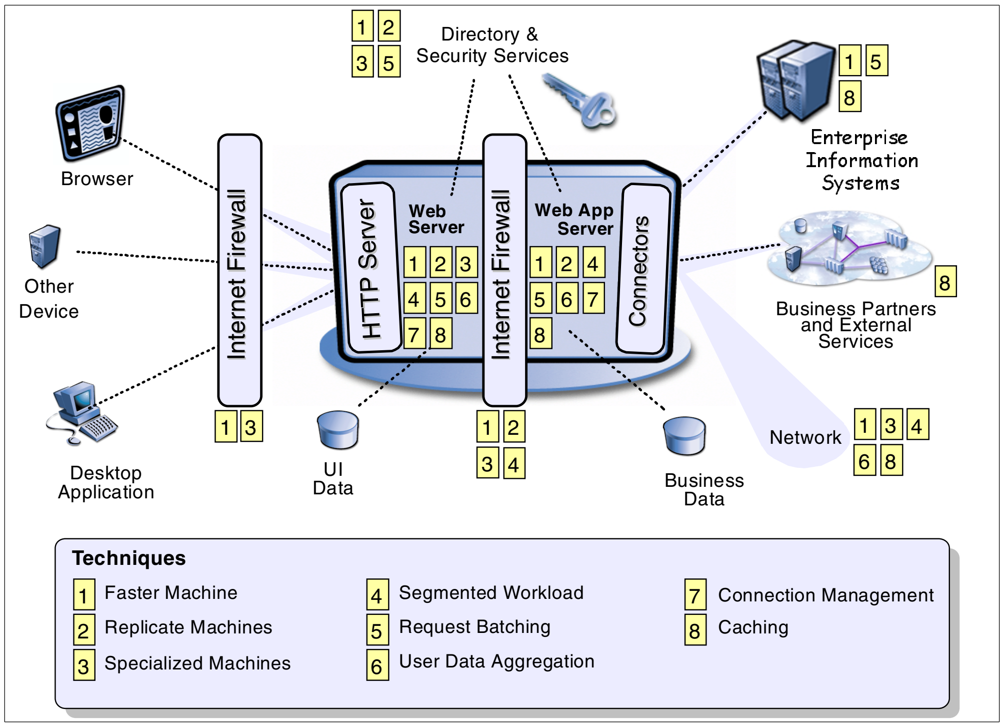


Figure 26. Scalability techniques

4.3.3 General recommendations

In this final section we pass on some of the experiences that we have gained - in some cases, the hard way:

- An e-business solution will have good performance only when attention is paid to each component of the solution: the client, the network, the servers, and so forth.
- It is key to plan for future growth in your initial design. Growth rates in the e-business arena are notoriously high. We strongly suggest that the best approach is to adopt open standards and to design a solution that is both modular and as platform-independent as possible. This will generally leave you with a solution that can be scaled either vertically (onto more powerful platforms) or horizontally (using multiple systems in parallel) as required. Using proprietary interfaces, system utilities, tools, or extensions will increase the chance that your solution will end up “stuck” on a platform that cannot be scaled.

Since developers often choose proprietary implementations in the belief that closed systems perform better than open systems, planning for future growth requires a good understanding of the longer term performance and scalability requirements of the solution, and also of the short-term performance gains that can be derived from proprietary technology.

- You must understand the detailed data flows within your proposed system at the time you are building your initial design. When you combine this analysis with an end-to-end performance budget and an iterative approach to development you should be able to set performance expectations correctly and structure your applications and systems to meet them.
- Wherever possible, use the latest release of infrastructure and system software. In general, because e-business uses technologies that are still evolving rapidly, each release of software provides better performance than the previous one; in some cases, dramatically better. Using the latest release of software is a key consideration for achieving good performance. However, you should pay careful attention to software prerequisites and corequisites, especially if your solution (as most do) is integrating with existing systems.
- Where possible, place your data as close to the client as possible; this will enable the solution to respond “faster”. If the data is published, this is covered by caching techniques. In the case of other types of data, replication techniques can be used to achieve the same ends.
- Consider the use of SSL connections carefully. There is a significant overhead in establishing such connections, and you will be able to reduce the load on your servers significantly by selectively securing objects. Sending all objects on a Web page using SSL allows the client browser to show the security “lock” icon. However, you should balance this advantage against your real security needs and the probable performance costs.
- You should instrument your applications to allow you to capture important performance information, but you *must* enable different levels of logging for test and production systems.
- Caching is a key technology to help you obtain performance and scalability. Cache information anywhere and everywhere possible. When in doubt, cache, cache, cache!
- Finally, before deployment you should test, test, and test again to ensure that your solution meets your business requirements, and especially your peak workloads. In truth, you will always test your solution; the question is whether you test your solution in private, or you let your customers test it in public. We strongly recommend you adopt the former approach; it is significantly less stressful, and often less expensive in the long run.

Chapter 5. Security

We increasingly rely on the electronic creation, transmission, and storage of personal, financial, and other confidential information. In order to give people convenient access to such information and allow them to easily take action anywhere anytime, the IBM WebSphere Everyplace Suite is designed to enable increasing number of mobile personal and professional transactions using a new class of intelligent and portable devices. By the nature of the pervasive computing, accessing and transferring sensitive information usually involves transmission over the Internet and other public networks. Therefore, creating a safe computing environment and providing the highest security for the confidential transactions become important issues.

This chapter addresses the security implementation within the WebSphere Everyplace Suite. First we will discuss the general security objectives for data communication. Secondly, the security design of the WebSphere Everyplace Suite will be presented. Thirdly, we will outline the three types of security implementations within the WebSphere Everyplace Suite, namely TCP/IP security, wireless security, and MQSeries Everyplace security. Finally, we will discuss firewall considerations within the WebSphere Everyplace Suite.

5.1 Background

Security for data communications has been well documented in the industry ever since the Internet became popular. Therefore we will not attempt to review every aspects of the security for computing but will only focus on the most common and relevant security objectives for data communications in our discussion.

Security objectives

The most common and relevant security objectives for data communications are:

Authentication: to verify the identity of the sender or the receiver of the data in communications. This is to make sure the clients or servers are really who they claim to be.

Confidentiality: as a synonym of secrecy or privacy, confidentiality means to prevent eavesdropping on data communications.

Integrity: to verify that data has not been altered in transit by a third party. This is to prevent forgery, tampering, and unauthorized alteration.

Authorization: to limit the improper use of the data and services by limiting user privileges to access information in order to minimize the chance of exposing sensitive information to malicious attack or unauthorized alteration.

Non-repudiation: to prevent the parties in a data transaction from denying their actions after the transactions are done. This is to enforce the accountability for electronic transactions.

These are the five objectives normally desired by online content and application services which involves 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.

5.1.1 Tools and solutions to achieve security objectives

There are many tools to achieve the most common security objectives, including *data encryption*, *message digest*, *digital certificate*, *packet filtering*, and *address concealing*.

Many implementations of these tools have led to the popular security *s*olutions or technologies such as the IEFT standard Transport Layer Security (TLS), formerly called Secure Socket Layer (SSL), which uses data encryption, message digest, digital certificate, etc. to achieve multiple security objectives, such as confidentiality, authentication, and data integrity. TLS is primarily used for TCP/IP networks.

Note

SSL does not provide non-repudiation automatically but if required it helps you to implement it at the application level.

For data communications over wireless networks using the WAP protocol, there is a solution similar to TLS called wireless transport layer security (WTLS). Other solutions include *proxy* and *firewall*, bothof which are used to achieve a *secure boundary* for the service domains. They use packet filtering and address concealing and many other security tools to protect the service domain on the edge from deliberate attacks over the public networks.

5.2 WES security

The IBM WebSphere Everyplace Suite 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 Suite. It exploits the single sign-on for user-friendly implementation of credential sharing across the services hosted by the Suite. The Edge 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 Suite uses the proxy technology in conjunction with firewall to define the secure boundary for the service domain.

5.2.1 Authentication

The user authentication function within the WebSphere Everyplace Suite is performed by the Everyplace Wireless Gateway and the Everyplace Authentication Server. On the other hand, the administrator authentication within the WebSphere Everyplace Suite is component dependent.

The WebSphere Everyplace Suite employs several industry standard technologies to perform authentication. The Everyplace Authentication 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.

Third-party gateway needs to support HTTP

In order to access the Edge Server, the clients or the third-party gateway that acts as the proxy for the client must support HTTP in order for the Everyplace Authentication Server to carry out the authentication process.

The Everyplace Wireless Gateway authenticates users from three types of different connections. Each type of user connection uses different a authentication process. For example, the Wireless Gateway uses the 2PKDP protocol with mutual authentication to authenticate wireless clients (non-WAP) at the WLP link layer. On the other hand, WAP clients are authenticated by the Authentication Server with a user ID and password using HTTP basic authorization in the Wireless Gateway. Also, WAP clients authenticate the server using WTLS at the transport layer.

Note: Currently, WTLS does not provide for client authentication.

The authentication process by the Wireless Gateway, and the coordination between the Wireless Gateway and the Everyplace Authentication Server on authentication, is discussed in this chapter, whereas the authentication process using the Everyplace Authentication Server is covered in Chapter 6, “Authentication” on page 69.

5.2.2 Confidentiality

The WebSphere Everyplace Suite utilizes a set of industry-standard security technologies to achieve the goal of confidentiality. Such technologies include the Secure Socket Layer (SSL) and wireless transport layer security (WTLS) for WAP clients. In addition,the Everyplace Wireless Gateway uses a modified version of the Point-to-Point Protocol (PPP) called the Wireless Optimized Link Protocol (WLP), formerly called the ArTour Link Protocol (ALP), to create secure tunnels to achieve confidentiality for wireless clients (non-WAP) with Wireless Client software applications installed.

5.2.3 Authorization

The access control in the WebSphere Everyplace Suite is achieved using HTTP proxy technology. To implement finer levels of access control, it may be done at individual WES components and application servers. One way to do it is integrating SecureWay Policy Director with the WES services.

5.2.4 Data integrity

The WebSphere Everyplace Suite achieves data integrity using features such as message digests and certificates included in the security technologies SSL and WTLS.

5.2.5 Non-repudiation

Since the WebSphere Everyplace Suite utilizes SSL and WTLS with certificates, the objectives of non-repudiation can be also achieved at the transaction level.

The WebSphere Everyplace Suite has implemented a set of industry-standard security technologies such as TLS/SSL and WTLS to achieve various security objectives. This section will discuss such implementation in detail. As shown in

Table 1, the WebSphere Everyplace Suite includes SSL, WTLS, WLP, and Proxy technologies in its components to enable many security measures. In addition, the WebSphere Everyplace Suite relies on properly configured firewalls, which is not shipped as a component of the Edge Server, to achieve the highest security.

This section discusses the confidentiality implementation by the WebSphere Everyplace Suite. We will also explain the authentication process by the Everyplace Wireless Gateway. In the next section we discuss the firewall considerations for the Edge Server. Authorization is mainly achieved by the Everyplace Authentication Server, which is covered in Chapter 6, “Authentication” on page 69.

Table 1. IBM WebSphere Everyplace Suite security implementation

Technology used in WES	**SSL**	**WTLS**	**PPP/WLP**	**Proxy**	**Firewall**
Components using the technology	EAS/WTE EWG	EWG	EWG	EAS/WTE	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
Secure Boundary	No	No	No	Yes	Yes

5.3 Security implementation

This section goes into more detail about how the WebSphere Everyplace Suite implements the security measures introduced in the previous section.

5.3.1 Single sign-on

The WebSphere Everyplace Suite allows users to connect to the Suite through either the Everyplace Wireless Gateway or the Everyplace Authentication Server. See Figure 27 on page 57.

Users from the Internet or third-party gateways are allowed to connect to the Edge Server only through the Everyplace Authentication Server.

Users may also access the WebSphere Everyplace Suite using three types of links through the Everyplace Wireless Gateway:

- Dial-up connection based on the Point-to-Point Protocol (PPP)
- Wireless Client connections over wireless or IP networks
- Wireless connection based on the Wireless Application Protocol (WAP)

The user authentication is conducted at both points of entries by the Everyplace Wireless Gateway and the Everyplace Authentication Server.

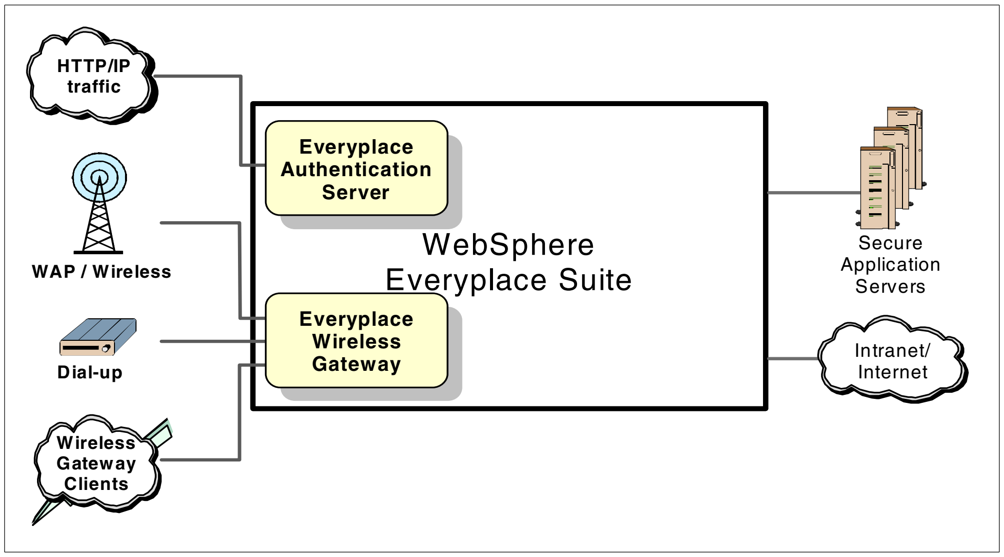

HTTP/IP traffic
WAP / Wireless
Dial-up
Wireless Gateway Clients
Everyplace Authentication Server
WebSphere Everyplace Suite
Everyplace Wireless Gateway
Secure Application Servers
Intranet/ Internet

Figure 27. Points of entry to the IBM WebSphere Everyplace Suite.

The WebSphere Everyplace Suite is designed to achieve single sign-on, namely to authenticate the users only once for their access to the services hosted by the Edge Server. This authentication design is achieved by sharing user credentials through a centralized repository. This centralized repository consists of a TPSM user or subscriber database,an Active Session Table (AST) database, and a SecureWay Directory database (LDAP). As shown in Figure 28 on page 58, both the Everyplace Wireless Gateway and the Everyplace Authentication Server use centralized RADIUS to authenticate users. They both deposit user active session entries into the AST database to share the user credential and profiles with the Edge Server services. Information sharing among the Edge Server services is achieved by their access to the AST and LDAP.

In addition to credential sharing across the WebSphere Everyplace Suite components, the single sign-on also requires the coordination between the two authentication agents, namely the Everyplace Wireless Gateway and the Everyplace Authentication Server.

To enable single sign-on access to the WebSphere Everyplace Suite services, the WebSphere Everyplace Suite uses the Everyplace Authentication Server as the central point of the authentication process. The Everyplace Authentication Server is positioned as the first entry point for all HTTP traffic from the Internet and third-party gateways. The Everyplace Wireless Gateway also routes all HTTP requests to the Everyplace Authentication Server which is the next non-firewall hop after the Wireless Gateway. Users authenticated by the Wireless Gateway will not be re-authenticated by the Everyplace Authentication Server.

The TPSM subscriber database and active session database are covered in Chapter 9, “Subscriber and device management” on page 135. See also Chapter 6, “Authentication” on page 69 for additional information on the authentication process. Using LDAP as a centralized repository for information sharing and single sign-on is discussed in Chapter 3, “Architecture” on page 23.

.

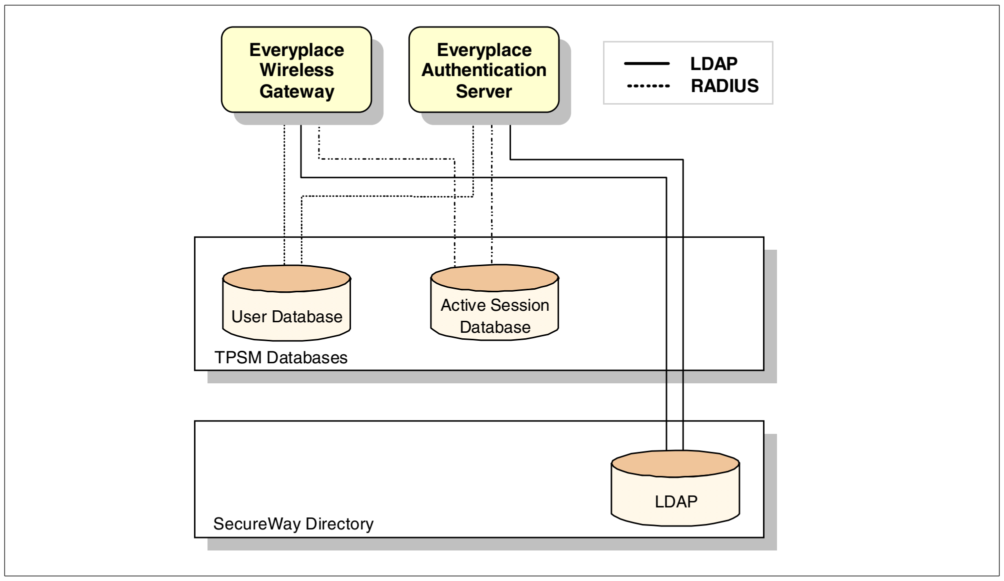


Figure 28. Single Sign-on implementation in the WebSphere Everyplace Suite.

5.3.2 Authentication

As we explained in 5.3.1, “Single sign-on” on page 56, three types of links can be used to access the Everyplace Wireless Gateway. Users from each link are authenticated using the appropriate protocols. After authentication, each connection is made secure using the proper encryptions provided by those protocols.

Devices with Wireless Client application software installed can access the WebSphere Everyplace Suite via the Everyplace Wireless Gateway. The client and server authentications are simultaneously done using a two-party key exchange protocol included in the Wireless Optimized Link Protocol (WLP).

The WAP clients can access the Wireless Gateway using the Wireless Application Protocol (WAP). Client authentication is done using the handshake protocol of Wireless Transport Layer Security (WTLS). During the handshake process, in addition to negotiating security algorithms and exchanging cipher secrets, the user also enters a proper user ID and password. Server authentication can be done using mini-certificates supported by WTLS. In addition, client authentication can also be done using HTTP basic authentication.

Note

Currently WAP client authentication is not implemented using WTLS mini-certificates.

The Everyplace Wireless Gateway supports cookies on behalf of its WAP clients. It can open SSL connection to destined Web and application servers on behalf of WAP clients if WAP clients requests an HTTPS connection.

WAP client connections authenticated by the Everyplace Wireless Gateway are assigned with the trusted Wireless Gateway IP address. Once the user is authenticated, the Wireless Gateway inserts the trusted Wireless Gateway IP address into the user’s HTTP request header and passes it on to the Everyplace Authentication Server.

When the Everyplace Authentication Server receives the HTTP request from the Wireless Gateway, it inspects the request header and acknowledges the trusted IP address. The Everyplace Authentication Server skips the authentication process since the user request is considered to be trusted. It looks up the session information in AST entered by the Wireless Gateway and creates a session ID and inserts device and network information in the header and passes the HTTP request on to the target Web and application server.

5.3.3 Secured Connections

Confidentiality is achieved by providing secured connections between clients and the Everyplace Authentication Server, as well as between Suite components. As illustrated in Figure 29, the most common deployment of the WebSphere Everyplace Suite has been highlighted to show segments of connections (labels 1 to 11 in the figure). Each segment can be configured to achieve confidentiality by enabling appropriate technology.

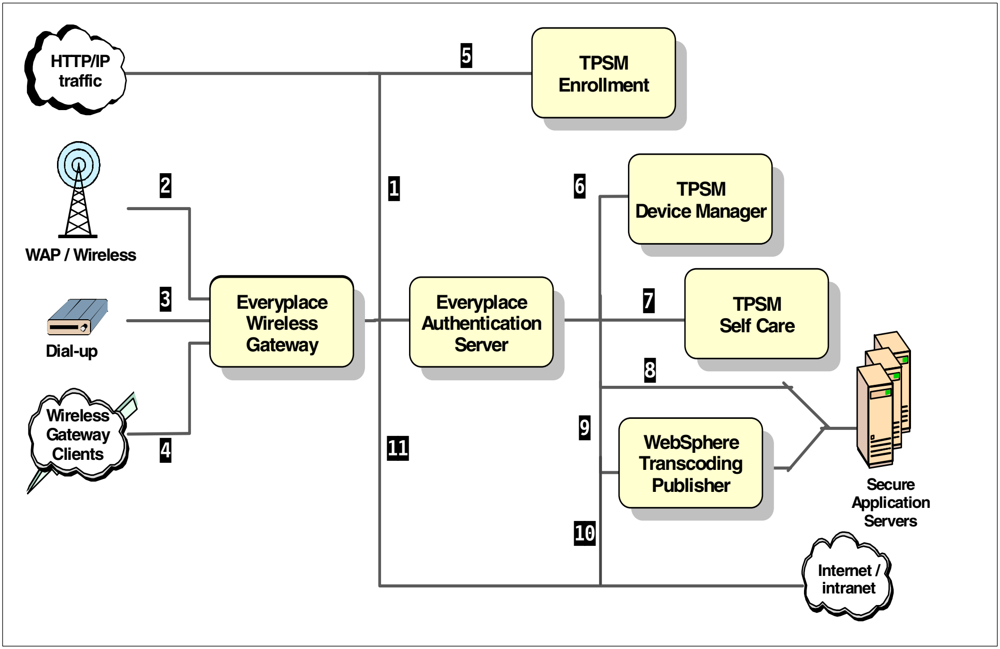


Figure 29. Confidentiality in the WES environment

1 Connection from the Internet to the Everyplace Authentication Server: for the HTTP clients from the Internet, SSL can be enabled between browsers and the Everyplace Authentication Server running on the Web Traffic Express server.

2 Connection between WAP clients and the Everyplace Wireless Gateway: the Wireless Gateway provides Wireless Transport Layer Security (WTLS) support for WAP devices. The WTLS secure connection is between the WAP device and the Wireless Gateway. The Wireless Gateway also provides the capability to enable SSL between the Wireless Gateway and the back-end servers. Since the secure connection is always broken in the Wireless Gateway, it is highly recommended that you enable the Wireless Gateway back-end SSL for these client devices.

3 Connection between dial-up clients and the Everyplace Wireless Gateway: the Wireless Gateway can use an SSL connection and create a secure tunnel for the clients.

4 Connection between wireless clients and the Everyplace Wireless Gateway: the Wireless Gateway allows for 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 HTTP-S traffic is encapsulated in the WLP protocol with the wireless client. The Wireless Gateway provides end-to-end SSL for these clients. The secure connection will not be broken in the Wireless Gateway in this case. For more details about wireless clients see Chapter 7, “Supporting wireless devices” on page 89.

You should also be aware that if SSL is enabled for the wireless clients using the WLP protocol, double encryption will take place. WLP encryption provides for symmetric encryption only using DES, RC5 or 3-DES algorithms, although SSL is a more robust implementation using PKI infrastructure (public key, digital signatures and certificates).

Note

The Everyplace Authentication Server in AP mode requires the HTTPS connection to be broken at the Authentication server box.

5 Connection between HTTP clients and the TPSM enrollment server: the TPSM enrollment server runs above the IBM HTTP server, which can be SSL-enabled. Since the subscriber enrollment involves transmitting sensitive private information across the public network, it is recommended that this connection be SSL-enabled.

6 Connection between the Everyplace Authentication Server and TPSM device manager: this connection is within the WebSphere Everyplace Suite domain and hence be considered secure.

7 Connection between the Everyplace Authentication Server and the TPSM Self Care Server: the Self Care would involve transmitting sensitive personal information. However, this connection is considered to be secure since it is within the WebSphere Everyplace Suite domain.

8 Connection between the Everyplace Authentication Server and secure application servers within the Intranet: even though it is considered to be secure, users have the option to enable SSL connections between the Everyplace Authentication Server and application servers.

9 Connection between the Everyplace Authentication Server and IBM WebSphere Transcoding Publisher: if the client requests require proper transcoding, the connection between the Everyplace Authentication Server and WebSphere Transcoding Publisher cannot be encrypted. Otherwise, WebSphere Transcoding Publisher would not be able to transcode.

10 Connection between the Everyplace Authentication Server and the Internet/intranet: this connection generally is considered prone to security compromise. It is recommended that SSL is enabled if possible.

11 Connection between the Everyplace Wireless Gateway and the Internet/intranet: the Wireless Gateway provides the capability to enable SSL between the Wireless Gateway and the Internet/intranet Web servers. Since the secure connection is always broken in the Wireless Gateway, it is highly recommended that wireless back-end SSL be enabled for these client devices.

5.3.4 Administration security

The administrators for the Everyplace Wireless Gateway can log in remotely using the Wireless Gatekeeper. The connection between the gatekeeper and the Everyplace Wireless Gateway can be SSL-enabled to ensure the highest security. In addition, the Everyplace Wireless Gateway can specify only one IP address from which a gatekeeper administrator can log in the gateway. This way, by locking the administrator to a trusted IP, the Wireless Gateway can minimize the security compromise.

WTE administration console. For security reasons, SSL can also be enabled for the connection between the WTE administration console and WTE.

Note also that most components in the WebSphere Everyplace Suite have access to the LDAP directory, where much sensitive information is stored for sharing by the Suite services. The connection between the components and the LDAP needs to be secure as well. Currently, access to LDAP uses 40-bit encryption via SSL. However, the connection between the Wireless Gateway and LDAP is not secure and SSL is not used for this connection.

5.3.5 MQSeries Everyplace security

The primary goal for IBM MQSeries Everyplace security is to provide privacy, data integrity, data compression, and authentication services for applications running on devices that are low in performance and memory. Generally, we expect to pass messages over unsecured external networks and to hold data on unsecured handheld devices. These expectations created the need to provide complete asynchronous and synchronous application-to-application security with guaranteed message delivery. Message security can be guaranteed regardless of security levels adopted by fixed networks, mobile service providers and gateways

MQe provides lightweight security for:

- Equipment that is limited in processing power and memory
- Low-bandwidth networks

MQe security is an optimized lightweight security that secures connections between device applications and back-end applications.

There are two different security levels provided by IBM MQSeries Everyplace. IBM MQSeries Everyplace standard edition includes 56-bit DES encryption while the IBM MQSeries Everyplace high-security version supports 128-bit encryption.

Our discussion assumes you are using the MQ Everyplace high-security version of MQ Everyplace. For additional information about differences between MQ Everyplace editions please refer to the MQ Everyplace online programmer's guide.

5.3.6 MQe security categories

To secure a message we have to secure it while it is waiting in the queue or transferred over the network. In MQe we can use three different categories of security: local, Q-based, and message levels. The scopes of these three levels are illustrated in Figure 30.

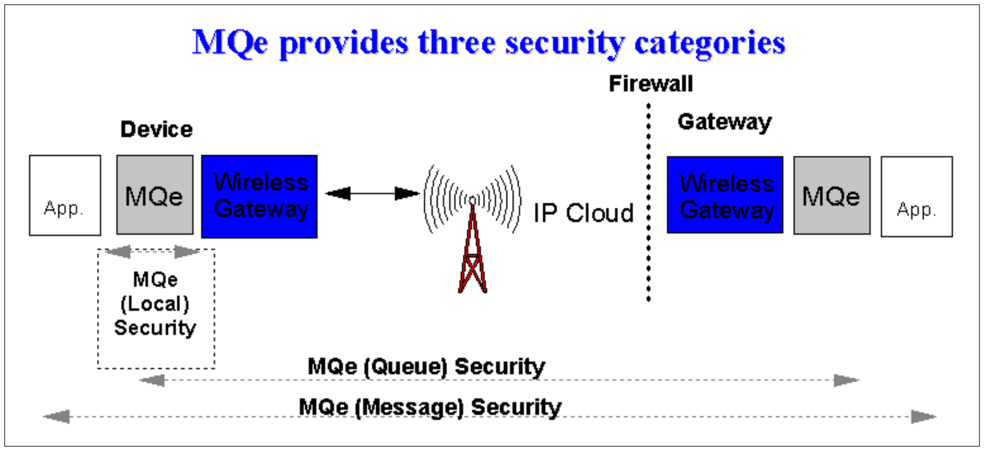
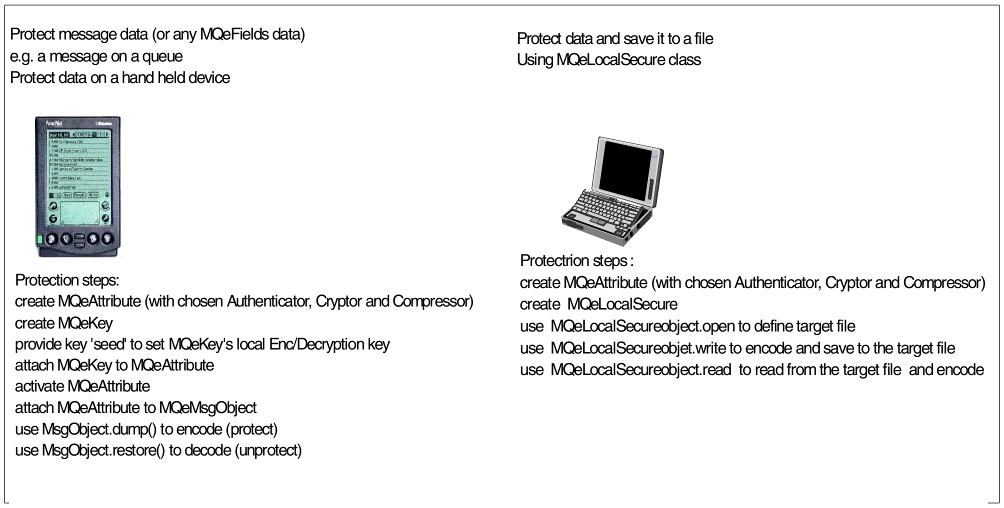


Figure 30. MQe security categories

5.3.6.1 Local security

In this category, to protect local data using a cryptor (secret key), we can:

1. Authenticate, encrypt/decrypt, compress/decompress data, and write/read it to/from a file.
2. Authenticate, encrypt/decrypt, compress/decompress data and write/read it to/from message data (or any MQeFields data). This can protect data and save it to a back-end such as a DB2 or LDAP server.



Protect message data (or any MQeFields data)
e.g. a message on a queue
Protect data on a hand held device

Protection steps:
create MQeAttribute (with chosen Authenticator, Cryptor and Compressor)
create MQeKey
provide key 'seed' to set MQeKey's local Enc/Decryption key
attach MQeKey to MQeAttribute
activate MQeAttribute
attach MQeAttribute to MQeMsgObject
use MsgObject.dump() to encode (protect)
use MsgObject.restore() to decode (unprotect)

Protect data and save it to a file
Using MQeLocalSecure class

Protectrion steps :
create MQeAttribute (with chosen Authenticator, Cryptor and Compressor)
create MQeLocalSecure
use MQeLocalSecureobject.open to define target file
use MQeLocalSecureobjet.write to encode and save to the target file
use MQeLocalSecureobject.read to read from the target file and encode

Figure 31. Local security options

5.3.6.2 Queue-based security

Queue-based security is appropriate for solutions designed to use synchronous queues. Queue-based security protects the message data being transferred between an initiating queue manager and a target queue manager queue. Using this category we automatically protect message data starting the moment we initiate queue manager and until it reaches the target queue. This protection is independent of whether the target queue is owned by a local or a remote queue manager.

As an example, we define a target queue with attributes to enable authentication, Triple-DES Cryptor (for encryption) and a compression method. When this target queue is accessed to put, get, or browse a message (using putMessages, getMessages, or browseMessages either locally or remotely), the queue attribute is automatically applied.

In this example, the application initiating the access has to satisfy access authentication before the operation is permitted. If access is permitted, the message data is automatically encrypted/decrypted using Triple DES and compressed/decompressed using the compression method selected. This means that when a secured target queue is remotely accessed (with put or get messages) security automatically ensures that the message data is protected as defined by the queue attribute, both during transfer between the initiating and remote queue manager and in the target queue backing storage.

5.3.6.3 Message-level security

Message-level security provides protection for message data between an initiating and a receiving MQe application.

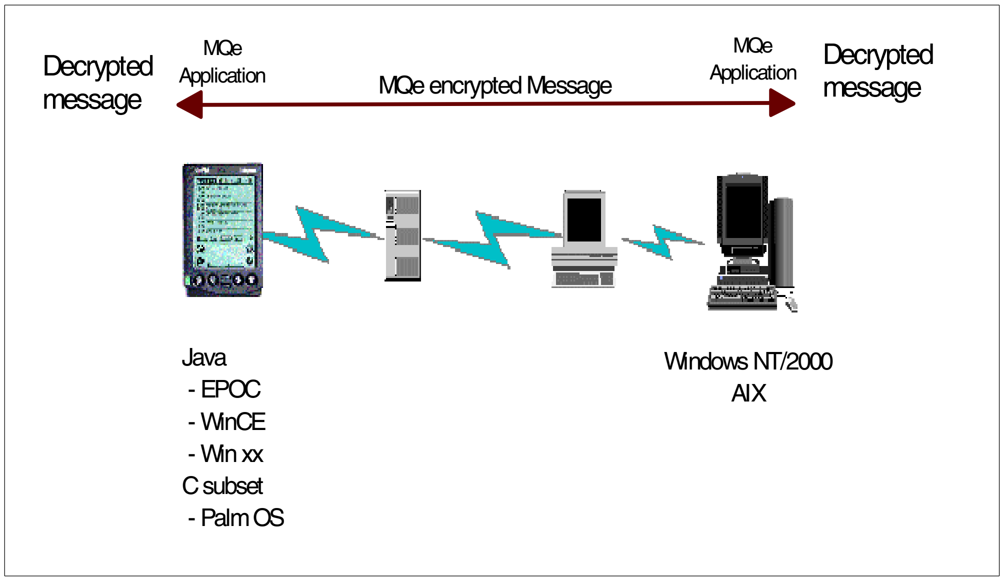


Figure 32. MQe message security offers end-to-end security

To use these security mechanisms we use secret keys, public/private keys to be communicated, and a certificate to identify the other party entity.

5.4 Firewall considerations

A number of general objectives for firewalls are:

- Allow only traffic flow that is determined to be in our interests.
- Give away a minimum of information about our private network.
- Keep track of firewall activity and be notified of suspicious behavior.

In the context of the deployment of Wireless Everyplace Suite (WES), the most generic deployment model, that suits the needs of most content providers, network operators, service providers, and enterprises can be depicted in Figure 33. The WES domain can be protected by two to three firewalls. For the purposes of the discussion, we label the firewalls *a*, *b*, *c*, and *d*, in the figure. Firewall *a* controls all access to the WES domain from third-party gateways or the Internet. FIrewall *b* is placed between the Everyplace Wireless Gateway and the devices connecting from the IP networks. Firewall *c* is optional and can be the same as firewall *a*. Additionally, the Everyplace Authentication Server (*d*) can use the underlying Edge Server - Caching Proxy server with multiple network adapters to achieve certain firewall functions as well.

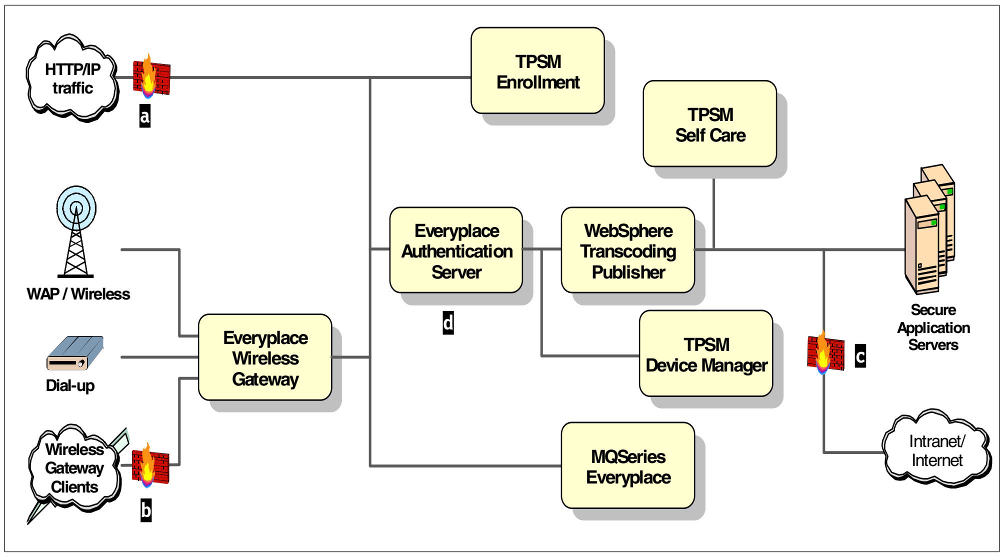


Figure 33. Firewall protection to the IBM WebSphere Everyplace Suite domain

Firewall *a* is configured to:

1. Allow HTTP requests destined for the WES services from IP addresses that are not owned by the Everyplace Wireless Gateway and route such requests to the authentication proxy.
2. Allow HTTP requests destined for the public Web servers and/or the enrollment server from IP addresses that are not owned by the Everyplace Wireless Gateway and route such requests to the Internet and/or enrollment server.
3. Filter the Wireless Gateway IP address range to eliminate possible conflicts.
4. Reject all other packets.

Firewall *b* is configured as follows:

1. Allow IP requests destined for the predefined Wireless Gateway IP ports from any IP address.
2. Allow HTTP requests for WES services.
3. Reject all other packets.

Firewall *c* is configured to:

1. Allow outbound HTTP requests destined for the public Web servers.

2. Allow inbound HTTP response for the active user session from the public Web servers.
3. Reject all other packets.

Firewall *d* (the Everyplace Authentication Server) is configured as follows:

1. Allow HTTP requests destined for the Edge Server services.
2. Use a separate network interface for the internal network.
3. Reject all other packets.

Part 2. Services

The chapters in this part of the redbook provide an overview of the IBM WebSphere Everyplace Suite core components and services, such as:

- Authentication Server, which supports single sign-on and a single point of entry to the Everyplace Suite domain.
- Wireless Gateway, which supports wireless clients, PPP, LAN, and WAP connections.
- WebSphere Transcoding Publisher, which provides transcoding capabilities running as a proxy server to adapt client markup language (for example, HTML and WML).
- Tivoli Personalized Services Manager, which supports functions such as user enrollment, self care, customer care, director, reporting, provisioning, billing and device management to maintain applications and data residing in pervasive devices.
- MQ Everyplace, which supports messaging applications in pervasive devices.

Chapter 6. Authentication

This chapter completes the discussion of how the IBM WebSphere Everyplace Suite authenticates its users. We will introduce the Everyplace Authentication Server as a product, and explain how it is used to perform user authentication for the WebSphere Everyplace Suite. The WebSphere Everyplace Suite adopts single sign-on for its users for all its functionality. Finally we will explain how it facilitates the single sign-on of the WebSphere Everyplace Suite.

6.1 Background

As we have discussed in Chapter 5, “Security” on page 53, the WebSphere Everyplace Suite allows users to connect to the Suite through either the Everyplace Wireless Gateway or the Everyplace Authentication Server (Figure 34).

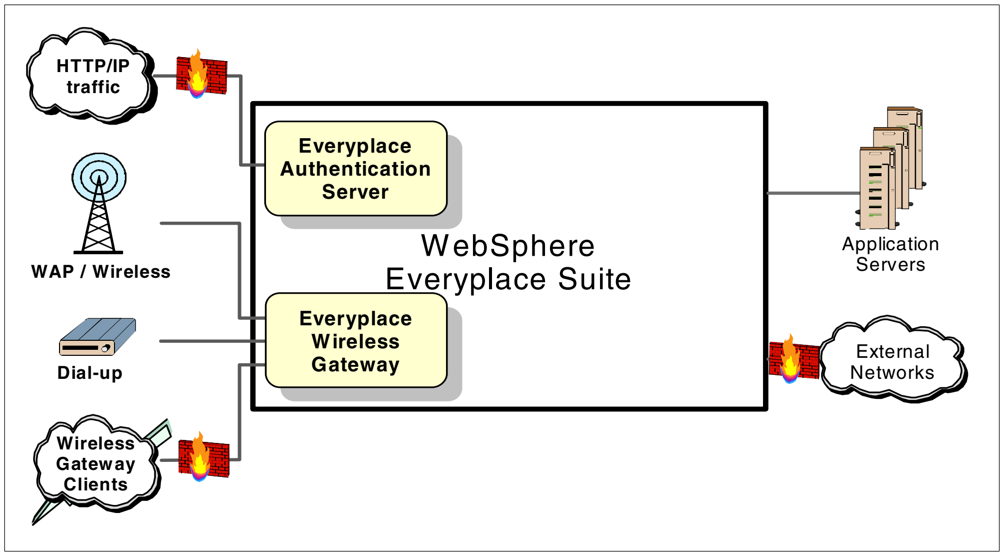


Figure 34. Points of entry to the IBM WebSphere Everyplace Suite

Users from the Internet or third-party gateways may connect to the Edge Server through the Everyplace Authentication Server.

Users may also access the Everyplace Wireless Gateway using three types of links:

- Dial-up connection based on the Point-to-Point Protocol (PPP)
- Wireless Client connections over wireless or IP networks
- Wireless connections based on the Wireless Application Protocol (WAP)

In either case, user connections must be authenticated before users can access the Edge Server. Authentication by the Wireless Gateway has been discussed in Chapter 5, “Security” on page 53. Here we will focus our discussion on the authentication done by the Everyplace Authentication Server for users from the Internet or third-party gateways.

The Everyplace Authentication Server is the point of entry to the Edge Server domain foe devices/users that do not connect through the Everyplace Wireless Gateway. It is the next, non-firewall hop for connections through the Everyplace Wireless Gateway. At least one Authentication Server is required in the Edge Server to enable integration of most Edge Server components.

Since no data link level authentication can be performed by the WebSphere Everyplace Suite for the connections from the Internet and third-party gateways, all authentication for these connections must take place at the HTTP level by the Everyplace Authentication Server. The client or the third-party gateway acting as the proxy for the client must support HTTP basic authentication, allowing the Everyplace Authentication Server to challenge the client for a user name and password for a particular domain.

The Everyplace Authentication Server runs on the Edge Server - Caching Proxy and is invoked as a caching proxy plug-in. Web Traffic Express is a prerequisite for the Everyplace Authentication Server on any machine where it is to be installed.

6.2 Everyplace Authentication Server

The Everyplace Authentication Server is a Web Traffic Express server with authentication plug-in enabled. It can be configured in one of the two flavors: *Authentication Proxy* or *Transparent Authentication Proxy*. For the Authentication Proxy, a Caching Proxy is configured as a protected reverse proxy server to enable the Authentication Proxy to act as the destined origin server. For a transparent proxy, WTE is configured as a protected proxy server. For more information about WTE as either a reverse proxy or protected proxy, please refer to Chapter 11, “Caching Proxy” on page 183.

The Authentication Proxy performs user authentication based on HTTP authentication headers and strictly enforces Edge Server single sign-on. That is, no other Suite component may do its own user authentication. In the meantime, users authenticated through the Authentication Proxy may not access content outside of the Edge Server.

The Authentication Proxy intercepts all HTTP requests destined for Edge Server services. It ensures all the requests have been authenticated and tagged with the active session key. If the requests are from a trusted IP address that is owned by the Everyplace Wireless Gateway, then the Authentication Proxy simply locates the associated active session from the Active Session Table, inserting the session key, 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, then it must have an authentication credential included within its WWW-auth header. This is done by the Authentication Proxy to challenge the user ID and password.

A Transparent Authentication Proxy also performs user authentication based on HTTP authentication headers. However, the Transparent Authentication Proxy allows other Suite components such as content and application servers to do their own user authentication. A Transparent Authentication Proxy also allows users to access material outside the Edge Server.

When an Edge Server is deployed using the Authentication Proxy, users do not know the existence of the proxy server for the services within the domain hosted by the WebSphere Everyplace Suite, such as domain xyz1.com in Figure 35. On the other hand, when an Edge Server is deployed using a transparent proxy, users have to explicitly use an HTTP proxy, such as tp.xyz1.com in Figure 35, to access the services within the domain hosted by the WebSphere Everyplace Suite. Users trying to access services outside the Edge Server, such as service hosted by abc.com, can do so through the transparent proxy.

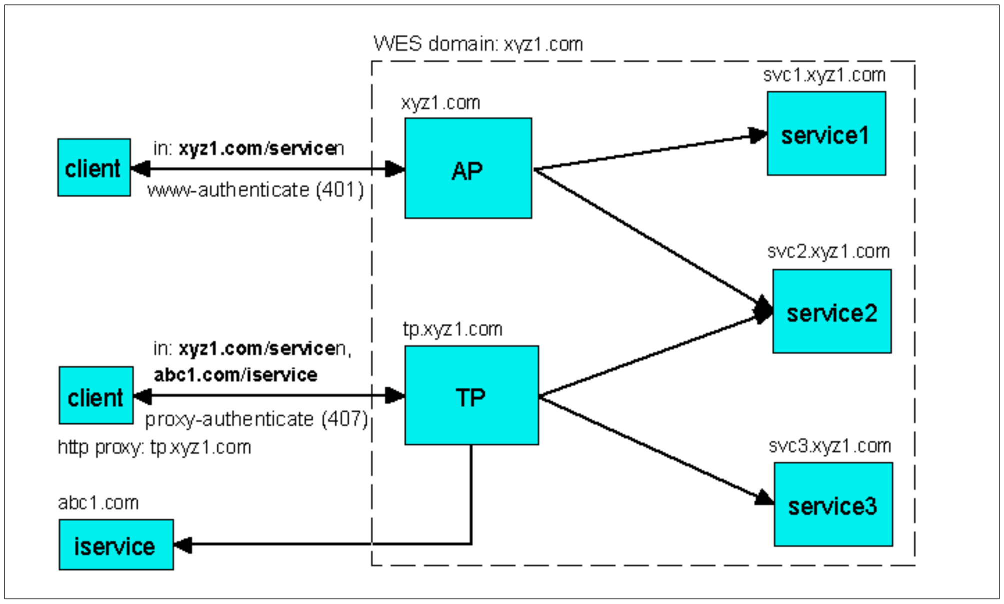


Figure 35. Two flavors of the Everyplace Authentication Server

6.2.1 Everyplace Authentication Server and single sign-on

The WebSphere Everyplace Suite is designed to achieve single sign-on, namely to authenticate the users only once for their access to the services hosted by the Edge Server. This authentication design is achieved by sharing user credentials through a centralized repository. This centralized repository consists of a RADIUS database, Active Session Table (AST) database, and a Lightweight Directory Access Protocol (LDAP) database. As shown in Figure 36 on page 72, both the Everyplace Wireless Gateway and the Everyplace Authentication Server use centralized RADIUS to authenticate users. They both deposit user active session entries into an AST database to share the user credentials and profiles with the Edge Server services. The information sharing among the Edge Server services is achieved by their access to the AST and LDAP.

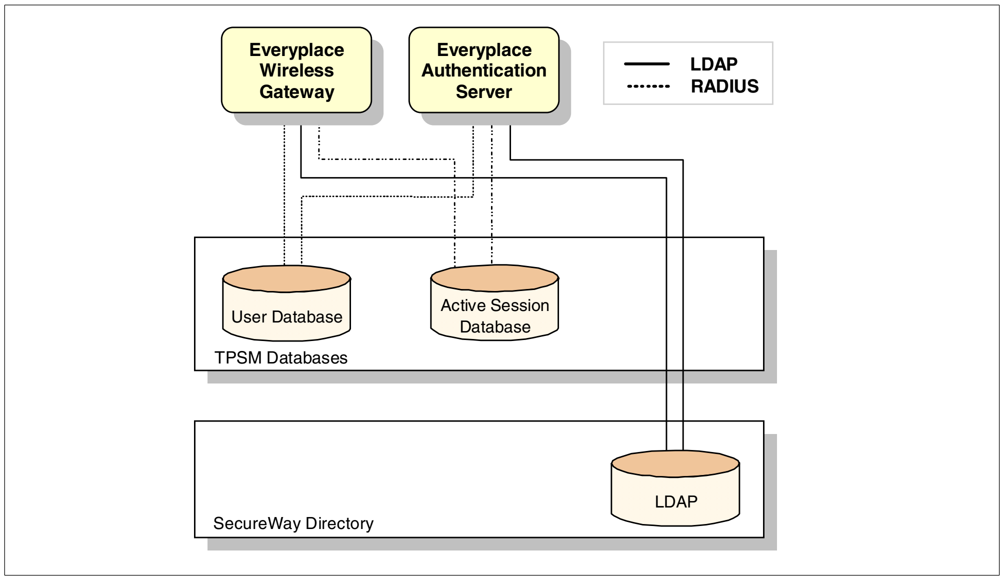


Figure 36. Single sign-on implementation in the WebSphere Everyplace Suite

To achieve the single sign-on for users to access the Edge Server services, the WebSphere Everyplace Suite uses the Everyplace Authentication Server as the central point of the authentication process. The Everyplace Authentication Server is positioned as the first entry point for all HTTP traffic from the Internet and third-party gateways. In the meantime, the Everyplace Wireless Gateway also routes all HTTP requests to the Everyplace Authentication Server, which is the next non-firewall hop after the Wireless Gateway. Users authenticated by the Wireless Gateway will not be authenticated by the Everyplace Authentication Server.

Since the Everyplace Authentication Server can be configured as a reverse proxy (Authentication Proxy), in addition to presenting a single domain name for the set of servers in the enterprise protection space, it provides security protection to the WebSphere Everyplace Suite domain by concealing the IP addresses for Web and application servers and allows the internal structure of an enterprise to be hidden from the attackers from the Internet.

The Everyplace Authentication Server is the central point for integration by the Edge Server. Figure 36 describes its interaction with LDAP, RADIUS server, and AST server. In addition, the Everyplace Authentication Server coordinates with the Everyplace Wireless Gateway in the authentication process and active session management.

As evident in Figure 36, the Everyplace Authentication Server receives the user HTTP requests from the Internet and third-party gateways. It inspects the HTTP header and looks for the user’s credentials. If the user is not yet authenticated, the Everyplace Authentication Server would challenge the user to enter a user ID in the appropriate realm with a password using HTTP basic authentication. Then it queries a RADIUS server for authentication. If the RADIUS server accepts the credentials, the Everyplace Authentication Server would generate a session ID and insert an active session entry in the Active Session.

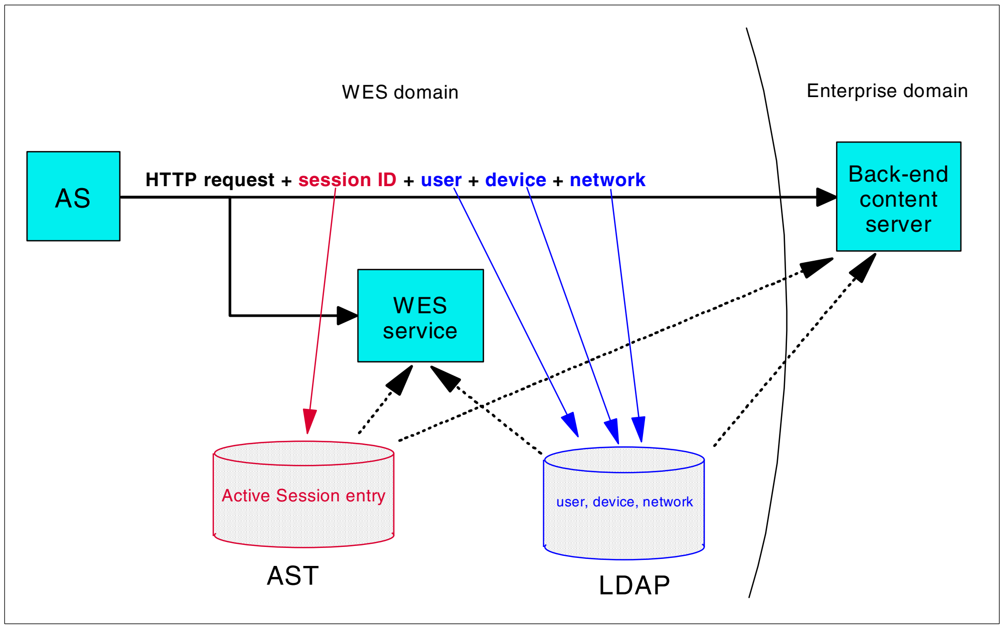


Figure 37. Everyplace Authentication Server enables the information sharing

For the user HTTP request, the Everyplace Authentication Server modifies its header and inserts the session ID, user, device, and network information in the header. In addition, it creates the user entries in the AST. The content and applications server downstream would use these pointers to retrieve information from the AST server and LDAP directory, which is necessary to complete the user requests.

The Everyplace Authentication Server also funnels tracking and logging information to applications included in TPSM for accounting and billing.

6.2.2 Coordination with the Wireless Gateway

As explained in Chapter 5, "Security" on page 53, both the Everyplace Authentication Server and the Everyplace Wireless Gateway authenticate user access. To provide user single sign-on, these two authentication agents need to coordinate the authentication process.

For clients entering the Edge Server via the Wireless Gateway, the Wireless Gateway performs the authentication by querying the RADIUS server. For those HTTP requests, the Wireless Gateway would route the user requests to the Everyplace Authentication Server for more information to be added to facilitate other Edge Server services rendering results for the user requests. In the process, for WAP clients, the Wireless Gateway inserts the trusted IP address in the HTTP header and creates an entry in the Active Session Table.

If the HTTP requests are routed to the Everyplace Authentication Server from the Wireless Gateway, the Everyplace Authentication Server inspects the HTTP header and looks for user's credentials. It would recognize the request is from a trusted IP address owned by the Wireless Gateway and consider the request is already authenticated. The Everyplace Authentication Server will not do the HTTP basic authentication again. It will insert more information on the device and network in the HTTP header and generate a new session ID for the user request and will insert it into the AST.

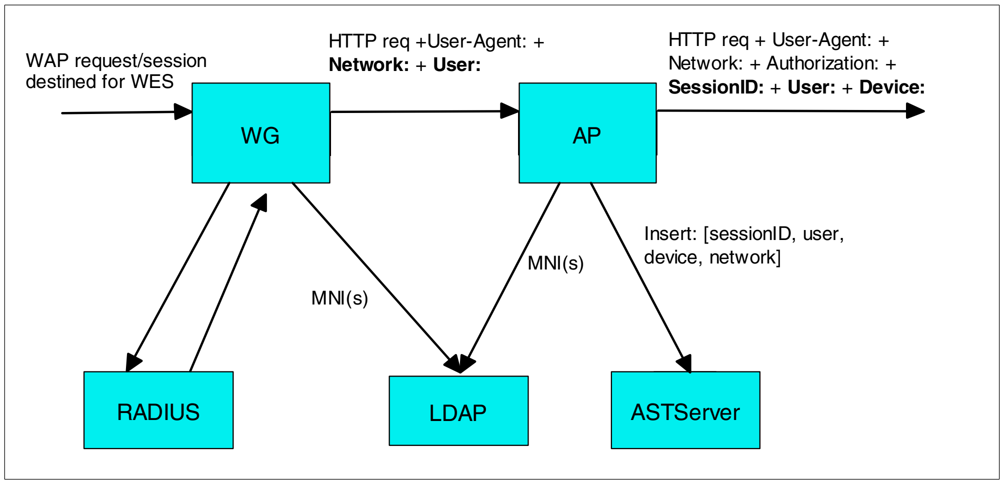

WAP request/session destined for WES
HTTP req +User-Agent: + **Network:** + **User:**
HTTP req + User-Agent: + Network: + Authorization: + **SessionID:** + **User:** + **Device:**
WG
AP
MNI(s)
MNI(s)
Insert: [sessionID, user, device, network]
RADIUS
LDAP
ASTServer

Figure 38. Handshake between the Wireless Gateway and Authentication Server (1)

Figure 38 shows the handling of requests from WAP clients connected through the Wireless Gateway as a special case for the coordination between the Everyplace Authentication Server and the Wireless Gateway. In this case, the WAP request will be translated by the Wireless Gateway into an HTTP request. The Wireless Gateway writes network and user information into the HTTP header along with a trusted IP address (not shown in the figure) and passes it on to the Everyplace Authentication Server. The Everyplace Authentication Server in turn will write a new session ID, user information, and device information into the HTTP header for the downstream services.

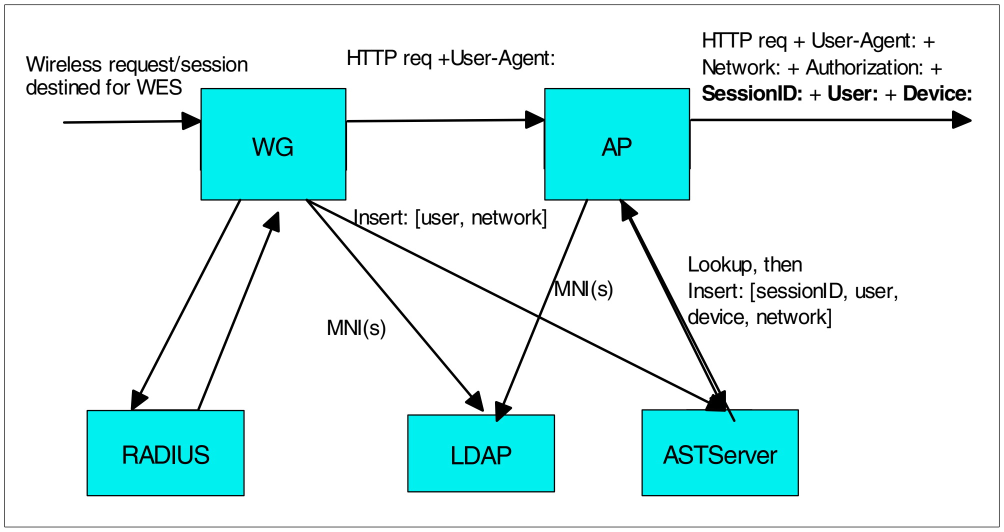

Wireless request/session destined for WES
HTTP req +User-Agent:
HTTP req + User-Agent: + Network: + Authorization: + **SessionID:** + **User:** + **Device:**
WG
AP
Insert: [user, network]
MNI(s)
MNI(s)
Lookup, then Insert: [sessionID, user, device, network]
RADIUS
LDAP
ASTServer

Figure 39. Handshake between the Wireless Gateway and Authentication Server (2)

For wireless access through PPP and wireless clients (Figure 39), the Wireless Gateway will not modify the data stream and insert the user and network information into the HTTP request. Instead, it forwards the HTTP request along to the Authentication Server while writing the user and network information into the

AST server. When the Authentication Server receives the HTTP request, it looks up the AST server for the user and network information, then it adds a device profile and creates a new session ID in the AST server.

6.2.3 Interface with Web Traffic Express

The Authentication Server operates as a set of plug-ins to Web Traffic Express (WTE). To enable the Authentication Server to run as a server hosted by the Caching Proxy, configurations specific to the Authentication Server reside as directives in the Caching Proxy's configuration file, ibmproxy.conf. These directives define the plug-in hooks, the proxy directives, and protection required for the Authentication Server.

Plug-in hooks

When Web Traffic Express is started, it will load its main configuration file, ibmproxy.conf which includes the defined plug-in exits for the Authentication Server, which will in turn start the Authentication Server. These exits are:

- **ServerInit** performs Authentication Server initialization
- **Authorization** performs the authentication and header insertion
- **ServerTerm** performs Authentication Server termination

The ServerInit and ServerTerm directives specify the Authentication Server initialization and termination code to be run when the Caching Proxy server is started. The Authorization exit enables the Authentication Server to take over the authorization process from the Caching Proxy. The Caching Proxy allows both authorization and authentication exits. The Caching Proxy defines only the Authorization directive in its configuration file to allow the Authentication Server to overtaken all authorization functions.

Proxy

The Proxy directive of the Caching Proxy is used when configuring the Authentication Server and serves to provide the next hop routing from the Authentication Server to the next component in the Everyplace Suite domain. For example, if we look at the sample Authentication Server configuration in the ibmproxy.conf file, then any requests for http://wesA that are received by the Caching Proxy proxy server will be proxied to http://A.wes.com. The proxy directive is not specified for the Transparent Authentication Proxy, since the next hop for the Transparent Authentication Proxy is an external Internet address, implicit in the URL of the request.

Protect

The Authentication Server, acting in either mode, is configured as a protected proxy. The protection setup is defined, using the Caching Proxy Protect directive which is needed to cause the Caching Proxy to generate authorization requests. If requests are directed to the Authentication Proxy, then a return code of 401, or a www-authentication request is generated and returned by the authorization exit of the Authentication Server. If requests are directed to the Transparent Authentication Server, then a return code of 407or a proxy authentication request is generated and returned by the authorization exit of the Authentication Server.

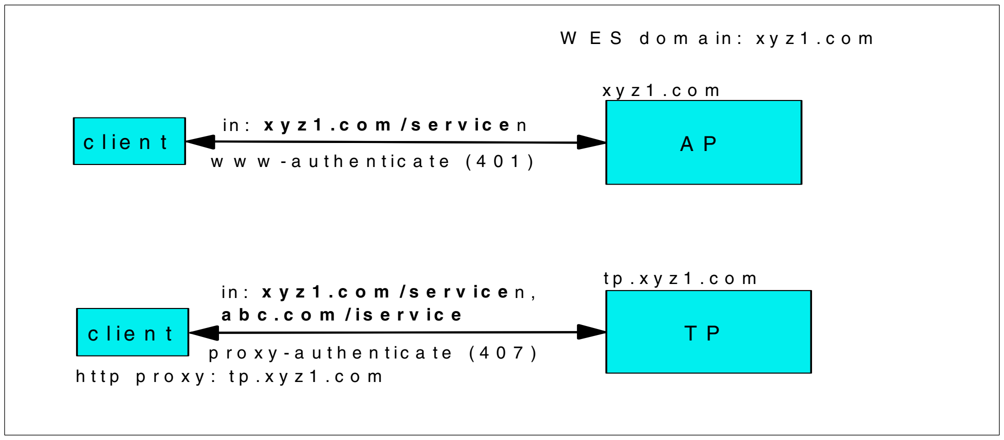


Figure 40. Authentication Proxy and Transparent Authentication Proxy

The Authentication Server automatically generates the challenge, prompting clients to return a user ID and password.

Both challenges will prompt the client to enter a user ID and password for whatever realm the ServerID specifies. In the example configuration, the browser would prompt for a user authentication for the WES realm or domain. Authentication is specified as basic since this is the only form of authentication that the Caching Proxy currently supports. The Mask directive states that all clients can access all resources within the Authentication Server.

If this is the first user request for resources within the Everyplace Suite domain, then the client will send a request for resources to the Load Balancer or to the Everyplace Authentication Server and the Load Balancer will intercept it. The Load Balancer will then dispatch it to the optimal Authentication Server. Detecting that the user is not authenticated, the Authentication Server will return a request for authentication details directly to the client. The Authentication Server will then authenticate the user, creating an Everyplace Suite session, and will pass this session information in the header to subsequent requests for resources in the Everyplace Suite.

You would ideally implement affinity with the Load Balancer if you want to take advantage of any local caching that the Authentication Server may do. If you are using local cache on EAS then you will want to maintain affinity to client and EAS. This allows the Authentication Server to take advantage of local caching of session information. The Authentication Server can cache the session ID and session creation timestamp locally, so that for subsequent requests the Authentication Server does not have to validation the session from AST. The local caching is recommended.

6.3 Authentication Process

To illustrate the authentication process done by the Everyplace Authentication Server, we will walk through a series of user requests in this section.

Figure 41 on page 78 details the authentication process conducted by the Everyplace Authentication Server 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 the Everyplace Authentication Server. The following six steps describe the process:

1. Request credentials for first request

   Assuming a user who is already enrolled in the WebSphere Everyplace Suite domain is trying to access the services hosted by the WebSphere Everyplace Suite, the user sends the first request to the Everyplace Authentication Server using an HTTP browser from the Internet.

   The Everyplace Authentication Server received the request. It inspects the HTTP request header and cannot find any credentials. Then 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.

   The Everyplace Authentication Server determines that the user has not been authenticated by the WebSphere Everyplace Suite yet. It returns a 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. It will prompt the user to enter a user ID and password and puts them in the authorization header and sends it to the Everyplace Authentication Server.

Note

In this release of the WebSphere Everyplace Suite, only HTTP *basic* authentication is supported by the Everyplace Authentication Server. The HTTP *digest* authentication method is not supported.

2. Look up session

   When the user request with the authorization header is returned to the Everyplace Authentication Server, the Everyplace Authentication Server notices the authorization header with a user ID and password. It will first check with the active session database to see whether the user already has an active session. Since this is the first request from the user, no active session exists. The Everyplace Authentication Server thus moves on to step 3.

3. Authenticate the credentials

   Since the user has already supplied the user ID and password while there is no active session for the user request, the Everyplace Authentication Server has to start the authentication and create an active session for the user if the authentication is successful.

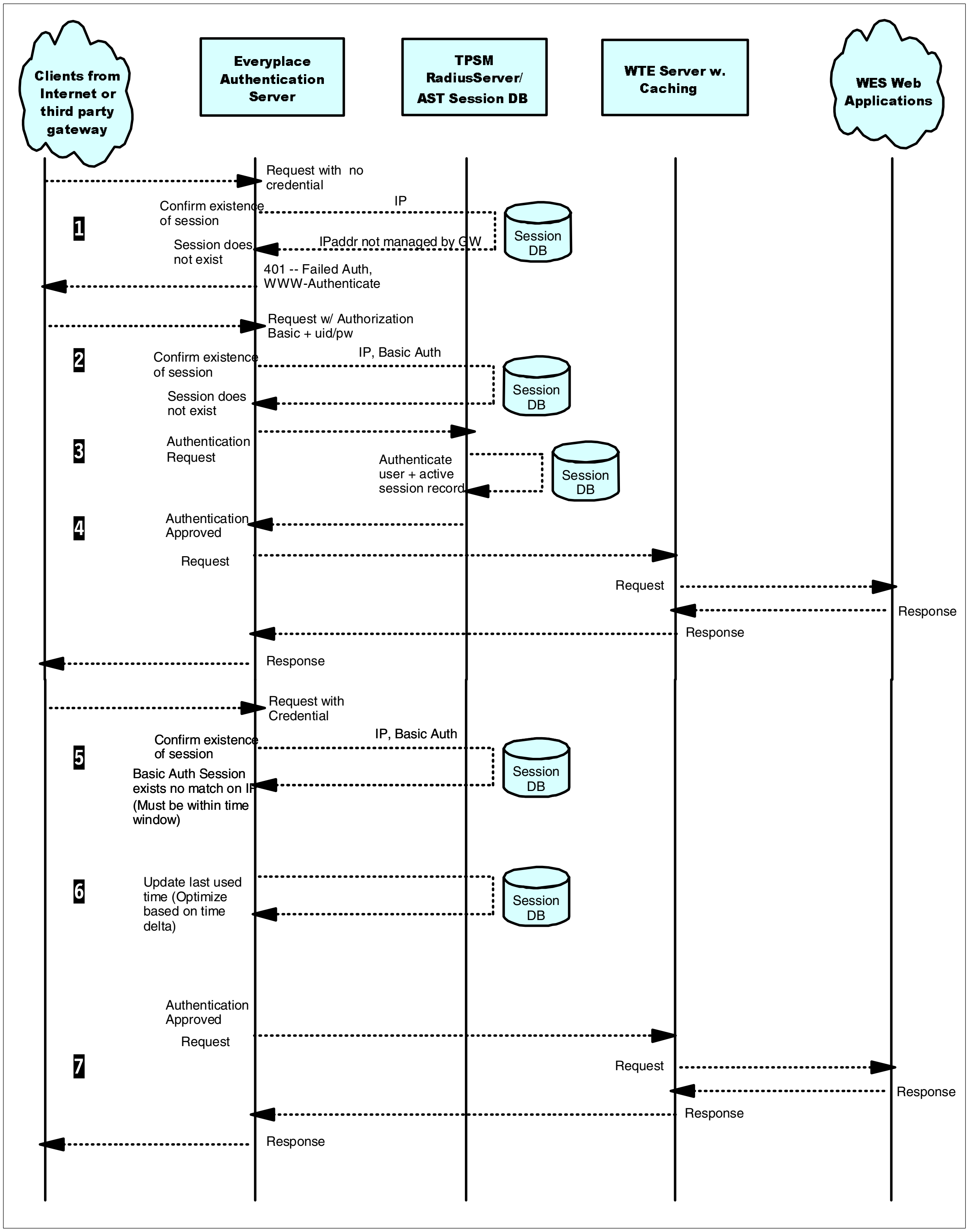

Clients from Internet or third party gateway
Everyplace Authentication Server
TPSM RadiusServer/ AST Session DB
WTE Server w. Caching
WES Web Applications
Request with no credential
1
Confirm existence of session
IP
Session DB
Session does not exist
IPaddr not managed by GW
401 -- Failed Auth, WWW-Authenticate
Request w/ Authorization Basic + uid/pw
2
Confirm existence of session
IP, Basic Auth
Session DB
Session does not exist
3
Authentication Request
Authenticate user + active session record
Session DB
4
Authentication Approved
Request
Request
Response
Response
Response
Request with Credential
5
Confirm existence of session
IP, Basic Auth
Session DB
Basic Auth Session exists no match on IP (Must be within time window)
6
Update last used time (Optimize based on time delta)
Session DB
Authentication Approved
Request
7
Request
Response
Response
Response

Figure 41. Authentication process for an enrolled user from the Internet

The Everyplace Authentication Server sends the authentication requests to the RADIUS server. The RADIUS server will check the user ID and password against the subscriber database. If the user ID and password match those in the subscriber database, the RADIUS server will return authentication as approved. It will also create an active session for the user.

4. Forward request and response

   Once the Everyplace Authentication Server receives a message from the RADIUS server indicating the authentication is approved, it will go ahead and direct the user request to the backend servers for fulfillment. It usually retrieves responses from the Caching Proxy and Web server and send the response back to the user.

5. Second request and look up session

   With the active session established for the user, the subsequent user requests will be handled repeatedly by step 5, 6, and 7, until the active session is expired or the user logs out.

   The second request would have the credentials in the HTTP header. The Everyplace Authentication Server will check against the active session database to see whether the user has an active session or not. Since the user already has a session established in step 3, the Everyplace Authentication Server will consider the user to be pre-authenticated and would not send an authentication request to RADIUS.

6. Update session information

   Instead, the Everyplace Authentication Server will update the record in the active session database for the time of the last visit.

7. Forward request and response

   After the update of the active session for the user, the Everyplace Authentication Server again will direct the request to the appropriate back-end server. The request can be handled by either the caching proxy or the Web application servers. The Everyplace Authentication Server would return the response to the user.

The subsequent user requests will be handled repeatedly using step 5, 6, and 7, until either the user chooses to log out or the active session expires after a predefined period of time.

6.4 Active session management

The TPSM AST server must be used for managing the Active Session Table and its entries. For more information about the AST server, please refer to Chapter 9, “Subscriber and device management” on page 135.

The Everyplace Authentication Server creates an active session and provides active session tracking. It also maintains a list of active sessions and provides information about sessions using headers, as shown in Figure 42.

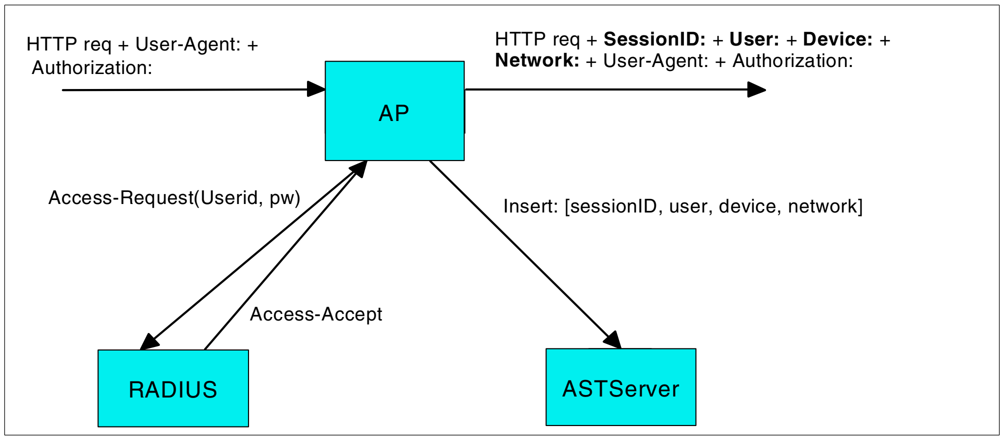

HTTP req + User-Agent: + Authorization:
AP
HTTP req + **SessionID:** + **User:** + **Device:** + **Network:** + User-Agent: + Authorization:
Access-Request(Userid, pw)
Insert: [sessionID, user, device, network]
Access-Accept
RADIUS
ASTServer

Figure 42. Everyplace Authentication Server manages an Active Session Table

6.4.1 Authentication Server session headers

The Everyplace Authentication Server session headers include the following.

6.4.1.1 Querying headers

The Authentication Server utilizes the Caching Proxy's variable access interface to query the HTTP headers that it receives with client requests. The Caching Proxy variables that EAS queries include:

- **REMOTE_USER** to get the user name of the requesting authenticated user.
- **PASSWORD** to get the decoded password basic authentication details.
- **HTTP_USER_AGENT** to get the user agent header for determining the device type and network type.
- **AUTH_STRING** to get the encoded authorization string.

6.4.1.2 Adding headers

The Authentication Server places session headers in each HTTP request that is bound for a hosted service within the Everyplace Suite domain. These headers are used to:

- Correlate individual HTTP requests with a given Authentication Server session.
- Look up the session record for the given session.
- Identify the user, device type, and network type associated with the session.

All requests passing through the Authentication Server are tagged with an Authentication Server HTTP header containing the following information:

- **Session ID**

  The session ID uniquely identifies an active session. The session ID is passed by the Authentication Server session header. By using the session ID, components do not have to know what individual active session entry fields uniquely define a session.

  It is included in the AST entry created by the call to the RADIUS server. It can also be used by other Everyplace Suite components as a convenience to look up an entry in the AST or as a correlator value for requests associated with a given session.

The format of the session will be:

X-IBM-PVC-Session = “X-IBM-PVC-Session”: “session ID”

- **Session location**

  The session location is a pointer to where the AST entry associated with a particular session is located. It consists of the host name of the AST server to which the AST entry was written plus the remote port number.

  The format of the session will be:

  X-IBM-PVC-Session-Location = “X-IBM-PVC-Session-Location”: “<hostname of AST Server>”

  To do a normal lookup of an AST entry, a TCP connection must first be created to the AST server pointed to by session location, and an AST lookup specifying a key of session is passed.

- **User ID**

  The user ID is the user identification, passed by the client device to the Authentication Server when prompted for basic authentication. It is queried by the Authentication Server from the HTTP header. It is also the same ID as known to TPSM through its enrollment process.

  The format normally consists of userid@realm, but realm is not required. If not specified, then TPSM assumes a default realm value.

- **Device type**

  This is the device type as mapped from the user-agent request header.

  The device type is queried from the HTTP header and used as a key to look up a device profile in LDAP.

  The device profiles are extracted from LDAP at initiation time along with the rules required to map to them from request headers.

- **Network type**

  This is based on the originating network of the request.

  The network type is queried from the HTTP header and is used as a key to look up a network profile in LDAP. The network profiles are extracted from LDAP at initialization time.

  If other application servers than WAS are used, they will need to recognize the HTTP header inserted by the Authentication Server and will pass it in raw form to the servlets.

6.4.1.3 Removing headers

The Authentication Server will remove headers from the request only if the request is destined for the Internet or a third-party service. The Transparent Authentication Proxy will remove the proxy-authorization headers from requests prior to forwarding a request.

The exception to this is when HTTPS requests are made for Internet access. The secure request will be proxied to the Transparent Authentication Proxy which will use SSL tunneling to forward the request to the destination content server. With SSL tunneling, a feature of the Caching Proxy, the requests are passed through the Transparent Authentication Proxy encrypted; therefore, the Authentication Server cannot process the header information and the header information is passed on.

Session ID

The session ID is a string generated when an AST entry is written, either directly by the Authentication Server or by the Wireless Gateway. The session ID is used to uniquely identify and differentiate an active session. The session ID is generated from several unique identifiers to ensure that a session is unique for a given user, device, and network.

6.4.1.4 Active session table maintenance

Format: AST Entry = [id, user, ip, create-time, device, network]

The AST clean-up daemon handles AST cleanup and cache maintenance. The daemon is responsible for periodically deleting old entries in the local cache and the AST. When the daemon starts, it will run against a list of known AST servers and issue a relational delete command for all AST entries that are older than the specified maximum sessions age. An AST entry is deleted when its age exceeds a time-out value. The Wireless Gateway manages deletion of sessions it originates and creates.

6.4.1.5 Active Session Cache

The Authentication Server enables you to locally cache session and authentication information. This serves to achieve performance objectives you may have by minimizing the number of RADIUS authentication requests and AST Server access requests. Both the session ID and create time of the sessions can be cached. The create time data is cached to enable the Authentication Server to know when the cache entry is stale, and when the AST entry needs to be re-inserted into the AST server.

If a request comes in that already has a matching session ID in the cache, then there already exists an AST entry for the session and the user is already authenticated, therefore permitting a bypass of the associated RADIUS/AST flows.

AST entries associated with a trusted IP address are not cached since there is no time out associated with this AST entry.

6.5 Scalability and availability

For large user-base applications, multiple Everyplace Authentication Servers, RADIUS servers, and AST servers will be needed. This can be achieved jointly by using techniques such as the Everyplace Authentication Server clustering, RADIUS backups, and AST partitioning, as shown in Figure 43.

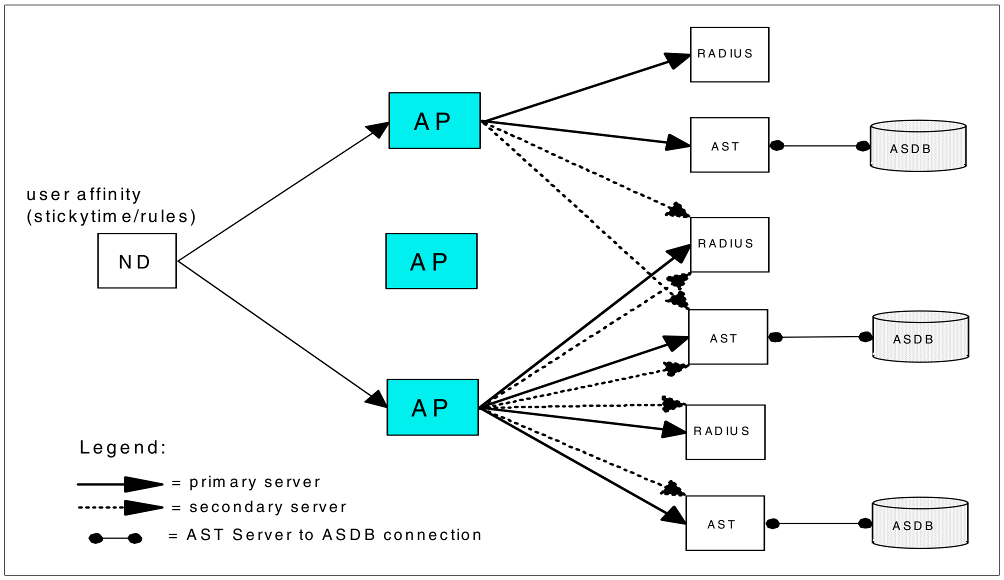


Figure 43. Scalability and availability for Authentication Servers

Multiple Everyplace Authentication Servers can be clustered using the Edge Server - Load Balancer to provide for upward scalability as the number of concurrent users increases. Local AST caching can help reduce the AST database access via the RADIUS server as the workload increases.

In the meantime, multiple RADIUS servers can be used, each as a backup server for others. Each Everyplace Authentication Server can be configured with a primary RADIUS server and one or more backup RADIUS servers. This helps prevent one "over-worked" central RADIUS for a large number of authentication requests. It also helps periodical maintenance as well. However, RADIUS servers cannot be clustered using the Edge Server - Load Balancer.

The AST is stored in a database. With the increasing number of users, scaling also becomes an issue for a single database. The AST database can be partitioned into physically separated and independent databases. The session location header is used by components to know which AST server contains the request's associated session information.

Note

RADIUS servers and AST servers cannot use the Edge Server - Load Balancer to form clusters.

6.6 Deployment scenarios

The deployment and configuration of the Everyplace Authentication Server for the WebSphere Everyplace Suite depends on the specific needs of the organization. This section discusses four kinds of deployment scenarios.

Before we proceed, let us review the configuration files. The configuration information for the Everyplace Authentication Server is maintained in three distinct repositories:

1. On the local file system in the WTE's configuration file (ibmproxy.conf).
2. On the local file system in the Everyplace Authentication Server configuration file (ibmwesap.conf).
3. In the LDAP directory.

The three repositories serve different purposes. The WTE configuration specific to the Everyplace Authentication Server resides as directives in ibmproxy.conf. As explained earlier, these items include the plug-in hooks, the proxy directives, and protection requirements.

Repository 2 contains information needed to access the LDAP server such as the base distinguished name for the WebSphere Everyplace Suite, the LDAP login distinguished name and password, and the LDAP server name and port. In addition, it contains the default setting for the Everyplace Authentication Server.

Repository 3 contains all other configuration parameters. For example, it contains entries common to all Everyplace Authentication Servers:

- RADIUS shared secret
- Maximum RADIUS retries
- Maximum RADIUS retry time-out (milliseconds)
- Maximum session age (minutes)
- Default retry-after delay (seconds)

In addition, it also contains entries unique to each Everyplace Authentication Server such as:

- Everyplace Authentication Server role (Transparent Authentication Proxy or Authentication Proxy)
- RADIUS Authentication Server hostname(s)
- AST server hostname(s) and port(s)
- Maximum session cache sizes
- AST daemon clean-up interval

6.6.1 TP and AP deployed

In order for clients to access both Internet services and services hosted by the WebSphere Everyplace Suite, the Everyplace Authentication Server can be configured in the way illustrated in Figure 45 on page 86. For this case, a client can send one of three requests: wesA for service A hosted by the WebSphere Everyplace Suite, wesB for service B hosted by the WebSphere Everyplace Suite, and an Internet requests.

Since Internet requests must be supported, an Everyplace Authentication Server acting as a Transparent Authentication Proxy is used, denoted as tp.wes.com in the figure. For the services hosted by the WebSphere Everyplace Suite domain, another Everyplace Authentication Server acting as an Authentication Proxy is used as the next hop after the Transparent Authentication Proxy.

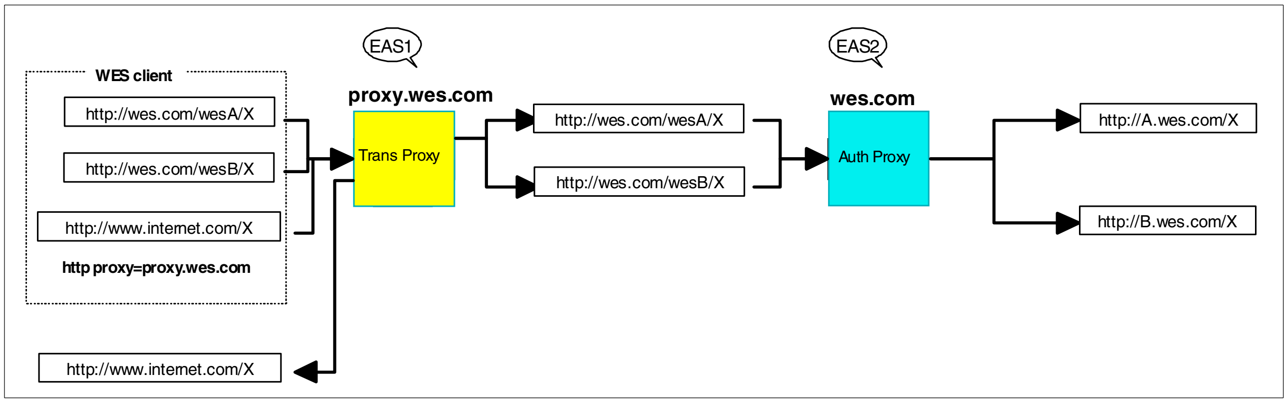


Figure 44. Everyplace Authentication Server scenario 1

The configuration information is shown for both Everyplace Authentication Servers as the following:

- EAS1: Transparent Authentication Proxy
  - ibmproxy.conf:

```
Authorization * /usr/bin.sdpauth.so:authRequest
ServerInit /usr/bin/spdauth.so.initAS /usr/lpp/wes/ibmwesap.conf
ServerTerm /usr/bin/sdpauth.so:termAS
Protect /* {
   ServerID WES
   AuthType Basic
   MASK Anybody@*
}
```

  - ibmwesap.conf:

```
ServerRole TransProxy
```

- EAS2: Authentication Proxy
  - ibmproxy.conf:

```
Authorization * /usr/bin.sdpauth.so:authRequest
ServerInit /usr/bin/spdauth.so.initAS /usr/lpp/wes/ibmwesap.conf
ServerTerm /usr/bin/sdpauth.so:termAS
Proxy /wesA http:/A.wes.com/*
Proxy /wesB http:/B.wes.com/*
Protect /* {
   ServerID WES
   AuthType Basic
   MASK Anybody@*
}
```

  - ibmwesap.conf:

```
ServerRole AuthProxy
```

Please note that only the Authentication Proxy uses the Proxy directive, which defines the next hop routing from the Authentication Proxy. The Transparent Authentication Proxy does not use the Proxy directive.

6.6.2 Only AP deployed

If the organization does not allow users to access the Internet from within the WebSphere Everyplace Suite, the deployment in Figure 45 can be used. The enterprise disallows traffic destined for outside the WebSphere Everyplace Suite domain by not using the Transparent Authentication Proxy.

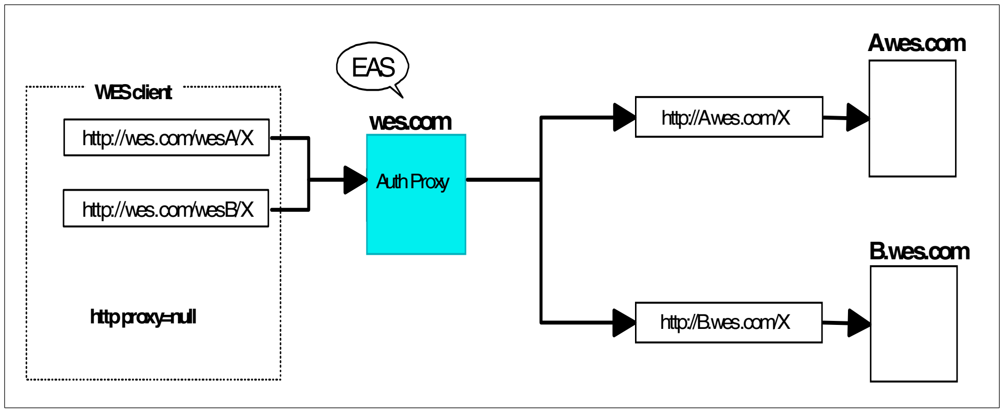


Figure 45. Everyplace Authentication Server scenario 2

In this case, the configuration of the Everyplace Authentication Server is the following:

- EAS: Authentication Proxy
  - ibmproxy.conf:

```
Authorization * /usr/bin.sdpauth.so:authRequest
ServerInit /usr/bin/spdauth.so.initAS /usr/lpp/wes/ibmwesap.conf
ServerTerm /usr/bin/sdpauth.so:termAS
Proxy /wesA http:/A.wes.com/*
Proxy /wesB http:/B.wes.com/*
Protect /* {
   ServerID WES
   AuthType Basic
   MASK Anybody@*
}
```

  - ibmwesap.conf:

```
ServerRole AuthProxy
```

6.6.3 Hybrid TP/AP deployed

We can also combine both Authentication Proxy and the Transparent Authentication Proxy into a single WTE-based Everyplace Authentication Server. This requires that you configure the HTTP proxy with the same name as the enterprise domain name. Figure 46 on page 87 shows this case, where the enterprise domain name is wes.com. The Everyplace Authentication Server is configured as a Transparent Authentication Proxy, and the WTE is configured to proxy all in-domain requests to the proper services while all external requests are forwarded to the Internet.

The configuration of this hybrid Everyplace Authentication Server is given as the following:

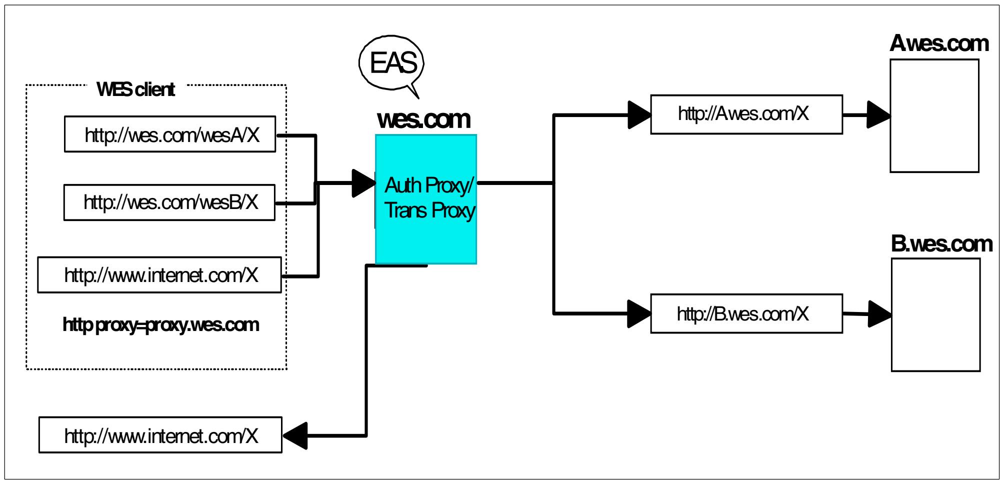


Figure 46. Everyplace Authentication Server scenario 3

- EAS: Hybrid Authentication Proxy and Transparent Authentication Proxy
  - ibmproxy.conf:

```
Authorization * /usr/bin.sdpauth.so:authRequest
ServerInit /usr/bin/spdauth.so.initAS /usr/lpp/wes/ibmwesap.conf
ServerTerm /usr/bin/sdpauth.so:termAS
Proxy /wesA http:/A.wes.com/*
Proxy /wesB http:/B.wes.com/*
Protect /* {
   ServerID WES
   AuthType Basic
   MASK Anybody@*
}
```

  - ibmwesap.conf:

```
ServerRole TransProxy
```

The major differences here are two-fold:

1. Unlike the first scenario where the Transparent Authentication Proxy is configured with proxy name tp.wes.com, the Transparent Authentication Proxy here is configured with the domain name wes.com.
2. Unlike the second scenario where the client HTTP proxy is null, here the Everyplace Authentication Server allows the clients to use HTTP proxy wes.com.

6.6.4 Configuration for transcoding

Another case is to configure the Everyplace Authentication Server in front of IBM WebSphere Transcoding Publisher to support client devices that require certain transcoding. As displayed in Figure 47, the IBM WebSphere Transcoding Publisher server is denoted as A.wes.com and services X and Y require transcoding before they can be presented to client devices.

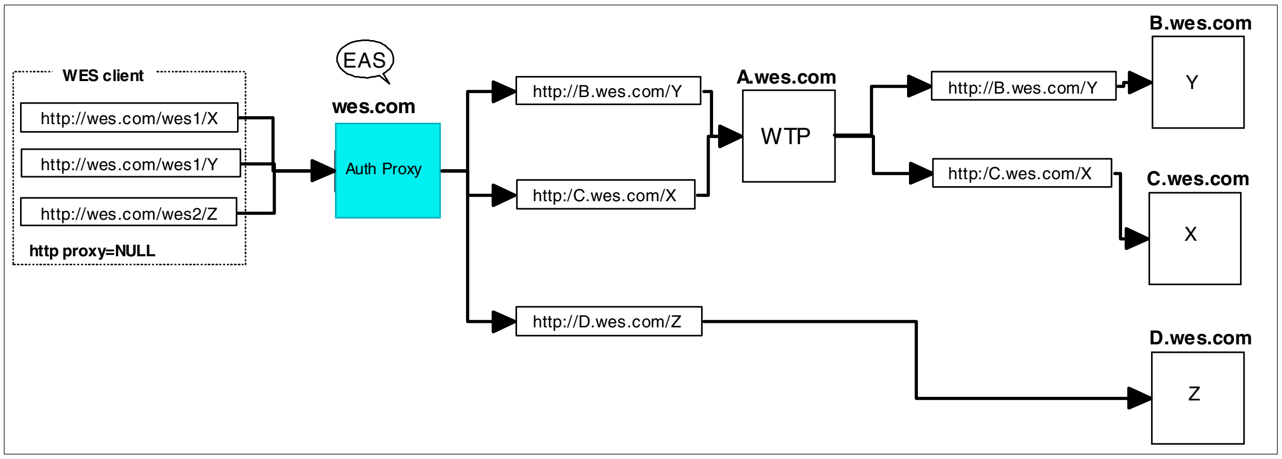


Figure 47. Everyplace Authentication Server scenario 4

The configuration for this case is very similar to the second scenario since the Everyplace Authentication Server configuration is at the URL level.

- EAS: Authentication Proxy for transcoding services
  - ibmproxy.conf:

```
Authorization * /usr/bin.sdpauth.so:authRequest
ServerInit /usr/bin/spdauth.so.initAS /usr/lpp/wes/ibmwesap.conf
ServerTerm /usr/bin/sdpauth.so:termAS
Proxy /wesA http:/A.wes.com/*
Proxy /wesB http:/B.wes.com/*
Protect /* {
   ServerID WES
   AuthType Basic
   MASK Anybody@*
}
```

  - ibmwesap.conf:

```
ServerRole AuthProxy
```

Chapter 7. Supporting wireless devices

As one of the core components of the IBM WebSphere Everyplace Suite, the IBM Everyplace Wireless Gateway (hereafter referred to as the Wireless Gateway) is a distributed, highly scalable connectivity platform that enables secure, optimized access by mobile client devices such as notebook with wireless modems, personal digital appliances (PDAs), and many WAP-enabled smart phones and communicators over a wide range of wireless networks as well as wireline networks such as local area networks (LANs) and wide area networks (WANs). In particular, the Wireless Gateway supports both IP and non-IP wireless bearer networks.

The Everyplace Wireless Gateway is an open-system communication platform that enables IP applications to run in wireless environments. It gives mobile workers access to host and network resources through radio and dial-up networks. Everyplace Wireless Gateway greatly reduces the cost, complexity, and time required to deploy mobile solutions, so that enterprise data and applications can be distributed to workers wherever and whenever they need them. The Everyplace Wireless Gateway extends the corporate network for e-business solutions and protects existing investment in software and information technology infrastructures.

This chapter helps you to understand the functionality included in the Wireless Gateway. We will focus on the architectures and technologies implemented in the Wireless Gateway. Issues about security, deployment considerations, and sample scenarios will be discussed in order to help you plan and configure the new Wireless Gateway functions in different environments.

7.1 Overview

With increasing numbers of business activities being conducted outside traditional office buildings, tools and services that allow mobile professionals to access and interact with information (and services) relevant to their enterprises are in great demand. On the other hand, Internet transactions are becoming increasingly mobile as well. It is forecasted that in the next few years, about two-thirds of all Internet transactions are expected to be generated by new wireless and mobile devices. In the emerging mobile business climate, wireless and mobile devices must work together seamlessly with both the Internet and enterprise intranet. Moreover, given the sensitive nature of business information, transactions must be conducted with the highest levels of security. The IBM Everyplace Wireless Gateway is designed and implemented to provide secure and seamless access for such mobile devices to both enterprise networks and internet applications.

The Wireless Gateway enables TCP/IP, the Point-to-Point Protocol (PPP) and WAP communications in a secure and optimized fashion (see Figure 48 on page 90).

Over wireless and wireline networks, Web and enterprise application programs using a standard TCP/IP interface can be accessed via the Wireless Gateway. The Wireless Gateway provides connections over a wide range of wireless, dial-up, and wireline networks and does so by shielding any network specific operations from the user application. In addition, the Wireless Gateway provides wireless network-specific enhancements to TCP/IP communications. Such enhancements as data compression, encryption, authentication, optimization, and retransmission are particularly beneficial to the wireless clients due to the limited bandwidth, high latency, low reliability, and security risks inherent in today's wireless networking technologies.

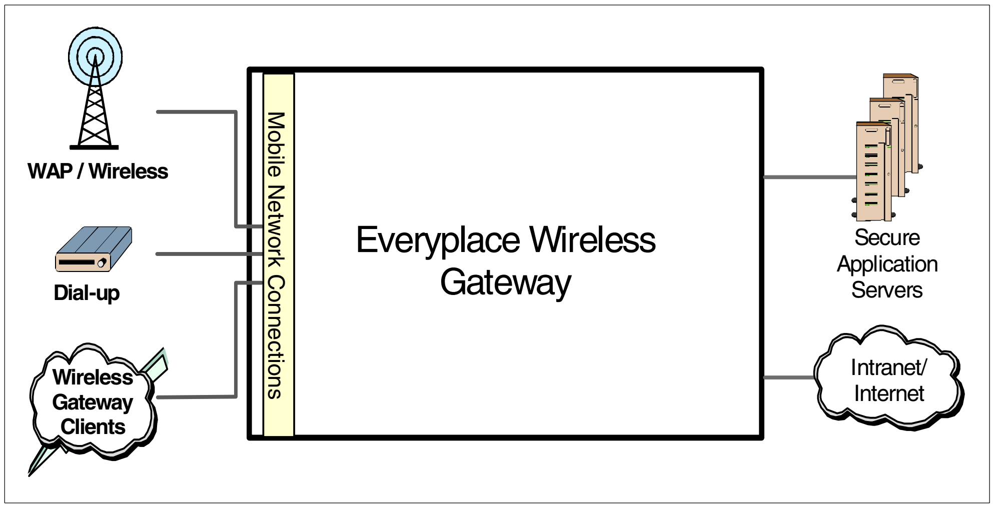

WAP / Wireless
Dial-up
Wireless Gateway Clients
Mobile Network Connections
Everyplace Wireless Gateway
Secure Application Servers
Intranet/ Internet

Figure 48. IBM Everyplace Wireless Gateway provides connectivity

In order to meet the increasing market demand for pervasive wireless devices such as phones or PDAs to access the Internet, the Wireless Gateway supports wireless communications. In particular, using an array of innovative implementations, the Wireless Gateway facilitates the lightweight, low-processing power, and low-bandwidth WAP clients to access the Web. It acts as a translator/encoder between the WAP clients that use the Wireless Session Protocol (WSP), and the Internet, which uses the HyperText Transport Protocol (HTTP). It also acts as an agent for WAP clients such as phones and PDAs by storing cookies and maintaining secure HTTPS connections on their behalf. In the course of providing the WAP communications, the Wireless Gateway implements high-strength security and high-performance caching.

The Wireless Gateway bridges devices over a variety of wireless and wired networks to the Web server/proxy and application server. Such networks can be:

- Public packet-radio networks such as CDPD and GPRS
- Cellular networks such as CDMA, GSM, and PCS 1900
- Dial networks such as PSTN, ISDN, and PPP
- Satellite networks such as Norcom
- Wired networks such as Ethernet and token-ring LANs

The Wireless Gateway accomplishes the protocol transformation from different bearer networks to the TCP/IP interface by using pluggable mobile network connections. It enables three categories of access to the Internet or enterprise applications: WAP-enabled wireless devices, dial-up devices, and wireless clients over either wireline or wireless networks. For a list of networks supported please refer to Appendix A, “Devices/networks supported by the Everyplace Suite” on page 221.

In this chapter, we will briefly describe the Wireless Application Protocol (WAP), the architecture of the Wireless Gateway, and the Wireless Gateway as a core component in the WebSphere Everyplace Suite (WES) environment. Then we discuss in more detail the Wireless Gateway, the client, and the Wireless Gateway administration console, called the Wireless Gatekeeper, as three components of the Everyplace Wireless Gateway. Subsequently, we will discuss the Wireless Gateway security and possible deployment. Finally, a sample scenario will be displayed.

7.2 Everyplace Wireless Gateway in the WebSphere Everyplace Suite

The implementation of WAP support within the IBM WebSphere Everyplace Suite requires some functionalities not traditionally provided by the IBM Everyplace Wireless Gateway. Moreover, most of the functionalities provided by the IBM Everyplace Wireless Gateway are implemented to support wireless devices such as notebooks with wireless modems that are not based on WAP. This section explains how IBM supports the Wireless Application Protocol (WAP) within the WebSphere Everyplace Suite as well as how the Wireless Gateway fits in the suite of WAP support.

7.2.1 Functions

As one of the core components of the IBM WebSphere Everyplace Suite, the IBM Everyplace Wireless Gateway adds connectivity, security,and optimization to the Suite. The Everyplace Wireless Gateway supports numerous world-wide network technologies including the Wireless Application Protocol (WAP). It provides the core WES functionalities by tightly integrating with many other components within the suite.

The Wireless Gateway provides a communication platform that enables Internet Protocol applications to run in wireless environments. The Wireless Gateway provides mobile devices containing wireless clients with wireless access to host and network resources through radio and dialup networks. The Wireless Gateway encrypts, compresses, and minimizes the data that passes through the wireless link, thereby increasing the speed of messaging.

Connectivity

It is the goal of the IBM WebSphere Everyplace Suite to provide information and services to clients over both wireline and wireless networks. The connectivity to the WES domain is provided by the IBM Everyplace Authentication Server (EAS) and the IBM Everyplace Wireless Gateway, as illustrated in Figure 27 on page 57.

While the wireline connection directly from the Internet as well as the wireless connections from third party gateways can be connected to the WES domain through the IBM Everyplace Authentication Server, the IBM Everyplace Wireless Gateway provides non-WAP wireless, WAP wireless, and other wireline connections such as dial-up connections to and from the IBM WebSphere Everyplace Suite. It establishes the communication connection to and from pervasive devices and translates those connections to IP and other standard protocols.

For non-WAP wireless clients, the Everyplace Wireless Gateway includes wireless client application software that extends TCP/IP communications to mobile devices running over a variety of network connections. Like its predecessor, the IBM SecureWay Wireless Gateway, the Wireless Gateway is designed to enable TCP/IP applications to run without modification efficiently, securely, and reliably over packet-radio networks that do not support TCP/IP. In order to preserve the operation of customers' existing TCP/IP applications, the Everyplace Wireless Gateway converts wireless protocols to TCP/IP requests then forwards the TCP/IP requests to the customers' existing TCP/IP networks and the TCP/IP applications. The Wireless Gateway server then converts the TCP/IP response from these applications to the wireless protocols supporting the devices making the request and sends the response over the protocol to the device.

With the increasing popularity of mobile access to the Internet using WAP-enabled phones, supporting WAP devices become the natural progression of the Wireless Gateway. For WAP-enabled mobile devices, the Everyplace Wireless Gateway supports WAP 1.2. The Wireless Gateway server accepts WAP requests over numerous network connections to preserve the operation of existing Web applications. The gateway server converts the request in WAP to an HTTP request and forward that request to the existing Web infrastructure, and in turn translates the HTTP response from the existing Web applications to a WAP response and sends this response to the WAP device.

The Everyplace Wireless Gateway also supports dial-up connections to the WebSphere Everyplace Suite. Such dial-up connections can be from a wireline network or a wireless network using supported wireless modems. The Wireless Gateway utilizes the Point-to-Point Protocol to authenticate the dial-up clients and provides secured tunneling to the TCP/IP applications hosted by the WES domain. However, in order for the clients to access the Everyplace Wireless Gateway, the Everyplace Wireless Client software application must be installed on the client devices (except for PPP connections).

Security
The Everyplace Wireless Gateway server and client application software work together to establish secure wireless and wireline communications between devices and the Wireless Gateway by providing two-way authentication, and data encryption for confidentiality.

The IBM Everyplace Wireless Gateway bridges pervasive devices to IP networks. When possible, it also maintains a VPN link between the pervasive devices and the WES services. Links to WES through the Wireless Gateway are considered to be trusted by the WebSphere Everyplace Suite domain, because the Wireless Gateway terminates and authenticates the data link layer of the network stack.

For example, HTTP requests generated by devices connected via the IBM Everyplace Wireless Gateway are considered to be “pre-authenticated” because they are associated with a trusted IP address that is controlled by the Wireless Gateway.

The Everyplace Wireless Gateway authenticates:

1. Dial-in connections from devices using the Point-to-Point Protocol (PPP) to make the connection.
2. Wireless connections from devices that support WAP.
3. Secure connection from devices using the Everyplace Wireless Client to make the connections. For devices that use Wireless Clients, customers must install the client software on the devices.

For the Everyplace Wireless Gateway client-to-server connections, the Wireless Gateway supports the Data Encryption Standard (DES), RC5, and 3-DES encryption world-wide. This results in a secure tunnel between client and server using the WLP protocol.

For WAP connections, the Everyplace Wireless Gateway utilizes the Wireless Transport Layer Security (WTLS) to perform gateway authentication and data encryption. WTLS is a WAP security implementation similar to the traditional Transport Layer Security (TLS) over TCP/IP networks. However, WTLS does not interoperate with TLS. More details about the Wireless Gateway security are provided in 7.7, “Wireless gateway security” on page 110.

An additional security measure prevents unwanted packet data from being transmitted. The Everyplace Wireless Gateway provides packet filtering for wireless clients and PPP connections as well as a mapping mechanism to filter out unsolicited or unwanted data. This not only reduces the chance of malicious attacks but also reduces the traffic on the wireless link.

Note

Packet filtering is not available for WAP connections.

Optimization

The Everyplace Wireless Gateway server and client application software work together to optimize the wireless communications between devices and the server. The client and server software compresses data to reduce unnecessary protocol headers and optimizes TCP communication to increase the effective data rate and lower the network cost.

The first optimization technique provided by the Everyplace Wireless Gateway is compression. Prior to the transmission of the IP packets, the size of each IP packet is reduced without any impact on the content of the packet. This compression increases the effective data rate of the wireless network and it also decreases the amount of data transmitted and therefore decreases the transmission cost in most cases.

The second optimization technique provided by the Everyplace Wireless Gateway also works for TCP traffic only. It is called TCP header reduction. This technique reduces the 40-byte TCP headers to an average size of 3-5 bytes, which decreases the amount of data to be transmitted over the wireless network.

The third technique is called TCP retransmission optimization. It’s been observed that TCP communication over wireless links often results in packet retransmission because of the high latency and small bandwidth available. The IBM Everyplace Wireless Gateway provides unique solutions to address such problems.

Finally there is a technique that reduces the air time on connection-oriented wireless network and PSTN. This technique is called short hold mode. With short hold mode enabled, there can be no established physical connection over the mobile network, but the client and server are virtually connected. If the client or gateway requests to transmit an IP packet, the client or gateway will re-establish the connection and start the transmission immediately. As a result, this technique may lower connection fees.

7.2.2 Integration

While contributing its unique functionalities, the IBM Everyplace Wireless Gateway is tightly integrated into the WebSphere Everyplace Suite, sharing a common WES environment and jointly achieving more pervasive computing objectives. As shown in Figure 49, the Wireless Gateway integrates with TPSM to update the user database, connects to the LDAP sharing user profiles, communicates with the RADIUS server for authentication, collaborates with the Everyplace Authentication Server for Active Session Table (AST) and user session information sharing, and works with the Edge Server - Caching Proxy to provide WAP push proxy and binary WML caching.

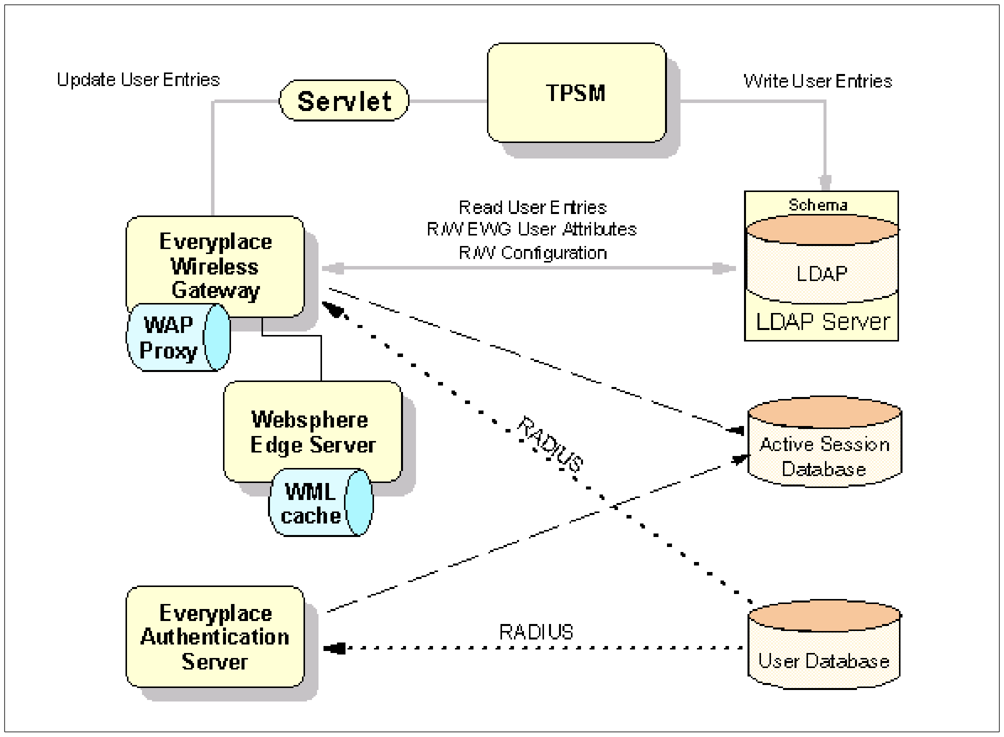


Figure 49. Integration of Everyplace Wireless Gateway

Within the WebSphere Everyplace Suite, the Tivoli Personalized Service Manager (TPSM) handles user enrollment and user database management. Typically, users are enrolled in the WebSphere Everyplace Suite through TPSM for subscribed Internet and/or wireless access to the WES services. It is desirable to have a centralized user database to simplify database management and enforce the consistency. Logically, only TPSM is responsible for creating and updating entries in the user database. For users from the wireline Internet, user profile management can be easily achieved by TPSM in a centralized fashion. However, clients over the wireless network should also be able to update their user profiles even they do not have direct access to TPSM services. This is achieved through the Everyplace Wireless Gateway by using a servlet that connects to TPSM and updates the user profiles. Moreover, for those service providers who have not implemented TPSM in the past, they can migrate the users already registered with the Wireless Gateway to TPSM through this servlet.

As displayed in Figure 49, the user profiles is stored in the WES directory service provided by the LDAP server. Only TPSM can directly write the user entries to

LDAP. The Wireless Gateway can modify the user entries through TPSM. Such user entries can be read by other components such as the WebSphere Application Server.

Configuration information about the Wireless Gateways and associated resources is stored in a database that conforms to the Lightweight Directory Access Protocol (LDAP). An LDAP database can reside on any attached host and can store information about more than one Wireless Gateway. Using LDAP, resources can be shared among Wireless Gateways.

Unlike the earlier versions of the IBM Secureway Wireless Gateway, which authenticates clients locally, the Everyplace Wireless Gateway within the WebSphere Everyplace Suite authenticates client access using the shared RADIUS server, shipped with the WebSphere Everyplace Suite as a standard component. As explained in earlier chapters, both the Everyplace Wireless Gateway and the Everyplace Authentication Server can carry out the authentication process for the WES domain utilizing the RADIUS server. For those clients authenticated by the Everyplace Wireless Gateway, the Wireless Gateway assigns a trusted IP address which is owned by the Wireless Gateway and known to the Everyplace Authentication Server. When HTTP requests are routed to the Everyplace Authentication Server from the Wireless Gateway, the trusted Wireless Gateway IP address is embedded in the header and acknowledged by the Everyplace Authentication Server; thus the Everyplace Authentication Server will not authenticate the client again.

The synergy between the Everyplace Wireless Gateway and the Everyplace Authentication Server also provides the Active Session Table (AST) and user session information sharing.

For wireless clients entering a WES domain through the Wireless Gateway, because the duration of the user sessions is defined by their connection to the Wireless Gateway, the Wireless Gateway is responsible for setting up and maintaining these user records within the Active Session Table. For WAP clients, the Authentication Server performs this function. In addition, since the users might request access to the services hosted by the WES domain, the Everyplace Authentication Server has to be able to modify the active session information and provides appropriate logging and tracing as well. Hence, there is collaboration between the Everyplace Wireless Gateway and the Everyplace Authentication Server over AST management. More details are provided in Chapter 6, “Authentication” on page 69.

To enhance the performance of the WAP services provided by the WebSphere Everyplace Suite, the Everyplace Wireless Gateway has been integrated with the Edge Server - Caching Proxy (WTE) as well. The Wireless Gateway shipped with the WebSphere Everyplace Suite now has a WTE plug-in to enable the caching of WAP data (WML) by the proxy server to improve performance, which traditionally can not be done by the WTE.

Note

Wireless Gateway WML caching is only possible with WebSphere Traffic Express (WTE).

7.3 The Everyplace Wireless Gateway

This section and the following two sections discuss the software application components included in the Everyplace Wireless Gateway: the Wireless Gateway, Wireless Client, and the administration console, called the Wireless Gatekeeper. Much of the pertinent information about the Wireless Gateway and Wireless Client has been covered in *Mobile Computing: The eNetwork Wireless Solution,* SG24-5299. We will primarily present the new and extended functionality of the gateway and client implemented after this earlier redbook.

7.3.1 Architecture of the Everyplace Wireless Gateway

The Everyplace Wireless Gateway provides a TCP/IP interface with data optimization and security to a variety of wireless and dial-up networks. The Wireless Gateway has been implemented for both AIX and SUN Solaris systems. As illustrated in Figure 50, the Wireless Gateway is comprised of a layered structure for the protocol transformation from bearer networks to TCP/IP, WAP, and WEB stacks. It also consists of many horizontal modules to interface with the Gatekeeper and to communicate with cross-functional components within the WebSphere Everyplace Suite for management, data storage, and authentication.

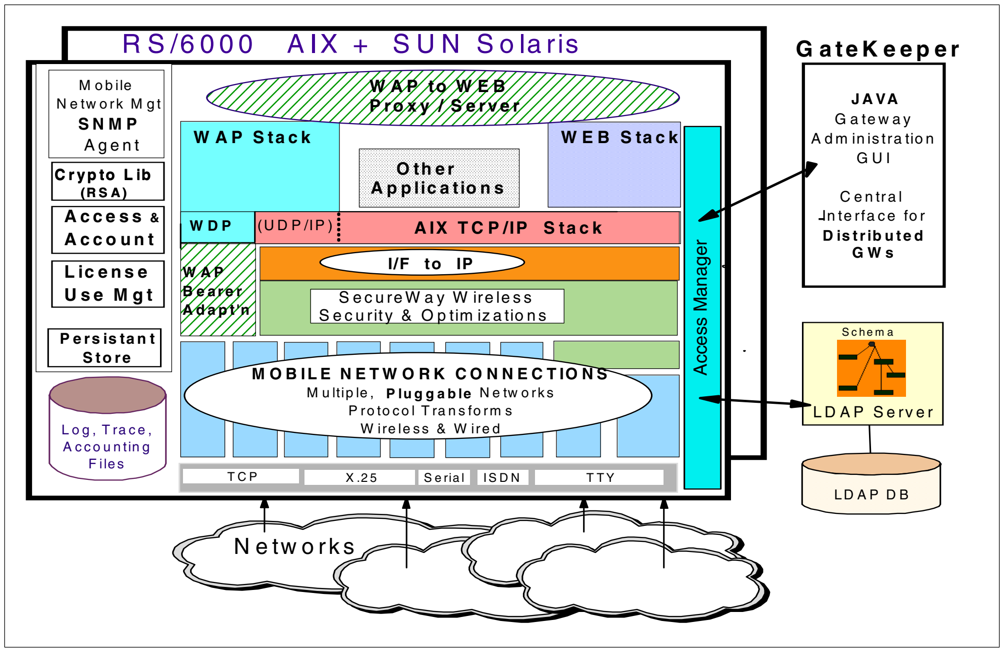

RS/6000 AIX + SUN Solaris
GateKeeper
Mobile Network Mgt SNMP Agent
Crypto Lib (RSA)
Access & Account
License Use Mgt
Persistant Store
Log, Trace, Accounting Files
WAP to WEB Proxy / Server
WAP Stack
Other Applications
WEB Stack
WDP
(UDP/IP)
AIX TCP/IP Stack
I/F to IP
WAP Bearer Adapt'n
SecureWay Wireless Security & Optimizations
MOBILE NETWORK CONNECTIONS
Multiple, Pluggable Networks
Protocol Transforms
Wireless & Wired
TCP
X.25
Serial
ISDN
TTY
Access Manager
Networks
JAVA Gateway Administration GUI
Central Interface for Distributed GWs
Schema
LDAP Server
LDAP DB

Figure 50. Everyplace Wireless Gateway architecture

Let us look at the layered implementation of the gateway first. At the very bottom is the mobile network device layer, which provides for access to the mobile networks such as wireless radio networks. Just above this layer is a set of pluggable modules called Mobile Network Connections (MNCs) that transforms different bearer network protocols into suitable data for further processing into IP, WDP, or UDP/IP protocols. The MNC modules contain network-specific functions and can be selectively installed for only those networks that the Wireless Gateway would support.

The layer above the Mobile Network Connections is the layer that distinguishes the Everyplace Wireless Gateway from other network access routers. This layer contains the IBM implementation of security such as data encryption and user authentication; optimization such as hear reduction, compression, and TCP optimization; and wireless client link, such as IP configuration and other client options. This layer also contains the IBM implementation of WAP bearer network adaptation, which allows the gateway to transform the WAP bearer protocols into WDP.

The next layer contains IP and WDP related functions. This layer supports application and presentation layers above itself to form WAP and Web stacks as well as other TCP/IP based applications.

Now let us examine other utility modules in the Everyplace Wireless Gateway. First, the Wireless Gateway contains modules and daemons to communicate with the Wireless Gatekeeper and LDAP. Such modules and daemons include the Access Manager and Cluster Manager. Secondly, the Gateway includes a set of modules that can connect to external RADIUS servers, Tivoli management software applications, and license user management software applications.

Access Manager facilitates the communication between the Gatekeeper and the persistent data storage, including both LDAP and other relational databases.

Note

In the Wireless Gateway, persistent storage implements a DB2 database while log, trace, and accounting information can be kept in either DB2 or flat files.

The Wireless Gatekeeper is an XML and Java-based graphic user interface tool to administer wireless IT resources. The Cluster Manager enables the clustering among multiple Everyplace Wireless Gateway servers to provide scalability and availability. Both Access Manager and Cluster Manager will be discussed later in more detail.

The Everyplace Wireless Gateway stores its configuration information in the LDAP service provided within the WebSphere Everyplace Suite. The LDAP database can reside on any WES domain host and can store information about more than one Wireless Gateway. Using LDAP, resources can be easily shared among Wireless Gateways.

For scalability and the provision of virtual private networks (VPN) and interfacing to existing corporate or subscriber access control, a RADIUS server is used by the gateway to authenticate the users to the WebSphere Everyplace Suite. A RADIUS server is shipped with TPSM of the WebSphere Everyplace Suite. The Wireless Gateway can communicate with Tivoli management tools in order to send SNMP traps (alerts).

7.3.2 Mobile network connections (MNC)

The Everyplace Wireless Gateway provides the solution to extend Internet Protocol (IP) connectivity across a diverse set of wireless and wireline networks, to enable TCP/IP applications to seamlessly access enterprise networks.

Each Mobile Network Connection (MNC) is the Wireless Gateway's interface to a specific type of network. Typically, an MNC consists of the communication line driver, the network protocol interpreter, and one or more ports. For each supported network provider, the customer has to install and configure an MNC. *Administrator's Guide for Wireless Gateway* contains a table that lists 15 types of MNCs supporting networks such as ARDIS, Dataradio, DataTAC, and Mobitex. Many of those networks are also supported by the earlier IBM SecureWay Wireless Gateway. The Everyplace Wireless Gateway adds support for more networks, such as the Norcom satellite network and many WAP bearer networks.

7.3.3 Wireless Gateway Clustering

In order to handle high traffic networks, clustering has been incorporated into the design of the Wireless Gateway. Clustering, in the context of the Everyplace Wireless Gateway, is the grouping of several machines (each running one instance of the Gateway), as illustrated in Figure 51 on page 99. 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.

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 AM supports the many-to-many relationship between Gatekeepers and Gateways. The AM 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 AM communicates between the Gateway to Gatekeeper using XML messages.

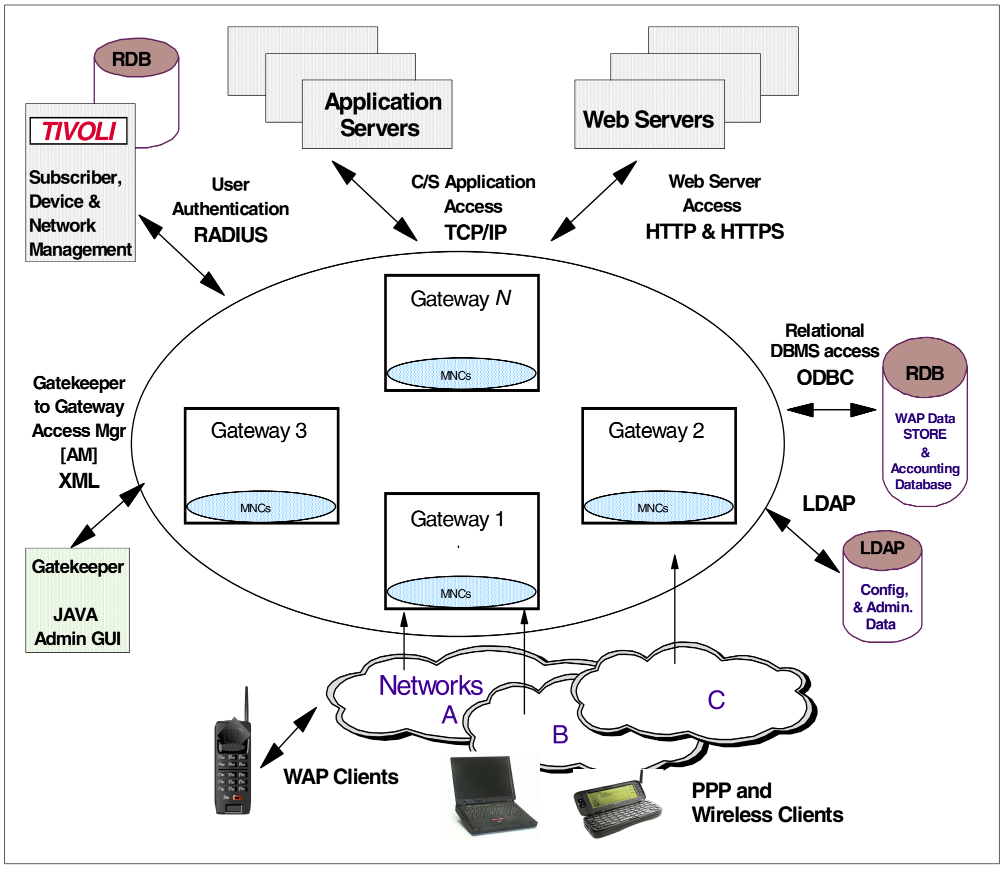

RDB
TIVOLI
Subscriber, Device & Network Management
Application Servers
Web Servers
User Authentication RADIUS
C/S Application Access TCP/IP
Web Server Access HTTP & HTTPS
Gateway N
MNCs
Gateway 3
Gateway 2
Gateway 1
Relational DBMS access ODBC
RDB
WAP Data STORE & Accounting Database
LDAP
LDAP
Config, & Admin. Data
Gatekeeper to Gateway Access Mgr [AM] XML
Gatekeeper JAVA Admin GUI
Networks A
B
C
WAP Clients
PPP and Wireless Clients

Figure 51. Distributed IBM Everyplace Wireless Gateway servers

Figure 51 shows all the components of a Gateway cluster. Tivoli, the application servers and the Web servers are all accessed through the WebSphere Everyplace Suite. The Gateway itself communicates with the Gatekeeper, a relational database, LDAP, and with outside networks via MNCs.

7.3.4 Cluster Manager (CM)

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, the Gatekeeper must be configured 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 more than one *principal* node, each with 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.

Note

In order to prevent possible prolonged downtime, the *principal* node should be backed up frequently.

The scenario in Figure 52 shows three networks: A, B, and C, and four Wireless Gateways (GW): 1, 2, 3, 4 respectively. WG1 has an MNC to network A while WG2 has MNCs to networks A and B. In Figure 52, WG1 is the *principal* node with WG3 and WG4 being its *slaves*. WG1 receives network traffic from A and distributes it over WG3, 4 and itself. WG2 does not participate in this cluster.

The scenario in Figure 53 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.

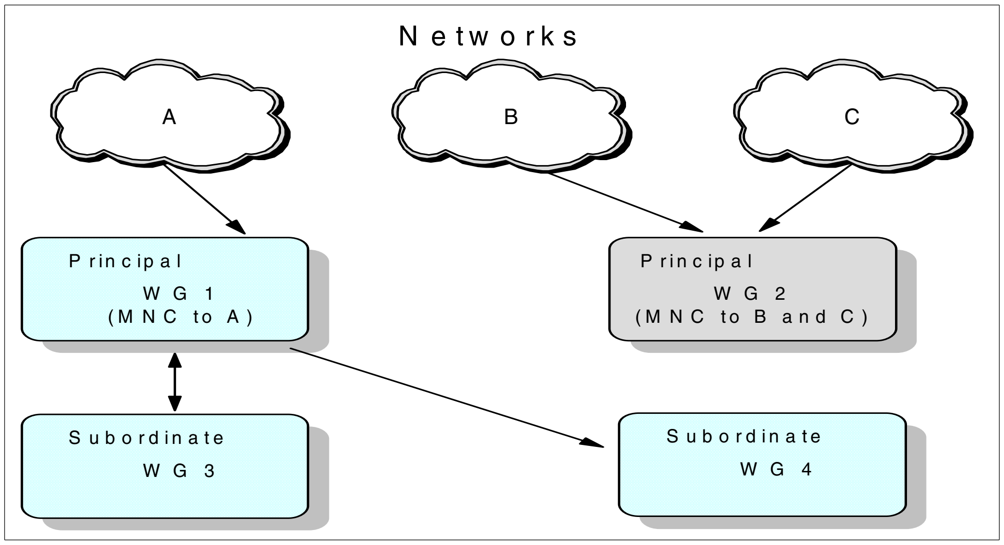


Figure 52. A Wireless Gateway clustering scenario 1

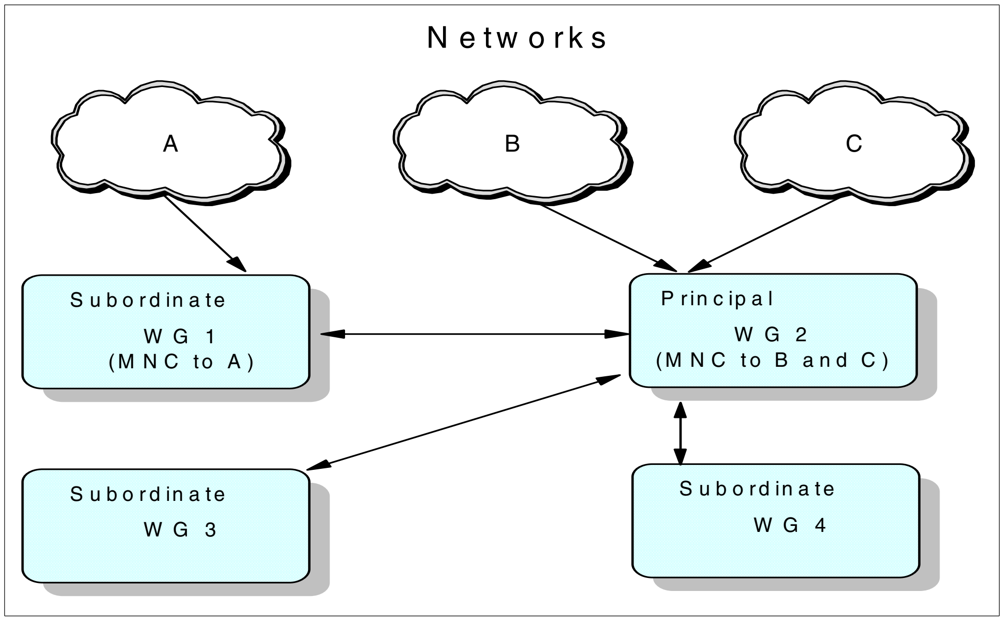


Figure 53. A Wireless Gateway clustering scenario 2

7.3.5 Access Manager and persistent data storage

The Access Manager is a daemon that manages communications among the gateway, the Wireless Gatekeeper, and the persistent data store. There is a secure Access Manager that can be optionally installed to use SSL to secure the communications between the Wireless Gatekeeper and the gateway.

The Everyplace Wireless Gateway does not use the local AIX ODM to be the persistent data storage, as did the former IBM SecureWay Wireless Gateway. Instead, the Everyplace Wireless Gateway uses LDAP directory service and relational database such as DB2 within the WebSphere Everyplace Suite to store the persistent data.

The configuration information is stored in a database that conforms to the Lightweight Directory Access Protocol (LDAP). IBM SecureWay Directory is shipped with the WebSphere Everyplace Suite. The LDAP service can reside on any host protected in the WES domain and can store information about more than one gateway. Using LDAP, resources can be easily shared among multiple Wireless Gateways.

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. Such relational database can reside on any host within the WES domain and can store session information for multiple Wireless Gateway servers.

In addition, the Everyplace Wireless Gateway also stores the accounting and billing information in an attached 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.

The Access Manager or the secure Access Manager is invoked when the Wireless Gatekeeper connects to the Wireless Gateway. It acts as both the query translator and the command launchpad. The communication between the Wireless Gatekeeper and the Wireless Gateway is coded in XML format. Typically, the Wireless Gatekeeper sends the query to the gateway and receives definitions from the gateway. The communication starts when the Wireless Gatekeeper logs on to the gateway. It sends an XML query to the Access Manager. The Access Manager converts the query into AIX commands and issues these commands to the Wireless Gateway or the persistent data store. Once the Access Manager receives the results of those commands from the Wireless Gateway or persistent data store, it converts them into XML and returns them to the Wireless Gatekeeper.

7.4 The Clients

7.4.1 The Wireless Client

Devices that are not WAP-enabled or PPP connected generally are not able to connect to the Wireless Gateway directly. For these devices, the Wireless Client is provided. The Wireless Client is an interface that allows non-WAP mobile devices to run TCP/IP applications over wireless networks. The Wireless Client runs below TCP/IP and handles all network-specific details inside a common interface layer. To the end user, the wireless network becomes just like any other network.

Figure 54 on page 103 depicts the Wireless Client below TCP/IP and before the wireless radio network. To the mobile device, the network connection is seamless; accessing the company intranet or manipulating an Internet application is the same as if you were on a wired connection.

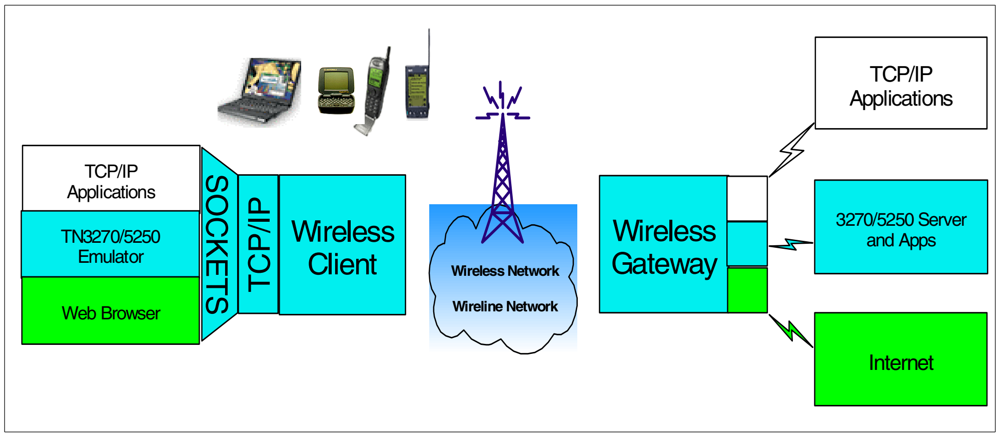


Figure 54. The Wireless Client in the context of the Wireless Gateway

A list of devices that are currently supported by the Everyplace Wireless Client software application is given in Appendix A, “Devices/networks supported by the Everyplace Suite” on page 221. For further information on the Wireless Client and details about installation refer to Chapter 4 in *Mobile Computing: The eNetwork Wireless Solution,* SG24-5299, and also to Chapter 3 in the *Everyplace Wireless Gateway Administrator’s Guide*.

7.4.2 WAP Clients

A WAP client is any mobile device that is WAP-enabled. Such devices connect to the Wireless Gateway using a microbrowser. Extra wireless client software is not needed in order to access the Wireless Gateway, although the Gateway needs to be configured as a WAP server in order to provide WAP services. The Gateway connects to WAP clients through the WAP-enabled MNC. WAP clients can be configured as users to the Wireless Gatekeeper or to a network/remote access server.

The Wireless Gateway supports any device that is WAP V1.1 compliant. Some examples of these devices are the Nokia 7110 and the Ericsson R380 and R390.

HDML phones from Phone.com are currently not supported. Support for these devices is planned for a future release of the WebSphere Everyplace Suite.

The Wireless Gateway is compatible with some WAP V1.0 phones as in the case of some European models. Extra transcoding from V1.0 to V1.1 is necessary for proper viewing in the microbrowser.

For additional information about connecting a specific device to a chosen wireless network, refer to manuals accompanying the device and to documentation provided by your wireless network provider.

7.5 Administration: the Wireless Gatekeeper

The Wireless Gatekeeper is a platform-independent graphical user interface that remotely administers one or more Wireless Gateways. With the Gatekeeper you can easily define and configure Gateway resources as well as register users or mobile devices, specify tracing, and perform other administrative tasks. The

Gatekeeper is a Java-based application that communicates with a Gateway using XML.

The Gatekeeper was designed to give customers an easy-to-use interface to administer their Wireless Gateways. The Gatekeeper was designed with the three following parameters:

1. *No AIX skills are needed to configure the Wireless Gateway.* This edition of the WebSphere Everyplace Suite runs only on the AIX platform. For the ease of use of customers, the Gatekeeper is a Java-based GUI that is multi-platform. No AIX skills are necessary to use the GUI.
2. *The Gateway can be configured remotely.* The Gatekeeper communicates with the Gateway using a SOCKS connection over TCP/IP. Customers can configure their Gateway from their desktop inside a LAN, or even from a remote laptop. TCP/IP communication between the Wireless Gateway and the Gatekeeper is required.
3. *The Gatekeeper will provide an easy-to-use GUI.* For non-DOS/UNIX/AIX/Linux users, a GUI interface is a godsend compared to using an AIX command line. The GUI for the Gatekeeper is designed to be intuitive and easy to learn. Of course the Gateway can be administered using an AIX command line for those who prefer it.

Certain security considerations are necessary when administering your Gateway outside a trusted network. Gatekeepers administrators can be locked to a given IP address. This may not be very secure if the Gatekeeper is coming through a proxy server or other such network configuration. In addition, for extra security, SSL can be configured between the Wireless Gatekeeper and the Wireless Gateway.

7.6 WAP gateway/proxy

The major functional expansion of the Everyplace Wireless Gateway is its support for the Wireless Application Protocol (WAP). With the emphasis on supporting pervasive devices, which often are WAP-enabled devices, the WebSphere Everyplace Suite focuses on utilizing the Wireless Gateway as a WAP gateway/proxy. Therefore, this section describes the Everyplace Wireless Gateway as a WAP gateway/proxy even though the Everyplace Wireless Gateway can function much more than just as a WAP gateway/proxy. Details on how to utilize the Everyplace Wireless Gateway for access to enterprise hosts and TCP/IP applications can be found *Mobile Computing: The eNetwork Wireless Solution,* SG24-5299.

7.6.1 WAP programming model

By adopting the WAP programming model similar to the traditional WWW programming model, the WAP community is able to leverage existing tools such as Web servers and XML tools, and to work on a proven architecture. As illustrated in Figure 55 on page 105, a generic WAP gateway bridges between the mass-market handheld mobile devices, which run with small mobile terminals such as phone.com's micro-browser, and the content servers, which are based on the traditional Web server technology. A basic WAP gateway serves at least the following two functions:

- Protocol translation: the Wireless Gateway translates requests from WAP protocol stacks (which will be explained later in Figure 63 on page 116) to the WWW protocol stacks such as HTTP and TCP/IP.
- Content encoding and decoding: the Wireless Gateway translates WAP content into a compact format for transport over the network.

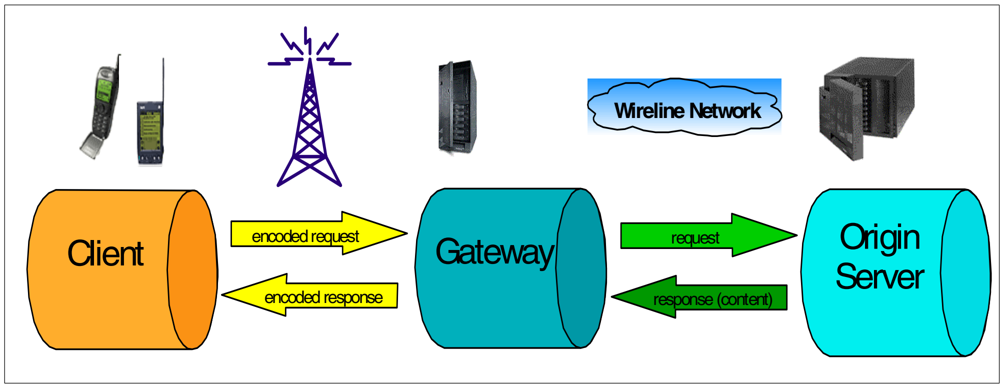


Figure 55. The generic WAP programming model proposed by the WAP Forum

The IBM WebSphere Everyplace Suite has implemented support for WAP clients with extensive considerations for performance, security, and optimization. The IBM Everyplace Wireless Gateway serves the bridging point between the WAP clients and the WES domain. It decodes the WAP requests into HTTP requests to the content servers and encodes the responses to the clients. To enhance performance, the IBM Everyplace Wireless Gateway has also implemented a binary WML plug-in to the traditional HTTP caching server. This feature requires and only works with the WebSphere Traffic Express (WTE) caching proxy.

Additional functionalities are included in the IBM Everyplace Wireless Gateway. The Wireless Gateway can store cookies on behalf of WAP clients. The Wireless Gateway can also open SSL links on behalf of WAP clients when the WAP clients request an HTTPS connection. With the increasing complexity of the Internet applications, the IBM implementation of the WAP gateway needs to be more than a simple WAP proxy.

7.6.2 IBM WAP implementation

Given the above discussion about the WAP programming model and the WAP architecture, we present the IBM implementation of the WAP support within IBM WebSphere Everyplace Suite in Figure 56.

IBM adopts multiple layers of intermediaries such as IBM WebSphere Transcoding Publisher and the Edge Server - Caching Proxy to channel the Web and/or enterprise content to WAP clients. Within the IBM WebSphere Everyplace Suite, the IBM Everyplace Wireless Gateway acts as the entry point for WAP devices. It provides the protocol translation from the IP or non-IP bearer networks to TCP/IP and translates the WAP request into HTTP requests. However, the WAP content caching and content transcoding from HTML/XML to WML are accomplished separately via the Edge Server - Caching Proxy and IBM WebSphere Transcoding Publisher, two major components in the IBM WebSphere Everyplace Suite. Such distributed implementation is beneficial to performance, security, and scalability.

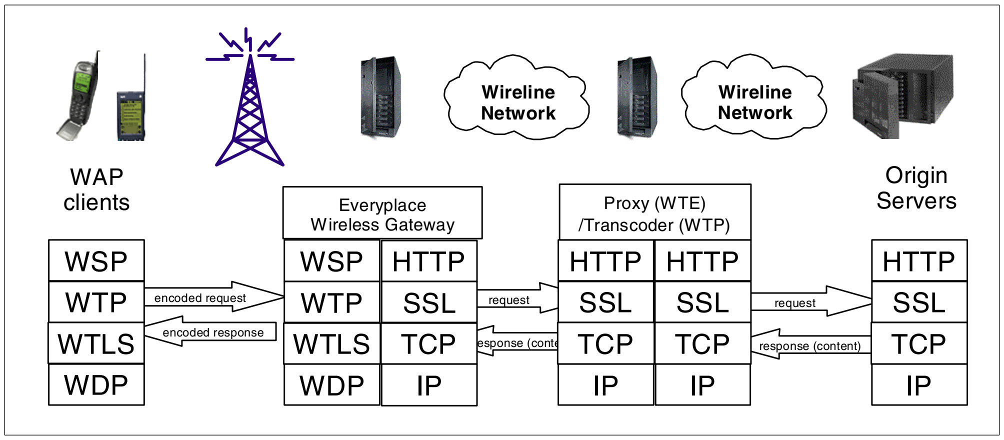


Figure 56. IBM implementation of WAP support

With the caching and content transcoding separated, the Wireless Gateway can focus on user/device authentication, network translation, and network optimization. Even a single Wireless Gateway can support hundreds of thousands of active WAP clients. On the other hand, the Web content caching is a well-known art, which has been satisfactorily delivered by the Edge Server - Caching Proxy. By simply providing a WTE plug-in, the Wireless Gateway can use WTE to cache binary WML data to avoid repetitive conversion and improve response time.

IBM WebSphere Transcoding Publisher is designed to support the content publishing model of write once and present many. It can translate the XML-based documentation into many different markup languages, including WML. Therefore, it is ideal to group the transcoding functionality into a new layer of content intermediary, which not only enhances performance but also helps better deployment of content publishing. Transcoding from HTML/XML to WML usually requires intensive computation power. Therefore, having an array of Transcoding Publisher servers in the back end suits large-scale deployment and better performance.

7.6.3 WAP basic functions

The IBM Everyplace WAP gateway is a scalable UNIX WAP gateway. It contains all the functional blocks for the basic WAP gateway functions, as shown in Figure 57 on page 107. It consists of the WAP network access, TCP/IP and HTTP stacks, WAP stack, WAP gateway bridging functions, and WAP gateway management functions.

The WAP network access provides a link to the network operators for data connectivity. Such access is defined as a mobile network connection (MNC), a resource of the Everyplace Wireless Gateway. WAP network access includes support to interface with both IP bearer networks and non-IP wireless datagram protocol (WDP) based bearer networks such as short message service (SMS) and unstructured supplementary services data (USSD). The IP circuit-switched access is achieved using IP via a Remote Access Server (RAS), which can be either integral or external to the WAP gateway. Standard LAN, ATM, and X.25 connections can be used to connect the RAS servers and WAP gateways.

The TCP/IP/HTTP stack includes the well-established TCP/IP, Secure Socket Layer (SSL) or transport layer security (TLS), and HTTP protocols.

The WAP stack is the WAP Forum-specified layered protocol implementation. Even though the WTLS is optional in the WAP Forum specification, the Everyplace Wireless Gateway uses it to provide server authentication and data encryption.

Note

WTLS support is required in WAP Version 1.1 or later.

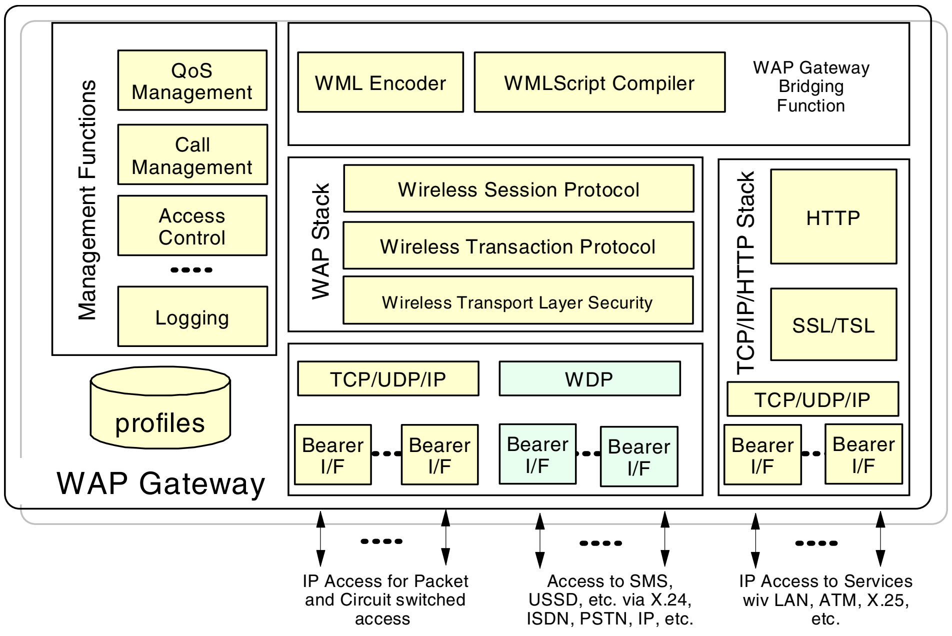

Management Functions
QoS Management
Call Management
Access Control
Logging
profiles
WAP Gateway
WML Encoder
WMLScript Compiler
WAP Gateway Bridging Function
WAP Stack
Wireless Session Protocol
Wireless Transaction Protocol
Wireless Transport Layer Security
TCP/IP/HTTP Stack
HTTP
SSL/TSL
TCP/UDP/IP
WDP
TCP/UDP/IP
Bearer I/F
Bearer I/F
Bearer I/F
Bearer I/F
Bearer I/F
Bearer I/F
IP Access for Packet and Circuit switched access
Access to SMS, USSD, etc. via X.24, ISDN, PSTN, IP, etc.
IP Access to Services wiv LAN, ATM, X.25, etc.

Figure 57. Basic WAP gateway functions

WAP gateway bridging functions include WML encoder, WMLscript compiler, and HTTP encoder. The WAP gateway bridging functions also include the ability to tokenize and cache the content so that users can re-use the content within a certain period of time defined by some valid lifetime criteria. This is a performance enhancement over the simple case where data is not cached and all WAP data being sourced is scrapped once it is delivered.

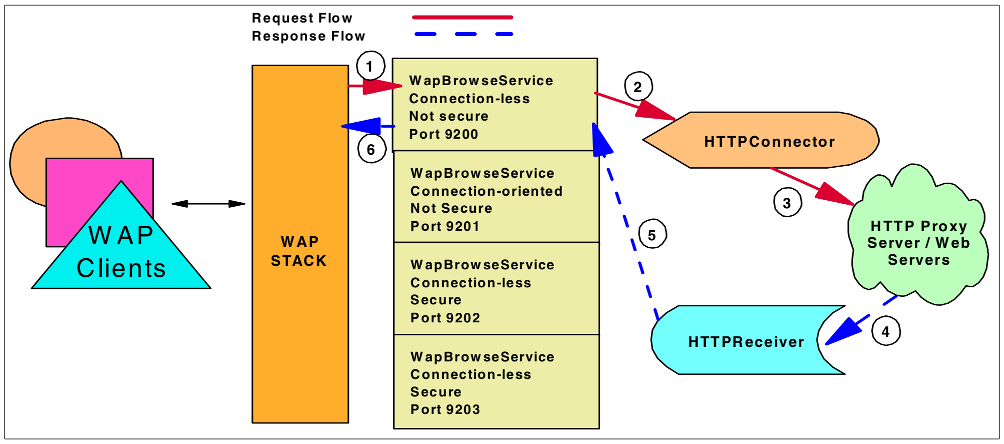
WAP gateway management functions comprise several managerial entities. The WAP gateway provides administration such as setting up accounts and system configuration. It also includes call management, which is the ability to recover the call seamlessly from a broken link. Call management is achieved through the use of link break detection, deadtime (no traffic), and timer-driven expiration of addresses. Finally, WAP gateway management functions also include the quality of service (QoS) management and logging tools.

7.6.4 WAP data flow

The WAP gateway converts the WSP session from the WAP client to HTTP for the Internet or intranet Web servers via the proxy. The gateway also performs the binary encoding and decoding of the content, and converts the HTTP response to WSP for delivery to the WAP clients.

The data flow for a typical WAP communication via the Everyplace Wireless Gateway is depicted in Figure 58. The process is as follows:

- Data is received from the WAP protocol stack.
- Request is validated and converted from binary WAP language to standard Web protocols. Additional HTTP request headers are added (such as cookies, WES integration headers, etc.)
- Request is forwarded to caching proxy which is a Caching Proxy with a Wireless Gateway plug-in.
- Response is received, cookies are processed, WML documents are converted to binary WML.
- HTTP headers are binary encoded.
- Response data is sent back to the WAP client.



Request Flow
Response Flow
WAP Clients
WAP STACK
1
6
WapBrowseService Connection-less Not secure Port 9200
WapBrowseService Connection-oriented Not Secure Port 9201
WapBrowseService Connection-less Secure Port 9202
WapBrowseService Connection-less Secure Port 9203
2
HTTPConnector
3
HTTP Proxy Server / Web Servers
4
HTTPReceiver
5

Figure 58. The data flow for the WAP requests.

7.6.5 WAP gateway advanced functions

In addition to the basic WAP functions, the Everyplace Wireless Gateway also supports advanced functions such as WAP push, WML caching, and short message service (SMS).

7.6.5.1 WAP push

The Everyplace Wireless Gateway supports the WAP 1.2 specified Push Access Protocol to allow external applications the ability to push various content down to WAP devices. With WAP push support, the service providers are able to transmit information to a device without a prior user-initiated action. Like the World Wide Web, the conventional WAP service is a user "*pull*" technology, meaning the client always initiates the transaction and pulls the information from the server. In contrast to the pull technology, WAP push has no explicit request from the client before the server transmits its content. In other words, "push" transactions are server-initiated. WAP push is very useful for providing better personalization and timely services.

The Push Access Protocol is used by a push initiator residing on an Internet server to access the Everyplace Wireless Gateway as a push proxy gateway. The push initiator originates the push content and submits it to the push proxy gateway for delivery to a user agent on a client. Figure 59 shows a WAP push configuration. The push initiator is able to submit a push, cancel a push, query for the status of a push, and query for wireless device capabilities. The push proxy gateway is able to notify the push initiator about the result of the push. For more detailed information about the Push Access Protocol, please refer to WAP Forum specifications.

Currently the Push Access Protocol is transported using HTTP. In the future, more transport such as SMTP will be supported as well. For the current HTTP-based push access, the HTTP POST method and response are used. Using HTTP, operation begins with an HTTP POST method containing the information for delivery to the push proxy or the push initiator. Upon receipt of the POST method, the receiving server replies with an HTTP response containing the response for the operation. For added security, the HTTP may be used with SSL.

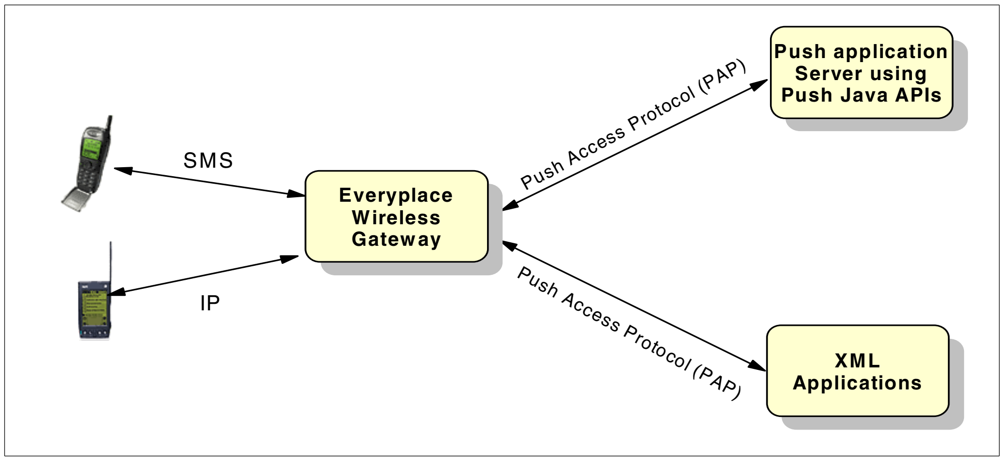

Push application Server using Push Java APIs
Push Access Protocol (PAP)
SMS
Everyplace Wireless Gateway
IP
Push Access Protocol (PAP)
XML Applications

Figure 59. Everyplace Wireless Gateway supports the Push Access Protocol

7.6.5.2 Support for cookies

The Everyplace Wireless Gateway also supports user cookies. Existing WAP phones do not store HTTP cookies; the Gateway will store these cookies on behalf of the WAP client and deliver them to Web servers when they are requested.

Cookies are stored in a shared database so that information persists across gateways and gateway start/stop.

7.6.5.3 Binary WML caching

Web servers can be accessed by the IBM Everyplace WAP gateway via the HTTP proxy. The Edge Server - Caching Proxy is shipped with the WebSphere Everyplace Suite as the HTTP caching proxy. The WAP gateway sends HTTP 1.1 style requests over TCP/IP and expects HTTP-encoded WML or WMLScript in return. Logically, a WAP gateway opens a TCP/IP socket to the WTE proxy for the client request and the WTE in turn opens a socket to the Web server or upstream proxy server.

It is desirable that the binary WML data can be cached for subsequent user requests. However, WTE is a HTTP proxy that does not support the WML caching by its own. In order to enable the WTE caching of the WML data and improve performance, the IBM WAP Gateway installs a special plug-in module on WTE to enable its caching of WAP data (binary WML).

SMS Support
The Everyplace Wireless Gateway now also supports WAP and non-WAP devices over short message service (SMS) networks. SMS is very popular in Europe and other countries outside the United States. Due to its popularity, even though SMS has not become an industry standard yet, the Everyplace Wireless Gateway has added additional SMS bearer network support.

7.7 Wireless gateway security

The Everyplace Wireless Gateway is one of the two points of entry for the IBM WebSphere Everyplace Suite. Safe and secure connections between the Wireless Gateway clients, the gateway(s), and the WES domain is of utmost importance. Wireless Gateway has implemented thorough security measures to achieve the goal of providing authentication and confidentiality for the WES domain, as displayed in Figure 60 on page 111.

The Wireless Gateway uses a modified Point-to-Point Protocol (PPP) called Wireless Optimized Link Protocol (WLP) to authenticate the connection between itself and Wireless Clients over either wireline or wireless networks. WLP uses a two-party key distribution protocol in which the Wireless Gateway and the Wireless Clients authenticate each other without sending a password over the air. Using WLP, the Wireless Gateway can provides an encrypted tunnel for the wireless clients to access the WES domain in a secure fashion. Authentication, confidentiality, and access control can be implemented. However, the non-repudiation can only be achieved by the client/server SSL connection since the Wireless Gateway cannot examine the data flow.

In addition and in order to integrate with TPSM, you can configure a Wireless Gateway to use Remote Authentication Dial-In User Service (RADIUS) third-party authentication. Acting as a proxy on behalf of Wireless Clients connected to it, the Wireless Gateway routes authentication requests to a RADIUS server. Just as another link in a chain makes the chain longer, request and response times to authenticate users are longer when you configure the Wireless Gateway to use a RADIUS server.

For scalability considerations, and the considerations for the provision of virtual private network (VPN) as well as the considerations of interfacing to existing corporate or subscriber access control, the HTTP authentication for WAP clients should be interfaced to external authentication processes such as RADIUS, as well as having a self-contained method for stand-alone use.

The connection between WAP devices and the Wireless Gateway(s) is secured using the Wireless Transport Layer Protocol (WTLS). WTLS supports mini-certificates for the public key exchange. WTLS support is required in the WAP Version 1.1 (or later) Forum specification and the Everyplace Wireless

Gateway uses it to provide server authentication and data encryption. It supports 1024-, 768-, and 512-bit RSA public key exchange, as well as RC5 symmetric key data encryption (40 or 56 bit) and the MAC algorithm SHA-1.

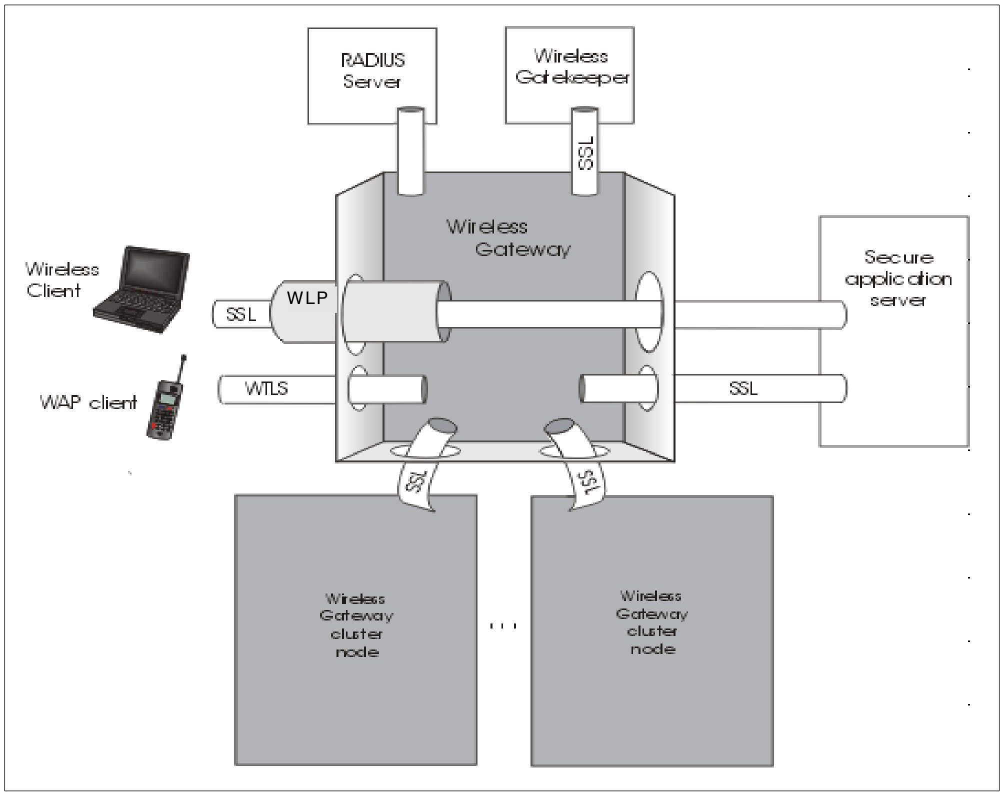


Figure 60. Security implementation in IBM Everyplace Wireless Gateway

WTLS offers authentication using certificates. It also provides confidentiality using data encryption. However, it cannot implement access control since it is based on stateless WDP and will be terminated at the Wireless Gateway.

Secure Sockets Layer (SSL) can be used to establish a secure connection between the Gateway and back-end Web servers, between the Gateway and the Gatekeeper, and between multiple Gateways within a cluster.

7.8 Deployment of the Everyplace Wireless Gateway

The IBM WebSphere Everyplace Suite is targeted for network operators, content providers, service providers such as ISP and ASP, and enterprises. Different customers may deploy the Everyplace Wireless Gateway differently based on their functionality needs. However, a general set of considerations exist for the deployment of the Everyplace Wireless Gateway. First we will summarize the considerations for performance, scalability, availability, and security. Such considerations and customers' individual functionality needs will dictate the exact configuration of the Everyplace Wireless Gateway. Secondly, we will discuss the WAP solution deployment models for network operators and content providers. The enterprise usage of the Wireless Gateway has been covered in detail in *Mobile Computing: The eNetwork Wireless Solution,* SG24-5299. Therefore we will only briefly discuss the deployment of the Everyplace Wireless Gateway for enterprises when we discuss the deployment for service providers.

7.8.1 Performance, scalability, availability, and security

As evident in the flexible design of the Everyplace Wireless Gateway architecture (see 7.3.1, “Architecture of the Everyplace Wireless Gateway” on page 96), the Wireless Gateway is built with the considerations of performance, scalability, availability, and security in mind. The overall system functions can be flexibly distributed. The design exploits standard architectures and many existing server products, for flexible and scalable configurations.

For example, in order to achieve scalability and the provision of virtual private networks (VPN) and interfacing to existing corporate or subscriber access control, RADIUS server is used by the gateway to authentication the users to the WebSphere Everyplace Suite. The RADIUS server is shipped with TPSM of the WebSphere Everyplace Suite.

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 that the distributed form be used separating security services, content adaptation, management services, base services, and connection services.

IBM recommends distributed configuration

In a distributed environment, customers can put most WebSphere Everyplace Suite components on their own physical machines to provide scalability, higher reliability and performance. Single function machine can be tuned more easily with fewer interruptions due to cross-product problems.

In order to achieve high performance in providing WAP and wireless services to clients, it is recommended that you use the Everyplace Wireless Gateway in conjunction with the Edge Server - Caching Proxy. The WTE HTTP proxy can enable both Web and WAP content caching for faster services.

Better performance and high availability can also be achieved by using the Everyplace Wireless Gateway cluster support. Having multiple Wireless Gateways in one cluster can improve the service response time and minimize the service downtime.

Similarly the Edge Server - Load Balancer can be used to support multiple HTTP proxy servers behind Wireless Gateways. However, 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 Network Dispatcher 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 network dispatcher’s load balancing feature is no longer applicable. In addition, WAP connection-oriented sessions and WTLS secure transactions may not be efficiently dispatched by the Network Dispatcher. 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 Network Dispatcher. The WAP session suspend and resume function is not reliably supported because the client will likely resume with a different IP address.

Note

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.

For scalability considerations, and the considerations for the provision of virtual private network (VPN) as well as the considerations of interfacing with existing corporate or subscriber access control, it is recommended that the Everyplace Wireless Gateway be configured to use external authentication processes such as RADIUS even though it has a self-contained authentication method for stand-alone use.

For security considerations, the connection between the Wireless Gateway and the Wireless Gatekeeper must be SSL-enabled. Further more, the connections among multiple Gateways in a cluster environment should be SSL-enabled as well.

7.8.2 WAP solution deployment

As explained in 7.6.2, "IBM WAP implementation" on page 105, IBM implements WAP support via the pattern of "client - gateway - transcoding and caching proxy - Web server". This pattern is made scalable with high performance.

The parties

In the most general sense, the parties involved in any WAP solution deployment includes the following:

- *User* with WAP client phone/device
- *Network Operator* providing wireless network connections
- *Content Provider* providing information and applications. This can be the case for enterprises providing internal information and applications to their mobile workers.
- "*Third Party*" acting in conjunction with or on behalf of the network operator and/or content provider. The "third party" can be a service provider such as an Internet Service Provider (ISP) or Application Service Provider (ASP).

Deployment models for network operator

Case #1:Wireless network operator becomes value-added provider of WAP service and content.

In this case, the wireless network operator host the total solutions based on the WebSphere Everyplace Suite. The Everyplace Wireless Gateway, transcoding/proxy servers, as well as back-end Web servers are hosted by the wireless network operators.

Case #2: Network operator hosts WAP service with external content providers.

In this case, the network operator provides WAP gateway and transcoding/proxy hosting itself while it offers its subscribers WAP services and content that are provided by third-party content providers. The network operator acts as aggregater and content transcoder.

Case #3: Third-party host WAP gateway and transcoding/proxy for the network operator, which also provides content and applications.

In this case, the wireless network operator offers its subscribers WAP services and content hosted by a third party. However, the network operator still hosts application services and content itself.

Deployment model for content provider as well as enterprises

Case #4: Network operator hosts WAP gateway only with external content intermediary and applications.

In this case, the network operator only provides the Everyplace Wireless Gateway as the WAP gateway for the protocol transformation and connectivity. The content, application, transcoding, and even WAP proxy is provided by external providers. The WAP proxy and content are not necessarily provided by the same party.

This case also applies to those enterprises that use the network providers for both network connectivity and the WAP gateway while providing their own content and content adaptation.

Case #5: Content provider hosts full WAP solutions.

The content provider can host a full line of WAP support from the WAP gateway, and WAP proxy and transcoding, to content and applications. The content provider uses the network provider only for wireless connectivity services.

This can also be the enterprise deployment model for users of WAP device to access its intranet.

For example, the content provider or enterprise can use a network provider's connection services over the IP network. In the dial-up example, the IP circuit-switched access is achieved using IP via the Remote Access Server (RAS) which can be either integral or external to the WAP gateway. Standard LAN, ATM, and X.25 connections can be used to connect the RAS servers from the network provider to WAP gateways.

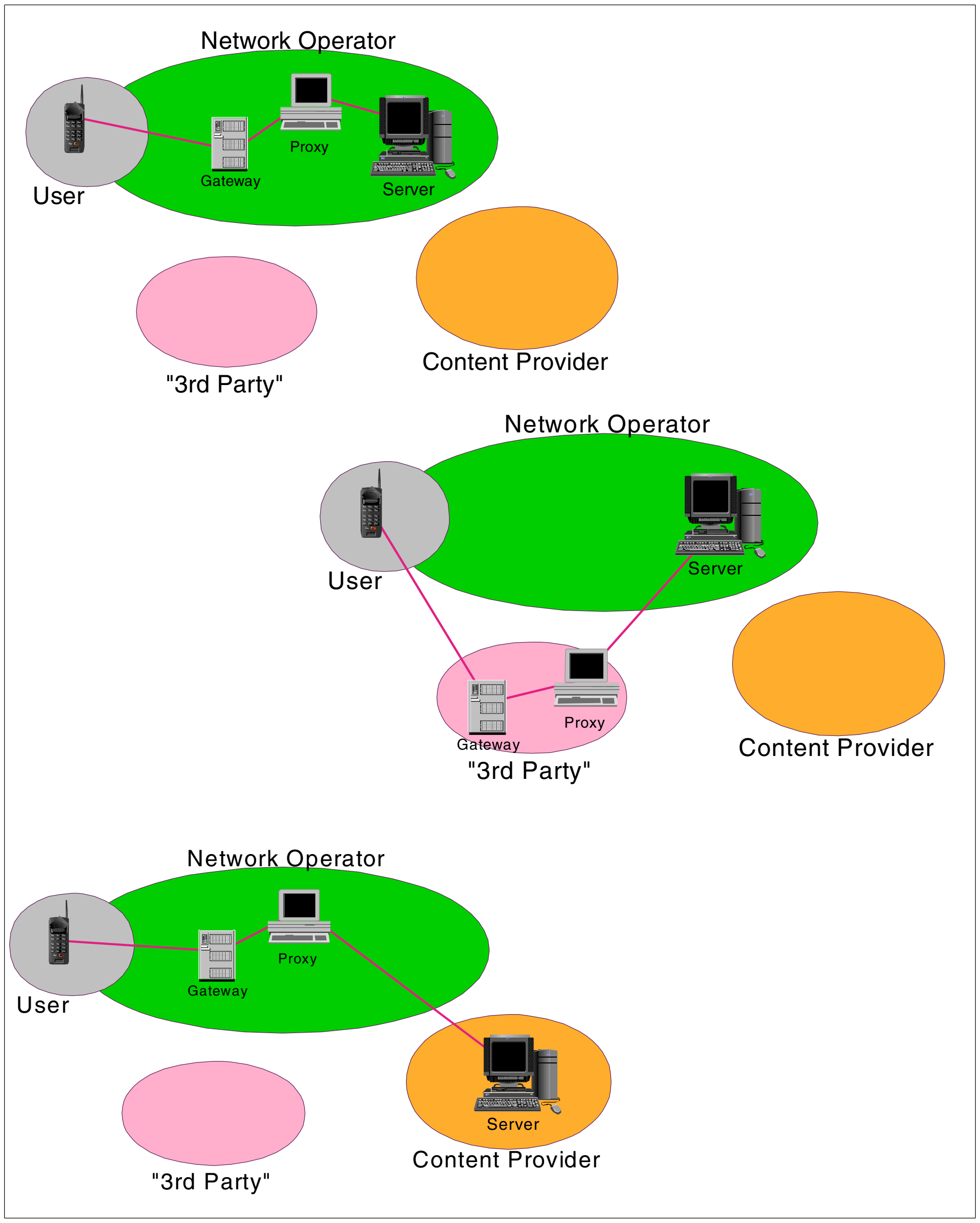


Figure 61. Typical scenarios of WAP deployment for network operators

Deployment model for service providers

Case #6: Service providers (third party) host WAP gateway/proxy with access to external content providers.

In this case, the service provider, either an Internet Service Provider (ISP) or Application Service Provider (ASP), hosts the WAP gateway, transcoding, and proxy servers while the contents are from external providers. The network connectivity is provided by network operators.

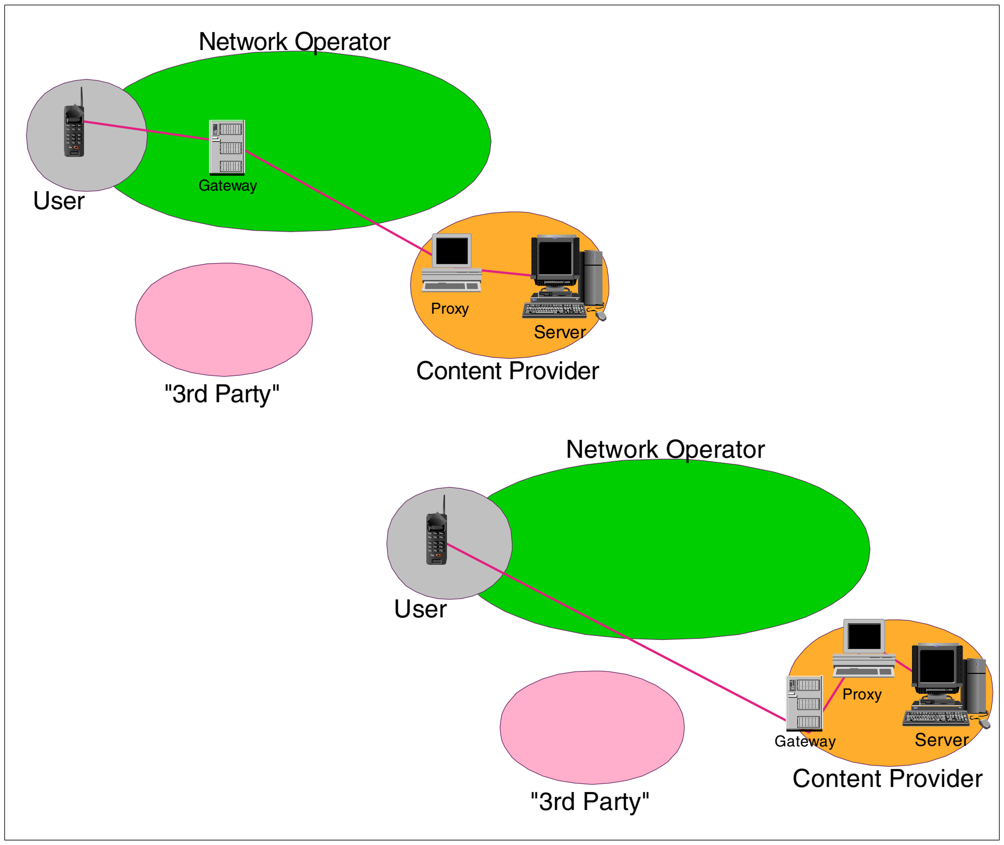


Figure 62. WAP deployment for content providers and enterprises

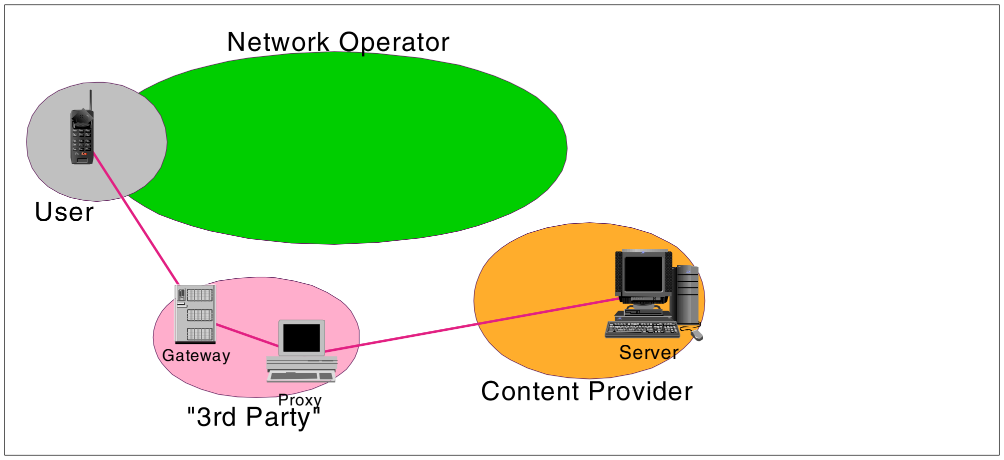


Figure 63. WAP deployment for typical service providers

7.8.3 Non-WAP solution deployment

Even though WAP support is the focal point of the WebSphere Everyplace Suite today, the Everyplace Wireless Gateway can support non-WAP clients for an array of enterprise applications and host services. The deployment of the Everyplace Wireless Gateway is in conjunction with the host intermediary such as IBM Emulator Express and IBM Web Express. Such deployment extends TCP/IP applications as well as legacy applications,such as 5250 (AS/400) and 3270 (CICS, IMS), to mobile and/or wireless users via mobile networks.

The scenarios similar to the six cases presented in 7.8.2, “WAP solution deployment” on page 113 can be derived for the non-WAP deployment of the Everyplace Wireless Gateway. The scenarios will again be drawn across network operators, content provider/enterprises, and service providers. We will not discuss them in detail here.

For guidelines on the deployment of the Everyplace Wireless Gateway for non-WAP wireless services, please refer to *Mobile Computing: The eNetwork Wireless Solution,* SG24-5299.

Chapter 8. Transcoding Web application content

This chapter provides a brief overview of the transcoding concept focusing on architecture and system design with *IBM WebSphere Transcoding Publisher*.

Information on the product and examples are limited to the context of this book. For further fundamentals and in-depth details on technologies and IBM WebSphere Transcoding Publisher see:

- The online product documentation
- *Extending Web Applications to the Pervasive World with IBM WebSphere Transcoding Publisher*, SG24-5965
- `http://www.ibm.com/software/webservers/transcoding/library.html`

8.1 Introduction to transcoding

Transcoding is the concept of transforming content. It's a general concept not bound to any specific markup languages, formats, or products. In today's pervasive world, transcoding is found useful - if not invaluable - in automated adaptation and conversion to:

- Adapt client markup languages (such as WML, HTML and all their variants).
- Enable system-to-system data exchange (such as XML feeds, B2B transactions).
- Preserve bandwidth where limited (such as dial-up and wireless connections).

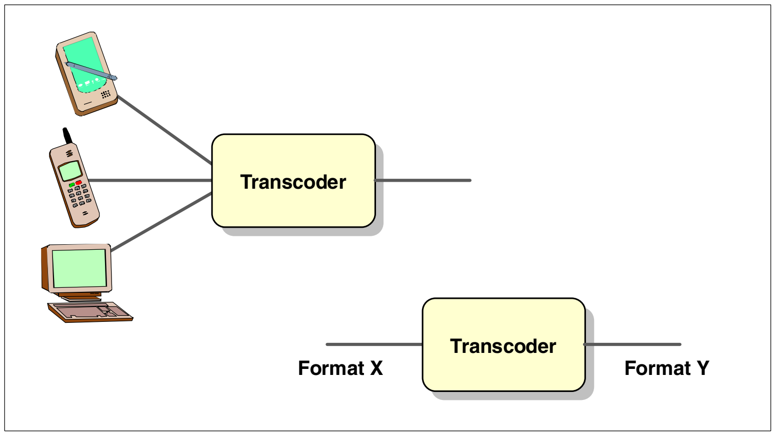


Figure 64. Use of transcoding in general

Some formats transcoded will be binary, such as images, while probably most transcoding will be applied to text streams. Parts will be different XML variants, parts will be non-XML formats for markup and data.

Transcoding is powerful and relieves the workload of designing a page version for each client type or writing interface code for heterogeneous systems. Magic can happen - for example, a very hot topic at the time of writing is automated conversion of HTML to WML. However, issues exist and some considerations should be done.

8.2 Overview of WebSphere Transcoding Publisher

The IBM transcoder implementation is named *IBM WebSphere Transcoding Publisher* (WTP). Currently the product is available for AIX, Linux, Solaris, Windows 2000, and Windows NT.

IBM WebSphere Transcoding Publisher can be used in three models:

- As a HTTP proxy in a network
- As a MIME type filter plug-in to the IBM WebSphere Application Server
- As JavaBeans enriching the functions of your own code

Note: WebSphere Transcoding Publisher is intended to be deployed as a proxy in the Everyplace Suite domain. It is not intended to be used as a servlet (WebSphere Application Server filters) or as JavaBeans within the Everyplace Suite domain.

The overall architecture of WebSphere Transcoding Publisher is shown below.

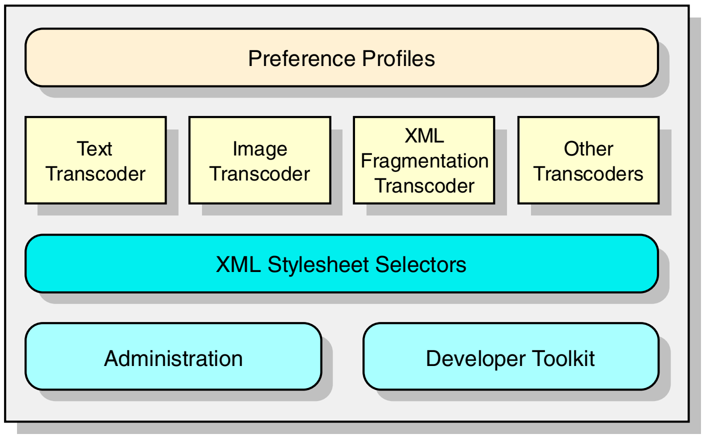


Figure 65. IBM WebSphere Transcoding Publisher architecture

A number of *preference profiles* are responsible for recognition of device, network and content types and to apply the right set of transcoders. More on this topic may be found in 8.5, “Preference profiles” on page 126.

WTP is hosting a number of *transcoder plug-in* units that can modify any type of content in practically any possible way. The default set of transcoders is extendible by custom-written transcoders implemented in Java. See 8.6, “Transcoders” on page 128.

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.

Included is an administration console and a toolkit for development and problem determination. Both will be described only as needed in this book. For more information refer to Appendix C, “Related publications” on page 227.

8.2.1 In the IBM WebSphere Everyplace Suite

This section describes WebSphere Transcoding Publisher used in the context of the Everyplace Suite:

- New features specific to the Everyplace Suite and general
- Proxy model only is supported
- Network and device recognition done by the IBM Everyplace Authentication Server

8.2.1.1 New features

The IBM WebSphere Everyplace Suite V1.1 includes IBM WebSphere Transcoding Publisher Version 1.1.3. Most significant changes since Version 1.1.1 are:

- **Infrastructure improvements**
  - Uses LDAP directory for settings (previously stored in *.properties files). The administration console is used exactly the same way.
  - Settings are shared between all instances of WTP using common LDAP schema.
  - Everyplace Suite Active Session support - to bypass device/network recognition if the IBM Everyplace Authentication Server already has done that.
  - *NO-OP* HTTP header support - the first transcoder will signal “job already done” to following ones, which then bypass transcoding.
- **Transcoding improvements**
  - Text transcoder includes HTML to I-Mode conversion.
  - Fragmentation engine now supports I-Mode in addition to WML.
  - Image transcoding includes GIF to WBMP and JPG to WBMP conversion.
  - New device profiles for WAP and I-Mode phones.

8.2.1.2 Proxy model

Using WebSphere Transcoding Publisher in the context of the IBM WebSphere Everyplace Suite supports the HTTP proxy model only.

Using the proxy model allows the freedom to use any Web application server to serve pages and other content types. From an architectural point of view the proxy model allows installation on separate machines for greater flexibility and scalability. However, performance and network load will be affected by adding an extra hop, and it will not be possible to transcode encrypted content.

8.2.1.3 Network and device recognition

In the Everyplace Suite the IBM Everyplace Authentication Server will perform network and device detection and store the result in a HTTP header for use by WTP and other subsequent servers.

Network differentiation is normally done by setting up multiple HTTP ports for distinguishing network types. In the Everyplace Suite, the IBM Everyplace Authentication Server will perform network detection and put the result in an HTTP header for WTP and other subsequent servers. Three 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 IBM Everyplace Authentication Server using the shared WTP configuration stored in LDAP. That means an administrator will not notice the difference.

8.3 Building with transcoders

A transcoder is a powerful building block. Used properly it can expand the reach of your services and applications with minimal effort. This section provides an overview of how the use of transcoding can (or should) affect both infrastructure and application design considerations.

8.3.1 Infrastructure design

It is most likely that you want to install more than one instance of the product to let a heavy workload be spread across several physical units.

Transcoding can also be utilized both at the “front” and at the “back” of your application server as illustrated in Figure 66 below.

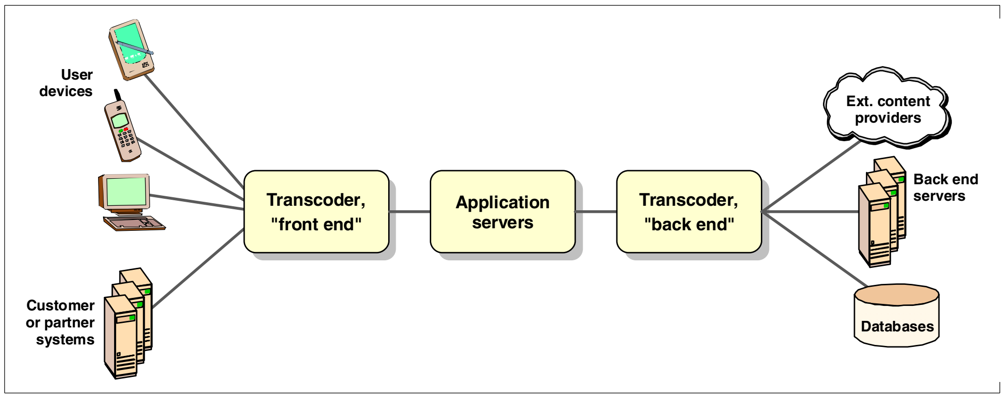


Figure 66. Transcoding in the front and back of an application server

In the 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 Suite.

The back-end transcoder helps adapting content from different sources to a format easier 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 Java beans (not supported by the Everyplace Suite), 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.

8.3.1.1 Optimization with a Caching Proxy

Every single transcoding done will take some time and consume some hardware resources, so in general it can be very beneficial to keep transcoded versions of content for reuse.

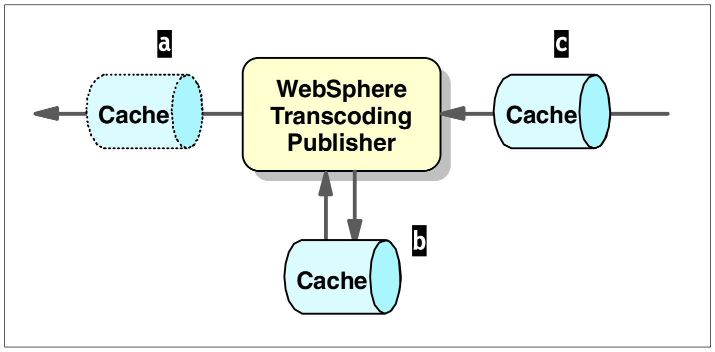


Figure 67. Caching with WebSphere Transcoding Publisher

WebSphere Transcoding Publisher can be configured to use an external cache directly (b in Figure 67 above). This allows caching of different transcoded representations of the same content element. Be careful when setting up a caching proxy along the HTTP stream in front of WTP 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.

Setting up caching for content served to clients can improve client response time and reduce the workload of both transcoder and application server. This is even more significant for client types with no cache.

Caching applied to a back-end transcoder either controlled by WTP (marked b in Figure 67) 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:

Note

Efficient and intelligent caching of external content can be set up with minimal effort using the caching server provided in the IBM WebSphere Everyplace Suite.

Cache control

A cache serving transcoded content and controlled by WTP will obey the same HTTP directives as usual, such as *Expires*, *Last-Modified* and *No-Cache*. That also means also transcoded content should be handled properly, considering both timeliness and optimal reuse.

Cache considerations

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.

General and product-specific documentation describing these considerations are also very useful and valid in the context of WTP. However, the quality measure is slightly different: Content adaptation with transcoding could for example result in four different versions of every page that should be cached individually. In advance, you should do calculations on cache efficiency by dividing the total number of hits by four groups and consider the number of pages multiplied by four. So the benefit of caching can be limited to the number of users accessing the same pages *from the same device*.

8.3.2 Application design

Setting up transcoders in every possible network path will not do it. Delivering high-quality and well-performing services with a reasonable effort requires careful considerations on application design.

Considerations of automated conversion of HTML (or similar client markup language) are divided into two major groups:

1. Supported features
2. Usability

Supported features

HTML served to devices with limited capabilities can be made usable by WebSphere Transcoding Publisher. Examples are reducing or removing colors, converting tables to plain text, removing frames, performing image translations, or replacing images with links, and more.

But how should a clickable image map be converted and used on a WAP phone? How would a transcoder handle applications based on Java Script, ActiveX objects, Java applets, etc., if the target device does not support this code?

Most advanced features are not suitable for transcoding. In general an application suitable for transcoding is built with more simple techniques. Sure, bells and whistles can be added for traditional browsers, but keep the core code running on the server with a thin HTML-only client.

Sometimes you don’t want to write transcoder-friendly applications; you want pixel-level control on every device. Or your want to utilize features supported on some specific devices only. The solution involves hard work building separate sets of ,for example, JSPs tailored for different device types.

A compromise can be to build only a few selected pages optimized for specific devices. This could be the home page and pages using “incompatible” stuff like Java applets, phone API, Palm software, and so on.

Usability

Designing applications for multiple devices requires considerations about input and output capabilities. Today’s common PC Web browser is at the high end with 1024x768 true-color screen, a mouse and some 102-keys available. The most minimalistic device used today is the first generation of WAP phones with two to four lines of monochrome text, 10 numeric keys and a few function buttons. (Who said a 3270 terminal was inadequate?) In between you will find Web TV, Palm Pilot, WorkPad, WinCE, and lots of other devices with various input and output capabilities.

Navigation and application flow clearly needs reconsideration. Some people have counted an average of 52 links for a public Web page. This is not recommended for applications to be used from a phone!

As an example take a page containing clickable news headlines. The PC Web browser could show 20-30 properly arranged headlines and still be usable. On the Palm Pilot, probably 10 headlines will do. From a phone, users may prefer five or so. Basically the number of selectable items can be reduced in two ways:

- *Categorization* - introducing one or more category selections in a flow
- *Personalization* - as a filter for passing the subset of relevant items only

The cost and benefit of each approach depend heavily on the application type, content, and users. No final answer can be provided.

Input forms introduce even more considerations in multi-device design. How many fields per page are acceptable? Does the device support more intelligent forms by allowing client-side scripting? Is text easily entered from the device, or should you rely more on selections? Using existing knowledge of the user can be most valuable, like letting the application suggest the customer's known home address for the goods delivery address.

Trade-off
Developing multiple versions of an application could be for example *two* versions; one for "large-screen" users and one for "small-screen" users. Usability can be improved significantly by implementing two different navigation schemes optimized for the two device groups. Device-specific adaptation is done generically by a transcoder. This is illustrated in Figure 68.

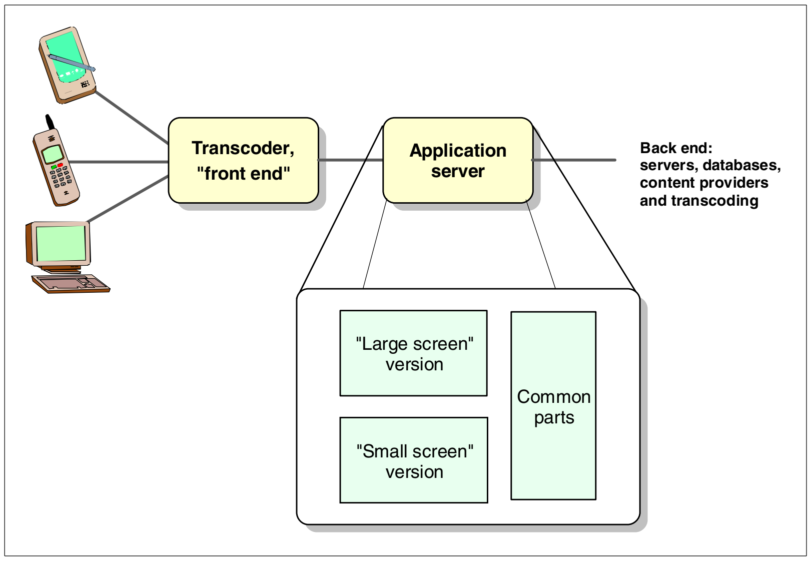


Figure 68. Application design for a compromise

Some devices with a "medium" user interface, such as the WorkPad or Palm Pilot could potentially use both versions. The choice can be left up to the user or determined by the application.

As an example of categorization, the application could present a complete menu to a “large-screen” user, while more levels of selection are introduced to the “small-screen” user, allowing each page to be simple. Techniques such as personalization could be applied to both versions. The general user interface of both versions should, of course, be carefully considered and designed; but by having two specialized versions fewer compromises have to be taken.

Technically the application could be implemented as two sets of JSPs, of which the large-screen version will generate standard HTML and the small-screen version will generate XML. Why not use XML for both versions? Or HMTL and WML? The decision has to be made as a trade-off keeping in mind the devices to be supported, format of the source data, the skills of the developers, capabilities of the application server used, solution flexibility, and last, but not least, performance considerations.

8.4 Installation and configuration

When the environment infrastructure and network topology is planned and sized, it’s time to install and configure products. WebSphere Transcoding Publisher is easily installed and configured by the Everyplace Suite installation program and needs no further configuration to run. However, you might want to adjust the configuration parameters or maybe to build and configure new transcoders as described in 8.6, “Transcoders” on page 128.

8.4.1 Installing WebSphere Transcoding Publisher

The IBM WebSphere Everyplace Suite installation program will silently install WebSphere Transcoding Publisher and configure it as follows:

- HTTP proxy model
- Caching proxy support not enabled
- Default transcoders for devices and networks

WebSphere Transcoding Publisher is now ready to work!

8.4.2 Configuring a Caching Proxy

IBM WebSphere Transcoding Publisher can be configured to use an external cache directly. In the IBM WebSphere Everyplace Suite the caching function should be delivered by the Edge Server - Caching Proxy. WTP is also tested with Squid, although any other standard HTTP caching proxy should work.

The Suite installation program will not configure the use of cache by default. The cache may need further configuration to be tuned to fit your unique deployment and use. Refer to the product documentation and *Extending Web Applications to the Pervasive World with IBM WebSphere Transcoding Publisher*, SG24-5965.

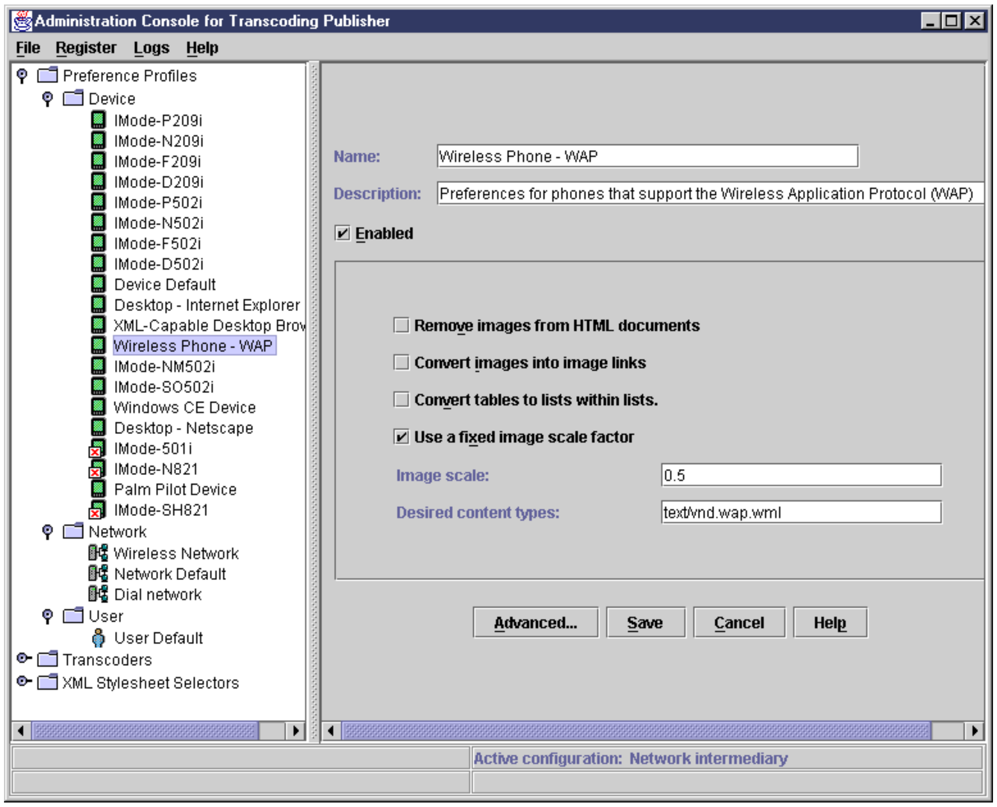
8.5 Preference profiles

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 LDAP. 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 - individual user setting could be set up using the portal toolkit pTk in TISM).

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

The existing preference profiles can be inspected and modified using the administration console shown in Figure 69.



Name: Wireless Phone - WAP

Description: Preferences for phones that support the Wireless Application Protocol (WAP)

☑ Enabled

☐ Remove images from HTML documents

☐ Convert images into image links

☐ Convert tables to lists within lists.

☑ Use a fixed image scale factor

Image scale: 0.5

Desired content types: text/vnd.wap.wml

Active configuration: Network intermediary

Figure 69. Preference profiles in WebSphere Transcoding Publisher

8.5.0.1 Modifying existing profiles

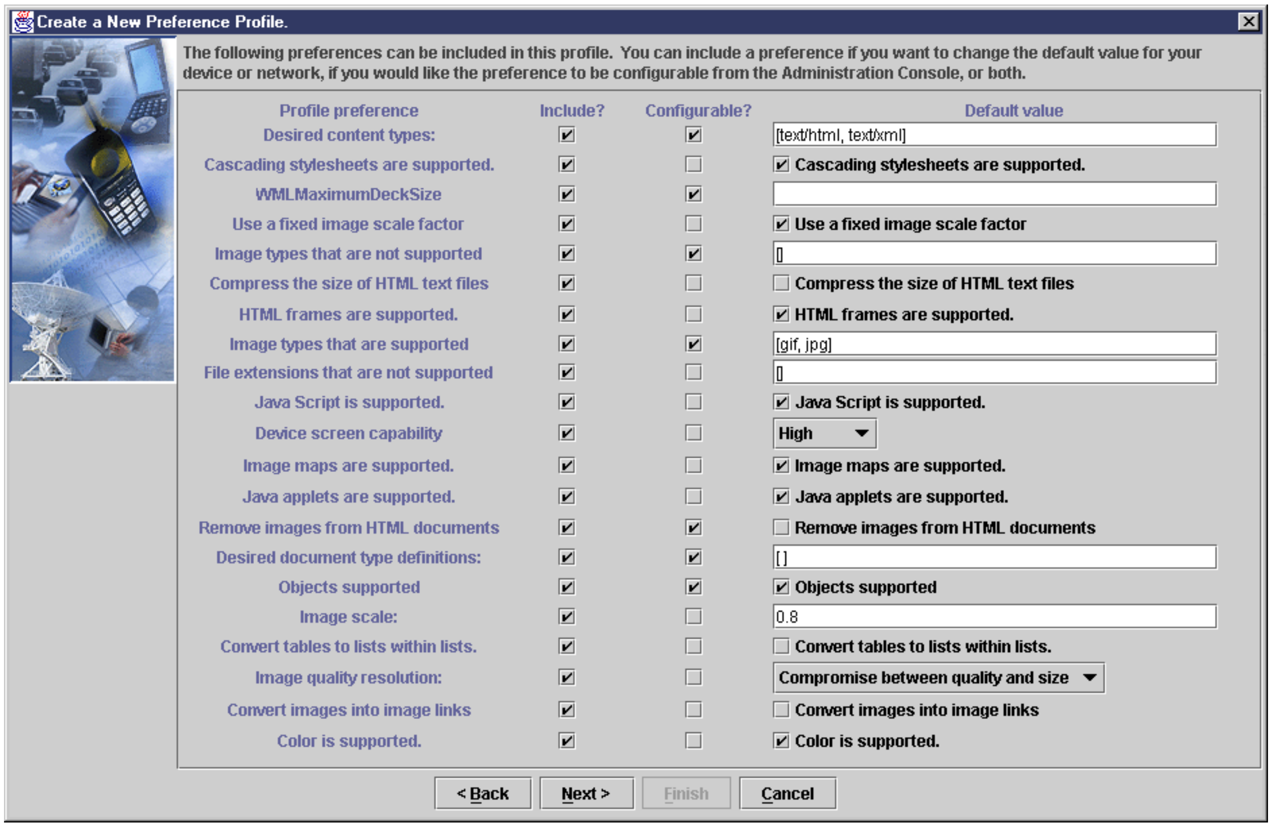
For most profiles a number of parameters can be configured to enable or adjust the transcoders it will invoke. The selection of configurable parameters is set up when creating a new preference profile and cannot be changed later. Refer to the WTP online *Developer's Guide* for information on how to re-create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

8.5.0.2 Creating new profiles

To create a new preference profile start the toolkit wizard Create Profile from the WebSphere Transcoding Publisher program folder or Start menu. You will be guided to supply:

1. A `*.prop`-filename for saving the profile.

2. Name and comment for the profile.
3. HTTP User-Agent string selection criteria (for device profiles only).
4. Preference selection and values for the transcoders to be invoked. Network profiles offer a subset of the preferences available to device profiles.



The following preferences can be included in this profile. You can include a preference if you want to change the default value for your device or network, if you would like the preference to be configurable from the Administration Console, or both.

Profile preference	Include?	Configurable?	Default value
Desired content types:	✓	✓	[text/html, text/xml]
Cascading stylesheets are supported.	✓	☐	✓ Cascading stylesheets are supported.
WMLMaximumDeckSize	✓	✓	
Use a fixed image scale factor	✓	☐	✓ Use a fixed image scale factor
Image types that are not supported	✓	✓	[]
Compress the size of HTML text files	✓	☐	☐ Compress the size of HTML text files
HTML frames are supported.	✓	☐	✓ HTML frames are supported.
Image types that are supported	✓	✓	[gif, jpg]
File extensions that are not supported	✓	☐	[]
Java Script is supported.	✓	☐	✓ Java Script is supported.
Device screen capability	✓	☐	High
Image maps are supported.	✓	☐	✓ Image maps are supported.
Java applets are supported.	✓	☐	✓ Java applets are supported.
Remove images from HTML documents	✓	✓	☐ Remove images from HTML documents
Desired document type definitions:	✓	✓	[ ]
Objects supported	✓	✓	✓ Objects supported
Image scale:	✓	☐	0.8
Convert tables to lists within lists.	✓	☐	☐ Convert tables to lists within lists.
Image quality resolution:	✓	☐	Compromise between quality and size
Convert images into image links	✓	☐	☐ Convert images into image links
Color is supported.	✓	☐	✓ Color is supported.

Figure 70. Preferences for a new device profile

The new profile now resides in a file system. It needs to be registered for WTP to know about it:

1. From the WTP Administration Console select **Register -> Preference Profile**.
2. Point out the `*.prop`-file.
3. Select to use it as a **Device** or **Network** profile.
4. Fill in values for parameters as required.

8.6 Transcoders

To get an impression of the capabilities of WTP and to understand how it can be extended, we first take a look at the two basic types of transcoders it uses: MEGs and XSL transforms.

• MEGs

A *MEG* is a content *M*onitor, *E*ditor or *G*enerator. 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 WTP implementation it takes the shape of a Java Servlet, and is then called a *MEGlet*.

See `http://www.almaden.ibm.com/cs/wbi` for more information on the MEG concept in general.

- 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 field and structural level (in contrast to 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.

This section briefly describes the built-in transcoding capabilities of WebSphere Transcoding Publisher followed by general guidelines and a few examples of extending the functionality.

8.6.1 Build-in transcoders

The set of transcoders supplied with WebSphere Transcoding Publisher consists of three powerful transcoders:

Image transcoders	Can convert between different image formats, reduce color depth and size and more.
Text transcoders	Can simplify HTML documents (for example replacing images with links and removing Java Script), apply XSL stylesheets to XML documents, and perform generic HTML-to-WML and HTML-to-I-Mode conversion.
XML fragmentation	Can split up WML decks into valid chunks small enough to be handled by a WML device. It is particularly useful when WML is generated automatically.

Configuration of the IBM-supplied transcoders is done in the preference profile as described in the previous section. For details on the three transcoders refer to the online *Developer's Guide*.

8.6.2 Extending the transcoder

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

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

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 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 WTP helpers is the choice for text clippers or other transforms.

Input content type	Preferred technique	Comments
Binary (such as images)	MEGlet	Java is very suitable, due to availability of existing helper libraries.

Examples of some transcoding tasks are:

- WML 1.1 to WML 1.0

  As WML is a well-formed XML, converting WML to earlier versions could be done using an XSL stylesheet. The WML language has undergone significant changes, but most will require replacing tags suitable for XSL.

- WML to HDML

  The input is XML while the output is not, so the conversion could be done from scratch using XSL. However, WTP includes a generic WML-to-HTML transcoder, which could be used for doing most of the job. Any final adjustment of the HTML needed to conform with HTML can be done by a MEGlet doing search and replace.

- Support for Web TV, including text size enlargement

  Typically a device with a large screen, such as a TV, will support standard HTML. Assuming you already are providing HTML content suited for PC Web browsers, a good solution could be to adjust this HTML. The built-in text clippers can do the job of removing unsupported features like applets and JavaScript. The only transcoding left to do is text size enlargements, which will require search and replace in non-XML content, so the choice is a MEGlet.

- Back-end conversion between XML variants

  XSL is specialized for this type of conversions. There is no obvious advantage in doing the job yourself with a Java toolkit for handling XML (such as the IBM XML4J parser). However, if the HTTP proxy model is not appropriate because the communication is non-HTTP, you could look for alternatives to WTP. As an example the second generation of IBM MQSeries Integrator can do XML conversion of MQ messages.

- Image conversion for color blind users

  To support users with red/green color blindness, you could wish to apply a general conversion of image colors. The transcoding will be done on binary data types, so a MEGlet is the right choice. This also enables use of all the existing image toolkits written for Java.

8.7 Development tools

A number of tools exist to help you develop applications and to work with WebSphere Transcoding Publisher. This section provides a brief overview of the Toolkit supplied with the product and selected tools available for download. For details about the Toolkit see *Extending Web Applications to the Pervasive World with IBM WebSphere Transcoding Publisher*, SG24-5965.

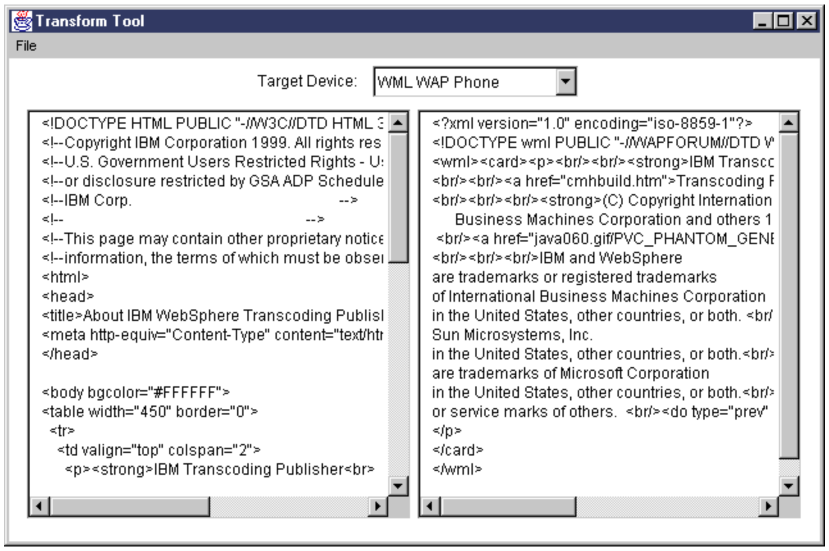
The Toolkit consists of:

- Transform Tool
- Request Viewer
- Preference Profile creator (see 8.5.0.2, “Creating new profiles” on page 127)
- Snoop tool

8.7.1 Transform tool

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 WTP, the tool will show transcoding of both text and images.



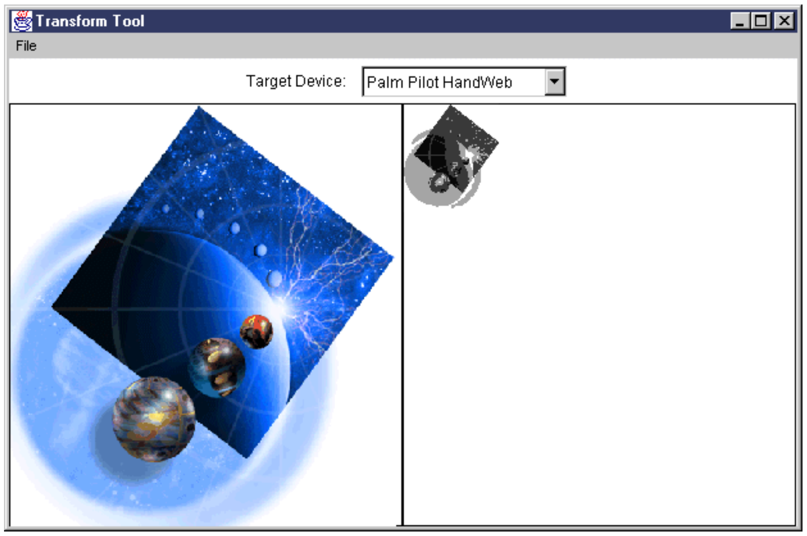
Target Device: WML WAP Phone

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3
<!--Copyright IBM Corporation 1999. All rights res
<!--U.S. Government Users Restricted Rights - U:
<!--or disclosure restricted by GSA ADP Schedule
<!--IBM Corp.                              -->
<!--                                       -->
<!--This page may contain other proprietary notice
<!--information, the terms of which must be obsei
<html>
<head>
<title>About IBM WebSphere Transcoding Publisl
<meta http-equiv="Content-Type" content="text/htr
</head>

<body bgcolor="#FFFFFF">
<table width="450" border="0">
  <tr>
    <td valign="top" colspan="2">
      <p><strong>IBM Transcoding Publisher<br>
```

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD W
<wml><card><p><br/><br/><strong>IBM Transcc
<br/><br/><a href="cmhbuild.htm">Transcoding F
<br/><br/><br/><strong>(C) Copyright Internation
    Business Machines Corporation and others 1
 <br/><a href="java060.gif/PVC_PHANTOM_GENE
<br/><br/><br/>IBM and WebSphere
are trademarks or registered trademarks
of International Business Machines Corporation
in the United States, other countries, or both. <br/
Sun Microsystems, Inc.
in the United States, other countries, or both.<br/>
are trademarks of Microsoft Corporation
in the United States, other countries, or both.<br/>
or service marks of others.  <br/><do type="prev"
</p>
</card>
</wml>
```

Figure 71. Transform Tool example with text



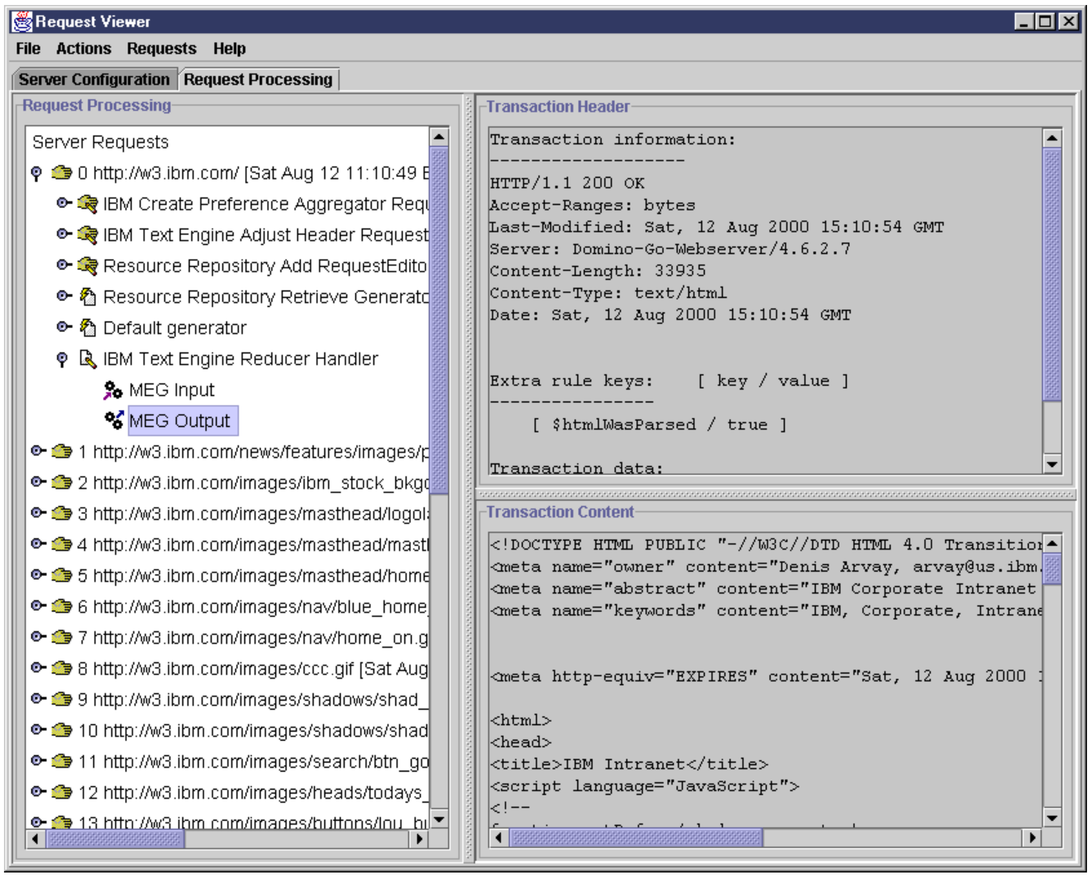
Target Device: Palm Pilot HandWeb

Figure 72. Transform Tool example with an image

8.7.2 Request viewer

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 WTP and the Request Viewer cannot be running at the same time, as the tool will run a complete debug version of WTP using the same ports.



```
Server Requests
0 http://w3.ibm.com/ [Sat Aug 12 11:10:49 E
   IBM Create Preference Aggregator Requ
   IBM Text Engine Adjust Header Request
   Resource Repository Add RequestEdito
   Resource Repository Retrieve Generato
   Default generator
   IBM Text Engine Reducer Handler
      MEG Input
      MEG Output
1 http://w3.ibm.com/news/features/images/p
2 http://w3.ibm.com/images/ibm_stock_bkgd
3 http://w3.ibm.com/images/masthead/logol
4 http://w3.ibm.com/images/masthead/mastl
5 http://w3.ibm.com/images/masthead/home
6 http://w3.ibm.com/images/nav/blue_home
7 http://w3.ibm.com/images/nav/home_on.g
8 http://w3.ibm.com/images/ccc.gif [Sat Aug
9 http://w3.ibm.com/images/shadows/shad_
10 http://w3.ibm.com/images/shadows/shad
11 http://w3.ibm.com/images/search/btn_go
12 http://w3.ibm.com/images/heads/todays_
13 http://w3.ibm.com/images/buttons/lou_bi
```

```
Transaction information:
--------------------
HTTP/1.1 200 OK
Accept-Ranges: bytes
Last-Modified: Sat, 12 Aug 2000 15:10:54 GMT
Server: Domino-Go-Webserver/4.6.2.7
Content-Length: 33935
Content-Type: text/html
Date: Sat, 12 Aug 2000 15:10:54 GMT


Extra rule keys:    [ key / value ]
----------------
    [ $htmlWasParsed / true ]

Transaction data:
```

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitio
<meta name="owner" content="Denis Arvay, arvay@us.ibm.
<meta name="abstract" content="IBM Corporate Intranet
<meta name="keywords" content="IBM, Corporate, Intrane


<meta http-equiv="EXPIRES" content="Sat, 12 Aug 2000 

<html>
<head>
<title>IBM Intranet</title>
<script language="JavaScript">
<!--
```

Figure 73. Request Viewer trace

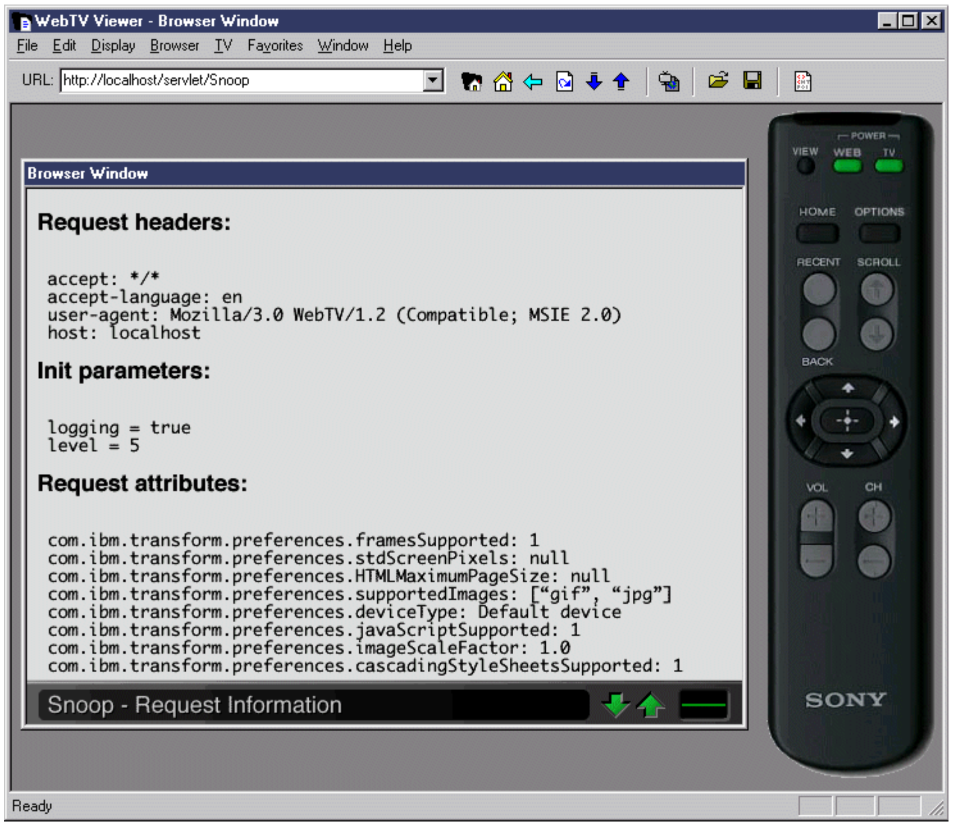
8.7.3 Snoop tool

The Snoop tool is a MEGlet generator similar to the SnoopServlet 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).

To enable Snoop you must register the MEGlet first:

1. From the WTP Administration Console select **Register -> Transcoder**
2. Point out /IBMTrans/toolkit/meglets/Snoop/Snoop.jar
3. Supply a name and comment
4. Activate the MEGlet
5. Refresh or restart the server

Now invoke Snoop by accessing any site via the Web or WAP asking for `/servlets/Snoop` ,which is defined as the trigger in Snoop.prof.


WebTV Viewer - Browser Window
File Edit Display Browser TV Favorites Window Help
URL: http://localhost/servlet/Snoop
Browser Window
Snoop - Request Information
Ready
SONY

Request headers:

```
accept: */*
accept-language: en
user-agent: Mozilla/3.0 WebTV/1.2 (Compatible; MSIE 2.0)
host: localhost
```

Init parameters:

```
logging = true
level = 5
```

Request attributes:

```
com.ibm.transform.preferences.framesSupported: 1
com.ibm.transform.preferences.stdScreenPixels: null
com.ibm.transform.preferences.HTMLMaximumPageSize: null
com.ibm.transform.preferences.supportedImages: ["gif", "jpg"]
com.ibm.transform.preferences.deviceType: Default device
com.ibm.transform.preferences.javaScriptSupported: 1
com.ibm.transform.preferences.imageScaleFactor: 1.0
com.ibm.transform.preferences.cascadingStyleSheetsSupported: 1
```

Figure 74. Example of Snoop called from WebTV

8.7.4 Other resources and tools

The processing of XML/XSL in WebSphere Transcoding Publisher is based on the IBM XML Parser for Java (XML4J). Documentation and code for this can be found at the IBM Alphaworks site `http://www.alphaworks.ibm.com/`. Here you will find lots of other useful tools, including:

- *XSL Trace*

  Step through XSL scripts and see the transformation rules as they are created and the XML or HTML as it is generated.

- *Visual XML Transformation Tool*

  Will take Document Type Definitions (DTDs) describing the source XML documents as input. The user visually constructs the new XML document. The tool will generate an XSL script for transforming the source documents to the target document and also the DTD for the target document. Optionally you can also unit test the XSL script from within the tool.

- *Visual-DTD*

  A visual editor for editing and viewing DTDs. It will generate DTDs, and W3C XML schemas as the standard evolves.

Chapter 9. Subscriber and device management

Network access and/or portal service providers have to face two types of management challenges. These are the usual *systems* management to monitor and maintain physical machines, networks and software, and also management of *subscribers* and, if necessary, their various wireless and wireline *devices*.

Requirements go beyond simple storage and management of profiles for subscribers and devices; business needs require both subscribers and Customer Service Representatives to perform enrollment, profile updates and maintenance. Accounting information needs to be created, maintained and finally provisioned[1] to a billing system. Other types of information, such as subscriber data, need to be exchanged with other systems in various ways, so openness for both API access and provisioning is required.

Application hosting is becoming increasingly popular. Technically this raises a requirement to maintain separately branded offerings simultaneously, and provide each brand with a unique marketplace identity, while sharing the same underlying infrastructure. Also a given implementation must be easily extended to new business models, new service offerings, and a dramatically increased number of subscribers.

Most e-businesses require a complete platform for subscriber and device management that can be implemented quickly. At the same time it must be flexible and scalable to support innovative business models and exponential growth.

9.1 Tivoli Personalized Services Manager overview

Tivoli Personalized Services Manager (TPSM), is an integrated framework of software components that satisfies the needs described above. It is based on open industry standards - the Java language and the Java programming model. The framework is very flexible ,allowing you to choose, tailor, or substitute any TPSM component. The product supports different business models and allows unique, separately branded offerings sharing the same infrastructure. The components are flexible and scalable, allowing for extreme change and growth.

To understand Tivoli Personalized Services Manager, you have to look at several levels of the comprehensive framework. The following sections provide four different overviews: logical, functional, architecture, and deployment. Following these product overviews, we look at TPSM in the context of the Everyplace Suite. Finally, the remainder of this chapter provides more details on the functions and components of TPSM, focusing on architecture and use within the Everyplace Suite. For more general information and development guides, refer to the TPSM product documentation.

[1] In this chapter the term *provisioning* is used with the same meaning as in the Tivoli Personalized Services Manager product; “notification to an external application, triggered by an event”. This can be seen as a “reverse API” actively invoking code outside the product boundaries.

Product names

Even though Tivoli Personalized Services Manager 1.1 has a low version number, it is a mature and proven platform for subscriber and device management. Requirements and experiences from field deployments have led to numerous improvements and extensions of the product. The product dates from 1998, and has had various names and version numbers:

- IBM Intelligent Subscriber Management System (ISMS) 1.0, 1.1 and 1.2
- Tivoli Subscription Manager (TSM) 2.0 and 2.1
- Tivoli Personalized Services Manager (TPSM) 1.0 and now 1.1

A product named Tivoli Internet Services Manager (TISM) is also available sharing the same version numbers. TISM is a subset of TPSM without the Tivoli Device Manager Server. All other functions and components are identical.

The vendor name has changed from IBM to Tivoli as a consequence of the merger of the two companies.

9.1.1 Logical overview

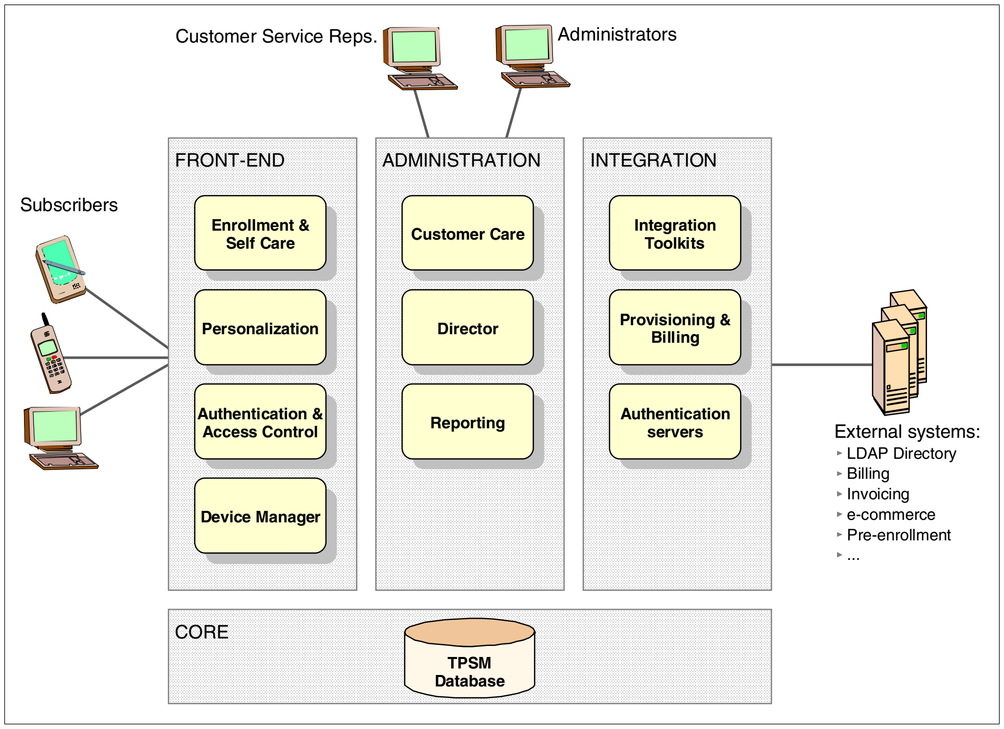


Figure 75. TPSM logical overview

TPSM can be divided into four logical areas, as illustrated in Figure 75:

Front-end The front-end is a framework of flexible components for building and running end-user applications that interact either directly or indirectly with subscriber and device profiles.

Administration Besides tools for traditional system administration performed by administrators, TPSM offers applications to allow Customer Service Representatives to manage subscribers.

Integration	TPSM offers a number of integration toolkits that operate on different levels, letting other systems access TPSM and for letting TSPM provision data to other systems.
Core	The core of TPSM is its database containing subscribers, device profiles, and associated data.

9.1.2 Functional overview

A complete solution for subscriber and device management provides so many diverse functions that the method of describing them becomes non-trivial. Instead of basing the structure on the logical or architectural overview, we have chosen to divide the TPSM functionality into six functional areas that are more related to actual tasks or areas of interest. The six areas are introduced and will hereafter form the structure of the remaining parts of this chapter.

Business and data model

To architect and develop solutions using TPSM, you need a thorough understanding of the central repository and its structure. The TPSM database contains relatively static data on subscribers and devices, but also very dynamic data describing current sessions and billing information. An overview is found in 9.2, "Business and data model" on page 142.

Subscriber and System management

The TPSM management tasks and tools cover a wide spectrum of functions ranging from traditional system configuration, through business model definition, service representative work, and subscriber self service.

Tasks are performed by people in different roles, so TPSM provides a number of management components to meet their needs:

- *Director* and *Reporting* for administrators
- *Customer Care* and *Reporting* for Customer Service Representatives
- *Enrollment* and *Self Care* components for letting subscribers serve themselves

More details on these components may be found in 9.3, "Subscriber and system management" on page 145.

Pervasive device management

To enable disconnected or specialized use of pervasive devices, you may wish to let applications and/or data reside on a device. *Tivoli Device Manager Server* is a solution for distributing, tracking and maintaining such software on pervasive devices in or outside your enterprise. See 9.4, "Pervasive device management" on page 150.

Authentication and Access Control

TPSM includes a sophisticated and flexible security model providing authentication based on subscriber information in the central repository for use by not only TPSM, but other applications and the *Network Access Devices* as well.

Access control to individual or groups of pages for subscribers can be built in to your application using the TPSM toolkits, or TPSM *Premium Content* using URL-based protection.

See 9.5, "Authentication and access control" on page 158.

Personalization Services
Application and content delivery can be personalized to match the profile of individual subscribers. The TPSM *Portal Toolkit* (pTk) is a framework for serving individual portal pages, targeting subscribers with special information and banner ads.

Another tool to build up portal pages is the *pTk JSP Components Framework*. This is an extendible collection of Java applications including calendar, address book and more, allowing rapid development of personalized services. Refer to 9.6, “Personalization services” on page 161 for details.

Integration and Provisioning
For integration with external databases and systems, such as an existing customer database, a billing system, LDAP directory or an e-mail system, TPSM has an *Integration Toolkit* (iTk) consisting of the following:

- *iTk Core* - provides database access with validation and utilities.
- *iTk Business Objects* - represents subscribers, deals, accounts, etc.
- *iTk Provisioning* - offers event notification to external applications.
- *iTk Billing* - provides an interface for use by an external billing system.

More on these toolkits is found in 9.7, “Integration and provisioning” on page 168.

9.1.3 Architecture

It is important to understand that TPSM is a framework consisting of front-end components for building custom applications, administrative tools for configuring the system, and a set of integration components and specialized servers.

Note

Most of the TPSM components are optional, so you have to include only the components that are useful in your implementation. If you don’t need a specific function, leave it out. If you prefer a non-TPSM implementation, you may substitute the corresponding component.

The front-end consists of end-user Web applications, so you probably will do most of the customization and development around these. As shown in Figure 76 on page 139, there are basically four sets of application pages: *Enrollment*, *Self Care*, *Premium Content* (access controlled pages) and a *personalized portal*. For each, TPSM provides a set of Java servlets and template JSPs. To get the desired functionality and look-and-feel of an application you can both:

- Develop your own JSPs, or modify the samples provided.
- Change setting in the property files of the front-end components.

The TPSM components, such as authorization and personalization, can also be used for your own business applications.

The front-end component framework consist of servlets and JSPs, while the administrative tools are primarily applets. The only non-Java components in TPSM are the RADIUS server, the enrollment configuration of Windows dial-up, and the database installation scripts.

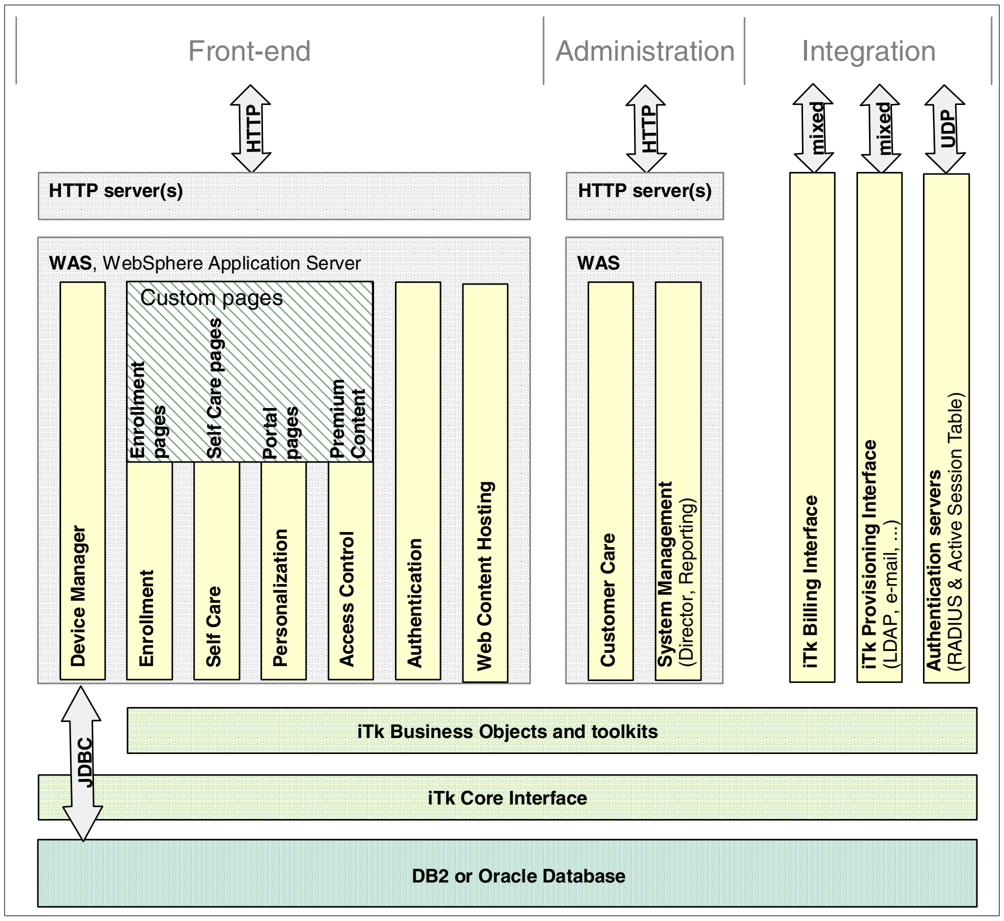


Figure 76. TPSM architecture

As illustrated in Figure 76, the TPSM front-end and administrative components run on IBM WebSphere Application Server using HTTP servers for communication. In the Everyplace Suite, that is the *IBM HTTP Server*. Each component uses its own instance of the HTTP server as a virtual host assigned to different IP addresses or port numbers. The actual configuration can be hidden from end users by proxy mappings typically done in the Edge Server - Caching Proxy, which hosts the IBM Everyplace Authentication Server.

9.1.4 Deployment

This section provides an overview of considerations when deploying TPSM to give you a few guidelines for the physical implementation. Requirements and policies for security, availability, scalability and personal preferences will dictate the exact deployment that is right for you.

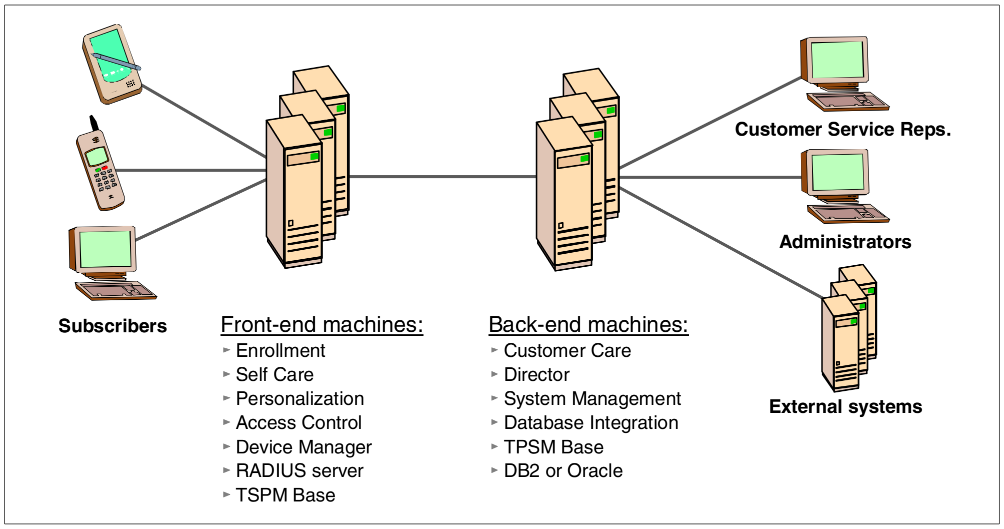


Figure 77. TPSM deployment in two groups

All TPSM components can be deployed on a single machine. However, in most implementations the components are split into two groups to divide the workload, and to allow separation by a firewall, if better protection of the back-end servers is needed. This is illustrated in Figure 77. All functions accessible by subscribers are located on the front-end machines, while the databases and administrative functions are deployed on the back-end machines. Note that the RADIUS server is typically running on the front-end for improved performance and enhanced security, even though its database resides on the back-end.

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.

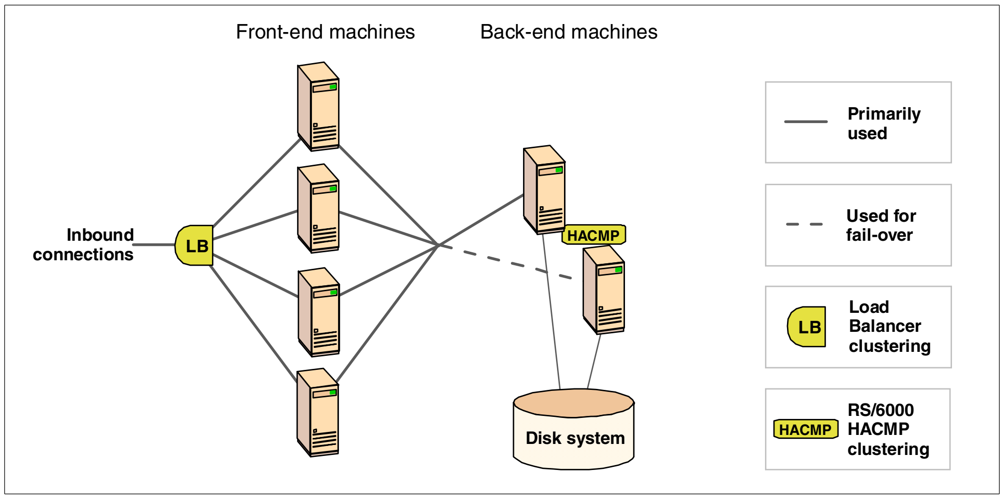


Figure 78. Deployment for scalability and availability

Scalability and availability can be addressed with an implementation as illustrated in Figure 78. Two or more duplicated front-end machines are set up in an *Edge*

Server - Load Balancer cluster. The back-end has been equipped with a set of two properly sized RS/6000 machines with HACMP hardware failover and a shared high availability and high performance disk system. Considerations and choice of techniques are discussed in Chapter 4, “Performance and scalability” on page 37 and 3.6, “Availability - dispatchers and clusters” on page 32.

9.1.5 TPSM in the Everyplace Suite

The IBM WebSphere Everyplace Suite makes use of TPSM to provide core subscriber authentication and management functions. The following components are required:

- TPSM database
- Integration Toolkit
- Customer Care
- System Management
- Authentication servers (RADIUS and Active Session Table)
- iTk Provisioning

The remaining components of TPSM are optional and can be used in building and running a specific solution.

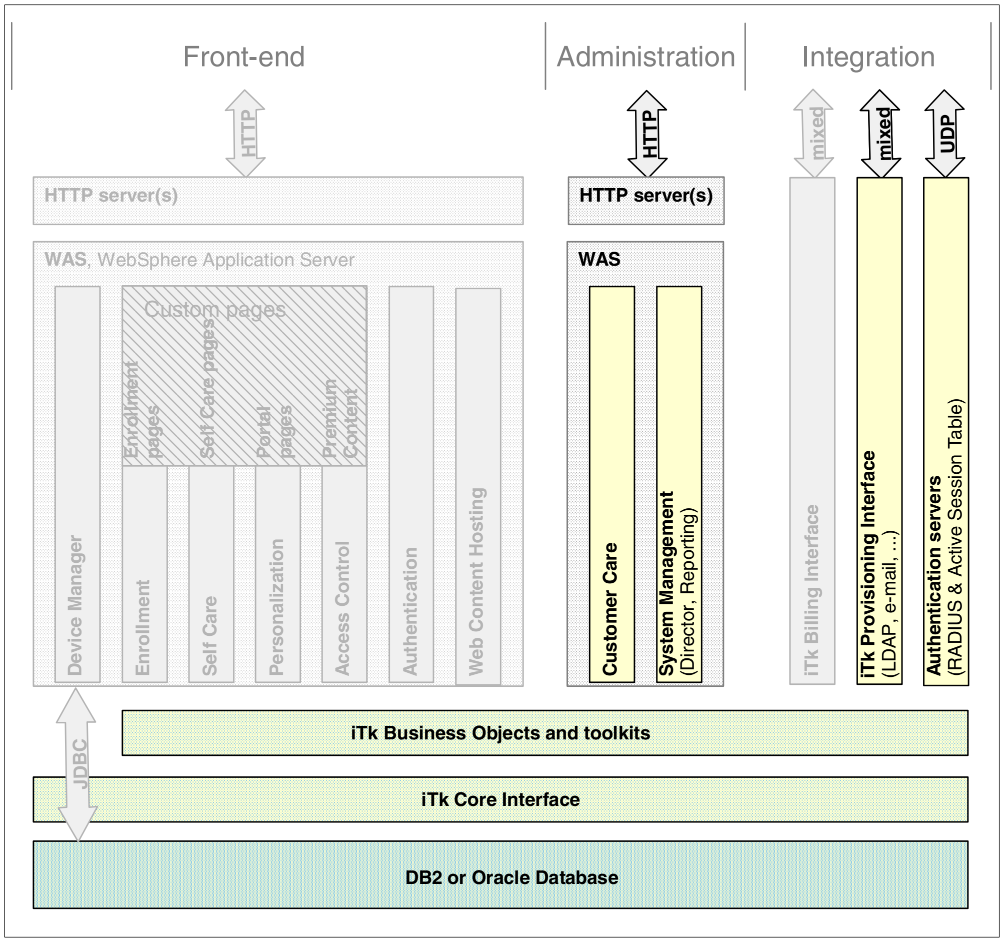


Figure 79. TPSM components highlighted are required by the Everyplace Suite

It is most likely that you will build applications that will access or update information in the TPSM database. This could be for personalization of pages, implementing security at the application level, enabling data synchronization with other systems or enabling profile management for subscribers.

TPSM provides integration at three different "levels":

Component level TPSM provides frameworks and samples for enrollment, self care, personalization etc. With these components basic functions are taken care of, and you can concentrate on the business application.

Toolkit level The four flavors of the Integration Toolkits (iTk) are the most flexible way of working with nearly the full set of TPSM data in any way. In addition, the Provisioning iTk allows provisioning outbound from TPSM to LDAP, e-mail, or other systems.

Data level A full range of DBMS product options exists to take advantage of the rich TPSM data model, for example, database replication, data warehousing options, reporting tools, etc.

In the Everyplace Suite, provisioning is set up to mirror all subscribers into the IBM SecureWay Directory (LDAP). This allows any system to look up subscriber information using either LDAP or one of the TPSM interfaces. However, note that the TPSM database is the master of the mirroring, so any update of subscriber information must be done in TPSM - not using LDAP[2].

9.2 Business and data model

Describing all the business models supported by TPSM will require thousands of words and just as many examples. Instead we start by introducing the rich data model forming the core of TPSM. From this the business models can be intuitively derived. A solution architect should find this selective overview of the data model useful as background for discussing the capabilities of TPSM from a business perspective. For a straight business-value proposition, refer to the sales and marketing material available on the IBM and Tivoli Internet and intranet sites.

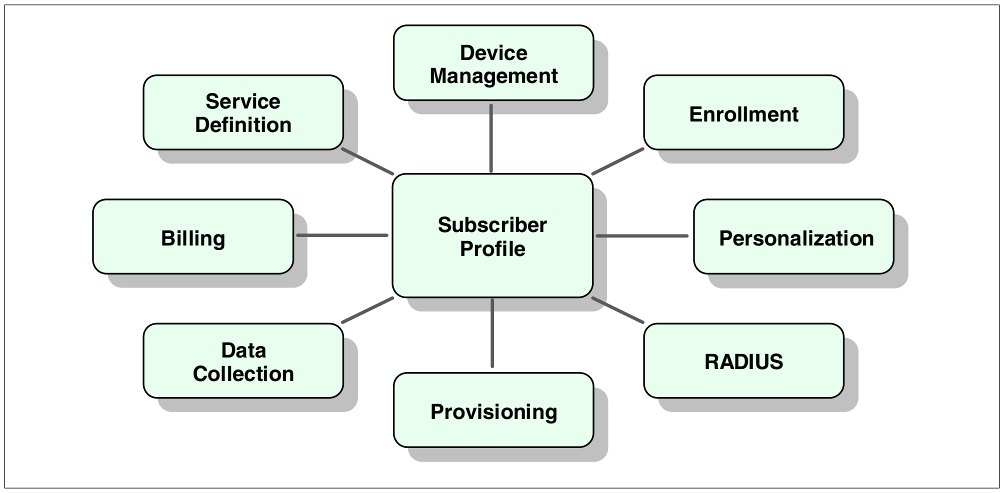


Figure 80. TPSM data model overview

[2] The IBM Everyplace Wireless Gateway uses LDAP as its subscriber database. To allow a Wireless Gateway administrator "Gatekeeper" to perform updates in the Everyplace Suite, it has been equipped with a servlet that performs the update in TPSM.

Figure 80 is a high-level overview of the TPSM data model. The *Subscriber* profile metadata is the core of the model, on which this book will focus. Some of the other groups are covered very briefly in other sections of this chapter, while any further information must be found in the product documentation.

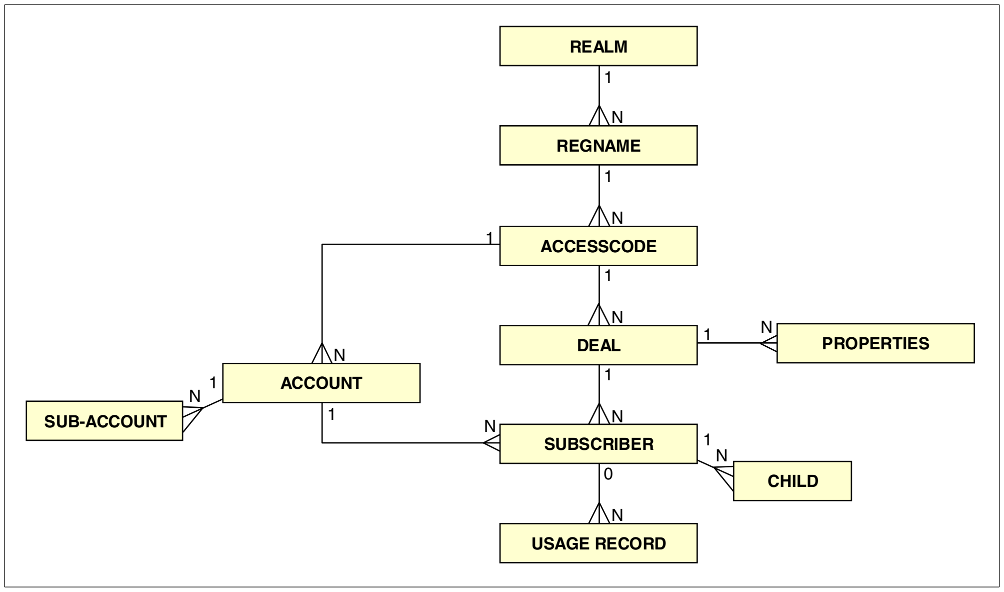


Figure 81. TPSM core data model

Core data divides into the data types *Realm*, *Subscriber* and *Account*. Around these three types a few other types are used, as illustrated in Figure 81 and explained in the following. The figure illustrates relations between the data types, where the numbers indicate if the relationships are 1-to-1, 1-to-many, or optional 1-to-many (indicated with a zero).

Realm

The top level administrative domain is the *Realm*. 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 TPSM 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. For example, a group of Customer Service Representatives can be allowed to work on `a.com` and `b.com`, but not `c.com`. Applications and services such as Enrollment and Self Care can be shared across *Realms*, or made specific to a *Realm* with the option of individual branding.

Regname

High-level access control and policy are defined in a *Regname* or *Registration Name*. A Registration Name must have one or more *Access Codes* (see next section). The Registration Name can be defined to be either “Generic” or “Multiple Access”:

Generic	Only one Access Code is permitted for the Registration Name and this is created automatically when the Registration Name is created. It is called the “DEFAULT” Access Code.

Multiple Access	Multiple *Access Codes* may be defined. An Access Code is linked to a Sales Channel for tracking how a subscriber was targeted. A Deal is associated with an Access Code.

Access Code

An Access Code defines a set of available services (*Deals*, see next section) and accounting policies at a high level. It normally maps to the sales channel to which the subscriber has enrolled. This could be using a CD-ROM, using the Internet, through a Customer Service Representative, or by bulk mass import from an existing system.

Deal

A *Deal* represents a set of services and corresponding accounting policies. For example a Deal could be "$5 per month for unlimited access, first month free" or "1¢ per hour connected, with no access to Premium Content" or "One time charge, must be paid with VISA or MasterCard". If more than one Deal is assigned to an Access Code, the user must select a deal during the enrollment process.

The definition is very flexible and allows the administrator to associate a Deal with a set of custom properties. Deals can also provide linkage to external billing systems.

Properties

The most common use of *Properties* is to store measurable parts of a Deal. That could be the number of SubAccounts allowed, amount of disk space assigned or for mail storage, or any other measure specific to the service provider.

Subscriber

This is the metadata for storing the subscriber profile containing the user name, password, full name, address, personalization preferences and other unique information.

A number of *Children* can be created and associated with the Subscriber and his Account. These could be family members having their own set of user name, password and personalization settings.

Account

A residential user (referred to as a consumer) will have a single subscriber profile associated with a single Account containing billing information. For a corporate account with a number of employees, there will be a single Account and several Subscribers, as each enrolled employee will have one. See the illustration in Figure 82.

For a business account, a *Sub-Account* can be created to represent subsidiaries or other units within the company.

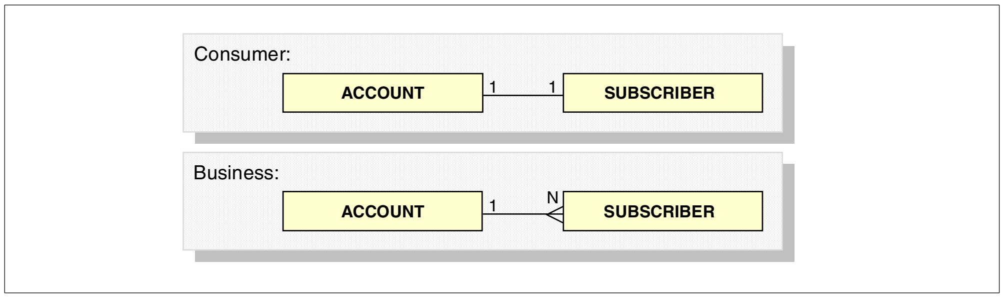


Figure 82. Relation between Account and Subscriber

Usage Record

The Network Access Device (NAS), as the Wireless Gateway in the Everyplace Suite, will send a message every time a connection is initiated or has ended. This is done using the RADIUS protocol. The information lets TPSM create Usage Records for the subscriber, which together with the Deal policy and Account information will form the basis for billing.

9.3 Subscriber and system management

Probably the most central components of TPSM are the ones directly related to subscriber management. The term covers a number of functions including subscriber enrollment, profile/preference updating and status reporting, which are necessary to maintain service delivery.

TPSM provides several subscriber management components that are optimally suited for each of the following three groups of users:

1. Subscribers, who will do self service:
   - *Enrollment*
   - *Self Care*
2. Customer Service Representatives performing customer care by phone or mail:
   - *Customer Care*
   - *Reporting*
3. Administrators who will manage system configuration, rule administration, mass import/export, reporting and similar tasks:
   - *Director*
   - *Reporting*

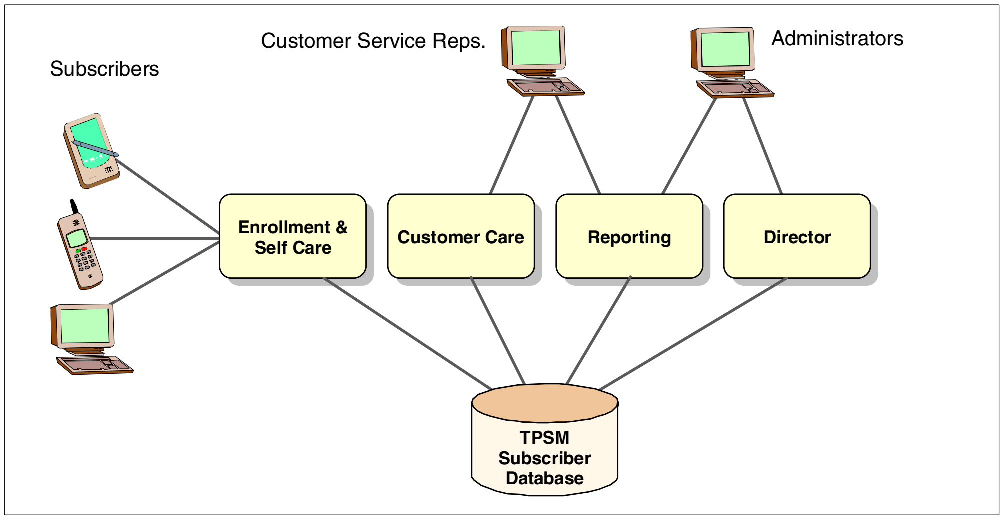


Figure 83. Subscriber management components

Even though similar tasks can be performed by all three user groups, the components are technically diverse; the subscriber services are a part of the front-end framework helping you build up Web applications, Customer Care is an out-of-the-box GUI application optimized for frequent use, while the administrator tools are GUI based with some functions to be done from a command line.

9.3.1 Enrollment

The flexible TPSM enrollment process consists of a servlet, a configuration file, a set of JSPs and graphics. The JSPs and images can be modified to display whatever logo, enticement text, deals offered and payment methods that you devise. Panels may be added, removed, or rearranged as desired.

Like all of the TPSM front-end services, the Enrollment can be adopted and served to any device; thus is not restricted to use from a PC Web browser.

Enrolling from any device

If network access itself is a part of the service offered, how do you connect for the first time? Some devices, such as screen phones or cable TV devices, are usually preconfigured to connect to a service provider’s Web site to retrieve further configuration settings, applications, or data. Other devices need initial configuration setting or software to connect as desired. Examples are *Over-The-Air* (OTA) provisioning of a WAP phone, or PDA software downloaded from a PC. TPSM provides tools for that, as described in 9.4, “Pervasive device management” on page 150.

If the service you provide includes network access from a PC running Windows, the enrollment component can output a Microsoft InterNet Signup file (INS), which is downloaded to the subscriber's PC. This file contains necessary connection settings such as dial-in number, DNS addresses, and account-specific parameters. Other mechanisms must be used for non-Windows or specialized purposes.

9.3.1.1 Ways of enrolling

This section lists the different options of enrollment:

Web Enrollment

For users already connected to the network (Internet or intranet), the provided set of templates can be used to build screens prompting for desired information, which finally will be validated and saved into the TPSM subscriber database.

Banner Ad

Potential subscribers can click through a banner ad that leads them to the TPSM enrollment screens. Technically this is very similar to Web enrollment, but it allows for an Access Code to be passed along to the enrollment server when the banner ad is clicked. This provides a tracking mechanism for marketers and triggers the display of Deals and billing plans that are specific to the banner ad's sponsor.

Microsoft Referral Service

Desktop PC users running Microsoft Windows can use the Internet Connection Wizard. This wizard provides a temporary connection to the Internet and the Microsoft Referral Server, where the user is presented with a choice of ISPs. When an ISP running TPSM is selected, a series of enrollment screens prompt the user for signup information.

Branded CD

A more flexible alternative with the option of branding is to create a CD containing startup files, a browser, and an installation wizard. This software provides a temporary connection and can point to or include the desired signup screens. The CD package is printed with an access code that the subscriber enters during registration, or the software could connect for signup using a common set of credentials.

Customer Service Representative

Enrollment can be done by calling a Customer Service Representative, who will prompt the new subscriber for information and enter it using the online screens of the TPSM Customer Care application.

If network access is provided, the representative must guide the user through setting up his PC or other device by supplying a username and a temporary password. Alternatively a CD and/or instruction letter can be mailed to the new subscriber.

Bulk enrollment

The TPSM Integration offers a set of flexible APIs to allow enrollment from other sources. This can be used for mass import of existing subscribers at the time of TPSM deployment or for building an interface allowing other systems to register subscribers into the TPSM database.

9.3.1.2 Business account options

Not only can enrollment be done by individual subscribers (as illustrated in Figure 82 on page 145), but entire businesses can be allowed to enroll, creating a new Realm automatically. This way a company can create a branded site and let subscribers enroll into its own private Realm!

The delegation allows a company to let all or only selected people enroll as subscribers. How many options are given free for delegated Realm administration can be configured.

9.3.1.3 Virtual / branded enrollment

Each of the enrollment option conforms to the data model of TPSM, which means a user will be enrolled into a specified Realm, which can be uniquely branded, not being aware of other Realms hosted on the same installation.

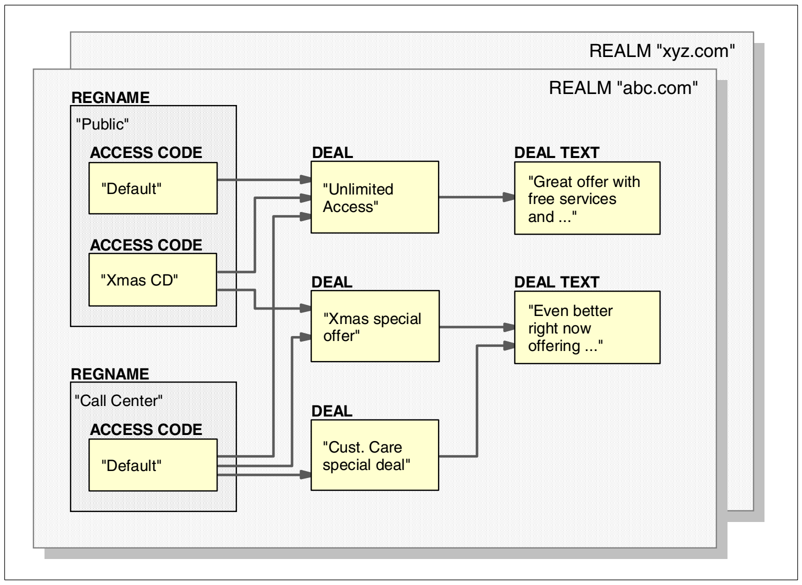


Figure 84. Sharing definitions between branded enrollments

Infrastructure and data definitions can be shared in several ways. Figure 84 is one example where *Deals* and their description text are shared across *Access Codes* in different *Regnames* and potentially different *Realms*.

9.3.1.4 Information verification

TPSM provides utilities for form and data validations. Everything from the validation of dates and values required to be within a given interval, to more complex functions, such as validating a credit card number, can be done. The built-in database integrity checks, for example, two subscribers from trying to sign up in the same Realm with the identical user names.

9.3.1.5 Fulfillment

Maybe an enrollment is fulfilled just by having registered all the information in the subscriber database and displaying a screen saying “Welcome Joe”. But you might also want to send e-mail, print a letter, or provision the enrollment to another system, such as billing. This can all be done using the TPSM Provisioning Toolkit described 9.7, “Integration and provisioning” on page 168.

9.3.2 Self care

A user registered in the TPSM subscriber database can be offered the ability to modify his own profile; this is called self care. When building the end-user applications, the self care component offers a set of tools to easily enable various self service tasks. As a front-end component, self care consists of set of Java servlets and template JSPs.

Basically any operation can be enabled for self care using the Integration Toolkit, iTk, but typically you will base them on the Self Care templates. Using these templates, subscribers are enabled to perform the following operations:

- Change password
- Change secret data item (used for identification when calling a service representative)
- View account status (how much is owed, how much time is spent online)
- Change billing plan and payment choices
- Add, modify, or cancel Premium Content subscriptions
- Add, modify, or delete a *Child*

Screens presented to the user can be modified and personalized as desired. This will be done primarily by modifying the JSP templates.

9.3.3 Customer care

A Customer Service Representative (CSR) will use the Customer Care component of TPSM to manage subscribers and their accounts.

After launching the Customer Care applet, a CSR will provide his or her personal login credentials. This identifies the CSR and grants the proper administrative privilege. For example this could be full rights to manage Realms `a.com` and `b.com`, but not `c.com`.

The tasks implemented in the Customer Care default template are slightly different for consumer and business accounts. The following are some examples of tasks that can be performed by Customer Care:

- Enroll a new subscriber
- Search for existing subscribers
- View and update subscriber profile/account information
- Create and delete Child subscribers
- Disconnect and reconnect a subscriber and/or children
- View subscriber billing summary
- Apply credits or payments to an account
- View a subscriber’s devices (if Device Manager is installed)
- Create various reports

Other TPSM and external applications can be linked into Customer Care to provide a common launch pad by a URL link or to do a tighter integration of both data and user interfaces.

9.3.4 Director

The Director tool is launched as an applet through a Web browser. Director is the administrator’s tool for configuring and managing the following:

- Premium Content

  All definition and configuration of Premium Content - see 9.5, “Authentication and access control” on page 158.

- Self Care options

  The Deal assigns a number of “options” to a subscriber. The Director tool allows an administrator to maintain a list of options that are subject to de-selection by a subscriber using the Self Care application.

- RADIUS server

  Configuration of RADIUS clients, RADIUS vendors, and RADIUS attributes.

- Registration definitions

  Director is the tool to build and maintain the tree of Realms, Regnames, Access Codes, Deals, Metrics, and Method of Payments.

- Access control and delegation

  Setting up login credentials, privileges, and delegation for administrators and Customer Care Representatives.

- Create various reports

  As described in the 9.3.5, “Reporting” on page 150.

9.3.5 Reporting

Administrators and Customer Care Representatives can query various types of reports to track the size of a subscriber base, subscriber activity, system liability and the volume of enrollments through the different enrollment channels. The Reporting component offers a number of reports of which some are run automatically, while others are started through the Reporting applet user interface.

Custom reports can be built in addition to the set of predefined reports. The predefined reports are as follows:

1. Daily Reports
   - Enrollment Activity
   - Session Activity
   - Hourly Activity
2. Weekly Reports
   - Member Level Activity
   - Subscriber Level Activity
3. Monthly Reports
   - Monthly Usage Summary
   - Cohort By Hours
   - Enrollments By Access Code
   - Monthly Enrollment Report
   - Disconnect Analysis Report

9.4 Pervasive device management

Delivering services to new classes of devices introduces new requirements for both “thin” and “fat” clients:

- **Thin client requirements**

  Pervasive devices with no software installed are “thin” clients. Examples are WAP devices, screen phones and of course the Web browser. An application service provider normally considers the thin client to be conveniently free of any maintenance. However, you still may need to distribute configuration parameters and other information, such as a “rest page” displayed on a screen phone when idle. This is particularly valuable if the service offered includes network connectivity.

- **Fat client requirements**

  When a device contains business application software, it’s referred to as “fat”. A fat client can be desired and/or required, if the users are supposed to work disconnected from any network, or if communication must be secured and optimized beyond the capabilities of the “browser” in a thin client. Offering applications and services for a fat client raises the need for regular software distribution and maintenance.

Most enterprises using PCs are facing these problems, and have probably implemented powerful tools to efficiently address them. However, two requirements introduced by the pervasive world may render most existing solutions unsuitable:

1. It works with PCs only, but support for pervasive devices is required.
2. It’s suitable for internal users only, but needs to reach users outside the enterprise.

Fat client example

An example of a fat client application is a travel package that’s downloaded to a PDA before travel departure. Using a browser from a PC or the PDA itself, the subscriber will visit the service provider’s self care page, sign up and pay a fee for this premium service. “Travel Package - Paris” contains:

- Restaurant guide of Paris composed according to personal preferences, and possibly enabled for GPS positioning
- Currency converter application loaded with current rates of French francs
- Dictionary with useful French words and phrases
- Directions, maps, useful URLs, phone numbers, Metro plans, etc.

The application modifies the time zone and default currency setting of the PDA according to the destination, and adds a new sheet to the PDA’s expense application for reporting during the trip.

9.4.1 Device manager overview

Issues raised are addressed by the *TPSM Device Manager*, a component extending the TPSM framework to distribute, keep track and maintain applications or data residing on pervasive devices. Device Manager is loosely coupled with the rest of TPSM to allow it to be used with third-party systems. As illustrated in Figure 76 on page 139, the Device Manager uses the iTk, but is accessing the database directly via JDBC.

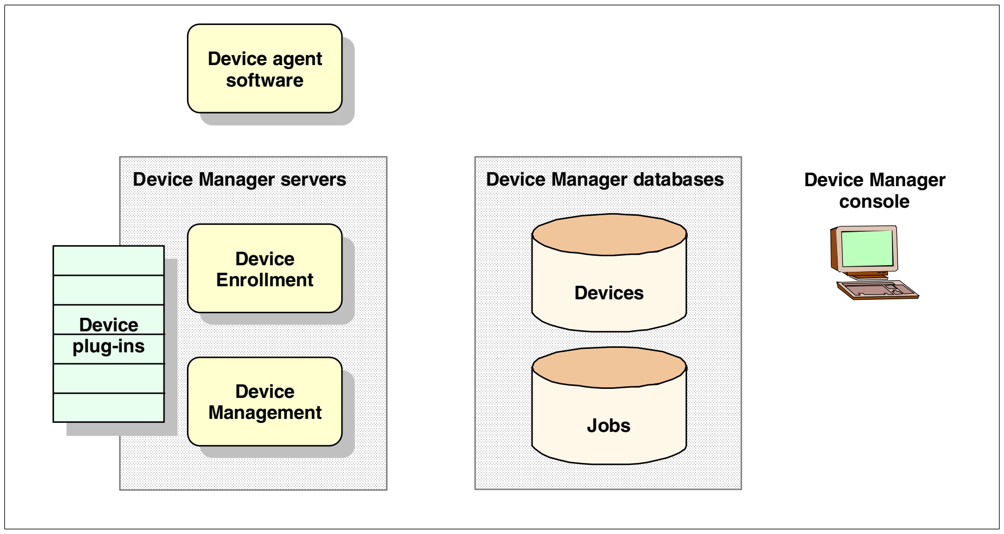


Figure 85. Device Manager overview

As illustrated in Figure 85, the Device Management component consists of a number of parts:

- **Device Manager server**

  This is the runtime component of Device Manager, which is implemented as Java servlets. The tasks it performs are split in two, which also reflect the architecture of the server: initial enrollment versus ongoing management of devices. A set of plug-ins are used both to interface with special protocols and to understand device specific characteristics. The set of plug-ins is extendible with new functions and with new device types. In addition to the plug-in architecture a number of APIs exist for integration with subscriber management and other systems.

- **Device agent software**

  The agent allows distribution of applications or data to “fat client” devices. One or more agents may be needed to support devices using different operating systems or if required by other technical or business needs.

- **Device databases**

  The central repository is implemented in relational database storing devices and device-related data resources including jobs scheduled for execution. The database can be shared with the subscriber management components of TPSM.

- **Device Manager console**

  The management console must be installed on a Windows workstation and is run as a Java applet from a Web browser. It allows an authorized administrator or Customer Care Representative to configure single devices, device types, jobs for distribution or configuration, and the actual software to be distributed.

9.4.2 Pervasive devices supported

The current release of Tivoli Device Manager is delivered with plug-ins supporting four device types:

- Palm Computing PDAs

  The PalmPilot family on the PDA market running PalmOS.
- Compaq Aero 8000 H/PC Pro

  Handheld PC using Microsoft Windows CE.
- IBM Internet Access Phone Model 500

  An ISDN screen phone with a built-in Web browser using point-to-point dial-up. Currently it is sold in Japan only.
- PvC Client Stack (Pervasive Computing Client Stack)

  Devices adhering to the PvC client stack architecture can range from in-vehicle information systems to home service gateways.

The pieces delivered to support each of these devices are summarized in Table 3 below. For details on supported features refer to the product documentation.

Table 3. Default Device Manager support for devices

Plug-in class	**PalmPilot**	**Compaq Aero**	**IAP500**	**PvC Client Stack**
Base device class	Yes	Yes	Yes	Yes
Software distribution[a]	Yes	Yes	Yes	Yes
Device configuration	Yes	Yes	Yes	Yes
Restpage management[b]			Yes	

a. Includes both a server-based plug-in class and a device agent software.
b. A restpage is a page displayed when the device is idle.

9.4.3 Initial enrollment/setup

This section describes the interactions that will occur between Device Manager and a device during the first contact. To illustrate the process, we use an example of a PalmPilot PDA enrollment. This example is more interesting than a simpler case with a device, such as a screen phone, which natively is connected to a network.

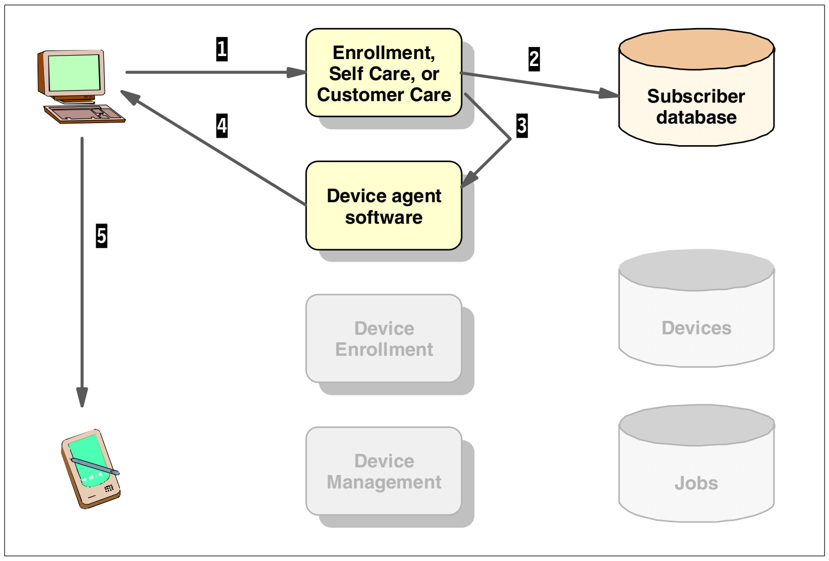


Figure 86. Initial device enrollment - Getting the device agent

The first step is to get the device agent and install it on the device. That process is shown in Figure 86 and does not involve the Device Manager server:

1 The subscriber contacts the service provider to initiate the process. That can be done using the Enrollment / Self Care applications, or by calling a Customer Service Representative.

2 If desired, the request can be registered in the subscriber profile. This could be used for tracking and/or to charge a Premium Content fee.

3 The Self Care or Customer Care application provides a link to download the agent, or creates e-mail with it.

4 The software containing the agent is sent or downloaded.

5 Finally, the agent is downloaded to the PDA using infrared or cable connection.

The subscriber now invokes the device agent that will connect to the server.

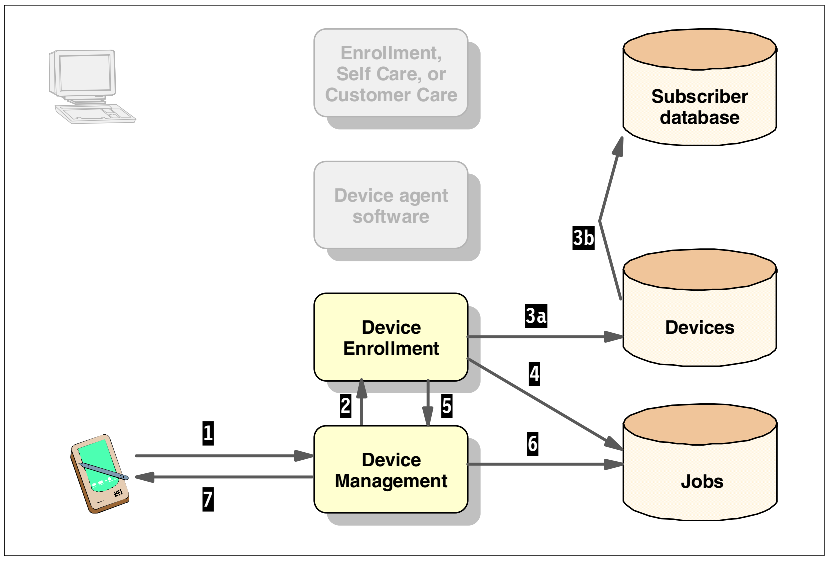


Figure 87. Initial device enrollment - Registering the device

The process of enrolling the device itself is as follows:

1 The device agent connects to Device Management.

2 Device is not known, so the request is redirected to Device Enrollment.

3 Device is registered in the Device Manager database, and is being associated with a subscriber record.

4 A device configuration job is scheduled for the device.

5 Device is redirected back to Device Management.

6 Now, the newly registered device is recognized, and because it has jobs scheduled, these jobs will be processed.

7 The actual configuration and/or software is downloaded to the device.

The subscriber now invokes the device agent, which is configured with the parameters necessary to connect to the server.

9.4.4 Software distribution

The day-to-day operation of software distribution is illustrated in the following example. Again we use a PDA as the example. However, it should be noted that the process is the same for any device using the Device Management agent.

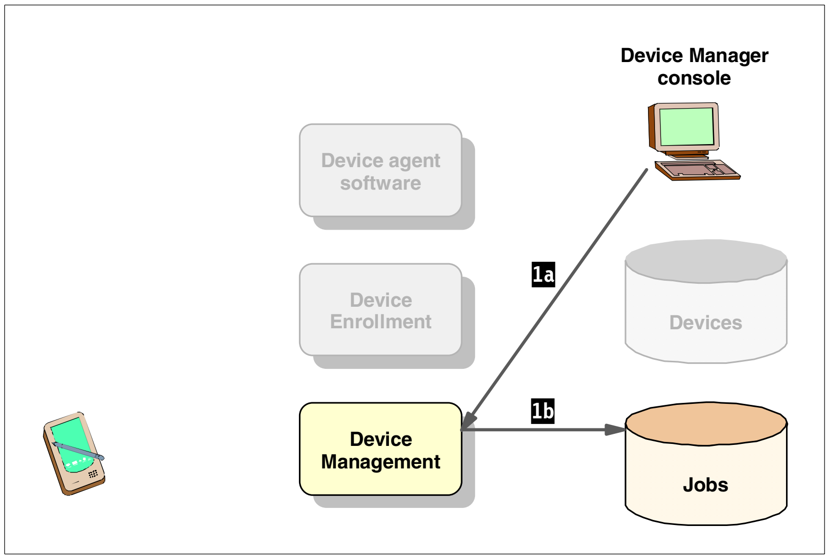


Figure 88. Software distribution - Submitting a job

An administrator or Customer Service Representative will use the graphical Device Manager console as shown in Figure 88:

1 The job is defined by specifying the task to be carried out and the criteria, for example device type, an individual subscriber and expiration date. It is now submitted to the job database.

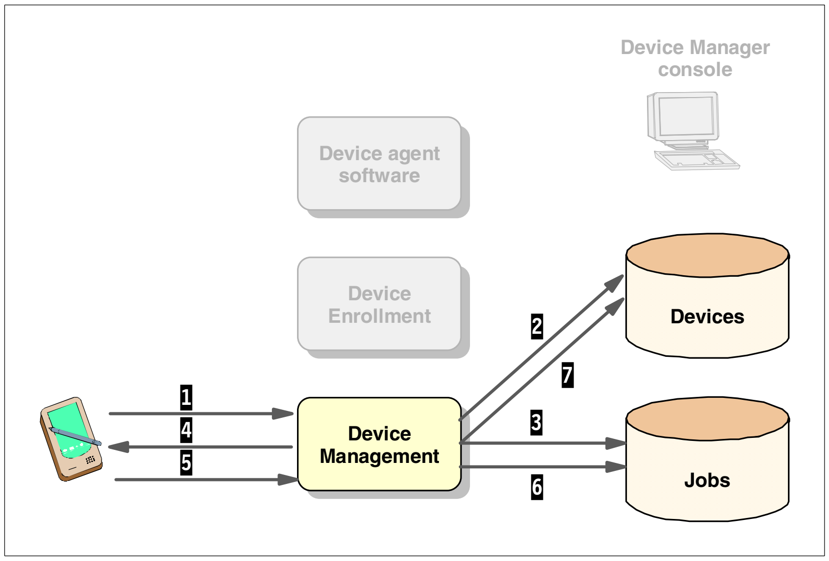


Figure 89. Software distribution - Device gets the software

Next time a device agent connects to the Device Manager server the following will happen:

1 The device agent contacts the Device Management server.

2 Device Manager makes sure the device is enrolled.

3 Device Manager searches for the job scheduled for the device and its type.

4 The jobs 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 is completed successfully.

6 The job database is updated to reflect that this job has run for that device.

7 The device database is updated to reflect that this software is installed on that device.

9.4.5 Deployment within the Everyplace Suite

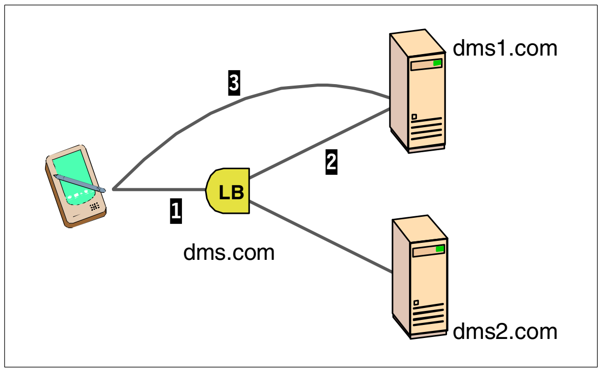


Figure 90. Device Manager redirections

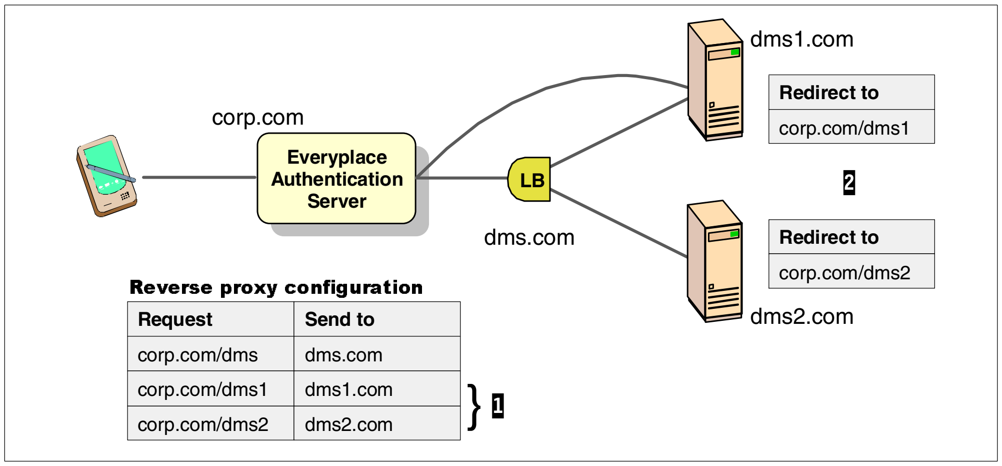
For load balancing and high availability, you may want to deploy multiple Device Manager servers in a Load Balancer cluster as illustrated in Figure 90. This is what will happen:

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 stateful 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 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 Suite implementation you will set up the IBM Everyplace Authentication Server as a reverse proxy in front of all application servers. To work properly, both Authentication Server and TPSM need a little extra configuration, because the actual server addresses are hidden to users.



Reverse proxy configuration

Request	Send to
corp.com/dms	dms.com
corp.com/dms1	dms1.com
corp.com/dms2	dms2.com

Figure 91. Device Manager in the Everyplace Suite

As illustrated in Figure 91, two things have to be configured to make this mechanism work in the Everyplace Suite environment:

1 The IBM Everyplace Authentication Server must be aware of the actual server addresses and their URL defined to users outside.

2 The Device Manager must be "Everyplace Suite 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 Authentication Server in the Everyplace Suite, it must be configured to direct those to the URL that is used externally. See Chapter 6, "Authentication" on page 69.

9.5 Authentication and access control

TPSM offers four components related to security:

- Authentication component
- Premium Content
- RADIUS server
- Active Session Table server

All of these are built on the shared TPSM subscriber database, and can be used independently of each other.

9.5.1 Authentication

9.5.1.1 Identification

The single sign-on architecture in the Everyplace Suite authenticates all sessions. The IBM Everyplace Authentication Server will validate every request and, if valid, put in an HTTP header with the authenticated identity of the user, device and network:

Table 4. HTTP headers used for identification

Header	Content
X-IBM-PVC-User	User name in the form “user@realm”
X-IBM-PVC-Device	Device type as mapped from the definition in the LDAP database
X-IBM-PVC-Network	`LAN`, `GENERIC_DIAL`, or `GENERIC_WIRELESS` based on the entry point

The identification of the user is directly given by the header information. However, for the identification to be reliable, the infrastructure must be set up to prevent bypassing the Authentication Server. All of the three HTTP header values can be used for personalizing the content served.

A new feature in TPSM 1.1 enables logon to a WAP device. It can work with the IBM Everyplace Wireless Gateway, the Nokia WAP Server, and other WAP gateways depending on the customer engagement agreement. The authentication process is achieved by MS-ISDN or IP address during auto-logon. It is then completed by the WML form. This feature is described in the TPSM System Administration guide.

9.5.1.2 TPSM authentication and delegation

TPSM provides a flexible single sign-on mechanism. It can be used with RADIUS authentication for dial-up/wireless connections, with HTTP basic authentication, or with HTML form authentication. In the context of the Everyplace Suite, authentication and single sign-on are delegated to the IBM Everyplace Authentication Server. This section describes how the TPSM authentication will work within the Everyplace Suite. Refer to the TPSM product documentation for a general discussion on the topic.

A DefaultAuthenticator class is provided which is extended by the PremiumAuthenticator and WapAuthenticator classes to provide specialized authentication. The DefaultAuthenticator class has been extended in TPSM 1.1 to provide integration with the Everyplace Suite.

The DefaultAuthenticator implements two authentication stages:

- **Stage 1**

  First it tries to identify the user with information extracted from the HTTP request headers. It supports:

  - Authentication by source IP address
  - Authentication by source IP address forwarded in any HTTP header
  - Authentication by user name forwarded in any HTTP header
  - Authentication by user ID forwarded in any HTTP header
  - None

- **Stage 2**

  Then, if stage 1 fails, it prompts the user for a login password:

  - Basic HTTP authentication
  - Authentication form using either HTTP or HTTPS. The form is device dependent. It can be HTML, WML, … It is also possible to brand it for virtual ISPs.

The IBM Everyplace Authentication Server will supply a username in the HTTP header as described in 9.5.1.1, “Identification” on page 158, so the DefaultAuthenticator will never have to reach Stage 2.

It implements two authentication modes:

- Multi-domain mode - the authentication process is distributed

  Each content server (portal, self care, premium) is protected by the authentication checker. There is one (or several) central authentication servers that receive authentication requests from checkers through HTTP redirects. When the user is authenticated, he is automatically redirected back to the content server.

- Single-domain mode - the authentication process is entirely done in the checker

  It does not use redirects. This mode is good for limited devices (such as some WAP mobiles not correctly supporting HTTP redirects), or when the authentication process is delegated to an external component such as the WES Authentication Proxy.

The single-domain mode can be faster but has limitations:

- Single sign-on is not supported
- Form authentication stage 2 is not supported.

9.5.2 Premium Content

Application-based access control is offered by TPSM Premium Content. The three key terms used are:

Tier	Represents a single collection of protected content; an “offering”.
Pattern	Represents a location of content described by its URL with wildcard allowed. At least one Pattern is required for each Tier.
Deal	The offering to be sold including a number of Tiers. Deal is defined in 9.2, “Business and data model” on page 142.

Basically this allows you to group and offer content based on its URL. The protection relies on the TPSM Authentication component to be invoked from every protected JSP. Setup and configuration of Tiers, Patterns, and Deals is done using TPSM Director.

9.5.3 RADIUS server

The RADIUS server component of TPSM serves two major purposes; authentication and accounting. The RADIUS protocol was originally developed for use by Network Access Devices (NAS), which is reflected in its name Remote Access Dial In User Service. It’s a widespread protocol for modems to authenticate user names and passwords of dial-in users when they attempt to connect. If the user credentials are valid, and the account is open, the RADIUS server will allow the session to begin.

Another important use of the RADIUS protocol and server is accounting. The NAS will inform its RADIUS server of both the start and end of a connection. Having this information, the server is able to track NAS usage for individual users. Today tracking and accounting by time connected makes sense, because the "line" is a limited and relatively expensive resource. But note that RADIUS is not made for traffic or package-based accounting that could have been used for a permanent network connection.

In the Everyplace Suite the IBM Everyplace Wireless Gateway has the role of a network access device, and uses the TPSM RADIUS server for authentication and accounting. Also the IBM Everyplace Authentication Server is having its users authenticated by this. However, accounting cannot be done in here, as there is no way for the Authentication Server to determine when an HTTP connection has come to its end. The Authentication Server does signal end-of-connection, but it is based on a time-out function and is not appropriate for billing.

9.5.4 Active Session Table server

To provide single sign-on within the Everyplace Suite, TPSM has been equipped with a highly dynamic database table, 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 Suite components.

When a new session is authenticated by either IBM Everyplace Wireless Gateway or IBM Everyplace Authentication Server, a unique record is created in the Active Session Table (AST). For subsequent requests, the record will provide all necessary information about credibility, the user, the device and network used:

- Sessions originating from the Wireless Gateway is authenticated and allocated a trusted IP address and/or an HTTP header identification. When the request passes the Authentication Server, it will use that as the key for AST lookups and provide the full set of Everyplace Suite headers.
- Session originating from an HTTP/IP connection via Authentication Server will be assigned a unique session ID placed in an HTTP cookie. The ID is the key used in AST.

The Active Session Table server is highly specialized and optimized to have minimal impact on system performance.

9.6 Personalization services

The first chapter of this book described the evolution of e-business and the increase of requirements in modern solutions. One of the central requirements is personalization. The vast majority of service applications and Web sites by using personalization will increase their value to both their users and owners. Examples could be a personal look and feel, targeted advertisement, or personalized applications. Any content can be personalized in any way, but in order to focus the discussion, we here provide a list of suggested criteria for personalization and examples of use.

Applications and services can be personalized based on the users:

- Explicit preferences, such as news categories selected by manually entered interests, or a shopping catalog displaying toys appropriate for the user's age entered.
- Implicit preferences, such as targeted banner ads based on a user's behavior by navigation on the site and purchases done.

- Business engagement, such as utilizing a customer's track record from existing Customer Relationship Management systems to show relevant up-sales or service information.
- Geographical location, such as switching site language based on nationality of the user's IP-name, or location known from the origin GSM cell[3] when using a WAP cell phone.
- Access device used, such as adapting the markup language and page layout to small screen devices.

Implementing personalization involves four areas:

User Identification	This can be done by explicit login, or transparently by a cookie, or by characteristics of a personal device (cell phone number, PDA serial number, or similar).
User Profile	A repository must exist for storage of implicit and/or explicit user information.
Content Selection	The discipline of matching the user with content. It ranges from a few simple lines of code to recommendation engines based on artificial intelligence.
Content Assembly	The process of putting together the personalized "page".

9.6.1 Personalization in the Everyplace Suite

9.6.1.1 Identification

The Everyplace Suite provides different types of user identification through the authentication done by the IBM Everyplace Wireless Gateway and the IBM Everyplace Authentication Server. Although TPSM also offers built-in authentication, its responsibilities are partly delegated to the single sign-on features of the Everyplace Suite.

9.6.1.2 User Profiles

The user profiles are stored and maintained by TPSM. The database schema defined for profiles is extendible to fit most specific needs.

9.6.1.3 Content Selection and Content Assembly

Everyplace Suite provides content selection and content assembly in a combination supplied by the TPSM Portal Toolkit (pTk). In addition to a programmable framework for personalization, the pTk also offers a set of JSP application components that can be used to provide additional services in a portal solution or any other solution. Finally, TPSM pTk includes an interface to the Ad Server produced and run by Double Click offering complete advertising campaign development and deployment.

The TPSM framework and its template applications use explicit user preferences for personalization. The framework is open and easy to extend for use of implicit preferences, business engagement, and geographical location information. For example, engines for rules and recommendations can be interfaced to the TPSM database and components, results of behavior tracking can be stored in the TPSM profiles, CRM systems can be interfaced, and so on.

[3] Knowing just which GSM cell is used will determine the location within 25-50km, but in cooperation with an operator the location *inside* the cell can be determined giving impressive accuracies, for example 100m. Refer to the relevant operators and the proposed WAP 1.2+ standard.

Everyplace Suite offers device-based adaptation by the IBM WebSphere Transcoding Publisher. Supplemental to this[4], device adaptation at application level can be developed using pTk. Information of the device type used is supplied in the HTTP request header by the IBM Everyplace Authentication Server.

9.6.2 Overview of pTk

The portal toolkit consists of a number of Java APIs and services, which will help development of personalized applications serving any markup language (HTML, WML, XML, etc.) using JSPs or Java servlets.

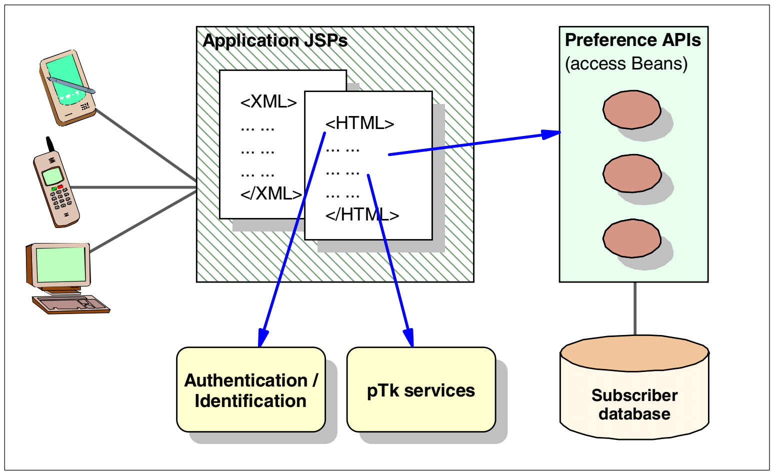


Figure 92. pTk overview

As illustrated in Figure 92, pTk offers services that are linked to or included in the JSPs of a business application. Alternatively, a servlet can use the pTk by calling the services and APIs.

9.6.3 Preference API

The key element of TPSM pTk is the persistent preference class hierarchy. Subscriber data is kept in the TPSM Subscriber database, and is accessed using the Access Beans. A number of data types are used, each mapped to a Java class.

[4] Refer to the discussion on automated versus programmed adaptation in 8.3.2, “Application design” on page 124.

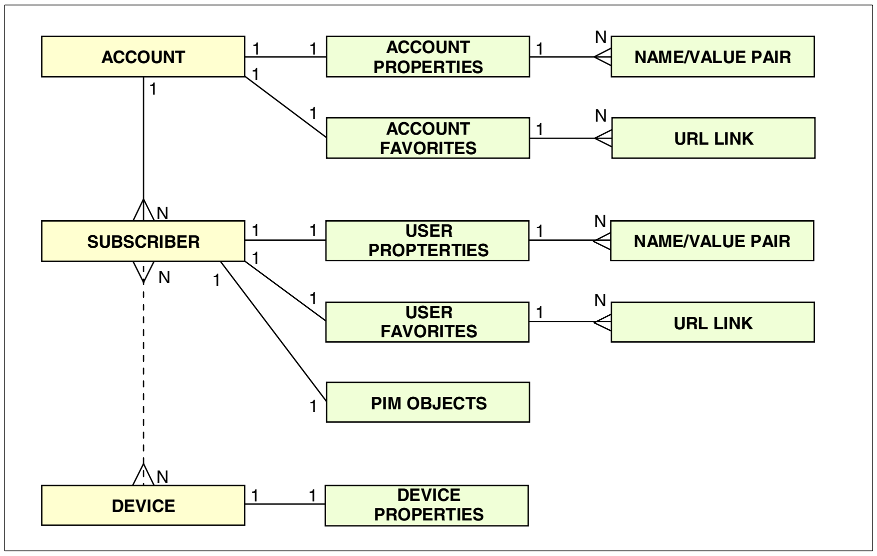


Figure 93. pTk preference data types/class

As the structure in Figure 93 shows, the preferences are tied to the three core data types; Account, Subscriber (or its Child), and Device (if Device Manager is installed). Properties are implemented as a number of name/value pairs for general use, while the specialized type “Favorites” consists of number of URL links organized in groups. Both Accounts and Subscribers can be assigned a set of Properties and a set of Favorites.

For the Subscriber (or its Child) a set of Personal Information Management (PIM) objects exists. These are used for storing data for the pTk components such as a personal address book or calendar. See 9.6.4, “JSP components of pTk” on page 165.

Devices are associated with a set of properties, which are also accessible using the pTk APIs. Device Manager must be installed before you can use the device class.

Using the Preference API

We include two pieces of JSP code to give a feeling of how the Access Beans are used to get and update preferences. Any JSP using the preference API should:

1. Include an import statement to make the classes available
2. Instantiate and populate beans using the getBean(request) method (not the BEAN tag)
3. Use access beans get/set methods

```
<%
User user = User.getBean(request);
NameValuePairs pairs = NameValuePairs.getBean(request);
FavoriteLinks favLinks = FavoriteLinks.getBean(request);
%>

<H1>Welcome <%= user.getFirstName() %></H1>
```

Figure 94. Preference API example: getting a user preference

```
<%
User user = User.getBean(request);
NameValuePairs pairs = NameValuePairs.getBean(request);
FavoriteLinks favLinks = FavoriteLinks.getBean(request);

favLinks.add(new Link("Business", "IBM Corp.", "http://www.ibm.com/"));
favLinks.update(request);
%>
```

Figure 95. Adding a URL link, and forcing a database write

9.6.4 JSP components of pTk

The JSP Components are introduced in TPSM Version 1.1. It's a set of building blocks that will allow rapid development. It also enforces a simple hierarchical JSP structure.

A component consists of a servlet and a number of JSPs for different "rendering" of output, for example to HTML and WML. It's intended to be included in the JSPs of an application as illustrated in Figure 96.

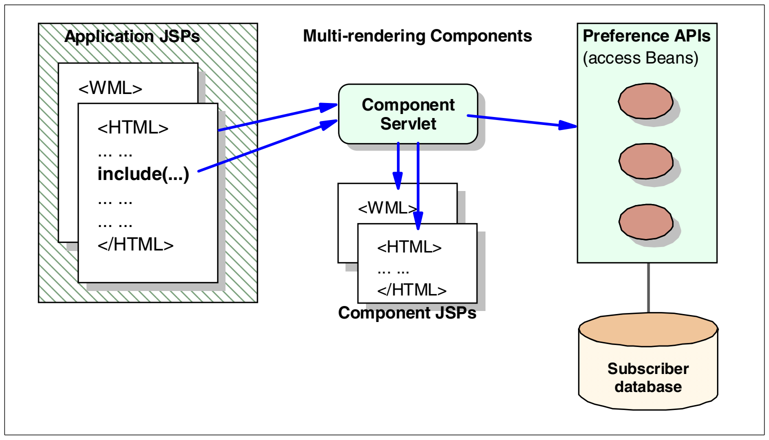


Figure 96. JSP components architecture

The JSP components are stateful, and some are persistent using the Preference API to store persistent information in the TPSM database. To effectively share common functions and characteristics, the components inherit from abstract classes. Their full object hierarchy is shown below.

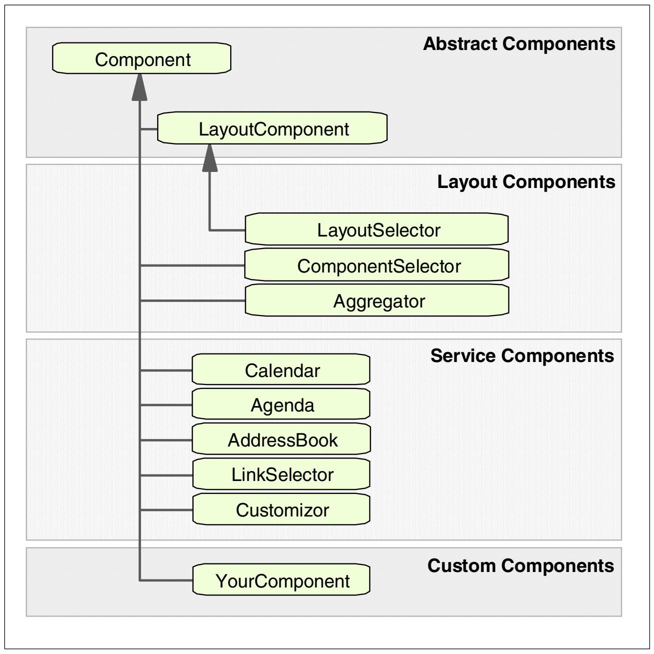


Figure 97. JSP Components hierarchy

Two types of components are delivered and deployable: Layout Components and Service Components. Let's have a brief look at their functions:

Layout Components

- LayoutSelector: Helps find the right JSP for the requesting device type. It is invoked directly, and hence not included in a JSP page.
- ComponentSelector: A container for applications of the user's interest (such as a "desktop").
- Aggregator: Will fetch and aggregate static content into a page.

Service Components

Service Components are "turn-key" applications or services within portal pages. You can develop your own service components, or use the default ones:

- Calendar: For display and navigation in weekly or monthly calendars. Multi-rendering enabled: HTML, WML, XML.
- Agenda: Display and manipulate user appointments, for one day or one week.
- AddressBook: Application to save and retrieve e-mail addresses.
- LinkSelector: Displays and manages URL links.
- Customizor: Lets the user specify any parameters for any set of forms.

Using the JSP components

To use a component from a JSP or a servlet, only a single line of code is required in the simplest example:

```
<%
Component.include (componentURL, instanceName, this, request, response);
%>
```

Figure 98. Generic statement for including a component

9.6.5 AdServer integration

TPSM enables you to target subscriber's pages for banner ads and special-interest notices, based on the information in the subscriber database. The ad services from Double Click are used to manage the campaign and supply banner graphics with a link.

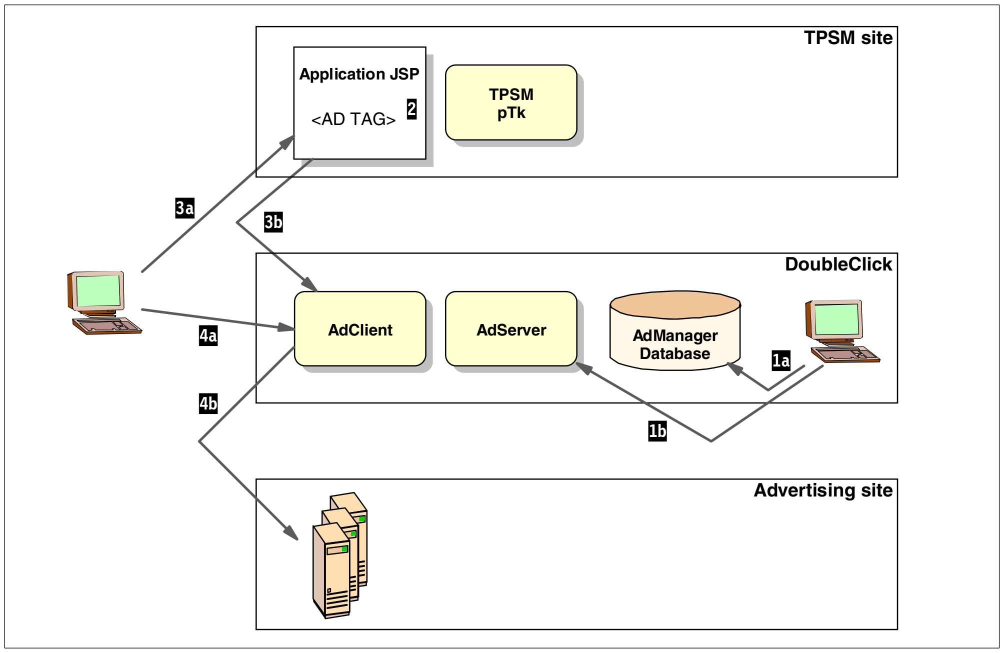


Figure 99. Integration with AdServer

The interaction is illustrated in Figure 99:

1 Advertising campaign is defined and then deployed with AdManager from Double Click.

2 Insert “Ad Tags” in the customized JSP pages.

3 The banner is displayed, having its graphics source at a Double Click Web server.

4 When the banner is selected by clicking it, the link leads the user to the advertising site.

Examples of the code fragments used are shown below.

```
<BEAN name="user" type="User" create="yes"></BEAN>
<%
user.read(request);
%>
<a href="http://ads.ibm.com:9000/click.ng/site=isp30&
gender=user.getGender()&age=user.getAge()">
<img src="http://ads.ibm.com:9000/image.ng/site=isp30?
gender=user.getGender()&age=user.getAge()">
</a>
```

Figure 100. Get user data with pTk Access Beans and make banner

9.7 Integration and provisioning

Today's key measures are openness and standards. No system in any business is expected to work in isolation; it must be easily extendible and easy to integrate.

The TPSM framework is based on widely accepted standards such as Java, and it provides a rich set of "APIs" for integration. The TPSM Integration Toolkit *(*iTk) consists of:

iTk Core	Provides database access with validation and utilities.
iTk Business Objects	Represent a subscriber, a deal, an account, etc.
iTk Provisioning	Event notification to external application.
iTk Billing	Interface for use by external billing systems.

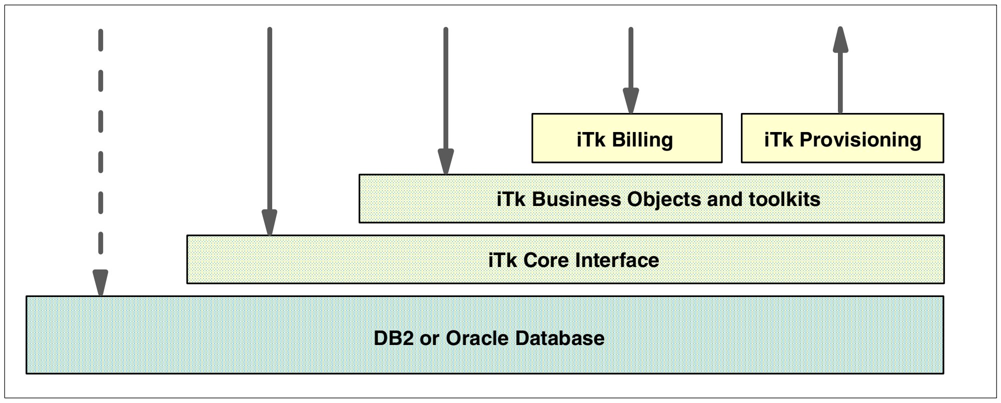


Figure 101. iTk parts in TPSM

The parts of iTk have different roles and dependencies. As illustrated in Figure 101 interaction can be done with each iTk part providing different levels of abstraction.

Together the iTk Core and Business Object interfaces form the foundation for all other TPSM components. Using JDBC, the Core interface provides abstraction of the underlying database implementation and infrastructure. This relieves a lot of potential pain for iTk users; what they see is just a clean Java-based interface.

If desired, the iTk can communicate using Java RMI. This allows the "client" and the "server" part of iTk to run in separate JVMs, which could be different machines. The performance penalty introduced by RMI is in the range of 10-30% depending on the number and types of transactions done.

The following sections will give a brief overview of the iTk parts.

9.7.1 iTk Core and Business Object interface

Basically, the iTk Core interface provides an interface for database transactions at a general low level. Two important functions are:

- To hide the underlying database (and any vendor-specific stuff)
- To provides database connection pooling for optimal response time

The iTk business objects are often the preferred level for interaction. There are objects to represent practically all the data types residing in the TPSM database; Realm, Access Code, Account, Subscriber, Child, Deal, Method of Payment, Properties, and so on.

Methods are provided to create, read, update and delete the objects. Of course both data integrity and user authorization checks are carried out before any operation is done. Validations available range from simple “stand-alone” checks, such as a number to be within a specified range, to complex validations involving a third party, such as a credit card account check.

The iTk Core interface and business objects can be invoked from TPSM applications such as Self Care, from any application JSP or servlet or from other systems.

9.7.2 iTk Provisioning

Provisioning allows external applications to register to TPSM in order to be notified of TPSM events such as subscriber enrollments, subscriber updates, account creation, account updates, etc. That allows the external applications to get TPSM updates for each event and keep the transaction status in the TPSM database.

This is very useful for maintaining subscriber or device data outside TPSM, such as in a customer database, or an LDAP Directory. Provisioning can also be used for more transactional events, such as sessions triggering external events.

In particular if there are large amounts of data in the TPSM database, provisioning is a much more effective than database level replication, if you need to do mirroring of one or more data sets.

9.7.3 iTk Billing

There are several options for billing with TPSM:

- The Billing iTk provides linkage from external billing systems to TPSM.
- The Provisioning iTk provides linkage from TPSM to external billing systems.
- The other iTks provide access to TPSM information that may be used by external billing systems.
- eBill is a lightweight billing engine that is integrated with the TPSM database. It can be used to provide simple billing functions.

Chapter 10. Pervasive messaging and queuing

This chapter discusses messaging and queuing. It introduces the IBM MQSeries products and the programming style, *messaging and queuing*, that underlies them, and describes the latest addition to the IBM MQSeries family of products, IBM MQSeries Everyplace, and how it integrates with Everyplace Suite.

If you are already familiar with messaging and queuing concepts and the IBM MQSeries family of products then you may wish to go straight to "MQSeries Everyplace" on page 176.

Briefly, messaging products enable programs to talk to each other across a network of unlike components - processors, operating systems, subsystems, and communication protocols - using a simple programming style called messaging and queuing: *messaging,* because programs communicate by sending each other data in messages rather than by calling each other directly, and *queuing,* because the messages are placed on queues in storage, so that programs can run independently of each other, at different speeds and times, in different locations, without needing a logical connection between them.

10.1 Introduction to messaging and queuing

There are three general styles of communication available for application-to-application communication; *conversational*, *remote procedure call* (RPC) and the *messaging and queuing* styles.

Conversational communication is where one application sends some data over a network to another application that is monitoring that network, and then waits for a response from its partner application via the network. Since the partners processing may take some time, the application may have a lengthy wait.

RPC is where an application calls a procedure on another system, and waits for the response to be returned. The application is typically isolated from the location of the procedure, and will probably be unaware that the procedure is not part of its own local application.

This two styles are generally considered to be *synchronous* styles of communication. They require all the components of the environment to be available before data exchange (and hence processing) can occur. Messaging and queuing on the other hand is an *asynchronous* style of communication, which does not depend on the availability of all its distributed components for processing to continue on any one system.

Typically, synchronously communicating applications are network-aware; they contain network-specific code, and must take into account the network, making and monitoring connections, and managing network or partner application failures. This generally means an application becomes very network dependent, making it difficult to accommodate changes. In addition, to form long-running conversations usually requires complex programming, increasing the difficulty of program development, and detracting from the real business logic.

On the other hand, asynchronously communicating applications tend to be decoupled both from each other and the network. There is no longer a direct connection between the communicating applications, and they are typically not network-aware. This means that the application does not need to be aware of setting up or using connections, handling network or partner failures, etc. The application can concentrate solely on business logic, which simplifies and speeds application development. As an additional benefit, if the underlying network changes (for example, from SNA to TCP/IP), then no change will be required in the application.

10.1.1 Messaging with IBM MQSeries

IBM MQSeries is centered around the idea of time-independent and connectionless communication of messages between distributed application components. A message is merely a collection of data - a string of bits and bytes that have meaning to a particular application - that is formatted by that application to convey that meaning in a recognizable fashion. Messages are exchanged between parts of a distributed application by placing the message on a queue. The application can be designed to use any number of queues, and will generally use each queue for a distinct purpose, or to convey a distinct type of message.

Generally, each queue is used to flow information from one part of the application to another part of the application in only one direction - one part of the application will put messages on a queue, and another part of the application will get messages off the queue. Two queues will usually be used to provide two-way communication between two parts of an application.

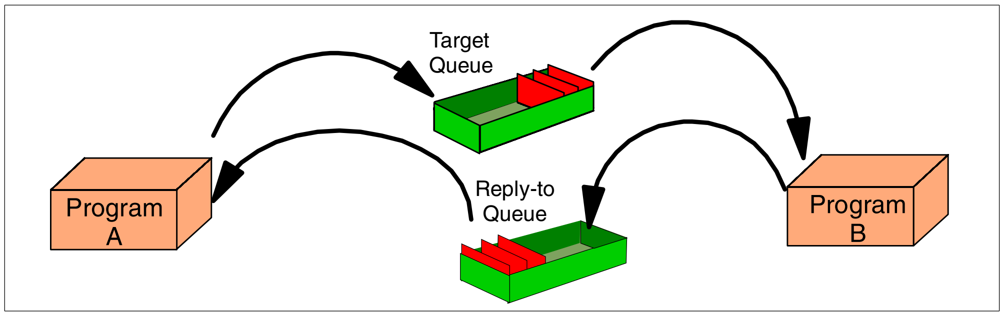


Figure 102. Two-way communication under a messaging and queuing paradigm

Queues are defined within, and controlled by, a queue manager. They can be created either administratively or programmatically by the applications that will use them. To access a queue, it is necessary to first connect to the queue manager that the queue is defined within. Typically it is possible for more than one queue manager to be created on the same host machine - in fact, different applications may use different queue managers to create, define, and control their queues if they wish. However, in general, a program can connect to only a single queue manager at any given time.

Queue managers on the same or different host machines can be interconnected (an administrative process) to allow messages to be transferred between queues on the different queue managers. This is a process known as distributed queuing, and requires that the queues be defined differently, causing the queue manager to send their messages over these connections. However, the applications that use these queues are able to remain unaware of these differences, and hence that they are even participating in distributed queuing.

Consequently, when operating within a distributed queuing environment the different parts of an application continue to see the same interface as in a local environment: their in-bound and out-bound queues. They are unaware that these queues now redirect their messages around the network in a time-independent fashion, and indeed are unable to even determine the final destination of their messages with any certainty.

A fundamental characteristic of message-queue based communication is that it places the focus on the flow of information rather than the flow of control. Unlike call-based communication models, the “requesting” program is not blocked while work is being performed. Rather, the requesting program merely puts information to a queue and then resumes with its own work. It has sent on the information that it has produced, but otherwise is ignorant of where, when, or how that information will be actually sent and processed.

In this way, the program that sends the information is completely independent of the program that receives and processes it. The receiving program can be busy at the time a message is put on the appropriate queue. The fact that a message has arrived doesn’t affect the receiving program’s current processing. In fact, the receiving program doesn’t even have to be running at the time the message is put on the queue. The receiving program can start running three hours or three weeks later, if that suits the business need. It could then even reformat the information and send it to several other applications via other queues.

The simple fact is that the sending program doesn’t have to know what the receiving program is, where it resides, or even what it does with the data that it is being passed.

10.1.2 Why messaging rather than browsing?

Browsing technologies (such as HTTP and typical deployments of WAP) and commercial messaging systems (such as MQSeries) each have their own advantages and disadvantages. Fortunately they tend to be very complementary technologies, and so IBM has provided both technologies in the Everyplace Suite, providing the solutions architect the maximum flexibility when deciding how to tackle any given problem.

When examining any given problem one of the two technologies will tend to be more appropriate, and the other less appropriate. Generally, it will always be possible to use either technology for any problem, but one will always tend to be a more natural choice,

So, how do you decide where to use MQSeries rather than a browser-based approach? The business problem, combined with the functional and non-functional requirements must be the ultimate arbitrator, but there are indicators that may tend to indicate that you should consider messaging rather than a browser approach.

Typically the advantages of MQSeries Everyplace are greatest in an environment where connectivity is either unavailable or intermittent - messages are stored until the connection is restored, the user is assured that the messages will be delivered, and delivered once only. In contrast, browsing technologies do not work well in this situation - when the connection is not available, they cannot operate. Also, if the connection fails during a transaction, resources and data can be left locked, and the end user can be left in doubt as to whether the request has been dealt with.

10.2 The MQSeries family of products

MQSeries is actually a family of products that implement and build on commercial messaging. Originally there was only MQSeries messaging, which provided heterogeneous connectivity, then as the product evolved, bridges to various other operating environments were added. Since then the product set has evolved further through the addition of message brokering and business integration technologies such as workflow support.

These products are often represented as a three-leveled pyramid (see Figure 103), to represent the increasing value that each can add to a business.

We will briefly introduce each product in turn in the following sections. However, if you require more detailed information on any of the MQSeries family of products, please visit the MQSeries Web site on the Internet, `http://www.software.ibm.com/ts/MQSeries.html` where product manuals, whitepapers, marketing materials and code samples are all freely available.

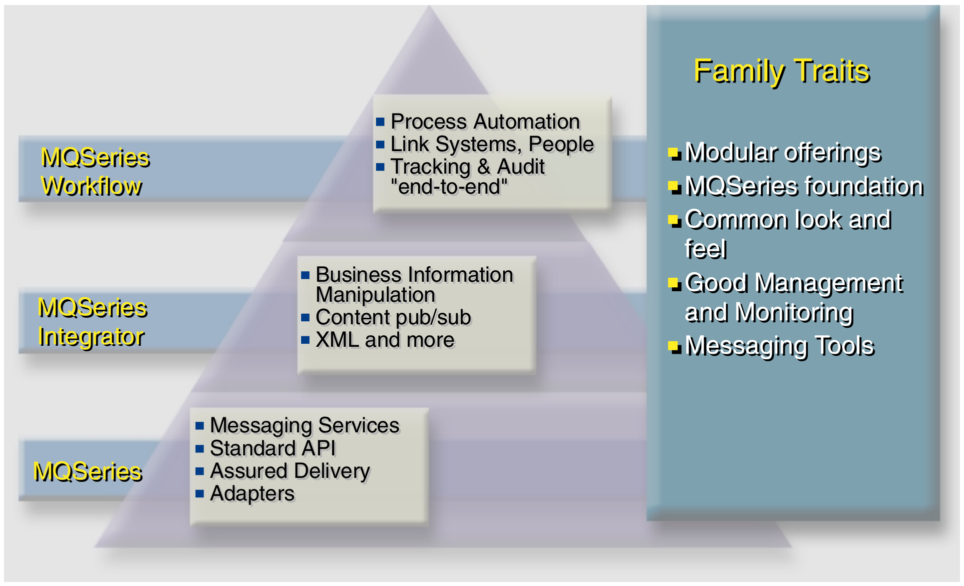


Figure 103. The MQSeries family of products

Finally, please note that the pyramid shape should not be taken to imply any specific relationship between the products; while both MQSeries Integrator and MQSeries Workflow are built on MQSeries messaging, and specify it as a prerequisite, they do not place corequisites on each other. It is perfectly acceptable to build a solution around MQSeries Workflow, without including MQSeries Integrator in that solution.

10.2.1 MQSeries

MQSeries is a commercial messaging system that provides assured, asynchronous, once-only delivery of data across a broad range of hardware and software platforms. It supports all the important communications protocols, and provides a means to route data across networks that use different, incompatible protocols. The assured nature of the messaging means that MQSeries can form the basis of critical communications infrastructures, and be safely entrusted with the delivery of high-value data.

Through the provision of various bridging and gateway products, MQSeries allows simple access to many standard systems and application environments, including Lotus Notes, Web browsers, Java applets, CICS, IMS, and SAP.

MQSeries can be deployed in a variety of ways to create many different solutions. It is possible to build a simple communications infrastructure, taking advantage of the ability to operate across multiple platforms and networks, or to connect many existing systems together into an integrated whole, allowing new applications to access those applications (perhaps via an Internet browser).

MQSeries is the world's leading commercial messaging system, currently holding two-thirds of the market. It is the defacto standard for messaging on more than 35 platforms, and has been deployed by more than 5000 customers world-wide.

It is the basis for the remainder of the IBM MQSeries family.

10.2.2 MQSeries Integrator

MQSeries has helped to popularize heterogeneous asynchronous messaging, and today MQSeries applications can communicate with other applications located anywhere in a customer's enterprise. Indeed, it is even possible for applications to exchange data with applications from other companies. As customers continually integrate different applications, data sources, and sometimes even entire systems to gain competitive advantage, they raise massive challenges for their IT departments.

As the number of data sources and sinks increases, the number of interfaces that have to be managed, maintained, and enhanced becomes exponentially larger. Strategies that rely on making all the application systems compatible with each other are doomed to failure by simple mathematics.

So, as customers increase the complexity of the tasks that they undertake, putting their data to more productive use, they discover a need to process and transform the message data as it passes through the MQSeries network, before it reaches the next MQSeries application. The purpose of a message broker is to carry out this transformation and processing of messages that are en-route to their destinations.

MQSeries Integrator is powerful message-brokering software that automatically distributes information to those applications that need it, dramatically simplifying the connections between different applications. Using MQSeries for transporting between different computing platforms, MQSeries Integrator routes information according to enterprise-defined rules, transforming and reformatting it to suit the receiving application.

This protects a customer's investment in their existing applications, and allows them to remain focussed on using MQSeries family messaging technologies to transform their business processes, without constantly enhancing and updating their existing applications.

10.2.3 MQSeries Workflow

Business processes are among a company's most valuable assets. They encapsulate all the business experience of the enterprise, and their definition is key to the company's success. However, as companies grow, their processes become more complex, and planning and managing activities, resources and applications become more challenging.

IBM MQSeries Workflow is a workflow management system that helps an organization to optimize its business processes. The organization uses MQSeries Workflow to completely define, refine, manage, and execute its business processes.

While the organization focuses on its key business objectives, MQSeries Workflow manages the processes through the execution of software whose order of execution is driven entirely by a computer representation of the workflow logic. MQSeries Workflow makes it easier for the organization's staff to work effectively within its processes. They receive the information they need, when they need it, and they can work with user-friendly front-end applications that connect seamlessly to core business applications. This automation of processes results in a better level of service, higher productivity, and simpler interaction between people and applications, and between distributed applications across multiple platforms.

10.3 MQSeries Everyplace

IBM MQSeries Everyplace is the latest member of the MQSeries family of products, joining the set of base commercial messaging products.

It is designed to satisfy the messaging needs of lightweight devices, such as sensors, phones, PDAs and laptop computers, as well as supporting mobility and the requirements that arise from the use of fragile communication networks. It provides the standard MQSeries quality of service (that is, once-only assured delivery) and can exchange messages with other MQSeries family members. Since it is expected to be deployed outside the corporate intranet, it also provides sophisticated security capabilities.

Lightweight devices require the messaging subsystem to be frugal in its use of system resources, and consequently MQSeries Everyplace offers tailored function and interfaces appropriate to its customer set; it does not aim to provide all the capabilities offered by other members of the family. On the other hand, it does include a number of unique capabilities in order to support its particular classes of user, such as comprehensive security provision and object messaging together with a rich set of messaging functions.

10.3.1 Key considerations

MQSeries Everyplace is *not* like standard MQSeries. The skills that are gained from working with standard MQSeries (beyond high-level architectural knowledge of messaging and queuing) are *not* directly transferable.

MQSeries Everyplace is a Java toolkit. To do anything serious with this product requires a significant amount of Java programming. Samples are supplied with the product, but in real terms it is necessary to write Java applications before you can even create a queue manager, let alone move messages.

The manuals supplied with the product are excellent. You are strongly advised to read them thoroughly to bridge any skills gaps before you commence designing or implementing solutions using this technology. We believe that the amount of flexibility and number of configuration options that are provided by the product make this an essential step.

Tracing and error handling are entirely under the application programmer's control. If you choose not to implement any, then you will not get any. This is a double-edged sword - it can improve the performance and minimize the size of your applications, which is good for deployment on pervasive devices, but it can (and will) make debugging any problems extremely difficult after deployment.

You may have multiple queue managers on a system, but you may only have a single queue manager within any given *Java Virtual Machine* (JVM).

10.3.2 IBM MQSeries and IBM MQSeries Everyplace

In many ways MQSeries Everyplace is very similar to standard MQSeries. Messaging is carried out using queues, which are maintained within queue managers. Applications communicate with each other by connecting to a queue manager, and get or put messages to a queue. Messages that are put to a queue on a remote queue manager are sent over the network using channels, and may travel through one or more intermediate queue managers, just as in standard MQSeries.

The essentials of messaging and queuing are all present and fully supported. MQSeries Everyplace extends the scope of the messaging members of the MQSeries family:

- It expands messaging capabilities to the set of low-end devices, such as PDAs, telephones, and sensors, allowing them to participate in an MQSeries messaging network. MQSeries Everyplace offers the same once-only assured delivery and permits the two-way exchange of messages with other members of the family.
- It is designed to operate efficiently in hostile communications environments where networks are unstable, or where bandwidth is tightly constrained. Thus it has an efficient wire protocol and automated recovery from communication link failures.
- It supports the mobile user, allowing network connectivity points to change as devices roam. It also allows control of behavior in conditions where battery resources and networks are failing or constrained.
- It minimizes administration tasks for the user, such that the presence of MQSeries Everyplace on a device can be largely hidden - making MQSeries Everyplace a suitable base on which to build utility-style applications.
- It includes extensive authentication and encryption facilities, making it suitable for applications outside firewalls.

However, in other ways MQSeries Everyplace is very different. Whereas standard MQSeries is typically shipped as a full product, which is then installed and configured, MQSeries Everyplace is supplied in a form more akin to a Java toolkit[1]. As a consequence, it is a completely object-oriented implementation, rather than being object-oriented wrappers over a procedural base, as is the case with the existing MQSeries products.

[1] Apart from the Palm version, which is shipped as a Palm application, written in native C code.

However, as a toolkit you have to program to even create a queue manager. Samples are supplied, but these are just that - samples - they are not part of the main product, and should not be relied on.

10.3.3 Architectures

The most important feature of MQSeries Everyplace is that it may be configured in many different ways. Much flexibility is provided, allowing you to create queue managers with or without queues, listeners or channel managers. The primary configurations we will consider are where the queue manager either has a listener or channel manager, or not. If it does not have either, then it is classed as a device configuration. If it has, then it is classed as a gateway configuration.

Device configurations combine many of the attributes of MQSeries clients and MQSeries servers. For example, they can access queues held remotely, a feature normally only present in MQSeries clients, but if local queuing has been enabled then this configuration will also be able to store messages locally. If local queuing is not enabled then this is known as an MQSeries Everyplace stub queue manager, and is directly analogous to an MQSeries Client. Clearly, given the name, we expect device configurations to be particularly popular for deployment on pervasive devices.

Device configurations can communicate directly with each other, offering a peer-to-peer messaging capability, or through *dynamic channels*, which are different from MQSeries client and messaging channels. Dynamic channels are bi-directional and support the full range of functions provided by MQSeries Everyplace, including both synchronous and asynchronous messaging. However, device configurations can only accept at most a single incoming connection.

They are typically deployed as either *clients* talking to a server, or as *peers* in a peer-to-peer configuration. See (b), (c) and (d) in Figure 104 on page 178 for examples of these deployment configurations.

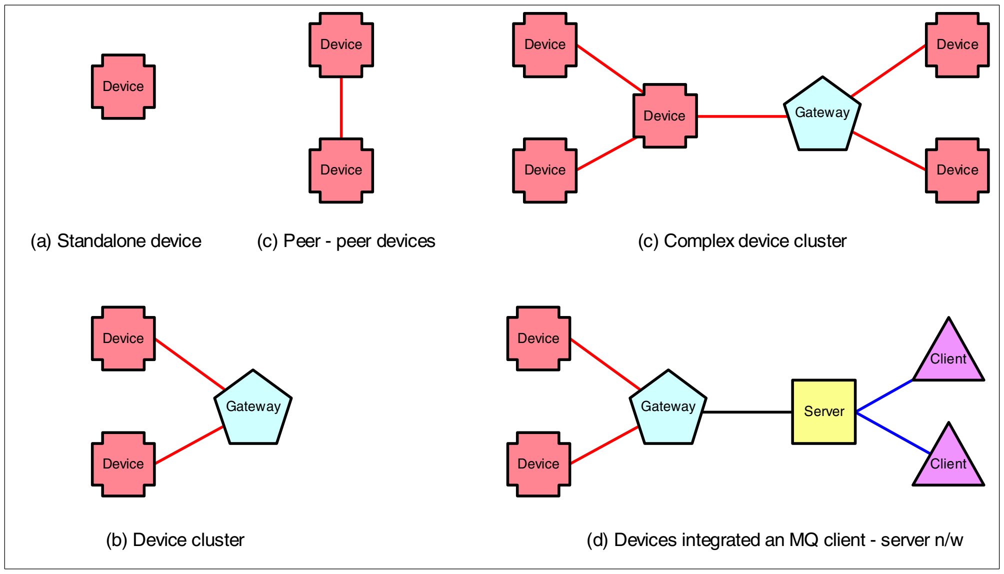


Figure 104. Possible MQSeries Everyplace deployment configurations

Gateway configurations on the other hand, will probably always have local queuing enabled, and can accept multiple incoming dynamic channels to enable them to support device configurations, and may be configured as either a *server* or a *servlet*. Gateways are used to attach MQSeries Everyplace device configurations. Gateways are also the mechanism through which MQSeries Everyplace devices are attached to an MQSeries network, as they can optionally support an MQSeries client channel, allowing them to communicate with an MQSeries server. See (e) in Figure 104 on page 178 for an example of a bridged configuration.

The key difference between the server and servlet implementations is that a servlet needs the features of an HTTP server to provide its channel connection features, whereas the server does not.

In summary:

Client:	Can initiate connection requests
Peer:	Can initiate connection requests Can accept a single connection request
Server:	Can accept multiple connection requests May provide a bridging facility to standard MQSeries
Servlet:	Can accept multiple connection requests The Listener function is provided by an HTTP server May provide a bridging facility to standard MQSeries

10.3.4 Bridging to MQSeries networks

MQSeries Everyplace provides seamless connectivity to MQSeries networks through gateway configurations. Messages flowing through these interfaces will be automatically translated from the highly optimized form used by MQSeries Everyplace to the normal MQSeries message formats, and vice versa.

When combined with the comprehensive security provided by MQSeries Everyplace, this allows existing MQSeries enterprise systems to extend their reach out of the enterprise network to highly mobile, pervasive systems.

This in turn provides the ability to integrate new classes of pervasive computing devices into the core enterprise applications, participating in the *Enterprise Application Integration* (EAI) solutions that are currently being enabled by the use of MQSeries, MQSeries Integrator and MQSeries Workflow.

10.3.5 Everyplace Suite considerations

At the current time, MQSeries Everyplace and Everyplace Suite are not tightly integrated in a physical, product sense. This is largely because most of Everyplace Suite is directly targeted at browser-based technologies, an arena where MQSeries Everyplace does not directly play.

They are, however, integrated in the sense that they are provided as part of the same product suite. This is because many pervasive solutions are not ideal candidates for browser-based technologies, and due to the complementary natures of Web browsing and messaging and queuing, it provides the additional architectural alternatives that are often required.

Key areas where you will need to pay particular attention to when integrating pervasive messaging into an Everyplace Suite solution are centered around the differing security models, the mechanisms used for providing scalability and availability, and the entry points for MQSeries Everyplace traffic into your chosen architecture:

- The Everyplace Suite security model is very distinct from the model employed by MQSeries Everyplace, and the two are not directly compatible. We expect there to be a series of initiatives to better integrate MQSeries Everyplace security with WES over time. However, both are sufficiently strong to stand on their own.
- The Everyplace Suite depends on extensive use of the Caching Proxy and Load Balancer to provide scalability and availability. However, MQSeries Everyplace is not ideally suited to either of these products because of its inherent difference from an Internet browsing technology. We recommend instead that you investigate using technologies such as HACMP to provide appropriate levels of availability.
- MQSeries can be introduced into your environment through the Wireless Gateway, using either TCP/IP or HTTP as a bearer, or via some other network entry point using either TCP/IP or HTTP as the bearer.
  Although using TCP/IP provides better performance, using HTTP as your bearer will automatically provide your MQSeries Everyplace traffic with a degree of authentication that will allow it to integrate with the Authentication Server.
  Using this scenario you can position your MQSeries Everyplace gateway servers behind the Authentication Server, providing them with an additional degree of WES security, and positioning yourself for future integration of MQSeries Everyplace and Everyplace Suite security models.

Part 3. Optimization

The IBM WebSphere Everyplace Suite includes the WebSphere Edge Server for traffic optimization. In this part of the redbook we provide an overview of the architecture and system design features provided in the two components included in the WebSphere Edge Server:

- IBM WebSphere Traffic Express (WTE), the caching proxy. It provides functionality of a proxy server and a caching proxy and can also act as a filter.
- The IBM Load Balancer. Formerly called the eNetwork Dispatcher, it provides dynamic load balancing for performance, availability and scalability.

Chapter 11. Caching Proxy

This chapter provides an overview of the architecture and system design features of the Caching Proxy component of IBM WebSphere Edge Server, that can be used within WES.

A discussion of the functionality of the Caching Proxy is limited to the context of this book. Details on the full functionality and configuration settings, provided in the Caching Proxy, can be found in the following resources:

- The online product documentation
- *WebSphere Edge Server for Multiplatforms Getting Started Guide Version 1.0,* SC09-4566
- *WebSphere Edge Server for Multiplatforms Web Traffic Express User's Guide Version 1.0* , GC09-4567
- *WebSphere Edge Server for Multiplatforms Web Traffic Express Programming Guide Version 1.0,* GC09-4568
- `http://www.ibm.com/software/webservers/edgeserver/library.html`

Changing names

The Caching Proxy component of the IBM WebSphere Edge Server for Multiplatforms V 1.0 is the new offering from IBM to deliver performance and scalability.

The Edge Server includes enhanced versions of both the caching proxy and load balancing components of IBM WebSphere Performance Pack V3.0.

IBM WebSphere Performance Pack V3.0 includes:

- IBM Web Traffic Express V3.0
- IBM Network Dispatcher V3.0
- IBM AFS Enterprise File Systems V3.5

The first two components have been enhanced and re-branded to produce the IBM WebSphere Edge Server which includes:

- Edge Server - Caching Proxy (IBM Web Traffic Express V3.5)
- Edge Server - Load Balancer (Edge Server - Load Balancer V3.0)

Despite this rebranding the basic concepts in each remain the same.

11.1 Introduction to a Caching Proxy

Traffic flowing between a client and content server can sometimes become corrupt. A proxy server can be used as an intermediary between a client and a content server to ensure such traffic flow corruption does not occur. If implemented, the proxy server will assume responsibility for retrieving and returning data from a content server, and returning it to a client. As part of this responsibility, it can also check the format of the traffic and reformat any corrupt data that is proxied through it, before delivering it to either the client or the content server.

Internet bandwidth availability is limited and expensive. Yet demand for information on the Internet doubles every three months, causing data congestion and delays in the Internet backbone, ISP networks, and corporate intranet networks. Many people continually retrieve the same Web content. This repetition wastes precious Internet and 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 server-to-client traffic is substantially greater than that of inbound traffic. For example, HTML and imbedded images sent from a server are typically at least 10 times the size of the client URLs that request them.

Caching can be implemented to alleviate these problems. If 100 people request the daily news then this will result in 100 separate inbound and outbound requests. If you store, or cache, the first retrieved copy of the news page, the subsequent 99 client requests could retrieve the page from the cache instead of going directly to the content server. Caching the most requested Web content can greatly reduce the load on the Internet and network bandwidth, and ultimately deliver the page faster to the client.

See 4.3.1.8, “Caching” on page 48 for a discussion on the role that caching provides in delivering performance for e-business applications.

11.2 What is a Caching Proxy?

The Caching Proxy provides a way to address the needs and demands of e-business applications and consumer expectations for performance and scalability. The Caching Proxy enables e-businesses to deploy scalable and high-performing Web sites. It helps deliver consistently rapid response rates, reduce the load on the Internet and network bandwidths, and increase server content availability.

The Caching Proxy consists of three main components; it is a proxy server, it is a cache, and it also does content filtering. This next section will provide an overview of these three components.

11.2.1 Proxy server

The Caching Proxy provides the functionality of a *proxy* server for various different 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.

11.2.2 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.

11.2.3 Content filter

The Caching Proxy provides more than proxying and caching, it also acts 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.

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

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

11.3 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.

The Caching Proxy also provides other features that can be used to extend any of these deployment methods:

- **Protection**

  Using the protection capabilities that the Caching Proxy provides, all three scenarios can be configured to protect the proxy server itself and its resources.

- **Extending Caching Proxy functionality**

  The Caching Proxy also provides application programming interfaces (APIs) to extend and customize its functionality. It also provides an interface to process and format information contained in the HTTP headers that flow with client requests and server responses.

The next section provides a detailed description of these Caching Proxy functions.

11.3.1 How forward proxy works

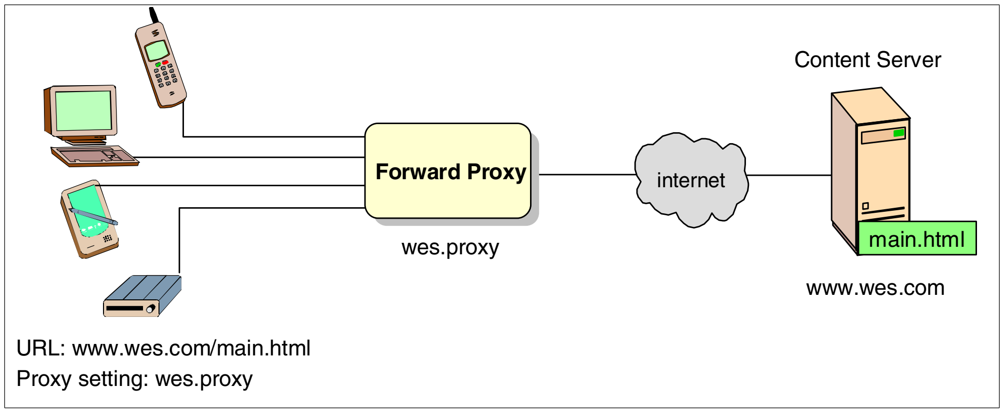


Figure 105. Caching Proxy acting as a forward proxy

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 direct to the forward proxy server using the proxy settings configured in the client's browser, as Figure 105 shows. Having received the request, the Caching Proxy obtains the requested hostname 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.

Note

When the Caching Proxy is configured as a forward proxy server it can only be accessed by those clients and devices where proxy settings can be configured.

11.3.2 How transparent proxy works

For some clients, such 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.

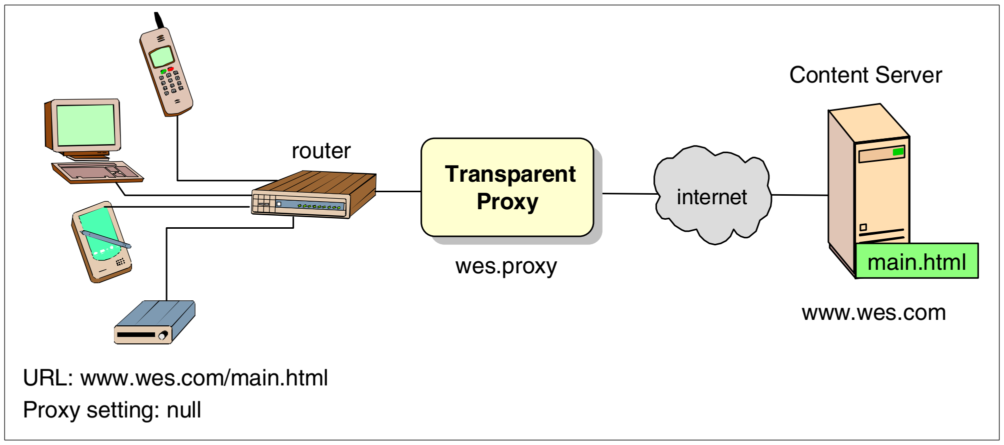


Figure 106. Caching Proxy acting as a transparent proxy

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 Linix, and applies only to HTTP requests that do not require authentication.

11.3.3 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.

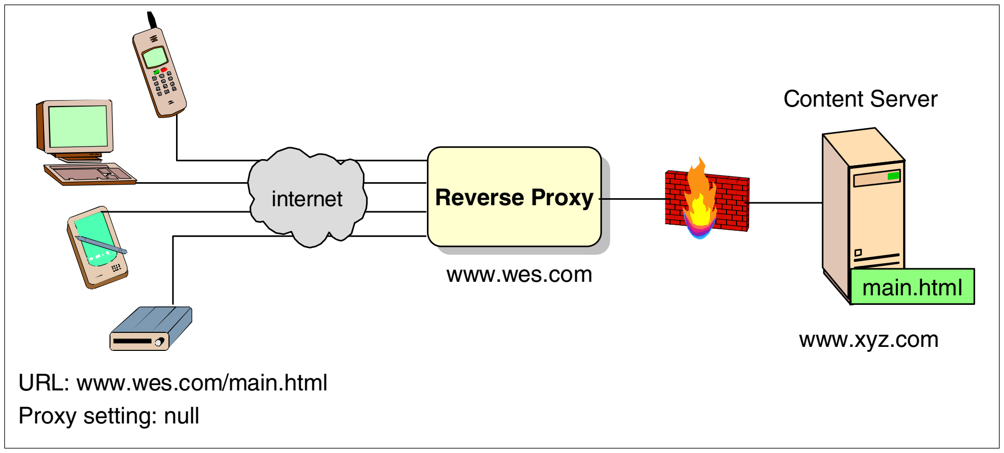


Figure 107. Caching Proxy acting as a reserve proxy

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 107, the client points its browser 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.

11.3.4 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.

There are three directives that define the file access protection to the Caching Proxy server:

- **Protection** - defines a protection setup
- **Protect** - sets protection by linking a request template to a protection setup
- **DefProt** - sets a default protection setup for a particular template request.

Note

A request is the part of the full URL that follows the host name. For example if the full URL is `http://www.wes.com/main.html` then `www.wes.com` is the host name and `main.html` is the request.

A request template specifies files that are subject to protection. Protection is activated based on request templates.

The most common implementations of protection use either the Protection directive with the Protect directive, or the Protect directive on its own. These two protection scenarios are *named* protection setup and *inline* protection setup.

For Inline protection, this would represent a simple configuration for a proxy server `www.wes.com`:.

:

```
Protect   *  {
   ServerID  ProxyServr
   AuthType  Basic
   Mask      All@(*)
             }
```

Figure 108. Sample protection setup for a proxy server - www.wes.com

Only the Protect directive is specified when configuring inline protection, and it is specified per request template. For every client request received by the proxy server, the Caching Proxy will first check the protection setup directives in the ibmproxy.conf.

The protection setup defined in Figure 108 indicates that every request received by the proxy server from any client has to be authenticated using basic authentication. Using this configuration, if the Caching Proxy receives a client request for `http://www.wes.com/main.html`, the proxy server would send an authentication request for the realm specified by the ServerID directive, ProxyServer. The AuthType subdirective specifies the type of authentication, which in the sample configuration is basic authentication. The Mask directive specifies that the protection setup is to be applied to all clients. The client will receive a browser authentication prompt to enter a user name and password for the realm. The client then returns the requested information back to the proxy server, for validation of authentication details. If the information is valid, the client is granted access to all resources in the ProxyServer realm.

Note

Basic authentication user names and passwords are encoded with base-64 format but are not encrypted. Therefore they flow in the clear to the proxy server, representing a possible security exposure.

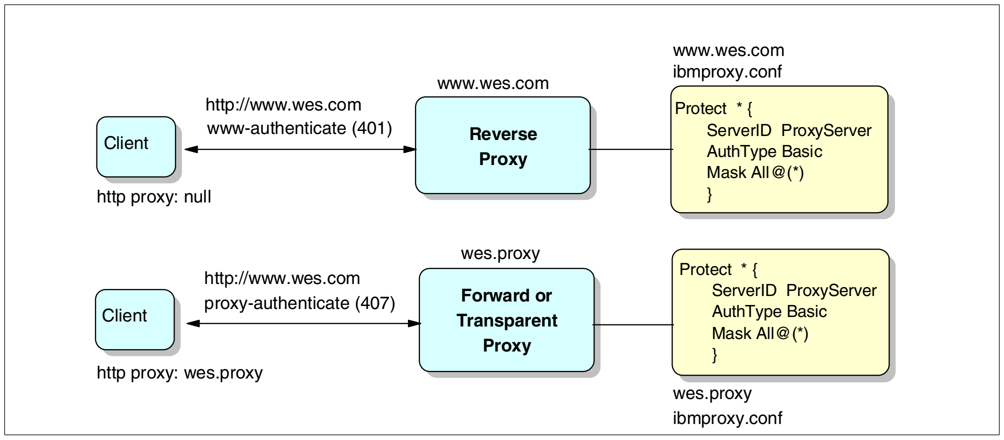


Figure 109. Authentication flows for different deployments of Caching Proxy

The protection directives, both inline and named, have the goal of limiting access to the proxy server and its resources to only authorized clients. Depending on your deployment of the proxy server, the protection process will generate different HTTP responses. As Figure 109 on page 189 illustrates, if protection setup directives are specified for the Caching Proxy when it is deployed as either a forward or transparent proxy server, then the Caching Proxy will generate and return a Proxy-Authenticate request to the client. When the Caching Proxy is configured as a reverse proxy server, and protection setup directives are defined, then the Caching Proxy will generate and return a WWW-Authentication request to the client, prompting the client to enter a valid user name and password.

For full details of named and inline protection refer to the *WebSphere Edge Server for Multiplatforms Web Traffic Express User's Guide.*

For details on protection configuration for the Authentication Server refer to *Everyplace Suite Getting Started Guide*.

11.3.5 Caching Proxy API interface

The Caching Proxy provides an API interface to extend the Caching Proxy's functionality. You may write extensions to perform customized processes such as enhancing authentication or authorization, add error handling routines to track problems, or to detect and track information that comes in from client requests.

When each client request is received by the Caching Proxy, the server performs a series of base steps. You can customize these steps by using the APIs within the Caching Proxy to define your own application plug-in module. This enables you to instruct the server to call your application functions at appropriate processing steps using API directives configured in the ibmproxy.conf.

The Caching Proxy provides a set of API directives, or exits, that can be used when extending Caching Proxy function within your own application.

The most relevant directives to the Everyplace Suite are those utilized by the Authentication Server.

Server Initialization	Performs initialization before any requests are accepted
Server Termination	Performs cleanup and shutdown of resources allocated during server initialization
Authentication	Enables verification of user's privileges to access resources
Authorization	Performs authentication and generates the WWW-Authenticate headers

See *Web Traffic Express User's Guide* and *Web Traffic Express Programming Guide* for details of the exits provided in the Caching Proxy.

If your Caching Proxy API application provides its own *authorization* process, it will override the default server authorization and authentication. If you have the Authorization directive in the configuration file then the application functions associated with them must also handle any necessary authentication.

If an application does not provide its own authorization, you can still perform *authentication* by writing your own authentication application function. To enable this the Authentication directive must be specified in the configuration file.

11.3.6 HTTP Headers

When a client sends a request to a content server, an HTTP header is automatically generated and sent with the request. This HTTP header will contain information about the client requester. It is passed with each HTTP request and response between and client and content server. Its function is to provide additional information to the content server about the client and request. For example, the browser type and the content type parameters specify to the content server how to interpret the request.

An HTTP header would typically contain the following information:

- User-agent consisting of the browser type and operating system type, for example Netscape Navigator V4.7
- Client IP address of the client requester
- A Referrer that provides the destination server with the URL of the referrer link
- HTTP response code and reason, for example `200 OK`
- Authentication details regarding the type of authentication and the credentials used
- Cookies and session information

The Caching Proxy provides an extended list of variables that provide information about the client and server, which can be inserted to or removed from the HTTP header. This functionality can be used by the Caching Proxy API to insert more information into the header of HTTP requests to a content server. For example, an API performing client authentication could prompt the client for a user name and password. The API would then valid the client credentials against LDAP. Having authenticated the client, the Caching Proxy will then insert the authentication credentials and any another authentication information into the HTTP header. The content server receiving the request, with the extended HTTP header could then read this information directly from the HTTP header rather than retrieving the source again.

See *Web Traffic Express Programming Guide* for more full details of the Caching Proxy's API interface.

11.3.7 Secure proxy connections

In a Web environment, the SSL protocol enables secure connections. An SSL connection is established directly between a browser and a content server and used to encrypt the data flow through the connection.

The Caching Proxy provides SSL support in two ways:

- SSL tunneling
- Reverse proxy SSL support

11.3.7.1 SSL tunneling

When the Caching Proxy acts as either a forward or reverse proxy server, secure connections are tunneled through the proxy server. With SSL tunneling, the client sends a secure request to a content server. The Caching Proxy receives the client request and establishes a connection to the destination server to pass the encrypted client request to the content server.

SSL requests are not tunneled through a transparent proxy. The router that redirects requests to the proxy server only directs requests destined for port 80.

The router does not redirect requests going to port 443 for port listening for SSL connections; instead it goes directly to the destination server and is not proxied. Therefore the transparent proxy server does not see any SSL traffic.

11.3.7.2 Reverse proxy SSL support

The Caching Proxy has the functionality to control, and enable, secure connections between a browser and the reverse proxy server, between the reverse proxy server and content server, or both.

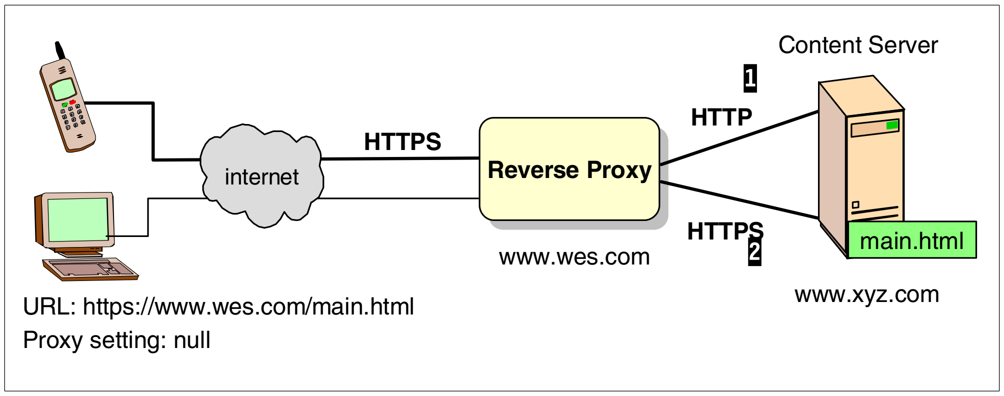


Figure 110. SSL reverse proxy support - secure connections starting at the client

The Caching Proxy allows secure connections to be established between itself and a client browser. All secure requests sent to the reverse proxy server can then be decrypted by the Caching Proxy, and either be allowed to flow in the clear to the content server, as line 1 in Figure 110 illustrates, or re-encrypted by establishing a secure connection between the Caching Proxy and the content server, as line 2 in Figure 110 illustrates.

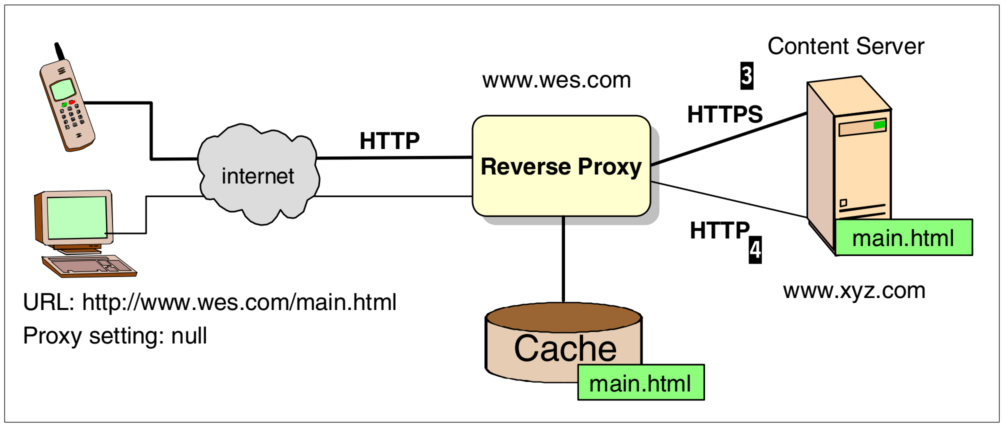


Figure 111. SSL reverse proxy support - secure connections starting at Caching Proxy

The Caching Proxy can also be used as a starting point for secure connections between the proxy and a content server. If the proxy server receives a request for a URL on a content server in the clear, the proxy server can then initiate a secure connection so that all traffic flowing between the two is encrypted. This is illustrated by Figure 111. The Caching Proxy will then decrypt all responses sent back to the client.

When an encrypted request is received by the reverse proxy server, the Caching Proxy can decrypt the request, and send the request to the content server. It may then cache the response content before encrypting and sending the secure response back to the client.

Using the Caching Proxy as an endpoint for any breaking secure connections can reduce the load on the content servers. It also prevents encrypted traffic flowing end-to-end between the client and content server, thereby reducing the demands for bandwidth and improving response rates. It is ideally used if the content server that the proxy server it is accessing is not behind a firewall.

5.2, “WES security” on page 54 provides further details on the SSL reverse proxy support that can be used with the Authentication Server.

11.4 Caching component

The average size of a typical URL (inbound) request is 400 bytes. The average size of an outbound response from a Web server (HTML page, GIF or JPEG image, etc) is 10 KB. If every client requests the same URL then the outbound response will use a large amount of Internet bandwidth, causing poor response rates on congested links. The Caching Proxy provides a cache function that can be used to store Web content. With a Caching Proxy, the requesting client gets the information faster and the network bandwidth is reduced.

The Caching Proxy can save or cache the Web content. It can then serve subsequent requests for the cached content from its local cache, thus eliminating repetitive traffic. This functionality makes the Caching Proxy a powerful tool to help deliver consistently fast response times.

Functionality of the Caching Proxy

The cache component of the Caching Proxy allows you to customize the caching of Web content. By using caching rules, you can define what Web content can actually be cached. Physical storage of cache content can also be customized. The Caching Proxy allows you to control the content contained in the cache by ensuring that the cached content is current and allows you to administer garbage collection to clean up the content. A caching agent can also be used to pre-load into cache specified files before they are requested by a client.

The Caching Proxy is highly scalable allowing multiple Caching Proxy servers to exist in the same cluster. With clustering, the cache component of each Caching Proxy can be shared through the use of Remote Cache Access (RCA), producing a larger logical cache.

This remainder of this section provides an overview of these caching features.

11.4.1 Caching rules

When a proxy server, configured either as forward, transparent or reverse proxy, retrieves a file from a content server, before it returns the requested file to the client, it checks a list of caching rules, both within the proxy server and in the retrieved file, determining what can be cached. 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. Figure 112 on page 194 demonstrates this process for a reverse proxy server.

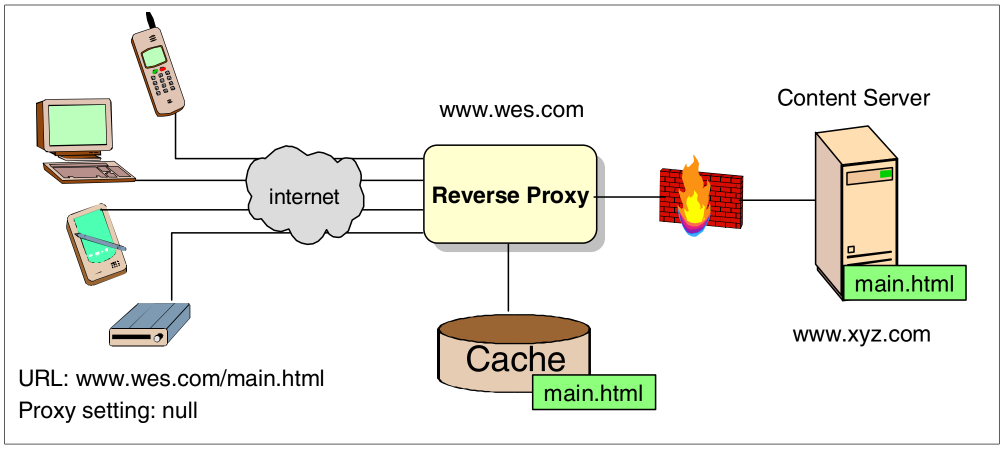


Figure 112. Caching Web content with the Caching Proxy

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 a client request is received for a file, and the file has expired, the Caching Proxy 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 a 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 *Web Traffic Express User's Guide* for full details of the caching and caching rules, and how they can be used and configured.

11.4.2 What to cache

Technically all Web content retrieved from a content server can be cached. However, caching Web content is ideally suited to static Web content. Static Web content generally does not change that frequently. The Caching Proxy provides the capabilities to allow you to determine what should be cached using the caching rules outlined in 11.4.1, "Caching rules" on page 193.

It is difficult to efficiently cache dynamic content since it is constructed by programs that execute at the time a request is made. For example, a search of IBM ThinkPad products will return a list of products based on your individual search criteria. Dynamic data tends to be specific and personalized to the individual client request. Therefore, unless you have an extremely popular search engine with millions of hits per minute, it is unlikely that the same search request will be submitted the same day. It is inefficient to cache such dynamic content.

Here are some general guidelines for dynamic content that should not be cached:

- The dynamic output of CGI scripts that is unique each time it is generated
- Any file passed on an encrypted connection
- Any file with a URL containing a ‘?’ unless query caching is specifically allowed
- Files returned from requests using HTTP methods other than GET, such as POST and PUT
- Any documents requiring authentication unless specifically allowed by the content server

The Caching Proxy allows you determine what dynamic content is cached, however specifying the dynamic cache rules should be carefully considered.

11.4.3 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:** cached files can be stored in the underlying file system. 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.

11.4.4 Caching agent

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 previous caching.

The Caching Proxy provides the functionality to pre-load 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.

11.5 Content Filtering Component

This section provides a brief overview of the filtering component of the Caching Proxy. The Caching Proxy provides filtering at the proxy level of Web content. The Caching Proxy implements this with the use of Platform for Internet Content Selection (PICS). When combined, the Caching Proxy and the Load Balancer components of the Edge Server, can use Content Based Routing to load balance traffic based on URLs. See 12.3.3, "Content Based Routing" on page 209 for more details.

11.5.1 What is content filtering?

The Caching Proxy supports content filtering to filter and block any Web content deemed inappropriate, such as language, nudity and violence, or viruses within requested objects.

Filtering in the Caching Proxy is based on PICS labels. PICS is a rating and labeling standard that users can use to determine what content they would like to see or exclude. When a client requests a file, the proxy server will retrieve the file from the content server. Prior to returning the file back to the client, the filter component of the Caching Proxy will send a request to a labels bureau to obtain labels for the requested file. The PICS label is associated to a document and provides a summary and rating of the content of the document. The PICS label will be defined and issued by a third-party service who will store them in a label bureau. The filter will then use the values contained in the PICS labels, to determine whether the content of the file will be passed to the client or blocked. If the file can be passed, then the cache component will determine whether the passed file can be cached.

The Caching Proxy provides the ability to apply filters in a variety of ways; by defining a list of URLs to be blocked, using PICS labels, and APIs for creating filtering applications.

Content filtering is typically implemented for Internet access to restrict what content users within an organization can request and receive from content servers across the Internet.

See *Web Traffic Express User's Guide* for more details on the content filtering function of the Caching Proxy.

11.6 High performance and scalability with the Caching Proxy

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.

A cache cluster addresses these issues by delivering scalability and high availability. The use of multiple Caching Proxies to form a caching 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 Load Balancer component of the Edge Server.

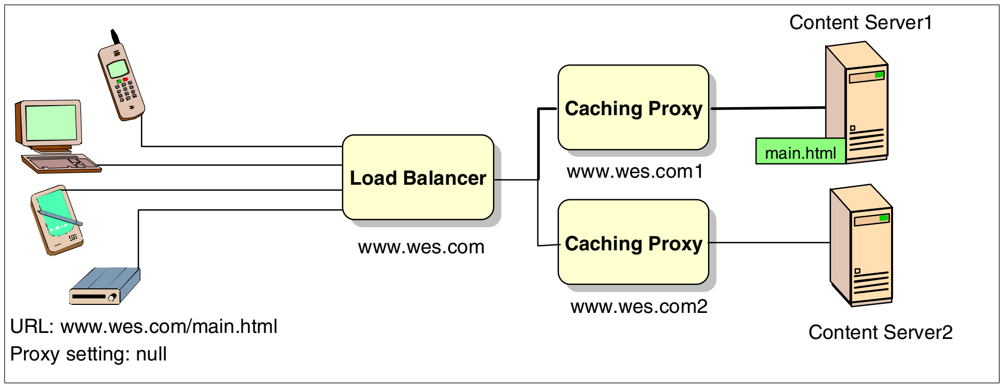


Figure 113. Providing high performance with the Caching Proxy and the Load Balancer

Figure 113 depicts a configuration in which the Load Balancer load balances a cluster of two Caching Proxy machines. The Load Balancer is configured with the cluster's dedicated hostname 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.

The Load Balancer may be used to provide high availability to the Caching Proxy. The 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 reroute requests to another proxy.

In a cluster scenario, individual Caching Proxies may be dynamically added or removed. This makes performing maintenance easy.

11.6.1 Using Remote Cache Access

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.

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.

11.6.2 Using proxy chaining

For large-scale implementations, Caching Proxy clusters can be chained together to improve availability and throughput. In a large implementation, enterprise and ISP, 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 a 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 Web Traffic Express,* SG24-5859.

11.7 Using the Caching Proxy in the Everyplace Suite

As the number of users entering the Everyplace Suite increases, your Everyplace Suite architecture must both perform and scale. To design a performing Everyplace Suite architecture, response rates must be high and path lengths must be minimized. See Chapter 4, “Performance and scalability” on page 37 for more details on these considerations.

The Caching Proxy functions and features that have been outlined can all be used in the Everyplace Suite to deploy a high-performing and scalable solution. Implemented effectively, the Caching Proxy can work with other Everyplace Suite components to help deliver consistently rapid response rates, reduce network traffic congestion, and reduce Internet and network bandwidths.

The Caching Proxy is implemented in the Everyplace Suite in two distinct methods;

- The Caching Proxy is a prerequisite component for Everyplace Suite components.
- The Caching Proxy can be used to optimize some Everyplace Suite component and to optimize the solution as a whole.

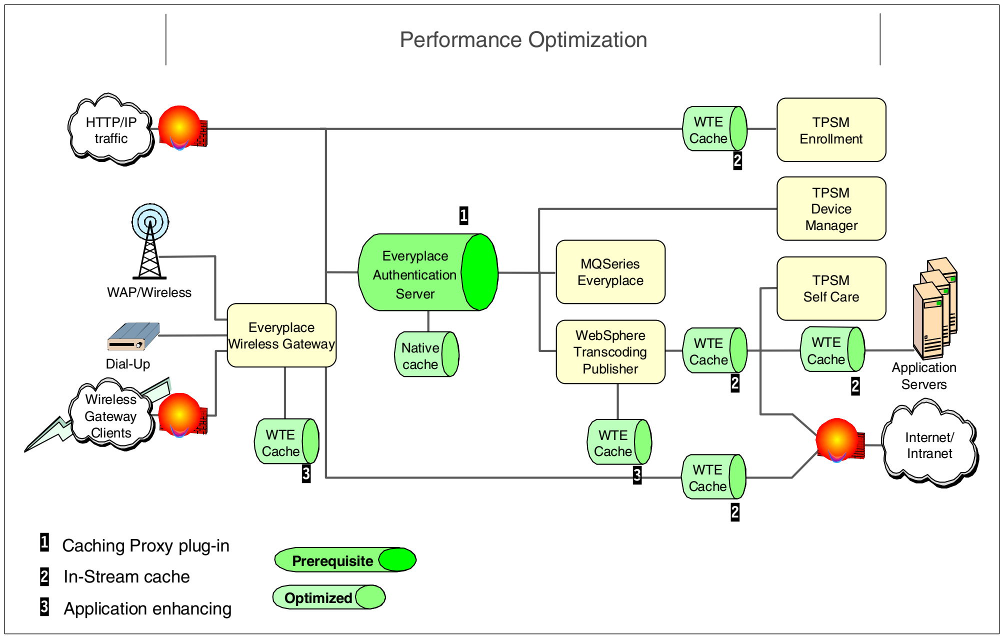


Figure 114. Everyplace Suite architecture - performance components

11.7.1 The Caching Proxy as a prerequisite

The Caching Proxy is a prerequisite component for the Authentication Server. The Authentication Server, is a Caching Proxy plug-in. Refer to 11.3.5, “Caching Proxy API interface” on page 190 for more details on Caching Proxy plug-ins.

The Authentication Server component of the Suite is written to be hosted and launched by the Caching Proxy. The Authentication Server makes use of several Caching Proxy functions. The Caching Proxy and the Authentication Server plug-in are configured as either a protected reverse proxy server or as a protected forward proxy server. To authenticate a client, the Authentication Server uses the HTTP header manipulation functions of the Caching Proxy. As an entry point into the Everyplace Suite domain, it can also act as the end point for secure connections using the new SSL reverse proxy support.

See Chapter 6, “Authentication” on page 69 for more details on how the Caching Proxy is used with the Authentication Server.

11.7.2 Using the Caching Proxy for optimization

The Caching Proxy also provides features and components that can be used to optimize certain Everyplace Suite components, and a WES implementation.

The functions and features that the Caching Proxy provides, such as caching and filtering, can be implemented independently of the Everyplace Suite components to optimize each individual component’s performance and functionality. Figure 113 on page 197 shows where Caching Proxy can be used for optimization in the Everyplace Suite. It can be used for both in-stream caching and to enhance other Everyplace Suite components.

11.7.2.1 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 (like the Internet) to generally optimize access to any site.

If you plan to implement in-stream caching in the Everyplace Suite, the Caching Proxy can be used at several points to cache content. Figure 113 on page 197 outlines where the Caching Proxy can be used for in-stream caching of HTTP traffic flowing within an Everyplace Suite domain. The figure outlines the potential uses of the Caching Proxy when the Everyplace Suite is architected for an adaptive portal. Refer to 3.2.3, “Adaptive (multi-modal) portal” on page 26 for more details on this architecture.

11.7.2.2 Application enhancing

When the Caching Proxy is used to enhance other components in the Everyplace Suite, it is used as a plug-in for specialized application caching.

The caching component of the Caching Proxy can be used to enhance:

- IBM Everyplace Wireless Gateway
- IBM WebSphere Transcoding Publisher

The IBM WAP Gateway installs a special plug-in module on the Caching Proxy to gain improved performance with caching binary WML. See Chapter 7, “Supporting wireless devices” on page 89 for more details on this Wireless Gateway feature.

WebSphere Transcoding Publisher also integrates with the Caching Proxy, and installs a special plug-in on WebSphere Transcoding Publisher to gain improved performance by caching transcoded data. See Chapter 8, “Transcoding Web application content” on page 119 for more details.

By using the Caching Proxy for both in-stream caching and application enhancing, the Caching Proxy can be used to optimize your solution as a whole.

11.7.3 Prerequisite and optimized Caching Proxy in Everyplace Suite

Using the three Everyplace Suite implementation types, this matrix outlines what Caching Proxy functions and features can be used by both the pre-requisite and optimized functions in the Everyplace Suite.

Table 5. Caching Proxy functions that can be used in the Everyplace Suite

Caching Proxy functions	Component host	In -stream cache	Application value-add
Proxy functions			
Forward proxy	@		
Transparent proxy	@		
Reverse proxy	X	@	
Secure connections	@	@	
Proxy Protection	X	@	
APIs	X	@	
Header configuration	X	@	
Proxy chaining		@	
Caching functions			
Enable Caching	@	@	@
Cache pre-loading	@	@	@
RCA		@	
Filter functions			
Content filtering			
Content-based rules		@	@

X - prerequisite use

@ - optimization use

Table 5 outlines which Caching Proxy functions are best placed to be implemented in the Everyplace Suite.

As this section has highlighted, your individual Everyplace Suite architecture and configuration will determine which of the many Caching Proxy functions you will need to deliver high performance and scalability for your Everyplace Suite solution.

Chapter 12. Load Balancer

This chapter provides an overview of the architecture and system design features of the Load Balancer component of IBM WebSphere Edge Server, which can be used within WES.

A discussion of the functionality of the Load Balancer is limited to the context of this book. Details on the full functionality and configuration settings, provided in the Load Balancer, can be found:

- The online product documentation
- *WebSphere Edge Server for Multiplatforms Getting Started Guide Version 1.0,* SC09-4566
- *IBM Load Balancer User's Guide Version 3.0 for Multiplatforms,* GC31-8496
- `http://www.ibm.com/software/webservers/edgeserver/library.html`

12.1 Changing names

The Edge Server - Load Balancer is the new offering from IBM to deliver availability and scalability.

The IBM WebSphere Edge Server includes enhanced versions of both the Caching Proxy and load balancing components of IBM WebSphere Performance Pack V3.0.

IBM WebSphere Performance Pack V3.0 includes:

- IBM Web Traffic Express V3.0
- IBM Network Dispatcher V3.0
- IBM AFS Enterprise File Systems V3.5

The first two components have been enhanced and re-branded to produce the Edge Server which includes:

- Edge Server - Caching Proxy (IBM Web Traffic Express V3.5)
- Edge Server - Load Balancer (Edge Server - Load Balancer V3.0)

Despite this rebranding the basic concepts in each remain the same.

12.2 Introduction to the Load Balancer

Load is the retrieval and processing of back-end Web content, generated from client browser requests for Web content stored on a content server. 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 to your organization the load can be spread over the multiple content servers. Scaling in such a way can directly improve the performance of high-demand content servers. Refer to 4.3.1.2, “Replicated machines (horizontal scaling)” on page 44 for a discussion of horizontal scaling. However increasing the number of machines that your organization has to deliver Web content is not enough to achieve availability and performance expectations. The client requests must be intelligently managed and balanced, over these replica machines to deliver availability expectations.

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 provides dynamic load balancing to deliver the following goals for servers:

- Performance
- Availability
- Scalability

12.2.1 Load Balancer delivers performance

The Load Balancer boosts the overall performance of a server by automatically finding the optimal server within a group of servers to handle each incoming request. The Load Balancer delivers performance because requests are not routed to overloaded servers, and no packet modifications are required to perform the balancing.

12.2.2 Load Balancer delivers availability

The Load Balancer delivers availability since requests are not routed to failed servers but only active servers. The goal of the Load Balancer is to prevent loss of service of a Web site content server by minimizing any single points of failure. Should any content server in the cluster require maintenance in the event of failure, then any servers within the load balancing cluster can be easily removed without losing of Web site service.

12.2.3 Load Balancer delivers scalability

Scaling can easily be introduced by using replicated machines to apply more resources to a heavily accessed Internet site. The Load Balancer delivers scalability by allowing the transparent addition of servers and the Load Balancer to load balance them, without affecting availability or performance.

12.3 Component overview

This section provides an overview of the main functions and features that the Load Balancer provides. For a more detailed discussion and configuration options refer to *IBM Load Balancer User’s Guide Version 3.0 for Multiplatforms.*

The Load Balancer consist 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
- Content Based Routing

12.3.1 The Dispatcher

Dispatcher is an IP packet-level load balancer to balance requests from TCP or UDP protocols. Using any of these protocols 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 will accept requests among HTTP, FTP, SSL Telnet, NNTP, POP3, SMTP or other TCP-based servers or stateless UDP-based servers.

The Dispatcher balances incoming requests and distributes them to the content server that 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 115 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.

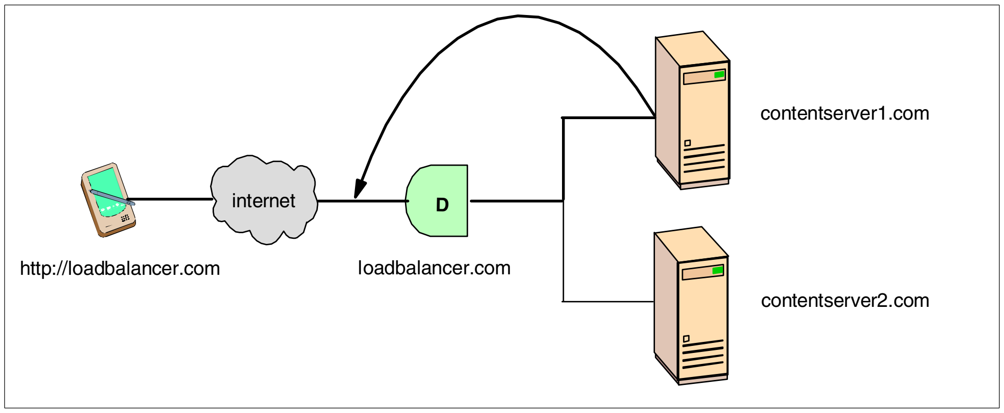


Figure 115. How the Dispatcher works

To enable load balancing for your cluster with the Dispatcher you first install the Dispatcher then you must configure the content servers that it is balancing.

In order to set up the Network Dispatcher itself you need at least two valid IP addresses. One address for the dispatcher itself (the nonforwarding address) and another for each cluster. In Figure 115, only one cluster is used. With this configuration the Dispatcher then becomes the site IP address to which your clients send all requests. This externally advertised address is referred to as the cluster address. The cluster address is associated with your host name and is the address that is load balanced by the Load Balancer. For the load balancing to work you also set the loopback address of the clustered content server to the cluster address so that these servers can accept requests addressed to the cluster address.

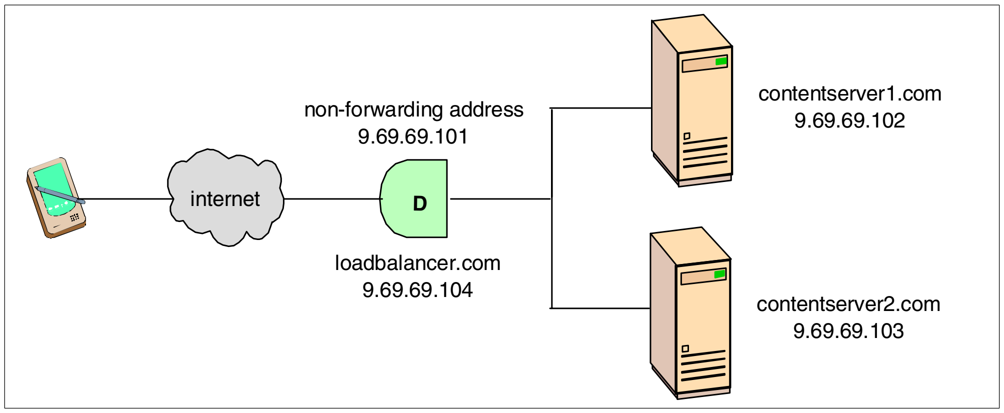


Figure 116. A simple Dispatcher deployment

With the configuration specified in Figure 116, the following will occur when a client request is received by the Dispatcher:

- The client sends its request to loadbalancer.com, the cluster address and the host name of the Dispatcher.
- The Dispatcher will receive the request directed to `http://loadbalancer.com`.
- The Dispatcher keeps a table of all the content servers that it is load balancing and selects one based on certain rules and weights. Having selected the optimal server, it will then forward the client request to that server. In this example it is directed to contentserver1.com.
- The contentserver1.com receives the request and takes a look at the destination IP address in the HTTP header of the request, 9.69.69.104. Contentserver1.com accepts the request because 9.69.69.104 is an alias on its loopback interface.
- The content server fulfills the requested URL and replies directly to the client IP address, extracting it from the HTTP header. It will also add the cluster host name, loadbalancer.com, to the response header as the request's source address. This means that all other client requests will still go through the Dispatcher.

The non-forwarding address, which in the example in Figure 116 is 9.69.69.101, is used to connect to the Dispatcher for management purposes, for example telnet or FTP. The content servers can also be accessed in this way. If you want you can conceal the real IP addresses of the servers in your cluster. This can be achieved by filtering all requests at the gateway router, generally positioned before the Load Balancer.

12.3.1.1 How does Load Balancer determine the optimal server?

Dispatcher utilizes three subcomponents to determine the optimal server in a cluster:

- The Executor
- The Manager
- The Advisor

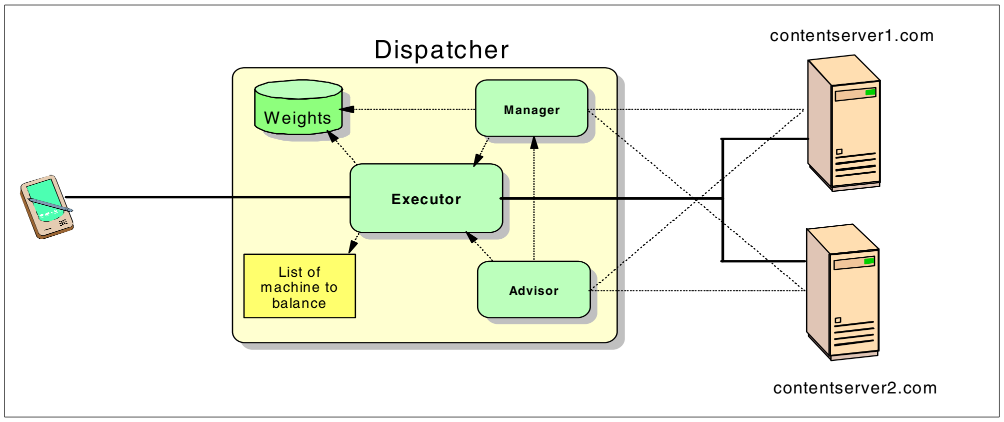


Figure 117. How the Dispatcher determines the optimal server in a cluster

The Executor
The core function of the dispatcher is the Executor. 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 connections 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
The Dispatcher has a Manager function, 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.

Advisors
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.

For more information on the functionality and configuration of Dispatcher refer to the *IBM Load Balancer User's Guide Version 3.0 for Multiplatforms.*

12.3.2 Interactive Session Support

Interactive Session Support (ISS) is a DNS load-balancing mechanism. It can be used to monitor and load traffic across a cluster of servers performing the same service, for example HTTP or FTP.

You can use the ISS component of the Load Balancer with or without a DNS name server.

12.3.2.1 ISS as a load balancer

If ISS is used with a domain name server, then ISS can be used as a DNS load-balancing mechanism. A load-monitoring daemon is installed on each server in a cluster. One of these servers will be elected as the main monitor. The load-monitor will periodically monitor the activity of the other servers, or agents, in the group to detect which agent is the least heavily loaded and communicate this information with other agents in the cluster. The load-monitor and agents in the cluster work closely together to reliably monitor and communicate with each other. ISS then balances the load on servers using this agent information. If it detects a failed server or overloaded server, then the load-monitor simply forwards traffic around it.

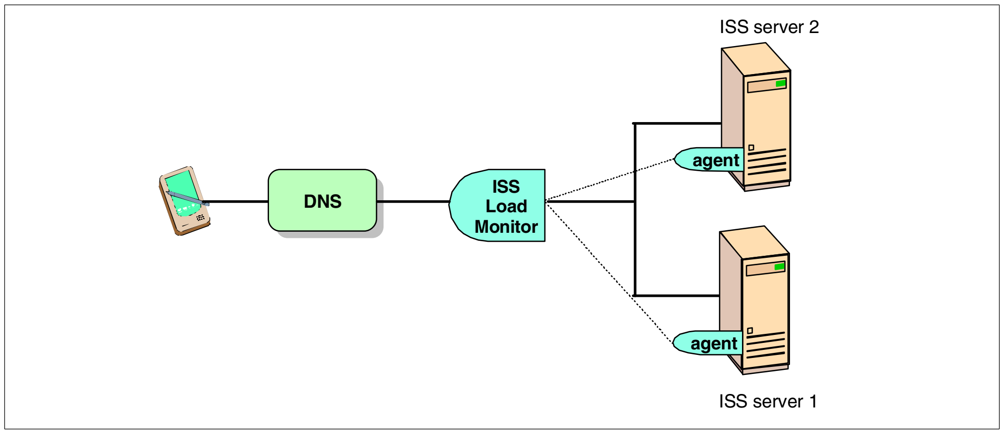


Figure 118. Load balancing using ISS

When a client submits a request for resolution of the DNS name of an ISS server, ISS resolves the name to the IP address of the server in the cluster and forwards this IP address to the client. The client will then resubmit the original request, and the load-monitor, based on the current health information of the agents, will forward the request to the healthiest server. During the next periodic monitor of the agents, the load-monitor will receive the updated health of the agent chosen to handle the request.

12.3.2.2 ISS as an information provider

Alternatively, you can use ISS without DNS. When this deployment is used, ISS will collect server load information and pass this to a load balancer, who will use the information about the status of each server to perform load balancing. ISS can provide this information to the Dispatcher or its own internal observers. ISS ensures that the information used by the Dispatcher or its own observers accurately reflects the load on the servers.

ISS and the Dispatcher can be used together to maximize load balancing for your cluster. ISS supplies the Dispatcher with server load information and the Dispatcher uses the information to load balance requests. Using ISS and the Dispatcher together in such a manner provides a two-tier load-balancing architecture.

Whether deployed with or without DNS, with ISS all servers in the cluster work together to eliminate any single point of failure on the cluster. For more information on ISS refer to the *IBM Load Balancer User's Guide Version 3.0 for Multiplatforms.*

12.3.3 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.

CBR 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, CBR provides the ability to proxy requests to specific servers based on the content requested. Using the rules above, CBR 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, CBR 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 with 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 CBR can be used with HTTP protocol on port 80 and 443.

For POP3 and IMAP requests, CBR is a proxy that chooses an appropriate server based on user name and password provided by the client.

For more information on the functionality and configuration of CBR, refer to the *IBM Load Balancer User's Guide Version 3.0 for Multiplatforms.*

12.3.4 Affinity

When requests are received by the dispatcher, 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 CBR components.

12.3.4.1 Client IP affinity

The Dispatcher and CBR can use this to maintain affinity between a client and one of the server in the cluster based on the IP address of the client. Client IP affinity is useful in an intranet environment but less so in an Internet where client IP addresses can be masked by the IP address of a proxy server or firewall. Using Client IP affinity, the Load Balancer assumes that all connections from the same IP source are potentially the same client. Another drawback with its use is that you can end up with server overload because of client IP address concentration.

12.3.4.2 Dispatcher affinity

Affinity can be used implemented with the Dispatcher component based on the port to which clients connect on.

- **Server affinity**

  When a client originally contacts the site, specifying a host name and port number, the request is load balanced to the chosen server by the Dispatcher in the normal way. With the affinity configured, any subsequent connections sent by the client on the same port will be dispatched to the same server until a configurable affinity time-out value expires. The Dispatcher allows you to configure this sticky option on a per-port basis. For example if you initially connect to `http://wes.com` on port 80, then all subsequent requests from your IP address to wes.com on 80 will be directed to the same server.

- **Cross-port affinity**

  Cross-port affinity covers requests coming in from multiple ports. If a client request is first received by one port, and the next request comes in on another port, cross-port affinity allows the dispatcher to send all client requests on different ports to the same initial server. For example if a clients initial request is `http://wes.com:80`, and the subsequent request is `https://wes.com:443`, then with cross-port affinity enabled the Dispatcher will recognize this request as coming from the same IP address, and will route the request to the original server where the client session information is held.

- **Rule-based affinity**

  Rule affinity override allows the stickiness of a particular port to be overridden for a particular server, even though the port is defined as sticky. You can use rules to define when and why requests are sent to servers in your cluster. For example, your cluster may have a a load-balancing rule defined to direct a particular client IP address to a content server on port 80. If that server is overloaded and rule-based affinity is defined, then the rule will be temporarily overridden and the request will be sent to a less busy server.

- **Server affinity API**

  Server-directed affinity API provides affinity under the control of the application rather than the Dispatcher’s load-balancing algorithm. Using this feature your application can decide where users need to be directed and tell the Load Balancer to set up an affinity relationship before they ever contact the site.

12.3.4.3 CBR affinity

Affinity can be used implemented with the CBR component based on the content of client requests.

- **Cookie affinity**

  Cookie affinity can be used to ensure that session state information is maintained between a client and the content server it connects to initially. Making a rule sticky would normally be used for those applications that store client state at the server, for example servlets and CGI. Using CBR, the Caching Proxy will determine the content of the client request, and the Load Balancer will then use this information to forward the request to the appropriate server. The chosen server is stored as a cookie, and placed by the Caching Proxy into the HTTP header of the server response to the client. If the HTTP header of a client's subsequent request to the Load Balancer contains the cookie and the cookie is still valid, then the client will maintain affinity with the initial server.

12.3.5 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 Dispatcher. The Dispatcher can then use this information to route requests for that particular application to an active application.

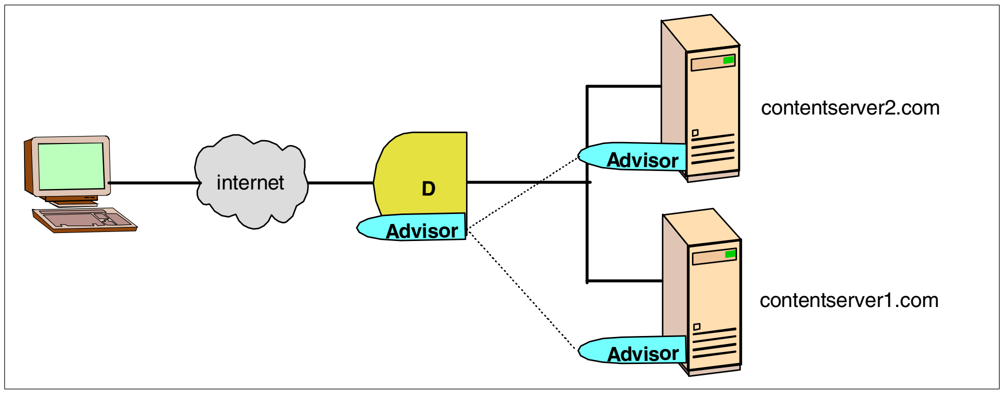


Figure 119. Using advisors to monitor the health of your applications

12.3.5.1 Standard advisors

The 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, WTE, ping, and WLM services.

12.3.5.2 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.

12.4 Providing high availability with the 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. This same principlel can be applied to the Load Balancer; 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 ISS components can provide high availability.

- **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.

- **ISS**

  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.

12.4.1 High availability using the Dispatcher

Two options are available to the Dispatcher to increase the availability of your cluster:

- Heartbeat high availability
- Mutual high availability

12.4.1.1 High availability using heartbeats

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.

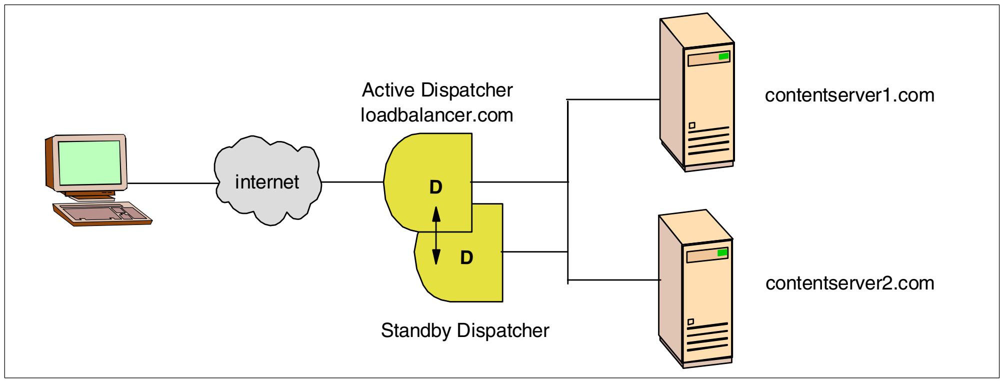


Figure 120. Providing high availability to the Dispatcher using heartbeats

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.

12.4.1.2 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.

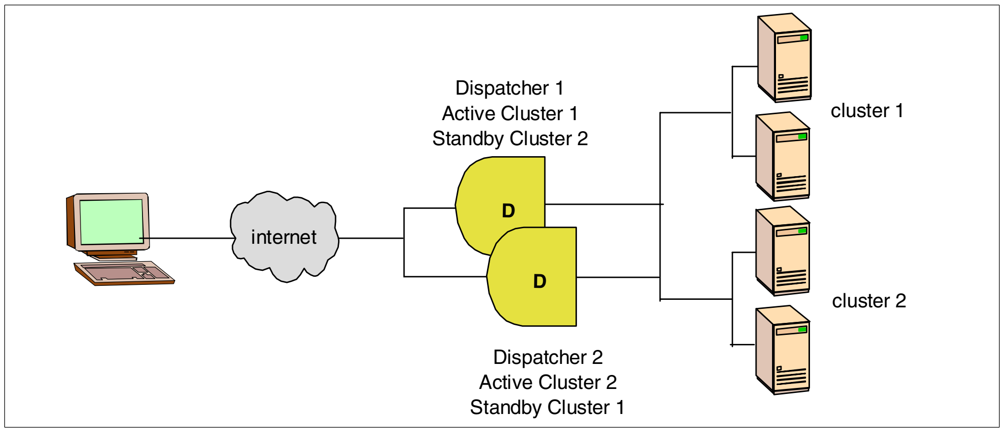


Figure 121. Providing high availability to the Dispatcher using mutual high availability

Colocating

The high availability options described in 12.4.1, "High availability using the Dispatcher" on page 213 generally involve purchasing extra machines in order to implement the benefits of high availability and fault tolerance that the Load Balancer provides. 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 colocating. Colocating 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 can choose to collocate your standby Dispatcher with one of the content servers.

12.4.1.3 ISS high availability

ISS is intrinsically highly available. All the servers in an ISS cluster work together to eliminate any single point of failure. You can configure one or more servers in a cluster to back up the monitor server. You do this by defining the priority number for each one. For the backup servers, or agents, if the load-monitor fails, the first backup server, based on priority, will then become the load-monitor.

12.5 Load Balancer in WES

The Load Balancer is used in the Everyplace Suite to balance requests in real time among Everyplace Suite components and application servers to increase availability and scalability of heavily accessed components.

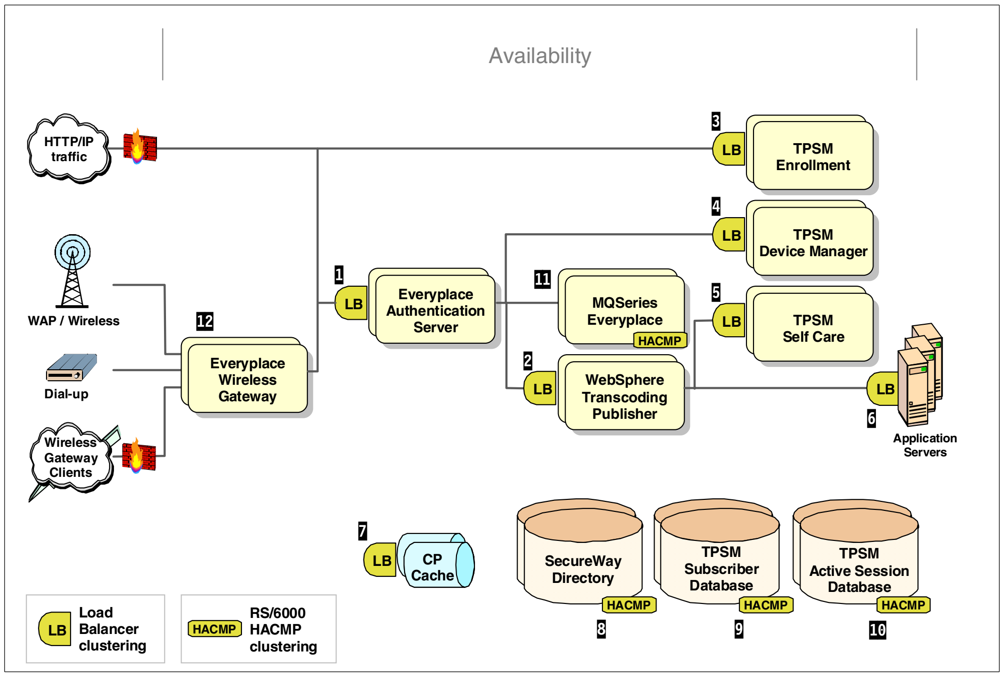


Figure 122. Everyplace Suite architecture - Availability components

As 4.1.3, “Scalability” on page 38, has discussed, using replicated machines is a way of applying resources to a given workload, improving the performance of the affected components. If you want to scale your Everyplace Suite solution, the Load Balancer can effectively and efficiently manage several replica Everyplace Suite components. It can also be used to balance the load between a number of replica or non-replica servers.

Using the Load Balancer in an Everyplace Suite solution will help to ensure that your solution can scale for a large number of clients and that availability is consistently delivered. The Load Balancer can be placed in front of several components in the Suite to balance high volumes of client requests to the individual Everyplace Suite component. See Figure 122.

In the Everyplace Suite, the Load Balancer can also be used to achieve availability by minimizing any Everyplace Suite component as a point of failure in your solution. As Figure 122 outlines, there may be multiple points in your solution architecture where the flow of traffic can fail. To distribute your solution’s load without interrupting its performance, you can implement a Load Balancer in front of several Everyplace Suite components to ensure that your solution works seamlessly if one Everyplace Suite component fails.

To ensure high availability of both the Everyplace Suite components and the Load Balancer itself, you could also implement the high availability features outlined in 12.4, “Providing high availability with the Load Balancer” on page 212. Using these, will help to ensure that no one Load Balancer nor Everyplace Suite component can fail and prevent the next hop in the user request to take place.

Figure 122 recommends where Everyplace Suite components are best placed to use the Load Balancer:

1 IBM Everyplace Authentication Server

2 IBM WebSphere Transcoding Publisher

3 Tivoli Personalized Services Manager, Enrollment

4 Tivoli Personalized Services Manager, Self Care

5 Tivoli Device Manager Server

6 Other application servers inside the Everyplace Suite domain

7 Edge Server - Caching Proxy (depending on its use and configuration)

For other Everyplace Suite components it is either not suitable to use the Load Balancer or the individual component may provided its own specialized load balancing. It is recommended that you use the specialized product load balancing or HACMP failover for the following components in the Everyplace Suite:

8 IBM SecureWay Directory

9 TPSM Subscriber database

10 TPSM Active Session Table database

11 IBM MQSeries Everyplace

12 IBM Everyplace Wireless Gateway

For a discussion on the availability options for these components refer to 3.6, “Availability - dispatchers and clusters” on page 32.

This next section outlines how the Load Balancer can be effectively utilized with each of the WES components that are suited for the Load Balancer.

12.5.1 Using the Load Balancer with the Authentication Server

The projected number of concurrent Everyplace Suite users will grow dramatically over time. In order to accommodate this growth, the Authentication Server can be clustered using the Load Balancer to provide for upward scalability as the number of concurrent users increases.

The Authentication Server acts as the entry point for all client requests to the Everyplace Suite. It is also the base URL of the Everyplace Suite domain. In other words, when clients connect to the Everyplace Suite, they connect using the Authentication Server’s host name. If a Load Balancer is positioned in front of a cluster of Authentication Servers, the Load Balancer then becomes the site IP address to which your clients send all requests. See 12.3, “Component overview” on page 204 for details on how the Load Balancer actually load balances.

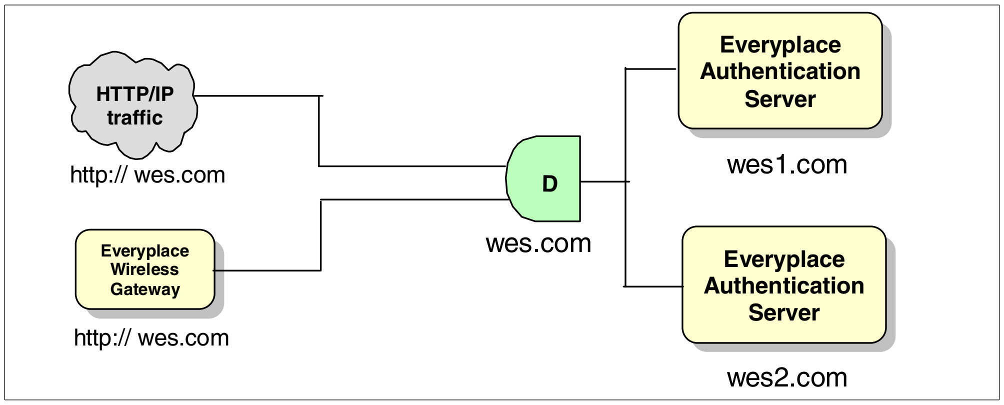

HTTP/IP traffic
http:// wes.com
Everyplace Wireless Gateway
http:// wes.com
D
wes.com
Everyplace Authentication Server
wes1.com
Everyplace Authentication Server
wes2.com

Figure 123. Using the Load Balancer with the Authentication Server

Figure 123 illustrates the architecture when a Load Balancer is deployed to balance requests for an Authentication Server cluster. If a client wishes to access an Everyplace Suite service, then the client will connect to `http://wes.com`. Client requests are received by the Load Balancer, which will determine which EAS in the cluster is best placed to handle the client authentication request. Refer to Chapter 6, "Authentication" on page 69, for more details on the authentication process used by the Authentication Server.

Affinity can also be used with the Authentication Server. Once authenticated, all subsequent user requests can be directed through the same Authentication Server 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 Authentication Server 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 AST to be validated. See Chapter 6, "Authentication" on page 69 for more details on where and why affinity could be used by the Authentication Server.

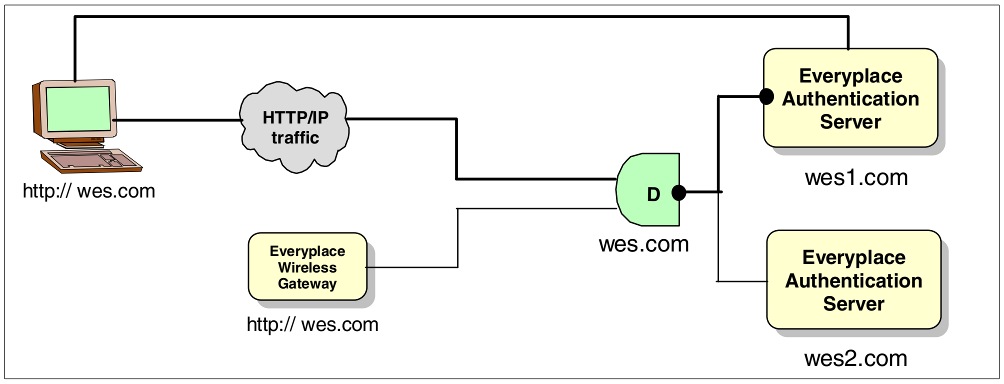

http:// wes.com
HTTP/IP traffic
Everyplace Wireless Gateway
http:// wes.com
D
wes.com
Everyplace Authentication Server
wes1.com
Everyplace Authentication Server
wes2.com

Figure 124. Affinity between the Load Balancer and the Authentication Server

Using advisors to monitor Authentication Server RADIUS and AST

Full availability will not be achieved when using the Load Balancer with the Authentication Server if either the RADIUS or AST servers are inactive. In this case a Customized Advisor could be used to monitor the health of the RADIUS and AST servers, and feed back this information to the main Load Balancer in front of the Authentication Server. When determining which the Authentication Server to dispatch a client request to, the Dispatcher could use the information returned from the customized advisors, residing on both the RADIUS and AST server, to find the optimal Authentication Server. Refer to 6.5, “Scalability and availability” on page 82 for more details.

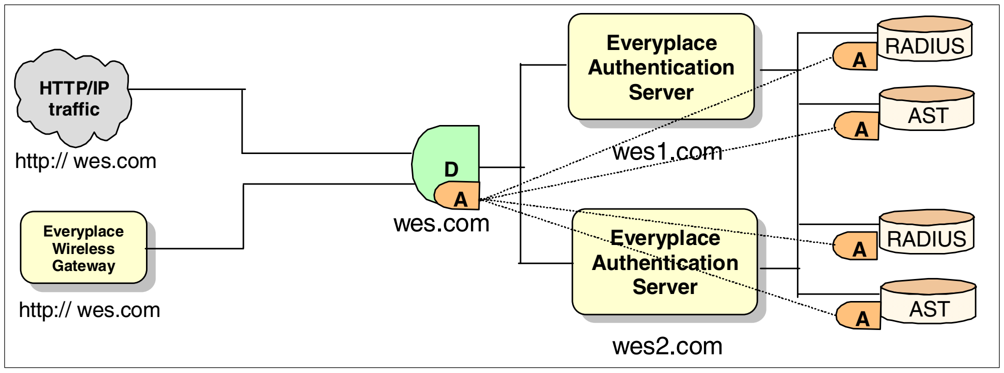


Figure 125. Using customized advisors with the Authentication Server

12.5.2 Using the Load Balancer with WebSphere Transcoding Publisher

As the number of users accessing the Everyplace Suite domain from different client devices grows, the demands for transcoding of application content to suit the different devices will increase. Also, the actual process of transcoding Web content is a high-intensive activity, and as the number of requests for transcoding increases WebSphere Transcoding Publisher’s time spent performing the 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.

The Load Balancer can be positioned in front of a cluster of WTP servers to resolve this potential bottleneck. It will accept requests from authenticated clients and dispatcher the request to the optimal server. Figure 126 on page 219 illustrates the hops from the Authentication Server to WebSphere Transcoding Publisher with the Load Balancer positioned in front to balance. The Authentication Server will direct all authenticate 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 Authentication Server, which will in turn forward the response back to the client.

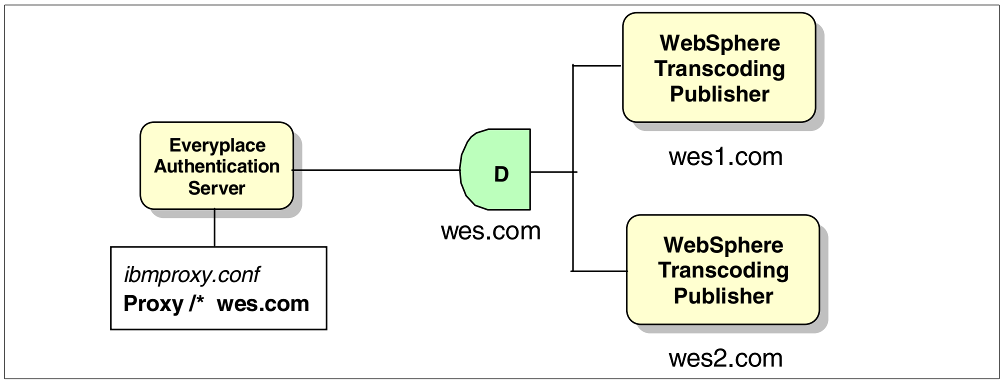

WebSphere Transcoding Publisher
wes1.com
Everyplace Authentication Server
D
wes.com
ibmproxy.conf
Proxy /* wes.com
WebSphere Transcoding Publisher
wes2.com

Figure 126. Using the Load Balancer with WebSphere Transcoding Publisher

No session state is maintained between client and WebSphere Transcoding Publisher,. Therefore, it makes little sense to implement an affinity between the Load Balancer and WebSphere Transcoding Publisher. CBR could be used to forward client requests for transcoded content to 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. Advisors, whether standard or customized, could be utilized to provide advanced monitoring of the health of the WebSphere Transcoding Publisher servers in the cluster.

12.5.3 Using the Load Balancer with application servers

TPSM has several components that are all part of the Everyplace Suite: TPSM Enrollment Server, TPSM Device Manager, and TPSM Self Care Server. For all application servers, as the number of Everyplace Suite users increase so the demand to access content on these application servers increases.

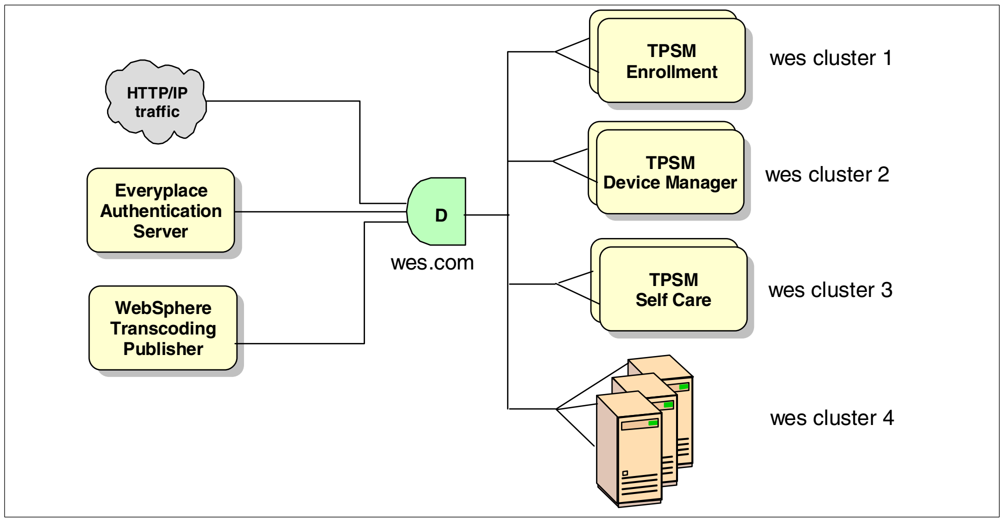


Figure 127. Using the Load Balancer with Everyplace Suite application servers

The Load Balancer can be positioned in front of any of these components to delivery both scalability and availability. As Figure 127 illustrates, the Load Balancer can be placed in front of TPSM Enrollment, Self Care Servers, and Device Manager, and any Everyplace Suite applications within your domain.

When the Load Balancer is deployed in this configuration, client requests are received by the Load Balancer, which in turn dispatches the request to the optimal server to handle the request, The application server will respond directly to the requesting component without going back through the Load Balancer.

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. See 9.4.5, “Deployment within the Everyplace Suite” on page 157 for more details.

If any of the application servers require stateful connections, then affinity can be used to maintain the connection. Advisors can also be utilized to determine not only the health of the application servers, but to monitor the application that resides there.

12.5.4 Using the Load Balancer in the Everyplace Suite

Availability requires more than simply introducing Load Balancers in front of every component. This next section will provide a discussion of over-balancing in the Everyplace Suite using the Load Balancer.

12.5.4.1 Over-balancing in the Everyplace Suite

As the previous sections have outlined, the Load Balancer can be effectively used with many of the Everyplace Suite components to provide availability and scalability by load balancing requests over horizontal servers. But availability, in the context of the Everyplace Suite, requires more than simply introducing dispatchers in front of every Everyplace Suite recommended component.

Replicating content server and introducing a Load Balancer in front of the cluster to balance the request for some deployments can create problems. Over-balancing is a problem that must be avoided.

Scalability and performance are the key objectives of the Load Balancer. 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 Load Balancers will create in the overall path. Each extra Load Balancer you introduce to your architecture will introduce a further hop that a request must travel through before reaching its final destination. The worst-case scenario of this over-balancing is that performance can be negatively impacted. Each Load Balancer introduces a further hop in the other path; with each hop the client is still waiting for a response and the response rate can be increased. This is contrary to exactly what the Edge Server is designed to solve. The number of Load Balancers that you deploy in your solution must be weighted against your goals for increasing performance.

Appendix A. Devices/networks supported by the Everyplace Suite

The Everyplace Suite runs on AIX Version 4.3.3 and Solaris Version 7 platforms only. The Everyplace Suite also requires that the Java Development Toolkit (JDK Version 1.8.8) be installed on servers installing any Everyplace Suite component. The following table displays the system requirements and prerequisites for each component.

Table 6. Prerequisites for WebSphere Everyplace Suite components

Everyplace Suite Component	**Requirements and Prerequisites**
Everyplace Authentication Server	Edge Server - Caching Proxy
Wireless GatewayWireless Gateway	• DB2 Enterprise Edition 7.1 or Oracle DB 8.1.5 (8i) • IBM X.25 co-processor card if you are connecting to any of the following packet radio networks: – ARDIS-X.25 – DataTAC-5000 – Mobitex – Modacom-SCR – MCA-bus RS/6000 machines: IBM X.25 Interface co-processor/2 card – ISA-bus RS/6000 machines: IBM X.25 Interface co-processor/1 card **Note:** If you connect to ARDIS, DataTAC-SCR, or Mobitex by IP instead of X.25, you do not need an X.25 Interface co-processor card. • Asynchronous adapter and modem if you connect to a circuit-switched cellular network using RS232 communication (for example, PSTN, GSM, or AMPS). • LAN adapter for connection to the IP network and to the RNG of a DataTAC-TCP, Mobitex-TCP, or RNC3000 radio network • AIXLinkX.25 1.1.5 (if using X.25 connectivity)
Wireless Gatekeeper	JRE 1.2.2
Everyplace Suite Administration Console	Netscape Navigator 4.08, or Netscape Communicator 4.5 or higher
Edge Server - Load Balancer	• GSKit (IKEYMAN) • Edge Server - Caching Proxy

Everyplace Suite Component	Requirements and Prerequisites
Tivoli Personalized Services Manager	• WebSphere Application Server, Standard Edition, 3.0.2 • DB2 UBB 7.1 or Oracle 8.1.5 RDMS • JDBC connector (Oracle JDBC) if Using Oracle • DBI/DB Perl module to Perl 5 • IBM HTTP Server • IGB RAM, 2GB hard disk space
Edge Server - Caching Proxy	• Network Dispatcher Admin package and Device driver • 4GB Ultra SCSI disk or 16GB SSA disk
Transcoding Publisher	SecureWay Directory 3.2

Supported Pervasive Devices and Network Types

The Everyplace Suite components support the following devices and platforms:

Table 7. Devices supported by WebSphere Everyplace Suite components

Everyplace Suite Component	**Supported Devices**	**Supported Platforms and Network Types**
Wireless Gateway	• IBM Workpad Z50 • HP 660LX • HP Jornada 820 • Compaq 2010C • Phillips Velo 500 • Sharp Mobilon Pro • Sharp Mobilon Pro • LG Phenom Express • NEC 770 Mobile Pro • IBM ThinkPad • Non-IBM Laptops • Palm VII • Palm V • WorkPad C3	• Windows CE 2.11 • Windows 95 • Windows 98 • Windows NT • Windows 2000 • Palm OS 3.x
Transcoding Publisher		• IE4 • IE5 • Netscape • PocketIE • Handweb 1.1 • Windows 3.2 • WindowsCE • Palm OS
Tivoli Personalized Services Manager	• Palm V • Palm III • Compaq Aero 800 • Compaq Stinger • ESBU Screen phone • PvC Client Stack	• Palm OS 3.0 or higher • Windows CE 3.0 • IAP500 • BeIA OS
MQSeries Everyplace	• Palm V • Psion 5MX (Pro) • Psion NetBook • HP Jornada 820 (CE)	• Palm OS 3.0 • Windows CE • Windows 95 • Windows 98 • Windows 2000

Table 8. Networks supported by Everyplace Wireless Gateway

MNC type	Description	Install this network support
ardis-tcp	ARDIS standard context routing using a TCP connection	Ardis
ardis-x25	ARDIS standard context routing using an X.25 connection	Ardis
dataradio-bdlc	Dataradio BDLC connection	Dataradio
dataradio-msc	Dataradio multi-site controller connection	Dataradio
datatac-5000	DataTAC 5000 using an X.25 connection	DataTAC
datatac-6000	DataTAC 5000 using a TCP/IP connection	DataTAC
dial-isdn	Integrated switched digital network connection	Dial
dial-pstn	Primary switched telephone network connection	Dial
dial-tcp	Dial connection through a IP-attached modem server	Dial
ip-lan	IP-based network, such as CDPD, frame relay, cable modem, or LAN	IP LAN
ip-wdp	IP-based network using wireless datagram protocol	Installed automatically with Wireless Gateway
mobitex	Mobitex international standard connection, such as Norcom Satellite	Mobitex
mobitex-tcp	Mobitex using a TCP connection	Mobitex
modacom-scr	Modacom standard context routing	Modacom
rnc-3000	Radio network controller	Motorola PMR

Appendix B. Special notices

AIX	AS/400
CICS	DB2
DB2 Universal Database	IBM
MQSeries	Netfinity

Appendix C. Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

C.1 IBM Redbooks

For information on ordering these publications see “How to get IBM Redbooks” on page 229.

- *IBM WebSphere Transcoding Publisher V1.1: Extending Web Applications to the Pervasive World*, SG24-5965
- *Mobile Computing: The eNetwork Wireless Solution*, SG24-5299
- *IBM WebSphere Performance Pack: Caching and Filtering with IBM Web Traffic Express*, SG24-5859
- *IBM WebSphere Performance Pack: Load Balancing with IBM SecureWay Network Dispatcher*, SG24-5858

C.2 IBM Redbooks collections

Redbooks are also available on the following CD-ROMs. Click the CD-ROMs button at ibm.com/redbooks for information about all the CD-ROMs offered, updates and formats.

CD-ROM Title	Collection Kit Number
IBM System/390 Redbooks Collection	SK2T-2177
IBM Networking Redbooks Collection	SK2T-6022
IBM Transaction Processing and Data Management Redbooks Collection	SK2T-8038
IBM Lotus Redbooks Collection	SK2T-8039
Tivoli Redbooks Collection	SK2T-8044
IBM AS/400 Redbooks Collection	SK2T-2849
IBM Netfinity Hardware and Software Redbooks Collection	SK2T-8046
IBM RS/6000 Redbooks Collection	SK2T-8043
IBM Application Development Redbooks Collection	SK2T-8037
IBM Enterprise Storage and Systems Management Solutions	SK3T-3694

C.3 Other resources

These publications are also relevant as further information sources:

- *IBM WebSphere Everyplace Suite Getting Started* (product online publication)
- *WebSphere Edge Server for Multiplatforms Getting Started Guide Version 1.0,* SC09-4566
- *WebSphere Edge Server for Multiplatforms Web Traffic Express User’s Guide Version 1.0,* GC09-4567
- *WebSphere Edge Server for Multiplatforms Web Traffic Express Programming Guide Version 1.0,* GC09-4568
- *Wireless Gateway for AIX Administrators Guide (product online publication)*

C.4 Referenced Web sites

These Web sites are also relevant as further information sources:

- http://www-4.ibm.com/software/network/mobile/library/
- http://www-4.ibm.com/software/ts/mqseries/everyplace/
- http://www-4.ibm.com/software/webservers/edgeserver/library.html
- http://www-4.ibm.com/software/webservers/transcoding/library.html/
- http://www-4.ibm.com/software/data/db2/library/
- http://www-4.ibm.com/software/webservers/appserv/library.html
- http://www-4.ibm.com/software/webservers/httpservers/library.html
- http://www-4.ibm.com/software/network/directory/library/
- http://www.ibm.com/pvc
- http://www.alphaworks.ibm.com
- http://www.almaden.ibm.com/cs/wbi
- http://www.ibm.com/software/webservers/transcoding/library.html
- http://www.software.ibm.com/ts/MQSeries.html
- http://www.wes.com/main.html
- http://www.ibm.com/software/webservers/edgeserver/library.html
- http://loadbalancer.com
- http://w3.itso.ibm.com
- http://w3.ibm.com
- http://www.elink.ibmlink.ibm.com/pbl/pbl
- http://www.software.ibm.com/ad/cb
- http://www.software.ibm.com/ebusiness/connector
- http://www.omg.org
- http://java.sun.com/products/ejb
- http://www.whatis.com/iiop.htm
- http://www.javasoft.com/beans/docs
- http://www.software.ibm.com/ebusiness/pm.html#J
- http://java.sun.com/products/jndi/index.html
- http://java.sun.com/products/jdk/rmi/index.html
- http://www.software.ibm.com/ebusiness/pm.html#S
- http://www.software.ibm.com/ebusiness/appsrvsw
- http://www.software.ibm.com/xml
- http://www.w3.org/TR/WD-xs

How to get IBM Redbooks

This section explains how both customers and IBM employees can find out about IBM Redbooks, redpieces, and CD-ROMs. A form for ordering books and CD-ROMs by fax or e-mail is also provided.

- **Redbooks Web Site** **ibm.com**/redbooks

  Search for, view, download, or order hardcopy/CD-ROM Redbooks from the Redbooks Web site. Also read redpieces and 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.

- **E-mail Orders**

  Send orders by e-mail including information from the IBM Redbooks fax order form to:

	e-mail address
In United States or Canada	pubscan@us.ibm.com
Outside North America	Contact information is in the "How to Order" section at this site: http://www.elink.ibmlink.ibm.com/pbl/pbl

- **Telephone Orders**

United States (toll free)	1-800-879-2755
Canada (toll free)	1-800-IBM-4YOU
Outside North America	Country coordinator phone number is in the "How to Order" section at this site: http://www.elink.ibmlink.ibm.com/pbl/pbl

- **Fax Orders**

United States (toll free)	1-800-445-9269
Canada	1-403-267-4455
Outside North America	Fax phone number is in the "How to Order" section at this site: http://www.elink.ibmlink.ibm.com/pbl/pbl

This information was current at the time of publication, but is continually subject to change. The latest information may be found at the Redbooks Web site.

IBM Intranet for Employees

IBM employees may register for information on workshops, residencies, and Redbooks by accessing the IBM Intranet Web site at http://w3.itso.ibm.com/ and clicking the ITSO Mailing List button. Look in the Materials repository for workshops, presentations, papers, and Web pages developed and written by the ITSO technical professionals; click the Additional Materials button. Employees may access MyNews at http://w3.ibm.com/ for redbook, residency, and workshop announcements.

IBM Redbooks fax order form

Please send me the following:

Title	**Order Number**	**Quantity**

First name	Last name	
Company		
Address		
City	Postal code	Country
Telephone number	Telefax number	VAT number

☐ Invoice to customer number

☐ Credit card number

Credit card expiration date	Card issued to	Signature

We accept American Express, Diners, Eurocard, Master Card, and Visa. Payment by credit card not available in all countries. Signature mandatory for credit card payment.

Glossary

ALP. ArTour Link Protocol. See WLP**.**

Applet. A Java applet is a small application program that is downloaded to and executed on a Web browser or network computer. A Java applet typically performs the type of operations that client code would perform in a client/server architecture. It edits input, controls the screen, and communicates transactions to a server, which in turn performs the data or database operations.

API. Application Program Interface.

Bean Managed Persistence. Bean Managed Persistence (BMP) is a term used to describe a type of entity EJB where the bean developer specifies how the bean is to be persisted to a database by writing Java code in the appropriate methods to perform the tasks required.

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 and giving thus better performance.

CICS. Customer Information Control System is an IBM application enabler system very popular in mainframes.

Component Broker. Component Broker is an IBM CORBA management server product that provides an object request broker (ORB) to facilitate the deployment of CORBA objects. The EJB deployment engine in WebSphere is based largely on similar services in Component Broker. See `http://www.software.ibm.com/ad/cb` for more information.

Connectors. The term connectors, or e-business connectors, is used to describe gateway products from IBM that allow access to enterprise data on back-end systems over the Internet. They include direct browser access to back-end systems such as DB2 through Net.data and also Java access through products such as the CICS gateway for Java. See `http://www.software.ibm.com/ebusiness/connectors.html` for more information.

Container Managed Persistence. Container Managed Persistence (CMP) is a term used to describe a type of entity EJB where the code to persist the bean to a database is generated at deployment time by the EJB container.

Clustering. Clustering is a technique used to provide scalability through the use of multiple copies of an application on the same or 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.

CORBA. Common Object Request Broker Architecture (CORBA) is a cross-platform, industry-standard distributed object protocol. CORBA is used to locate and use objects on a variety of platforms, written in a variety of languages across a network. See `http://www.omg.org` for more information on CORBA.

e-business. e-business is a term used by IBM to describe the use of Internet technologies to transform business processes. What this means in practice is using Internet clients such as Web browsers as front ends for applications that access back-end legacy systems to allow greater access. See `http://www.software.ibm.com/ebusiness` for more information.

Enterprise Java Beans. Despite the name, Enterprise Java Beans or 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 are rather 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.

IBM WebSphere Transcoding Publisher. IBM WebSphere Transcoding Publisher is a network software that modifies content presented to users based on the information associated with the request, such as device constraints, network constraints, user preferences, and organizational policies. Transforming content can reduce or eliminate the need to maintain multiple versions of data or applications for different device types and network service levels.

Image Transcoder. Image Transcoder is the transcoder that can scale, modify quality, and modify color levels in JPEG and GIF images. Additionally, the Image Transcoder can convert JPEGs to GIFs for devices that do not render JPEGs.

Intermediary. In a typical Web environment, information is simply sent from a server to a browser for display and interaction. However, there are many ways that adding an intermediary between the browser and the server can improve the system. For example, an intermediary can keep track of the information the user has viewed in order to make it easier to find information again. Or, an intermediary may enhance the information that the user sees by adding annotations and personalization beyond what the server was designed to do. Intermediaries turn the network into a “smart pipe” with applications that can enhance the information on the Web.

IIOP. Internet Inter ORB Protocol (IIOP) is an internet protocol used for CORBA object communication. For more information see `http://www.whatis.com/iiop.htm`.

Java Application. A Java application is a program written in Java that executes locally on a computer. It allows programming operations in addition to those used in applets which can make the code platform dependent. It can access local files, create and accept general network connections, and call native C or C++ functions in machine-specific libraries.

JavaBeans. JavaBeans are Java components designed to be used on client systems. They are Java classes that conform to certain coding standards. They can be described in terms of their properties, methods and events. JavaBeans may be packaged with a special descriptor class called a BeanInfo class and special property editor classes in a JAR file. Java Beans may or may not be visual components. See `http://www.javasoft.com/beans/docs` for more information.

JavaServer Pages. JavaServer Pages are HTML source files that include Java extensions to provide dynamic content and increased functionality. JavaServer Pages are compiled into Servlets before deployment. See `http://www.software.ibm.com/ebusiness/pm.html#JavaServer Pages`.

JDBC. JDBC is a Java API that allows Java programs to communicate with different database management systems in a platform-independent manner. Database vendors provide JDBC drivers for their platforms that implement the API for their database, allowing the Java developer to write applications to a consistent API no matter which database is used.

JNDI. Java Naming and Directory Interface (JNDI) is an API that allows Java programs to interface and query naming and directory services in order to find information about network resources. JNDI is used in WebSphere to provide a directory of Enterprise Java Beans. See `http://java.sun.com/products/jndi/index.html` for more information.

JSP. See JavaServer Pages.

MEG. A plug-in encapsulates a correlated set of transcoding components (MEGs). A MEG is one of Monitor, Editor (Request or Document) or Generator. More information on MEGs and WBI (Web Intermediaries) can be found from `http://www.almaden.ibm.com/cs/wbi`.

Mini-certificates. A simplified digital certificate specification used by WTLS.

MQe. See MQ Everyplace.

MQ Everyplace. The latest member of the MQSeries family of products. It is designed to satisfy the messaging needs of lightweight devices and the requirements that arise from the use of fragile communication networks.

PDA. Personal Digital Assistant.

Persistence. Persistence is a term used to describe the storage of objects in a database to allow them to persist over time rather than being destroyed when the application containing them terminates. Enterprise Java Bean containers such as WebSphere provide persistence services for EJBs deployed within them.

Preference Aggregator. Preference Aggregator is the process where the client’s device, network type, and any other information that might cause the transcoders to produce a different variant (or permutation) of the resource, are determined.

Preference Aggregation. Preference aggregation is the process where the client's device, network type, and any other information that might cause the transcoders to produce a different variant (or permutation) of the resource, are determined.

Preference profiles. IBM WebSphere Transcoding Publisher uses preference profiles to represent the characteristics of devices and networks, and a default user profile to represent organizational policies. Each profile defines IBM WebSphere Transcoding Publisher how to treat documents that will be delivered to that device or over that network.

A preference profile can represent a particular type of device, such as a WorkPad, or a particular network type, such as a wireless network.

Proxy. Transcoding Publisher connects through a proxy server that is configured with a firewall to manage network traffic and to protect your network from outside intrusion.

RMI. Remote Method Invocation (RMI) is a lightweight distributed object protocol that allows Java objects to call each other across a network. RMI is part of the core Java specification. See `http://java.sun.com/products/jdk/rmi/index.html` for more information.

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. They take as input the HTTP request from the client and output dynamically generated HTML. For more information on servlets see `http://www.software.ibm.com/ebusiness/pm.html#Servlets`.

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.

Stand-alone Network Proxy. User uses the IBM WebSphere Transcoding Publisher as a normal proxy in his browser and the data that flows from the original source will be transcoded in the proxy according to the device and network profile needed.

Stylesheet Transcoder. Stylesheet Transcoder is the transcoder that selects the stylesheet and applies it to an input Extensible Markup Language (XML) document to produce a version that is appropriate for the target device.

Text Transcoder. Text Transcoder is the transcoder that can modify elements of a text document based on device, network, and potentially user preference information. The primary use of this Text Transcoder is to modify Hypertext Markup Language (HTML) documents to remove unsupported elements, reduce space usage, replace features such as images or frames with links, and otherwise tailor documents to make them render more gracefully on constrained devices.

TLS. Transport Layer Security. The standard (IEFT) security protocol on the Internet. It is expected to eventually supersede SSL.

Transcoder. Transcoder is a program that modifies the content of a document.

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 according to the capacity of the network being used.

It is also the process where the MEGs related to modifying the request, generating the original resource and all of the document (or resource) editing (or transcoding) occurs.

WAP. Wireless Application Protocol.

WAS. IBM WebSphere Application Server.

Web Application Servers. 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`

WLP. A modified version of the Point-to-Point Protocol (PPP) used by the IBM Wireless Gateway to support wireless (non-WAP) client devices.

WTE. Web Traffic Express. An IBM caching proxy.

WTLS. Wireless Transport Layer Security. A simplified version of TLS designed specifically for WAP devices. It uses mini-certificates.

WTP. WebSphere Transcoding Publisher.

X.509. A digital certificate specification used by SSL and TLS.

XML. XML, or eXtensible Markup Language, is a platform-independent and application-independent way of describing data using tags. XML (a subset of SGML) is similar to HTML in that it uses tags to describe document elements but different in that the tags describe the structure of the data rather than how the data is to be presented to a client. XML has the facility 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.`

XSL Stylesheets. XSL stylesheets are documents that describe a mapping between XML documents and visual data that can be presented to a client in a browser. XSL was a draft standard when this book was being written. The draft can be found at `http://www.w3.org/TR/WD-xs.`

Index

IBM Redbooks review

Your feedback is valued by the Redbook authors. In particular we are interested in situations where a Redbook "made the difference" in a task or problem you encountered. Using one of the following methods, **please review the Redbook, addressing value, subject matter, structure, depth and quality as appropriate.**

- Use the online **Contact us** review redbook form found at **ibm.com**/redbooks
- Fax this form to: USA International Access Code + 1 914 432 8264
- Send your comments in an Internet note to redbook@us.ibm.com

Document Number **Redbook Title**	SG24-5995-00 An Introduction to IBM WebSphere Everyplace Suite Version 1.1 Accessing Web and Enterprise Applications
Review	
What other subjects would you like to see IBM Redbooks address?	
Please rate your overall satisfaction:	O Very Good O Good O Average O Poor
Please identify yourself as belonging to one of the following groups:	O Customer O Business Partner O Solution Developer O IBM, Lotus or Tivoli Employee O None of the above
Your email address: The data you provide here may be used to provide you with information from IBM or our business partners about our products, services or activities.	 O Please do not use the information collected here for future marketing or promotional contacts or other communications beyond the scope of this transaction.
Questions about IBM's privacy policy?	The following link explains how we protect your personal information. **ibm.com**/privacy/yourprivacy/

IBM

Redbooks

An Introduction to IBM WebSphere Everyplace Suite Version 1.1 Accessing Web and Enterprise Applications

(0.5” spine)
0.475”<->0.873”
250 <-> 459 pages

An Introduction to IBM WebSphere Everyplace Suite Version 1.1

Accessing Web and Enterprise Applications

Build business solutions to reach anyone, anywhere, anytime

Support wireless networks, WAP, dial-up and Internet connections

Single sign-on to multiple applications and services

This redbook is about building business solutions using the IBM WebSphere Everyplace Suite Version 1.1 product to enable Web and enterprise application access from pervasive computing devices. It helps you to understand this product and focuses on implemented architectures and technologies included in this release such as wireless communications, transcoding, security, caching proxy, load balancing, messaging, and single sign-on, among others.

IBM WebSphere Everyplace Suite is an integrated end-to-end software solution for mobile e-business. In this redbook, you will find information that will help you plan to successfully implement solutions that businesses must address to be able to access Web and enterprise applications from desktop browsers and the new class of client devices such as WAP phones, Palm Pilots, WorkPads, and others.

A basic knowledge of HTTP and WAP protocols as well as some understanding of Web and Java technologies (XML, HTML, WML, servlets, and JSPs) and the terminology used in Web and enterprise applications is assumed.

SG24-5995-00 ISBN 0738419346