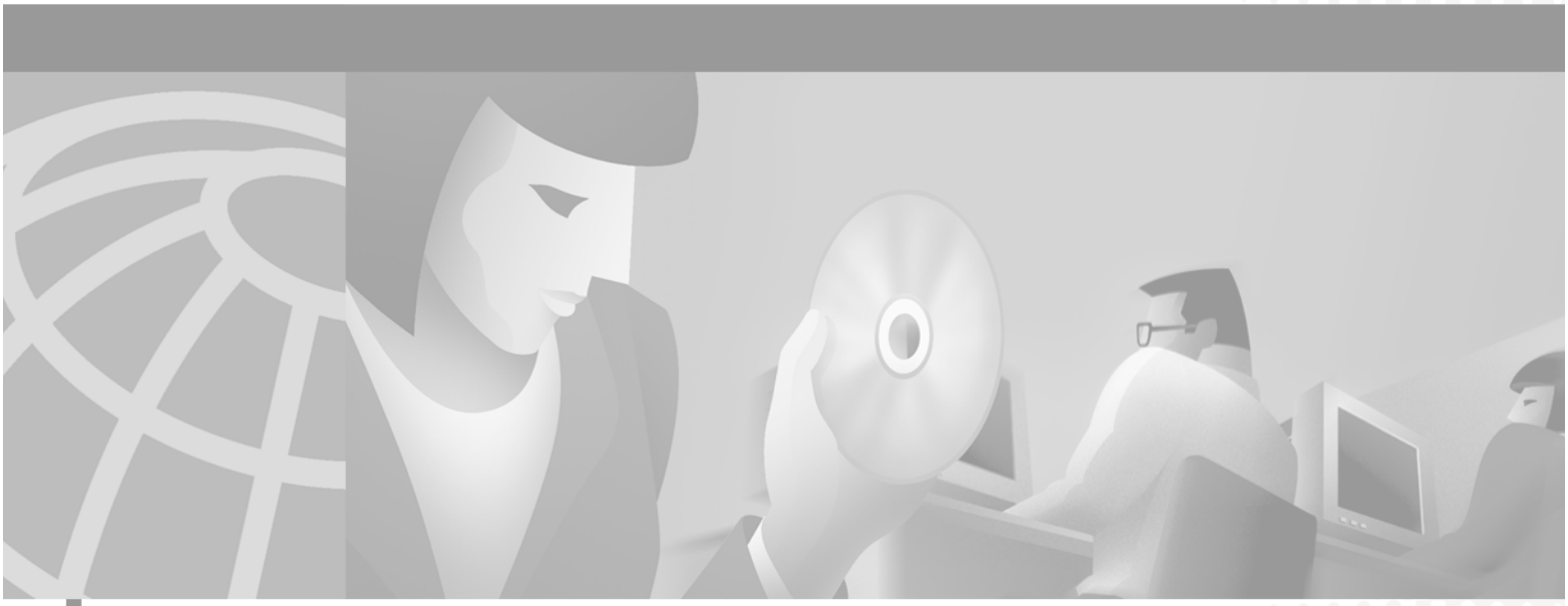


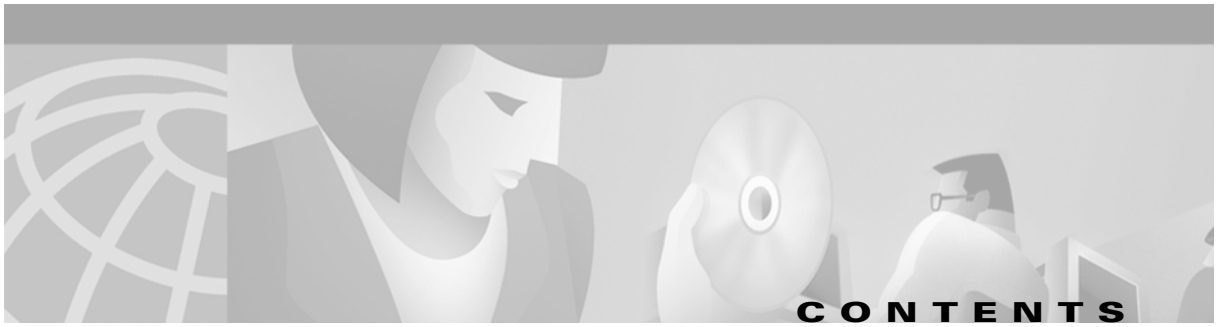
Cisco Subscriber Edge Services Manager Web Developer Guide

SESM Release 3.1(1)
August 2001

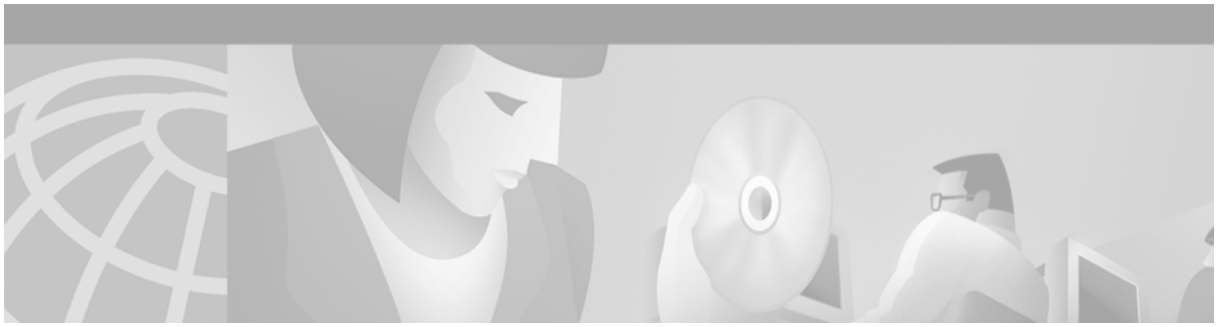
Corporate Headquarters
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134-1706
USA
http://www.cisco.com
Tel: 408 526-4000
800 553-NETS (6387)
Fax: 408 526-4100

Customer Order Number: DOC-7812940=
Customer Order Number: 78-12940-02

Cisco Subscriber Edge Services Manager Web Developer Guide



CONTENTS



About This Guide

This preface has information about the *Cisco Subscriber Edge Services Manager Web Developer Guide* and contains the following sections:

Document Objectives

This guide explains how to develop a Cisco Subscriber Edge Services Manager (Cisco SESM) web application. It describes SESM web application components and techniques.

Audience

This guide is intended for web designers and web developers responsible for the look-and-feel and branding of the service provider's web site. It is intended for web designers who will use the HTML design and integrate it with the SESM web components using JavaServer Pages.

Document Organization

This guide includes the following chapters:

Chapter	Title	Description
Chapter 1	SESM Web Development Overview	Provides an overview of a Cisco SESM system and an SESM web application.
Chapter 2	SESM Components and Techniques	Describes the SESM web application components and techniques.

Chapter	Title	Description
Chapter 3	New World Service Provider Web Application	Provides information on how a developer can use and modify the SESM components in the New World Service Provider sample web application.
Chapter 4	SESM Internationalization and Localization	Explains the SESM components and techniques that help a deployer internationalize and localize an SESM web application.
Index		

Document Conventions

The following conventions are used in this guide:

- **Boldface** font is used for user action, commands, and keywords.
- *Italic* font is used for emphasis, new terms, and elements such as a file name for which you supply a value.
- `Computer` font is used for code that appears on a JavaServer Page.

Note Means reader take note. Notes contain helpful suggestions or references to materials not contained in the manual.

Caution Means *reader be careful*. In this situation, you might do something that could result in equipment damage or loss of data.

Related Documentation

The following documents are relevant to Cisco SESM software and web development:

- *Release Notes for the Cisco Subscriber Edge Services Manager Release 3.1(1)*
- *Cisco Subscriber Edge Services Manager and Subscriber Policy Engine Installation and Configuration Guide*
- *Cisco Distributed Administration Tool Guide*
- *Cisco 6400 Feature Guide*

Obtaining Documentation

The following sections provide sources for obtaining documentation from Cisco Systems.

World Wide Web

You can access the most current Cisco documentation on the World Wide Web at the following sites:

- http://www.cisco.com
- http://www-china.cisco.com
- http://www-europe.cisco.com

Documentation CD-ROM

Cisco documentation and additional literature are available in a CD-ROM package, which ships with your product. The Documentation CD-ROM is updated monthly and may be more current than printed documentation. The CD-ROM package is available as a single unit or as an annual subscription.

Ordering Documentation

Cisco documentation is available in the following ways:

- Registered Cisco Direct Customers can order Cisco Product documentation from the Networking Products MarketPlace:

  http://www.cisco.com/cgi-bin/order/order_root.pl

- Registered Cisco.com users can order the Documentation CD-ROM through the online Subscription Store:

  http://www.cisco.com/go/subscription

- Nonregistered Cisco.com users can order documentation through a local account representative by calling Cisco corporate headquarters (California, USA) at 408 526-7208 or, in North America, by calling 800 553-NETS (6387).

Documentation Feedback

If you are reading Cisco product documentation on the World Wide Web, you can submit technical comments electronically. Click **Feedback** in the toolbar and select **Documentation**. After you complete the form, click **Submit** to send it to Cisco.

You can e-mail your comments to bug-doc@cisco.com.

To submit your comments by mail, use the response card behind the front cover of your document, or write to the following address:

Attn Document Resource Connection
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134-9883

We appreciate your comments.

Obtaining Technical Assistance

Cisco provides Cisco.com as a starting point for all technical assistance. Customers and partners can obtain documentation, troubleshooting tips, and sample configurations from online tools. For Cisco.com registered users, additional troubleshooting tools are available from the TAC website.

Cisco.com

Cisco.com is the foundation of a suite of interactive, networked services that provides immediate, open access to Cisco information and resources at anytime, from anywhere in the world. This highly integrated Internet application is a powerful, easy-to-use tool for doing business with Cisco.

Cisco.com provides a broad range of features and services to help customers and partners streamline business processes and improve productivity. Through Cisco.com, you can find information about Cisco and our networking solutions, services, and programs. In addition, you can resolve technical issues with online technical support, download and test software packages, and order Cisco learning materials and merchandise. Valuable online skill assessment, training, and certification programs are also available.

Customers and partners can self-register on Cisco.com to obtain additional personalized information and services. Registered users can order products, check on the status of an order, access technical support, and view benefits specific to their relationships with Cisco.

To access Cisco.com, go to the following website:

http://www.cisco.com

Technical Assistance Center

The Cisco TAC website is available to all customers who need technical assistance with a Cisco product or technology that is under warranty or covered by a maintenance contract.

Contacting TAC by Using the Cisco TAC Website

If you have a priority level 3 (P3) or priority level 4 (P4) problem, contact TAC by going to the TAC website:

http://www.cisco.com/tac

P3 and P4 level problems are defined as follows:

- P3—Your network performance is degraded. Network functionality is noticeably impaired, but most business operations continue.
- P4—You need information or assistance on Cisco product capabilities, product installation, or basic product configuration.

In each of the above cases, use the Cisco TAC website to quickly find answers to your questions.

To register for Cisco.com, go to the following website:

http://www.cisco.com/register/

If you cannot resolve your technical issue by using the TAC online resources, Cisco.com registered users can open a case online by using the TAC Case Open tool at the following website:

http://www.cisco.com/tac/caseopen

Contacting TAC by Telephone

If you have a priority level 1 (P1) or priority level 2 (P2) problem, contact TAC by telephone and immediately open a case. To obtain a directory of toll-free numbers for your country, go to the following website:

http://www.cisco.com/warp/public/687/Directory/DirTAC.shtml

P1 and P2 level problems are defined as follows:

- P1—Your production network is down, causing a critical impact to business operations if service is not restored quickly. No workaround is available.
- P2—Your production network is severely degraded, affecting significant aspects of your business operations. No workaround is available.

CHAPTER 1

SESM Web Development Overview

This chapter provides an overview of a Cisco Subscriber Edge Services Manager (Cisco SESM) system and a Cisco SESM web application. It also provides a high-level look at the web components and techniques used to develop a Cisco SESM web application.

Cisco SESM System

Cisco SESM is part of a Cisco solution that allows subscribers of DSL, cable, wireless, and dial-up to simultaneously access multiple services provided by different Internet service providers, application service providers, and Corporate Access Servers.

Cisco SESM allows a service provider to create a customized web application that provides a network portal for individual subscribers. Through the Cisco SESM's web-based network portals, subscribers can have simultaneous access to the Internet, corporate intranets, gaming, and other entertainment-based services. After logging on and being authenticated to the system, subscribers access their own personalized services by pointing and clicking.

Depending on the SESM software that is used, a deployed SESM web application can be configured for one of two modes:

- *RADIUS mode*—Service and subscriber information is stored in a RADIUS server.
- *DESS mode*—Service, subscriber, and policy information is stored in an LDAP-compliant directory, which is accessed with the Directory Enabled Service Selection (DESS) application programming interfaces of Cisco Subscriber Policy Engine (SPE).

A Cisco SESM web application allows subscribers to:

- Select or deselect services to which they are subscribed
- Subscribe to or unsubscribe from services they are authorized to access (DESS mode only)
- Change account details, such as address information and passwords (DESS mode only)
- Create subaccounts for other family members (DESS mode only)
- View the status of service connections
- View system messages

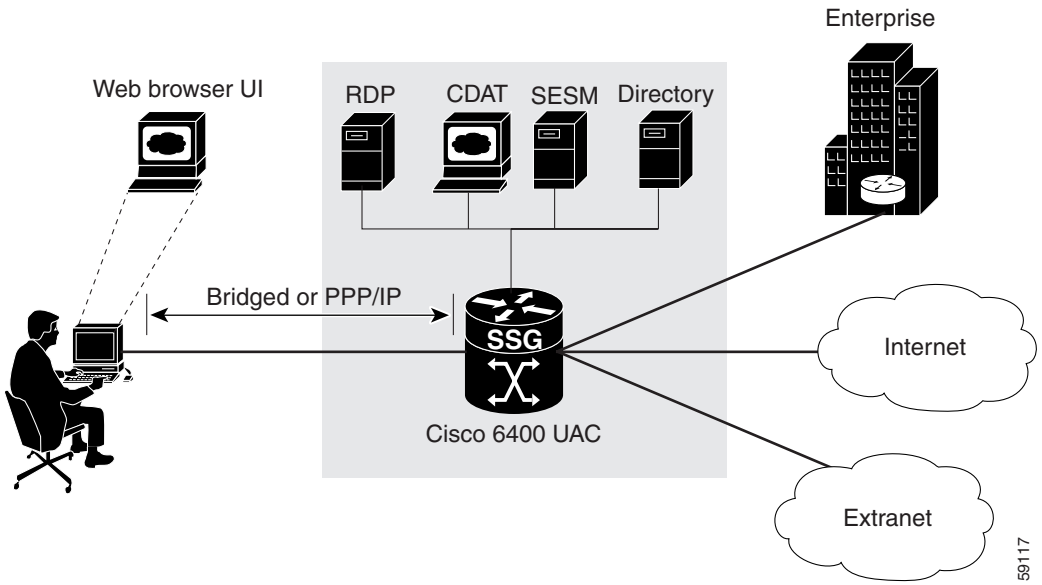
A Cisco SESM system includes one or more SESM servers running one or more Java 2 Enterprise Edition (J2EE) web servers.

The Cisco SESM system components (Figure 1-1) include the Cisco Service Selection Gateway (SSG), a software module that runs in a node route processor (NRP) router blade on a Cisco 6400 universal access concentrator (UAC).

Figure 1-1 shows an SESM system that includes the components for an LDAP-compliant directory implementation.

- RDP (RADIUS-DESS Proxy) is a proxy server that uses the DESS class libraries to translate RADIUS into Lightweight Directory Access Protocol (LDAP) so that service and subscriber information in an LDAP-compliant directory can be accessed.
- CDAT (Cisco Distributed Administration Tool) is a web-based facility for creating and maintaining service and subscriber information in an LDAP-compliant directory.

Figure 1-1 SESM System Components



For detailed information on SESM system components, see the *Cisco Subscriber Edge Services Manager and Subscriber Policy Engine Installation and Configuration Guide*.

Cisco SESM Web Applications

A Cisco SESM *web application* is a collection of associated web resources that can include JavaServer Pages (JSP), servlets, utility classes, static documents (such as HTML or WML files), images, and other data. An SESM web application includes:

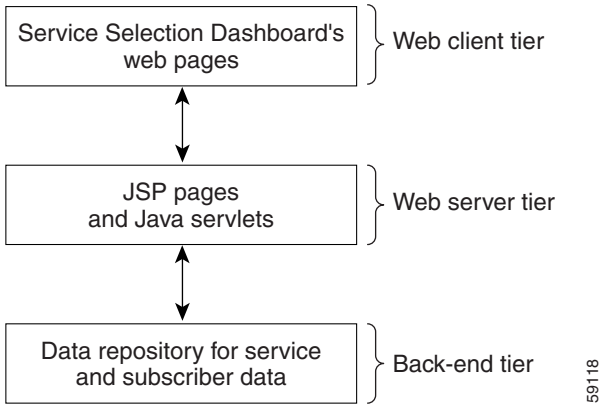
- A business model of the relationship between subscribers and services. In DESS mode, the business model supports service selection, service subscription, subscriber self-care, and service and subscriber management.
- Components that allow the SESM business model to communicate with other system components such as an SSG, a RADIUS AAA server, and an LDAP-compliant directory.
- Components for internationalization and localization of an SESM web application.
- Development guidelines that allow an SESM web application to be deployed and configured in the context of a J2EE web server.

The Jetty web server from Mortbay.com is installed with the SESM software—though a Cisco SESM web application can be deployed with any J2EE-compliant web server. From a development perspective, a Cisco SESM web application has three tiers:

- Web client tier—The subscriber accesses the SESM user interface through a web browser.
- J2EE web server tier—A set of JSP pages and Java servlets receives requests from the web client, processes the requests by communicating with back-end components, and renders and sends a reply to the client.
- Back-end tier—One or more data repositories (such as a RADIUS authentication, authorization, and accounting (AAA) server or an LDAP directory) store service and subscriber information.

Figure 1-2 illustrates this three-tiered perspective.

Figure 1-2 Three-tiered SESM Web Application



The interactions between an SESM web application and the back-end components are implemented with a set of specialized SESM programming interfaces. Currently, these interactions are preprogrammed in the JSP pages and configurable by the SESM deployer. Because these pieces of the web-application software do not require programming by the service-provider's developer, SESM web development can be completed in a much shorter period of time.

For the service provider, SESM web development involves two distinct roles:

- Web designer—Responsible for the HTML or WML design: the look, branding, and organization of the web site.
- Web developer—Responsible for integrating the HTML or WML design with the SESM web components using JSP pages.

The SESM software provides components and techniques for the customization and localization of the web pages. The web designer and web developer determine what customization is needed and use the SESM components and techniques to render the pages in the required manner.

Sample Cisco SESM Web Application

A complete sample Cisco SESM web application is included with the SESM software: New World Service Provider (NWSP). The NWSP web application gives the web developer a fully functional set of SESM web components that demonstrate much of what can be done in an SESM web application. The NWSP sample web application can also be used as the basis for creating a web application that meets the service provider's branding, look-and-feel, and other requirements.

Conventional Web Sites: A Problem

Most conventional web sites on the World Wide Web are structured in a manner that works well as long as certain assumptions about the user are met. For example, the user:

- Has a PC running Microsoft Windows.
- Has a browser that is a recent version of Microsoft Internet Explorer or Netscape Navigator.
- Understands English (with American spelling).

In the past, these assumptions were true for most users of conventional web sites.

Today, web-based interfaces must accommodate more variety on the World Wide Web in the form of:

- Diversity of devices including PCs, handheld computing devices, and smart phones
- Diversity of bandwidths and software technologies such as HTTP and WAP
- Diversity of markup languages including HTML, HDML, XML, and WML
- Diversity of languages and cultures

Sometimes it is required that the same information or services be provided to the same subscriber with a variety of applications, devices, and locales.

Customer circumstances such as the device, bandwidth, and language are outside of the control of the web site. Other customer characteristics are imposed by the web site itself. Is this a new or existing customer? With which brands is the web site associated? What level of service has the customer selected?

The problem with conventional web sites is that they cannot adapt their content and format to the variety of user characteristics that are currently on the web.

SESM Technology: A Solution

The set of characteristics (for example, device, brand, and locale) that define a subscriber is called the user's *shape*. The web components, directory hierarchy, and infrastructure of a Cisco SESM web application are designed to provide an SESM network portal that is customized for each user's shape.

Because an SESM web application is designed to detect and adapt to each user's shape, the Cisco SESM network portal can provide customer-tailored content and service offerings as well as a high degree of brand identity. As an example, an SESM web application can identify the location of the subscriber and provide location-specific pages and service offerings.

How does the Cisco SESM dynamically adapt to the user's shape? There are a number of possible strategies for providing customized content:

- Strategy 1—Make the existing web pages adapt for different types of users.
- Strategy 2—Create a separate set of web pages for each type of user.
- Strategy 3—A combination of strategy 1 and strategy 2.

Strategy 1 can make web pages complex, error prone, and difficult to maintain. For example, the presentation of a web page targeted for a Personal Digital Assistant (PDA) is different from the presentation of the same page targeted for a PC. These size differences often require dynamic HTML scripting to accommodate the differences in page content and layout. Implementing and maintaining these pages can be difficult.

Strategy 2 is often used when there is a requirement to support multiple languages. The entire web site is copied, and one language is replaced with another. For example, an English-language web site might be copied, and the English-language elements including text, currency symbols, formats of dates, and formats of numbers are replaced with Japanese-language elements. The end result is one web site for each language that is supported.

Strategy 2 ignores the commonality of elements on the multiple sets of web pages. As an example, the company logo might be the same on the pages in each web site but would need to be included in the set of resources in each web set. With strategy 2, when you change the logo on one web site, the change has to be replicated on the other web sites. This approach is time-consuming and error prone.

If you need to support English, Japanese, and Spanish, and desktop PCs, color PDAs, and monochrome PDAs, both strategy 1 and strategy 2 become more complicated. Strategy 1 requires complex dynamic HTML techniques, and strategy 2 requires nine separate web sites.

Strategy 3 is the approach to user customization that a Cisco SESM web application takes. This approach combines the most useful aspects of strategy 1 and strategy 2 while it attempts to minimize the drawbacks of both techniques. The web components, directory hierarchy, and infrastructure of a Cisco SESM web application provide easy-to-use mechanisms for customizing web content and format based on each user's shape.

- SESM web components—The SESM web components are designed so that they can be easily changed to meet the service provider's branding and look-and-feel requirements. The SESM web components include:
  - Easy-to-customize JSP pages
  - Templates and library items that allow an entire set of web pages to be updated when a template or library item is changed
  - Images and icons that are shipped with the native image-editor source files so they can be quickly modified
- SESM directory hierarchy—The SESM web application uses a *sparse tree directory structure*. With a sparse tree directory structure, though some web site resources may exist in more than one location (as in strategy 2), the need for multiple copies of the same resource is greatly reduced.
- SESM web application infrastructure—The SESM web application includes Java servlets, JSP pages, and specially designed Java classes that allow the SESM web site to be customized and localized for each subscriber.

For detailed information on each of the preceding elements of the SESM web application, see Chapter 2, "SESM Components and Techniques."

Hardware and Software Requirements for Development

On the development machine, the following software must be installed and set up correctly:

- Cisco SESM Release 3.1(1)
- Java 2 SDK, Standard Edition, Version 1.2.2 or later, which is available for downloading at:

  `http://java.sun.com/products/jdk/1.2/index.html`

The other hardware and software requirements for SESM web development include the same requirements that apply to deploying an SESM web application except that no separate Java Runtime Environment (JRE) is required if the Java 2 SDK is installed. For information on these requirements and how to install and configure the software, see the *Cisco Subscriber Edge Services Manager Installation and Configuration Guide*.

The development machine must have a J2EE-compliant web server installed and set up correctly. The Jetty web server is included with the SESM software. For information on installing and configuring a web server, refer to the instructions that apply to your web server.

Demo Mode

You can install or configure the SESM software for demonstration mode to observe how the NWSP web application works. Demonstration mode is also useful during some phases of web-application development because this mode does not require other system components such as a configured Cisco 6400 UAC and SSG. For more information on demonstration mode, see the "Demo Mode" section on page 2-24.

Environment Variables

The installation instructions for the Java 2 SDK describe how to set the required environment variables. For information on where to find the installation instructions, see the README file that is installed with the Java 2 SDK. Before you start Cisco SESM web development, check that the environment variables shown in Table 1-1 are set to the indicated locations.

Table 1-1 Java 2 SDK Environment Variables

Environment Variable	Value
`JDK_HOME`	The location of the Java 2 SDK Standard Edition installation if that SDK is used.
`PATH`	The location of the `bin` directory of the Java 2 SDK installation (for example, C:\jdk1.2.2\bin). This allows you to run the SDK executables from any directory.

If you use the CLASSPATH environment variable to tell the Java compiler where to find SESM-related class files, you need to set CLASSPATH to the location of these files. For information on the class libraries needed for compiling an SESM web application, see the "SESM Class Libraries" section on page 2-23.

Recommended Development Tools

The Cisco SESM software includes web components that were developed with Dreamweaver UltraDev 4, a visual editor for creating and managing web sites and pages, and Fireworks 4, a Web graphics design and production facility. We recommend these two state-of-the-art tools for SESM web development because with them the developer can create a customized SESM web application more quickly and easily.

Neither Dreamweaver UltraDev nor Fireworks is required to develop an SESM web application. However, you can more easily modify some SESM web components using Dreamweaver UltraDev and Fireworks. SESM web components that require Dreamweaver UltraDev and Fireworks include:

- Dreamweaver templates for creating and modifying page layout and appearance
- Dreamweaver library items for frequently reused images, text, and other objects
- Fireworks buttons with preprogrammed rollover behaviors

SESM images and icons are provided in Portable Networks Graphics (PNG) format so they can be easily customized in Fireworks. Image editors other than Fireworks may be limited in their ability to edit these PNG-formatted images and icons.

For information on Dreamweaver UltraDev 4 and Fireworks 4, see the Macromedia web site at:

```
http://www.macromedia.com
```

Both Dreamweaver UltraDev 4 and Fireworks 4 are available at the web site for a free 30-day evaluation.

Learning about SESM Web Application Development

The web developer should become familiar with the SESM components and techniques by reading this document and experimenting with a sample SESM application such as NWSP. Table 1-2 lists some topics with which the developer should be familiar.

Table 1-2 SESM Web Development Reading Path

For information on this topic	Read this Chapter or Section
Overview of SESM components and techniques	Chapter 1, "SESM Web Development Overview"
Using the SESM web components	"Using SESM Web Components" section on page 2-1
Customizing an SESM web application's look and feel	"Customizing an SESM Web Application" section on page 2-4
Rendering SESM web pages to match the user's shape	"Using a Sparse Tree Directory Structure" section on page 2-6 and "Using the Decorator Components" section on page 2-12
Developing an SESM web application	"Developing an SESM Web Application" section on page 2-22
Using the sample SESM web application components	Chapter 3, "New World Service Provider Web Application"
Internationalizing and localizing an SESM web application	Chapter 4, "SESM Internationalization and Localization"

For information on J2EE, web application development, and JSP pages, the web provides many resources including the Java Developer Connection at http://developer.java.sun.com/developer. Two recommended resources are:

- The book *Core Servlets and JavaServer Pages* (by Marty Hall, Sun Microsystems Press, 2000) is an excellent resource for learning about JSP pages.
- The *Java Servlet Specification* (Sun Microsystems, Inc., 2000) has some useful general information on web applications and other related topics. The PDF file containing the specification is at:

  ```
  http://java.sun.com/aboutJava/communityprocess/first/jsr053/servlet23_PFD.pdf
  ```

Java Servlet or JSP class library documentation is available in the /doc directory at the Java 2 SDK installation location.

If you plan to use Dreamweaver UltraDev or Fireworks as development tools and are not familiar with their use, those facilities have their own documentation, Help systems, and web resources. Some Dreamweaver templates in the NWSP sample web application use Fireworks 4 with Dreamweaver. A tutorial on using Fireworks and Dreamweaver together to edit web pages and graphics is at:

```
http://www.macromedia.com/support/fireworks/programs/fw_dw_tutorial
```

CHAPTER 2

SESM Components and Techniques

Use the Cisco SESM web application to dynamically render the look-and-feel of the user interface for each subscriber. This chapter describes the following topics:

The sections in the chapter progress, in order, from an explanation of the simplest, fastest-to-implement techniques to a description of full customization based on characteristics such as the device, brand, and locale of the subscriber.

The SESM web programming techniques described in this document are examples of how a developer might accomplish certain programming tasks. For illustration purposes, the explanations use the NWSP sample web application. Though there are certain programming tasks that must be performed in all SESM web applications, the developer decides techniques to use, techniques to modify, and techniques not to use based on the application's presentation and business requirements.

Using SESM Web Components

Each sample Cisco SESM web application (currently NWSP) includes a fully functional set of web components. This section provides a general description of those components. For specific information on the NWSP components, see Chapter 3, "New World Service Provider Web Application."

JavaServer Pages and Servlets

An SESM web application is implemented in a set of JSP pages and Java servlets. To develop an SESM web application, be familiar with the `Decorator` servlet and the JSP pages provided in a sample SESM web application such as NWSP. No servlet coding is required, and the Java coding in JSP pages is not extensive. The developer's JSP coding tasks are related to the decoration of the user shape. In NWSP, decoration of the user shape involves setting characteristics such as the device, brand, and locale for a specific subscriber.

For more information on JSP pages and servlets, see the "NWSP JavaServer Pages and Servlets" section on page 3-5.

JSP Tag Libraries

Use the Cisco SESM Localization tag library that helps reduce the complexity of localizing a web application. The Localization tag library uses a special SESM class (`L10nContext`) that improves upon the standard Java locale-related classes for use in a web application.

For more information on the Localization tag library, see the "Localization Tag Library" section on page 4-4.

Resource Bundles

The sample SESM web applications make use of resource bundles. Resource bundles contain locale-specific data that varies depending on the user's language and region, such as translatable text for status and error messages and for labels on GUI elements. A resource bundle allows a Cisco SESM deployer to separate localizable elements from the rest of the web application. The localizable elements are stored in a set of properties files, one for each language-region combination. Resource bundles allow the developer to design and write an SESM web application that can be easily localized for the subscriber's language and region. You can add additional resource bundles to a web application if a new locale is required.

For more information on resource bundles, see the "Resource Bundles" section on page 4-1.

Templates

The sample SESM web applications include one or more Dreamweaver templates. These files have the suffix `.dwt` and reside in the */webapp*/docroot/templates directory, where *webapp* is the name of the sample web application. Dreamweaver templates can be very useful for customizing or maintaining a web application's JSP pages when many pages have the same layout. By modifying a template and then updating the JSP pages that use the template, you can change the look and feel of an entire set of pages very quickly.

When a JSP page is derived from a template, the JSP page has locked regions that cannot be edited and editable regions that can be edited. The intention with locked regions is that if changes are required on a locked region, the changes are made in the template file. After the changes are made to the template, all files that use the template can then be automatically updated to incorporate the changes. You modify the template and update all occurrences on the web site using the Dreamweaver **update** commands in the Modify > Templates menu.

The use of Dreamweaver templates and automatic updating is a recommended approach but not a requirement.

If changes are required to individual JSP pages (as opposed to in a Dreamweaver template), the part of the individual JSP page that you can change appears between BeginEditable and EndEditable comments. For example:

```
<!-- #BeginEditable "main" -->
 <img src="../images/video_conferencing.gif" width="591" height="347">
<!-- #EndEditable -->
```

For more information on templates, see the "NWSP Templates" section on page 3-8 and the Dreamweaver UltraDev documentation.

Library Items

Dreamweaver library items contain Body elements such as images, text, and other objects that are reused throughout the JSP pages. For example, the NWSP sample web application contains library items for some user interface components, such as the buttons in the navigation bar. Library items have the file extension `.lbi` and are located in the */webapp*/docroot/library directory.

When you use a library item on a page, Dreamweaver inserts a copy of the HTML source code for the item into the file and creates a reference to the original, external item. This reference to the external item allows the contents of the library item to be updated wherever the item appears on the web site. To modify the external library item and update all occurrences on the web site, use the Dreamweaver **update** commands in the Modify > Library menu.

For more information on library items, see the "NWSP Library Items" section on page 3-13 and the Dreamweaver UltraDev documentation.

Images, Buttons, and Navigation Bars

Each sample SESM web application includes a complete set of customizable images, buttons, and a navigation bar. Fireworks graphics design tools were used to create the buttons for the NWSP service list and navigation bar, as well as many of the images.

Images

The images and icons in the NWSP web application are provided in two formats: GIF and PNG.

- The GIF-formatted image files are incorporated into the web pages. Their small file size makes the optimized GIF files suitable for downloading.
- The PNG-formatted files are used when the web designer edits the images and icons. Because PNG format is the native Fireworks file format, the images and embedded text are easily customizable. PNG files can also be read by other graphics applications, such as Photoshop.

Buttons

A Fireworks button is a type of rollover that has up to four different states for the user's pointer actions: Up, Over, Down, and Over While Down. When you export a button in Fireworks, the JavaScript that controls the button and the required HTML code is automatically created. When you import a Fireworks button library item into Dreamweaver, the JavaScript that controls the button and the required HTML code are automatically inserted into the JSP page or template.

In the NWSP sample web application, the navigation bar buttons are Fireworks buttons. For more NWSP navigation bar and Fireworks buttons, see the "Service and Settings Template" section on page 3-9 and the Fireworks documentation.

Navigation Bars

A Dreamweaver navigation bar (sometimes called a nav bar) is a set of buttons that appear on a series of related web pages and that provide a consistent mechanism for navigation between pages. For example, the NWSP web application contains a navigation bar (Figure 2-1) below the banner on most of its pages.

Figure 2-1 Navigation Bar

When modifying a Dreamweaver navigation bar, the developer can use the existing SESM buttons, create a new button, or use a button from the Fireworks button library. You can also use Fireworks to change the text on the existing SESM buttons. For more information on navigation bars, see the "Navigation Bar" section on page 3-12 and the Fireworks documentation.

Customizing an SESM Web Application

The simplest, fastest-to-implement approach to developing an SESM web application is to use the components in a sample SESM web application as a starting point and then:

1. Change the look-and-feel elements
2. Localize the web application

The customizable JSP pages and look-and-feel elements allow developers to create web pages that incorporate artistic flexibility and meet corporate identity standards without requiring extensive JSP or Java programming expertise.

The SESM software's use of resource bundles and properties files makes localization straightforward and easy to accomplish.

Changing the Look-and-Feel Elements

Many web developers will use the web components found in a sample SESM web application, such as the NWSP example, as starting point and customize those components. Simple customizations involve changing the look-and-feel elements of the components to meet the service provider's brand requirements.

In general, you can modify all static markup language elements and images (template text) that appear in a template, library item, or JSP page to meet the service provider's corporate standards. For example, you can change the static HTML elements for text and formatting in the set of NWSP web components. In these components, the developer can:

- Change the background colors and background images used in the JSPs.
- Modify or replace icons, images, and graphical elements used in the JSPs.
- Customize a style sheet. For example, the JSPs of NWSP use a Cascading Style Sheet that is included with the sample components. The style sheet defines several classes for textual elements that are used on the web pages.

In the NWSP sample web application, the Cascading Style Sheet is named `nwsp_styles.css` and is located in the /nwsp/docroot/styles directory.

Caution When you customize the web components in the NWSP sample web application, you must not change programmatic elements within the JSP pages in the /pages or /pages-ssm directories. On these JSP pages, do not change directives and code in scripting elements such as JSP expressions, scriptlets, and declarations. The programmatic elements in some of the JSP pages in the /decorator directory *do require* modifications. For more information, see the "Using the Decorator Components" section on page 2-12.

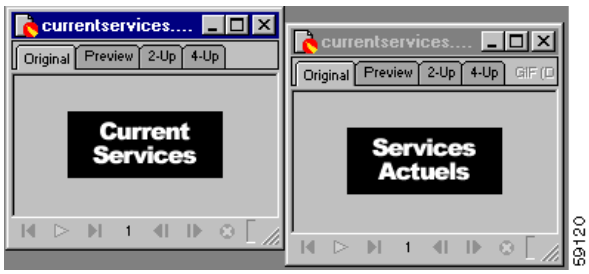
Localizing a Web Application

The SESM software provides a number of built-in mechanisms and techniques for internationalization and localization. An SESM web application can be implemented so that it dynamically detects each subscriber's language and country and renders SESM pages with resources appropriate for the language and country. For more information on these mechanisms and techniques, see the "Using a Sparse Tree Directory Structure" section on page 2-6 and the "Using the Decorator Components" section on page 2-12.

The sample SESM web components are intended for English language subscribers. The simplest form of localization is to create an SESM web site that uses a language other than English. Creating an SESM web site for another language can be accomplished with two sets of modifications.

The first set of modifications for the simplest form of localization involves changing SESM web application components so that they meet the needs of a language other than English. All JSP text that the subscriber will see as well as icons and images must be translated to accommodate the subscriber's language. For example, the `currentservices.gif` image in the NWSP sample contains the English language text "Current Services." If an SESM web application requires localization for the French language, the text in this image would need to be translated into the French language equivalent "Services Actuels" as shown in Figure 2-2.

Figure 2-2 Localizing currentservices.gif



The SESM web components include each of the images and icons in Portable Network Graphics (PNG) format, which is the Fireworks native format. The PNG images and icons are located in the /*webapp*/docroot/assets directory. You can change the text in a PNG image using Fireworks or another image editor and then export the GIF image to the *webapp*\docroot\images directory.

The second set of modifications for the simplest form of localization is that message text and other items in resource bundles must be translated, and an additional properties file must be created for the new language. The `createLocaleDimension.jspi` file includes the logic to determine and set the subscriber's locale. For information on resource bundles and setting the locale for a specific subscriber, see the "Resource Bundles" section on page 4-1 and "createLocaleDimension.jspi" section on page 2-17.

Using a Sparse Tree Directory Structure

A Cisco SESM web application can use a directory hierarchy that is structured for customizing the SESM web site for each user's shape. The Cisco SESM directory structure combines two hierarchies:

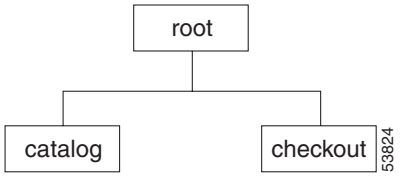
- Web site pages hierarchy
- User shape hierarchy

These two hierarchies are developed independently of each other. However, all web resource files are located within one web application because you combine the two hierarchies into one large hierarchy whose root is the web application. For this discussion, a web *resource* might be a JSP page, HTML file, GIF image, or another web application component.

Web Site Pages Hierarchy

A *web site pages hierarchy* is the directory structure of a conventional web site, which typically includes a single copy of each required resource. For example, consider a web application where the document root contains a page named welcome.html, and subdirectories are named /catalog and /checkout, containing catalog.html and checkout.html respectively. Figure 2-3 shows the web site pages hierarchy.

Figure 2-3 Web Site Pages Directory Hierarchy



User Shape Hierarchy

A *user shape* is a set of characteristics that define the web resources available for a specific subscriber. The shape of a user can include characteristics such as the following:

- Devices and browser software used to connect to the web site
- Branding for the web site, such as a brand for business use and a brand for personal use
- Language and country of the user
- Personalization of the web site for a specific user

The characteristics that define a user shape are application-specific and can include characteristics other than the preceding ones.

The user shapes that need to be accommodated by a given Cisco SESM web application determine the application's user shape hierarchy. A *user shape hierarchy* is the directory structure of an SESM web site, which may include one or more different instances of each required resource.

A Cisco SESM web application includes a set of Java classes that implement the user shape model. For example, a Cisco SESM web application makes use of a `Shape` class that encapsulates a user shape and allows the appropriate web resources to be used for a specific subscriber. For information on the programmatic details for implementing the user shape model, see the "Using the Decorator Components" section on page 2-12.

The following example describes how user shapes help determine the user shape hierarchy. Assume that the user shapes that need to be accommodated by a Cisco SESM web application have the following characteristics:

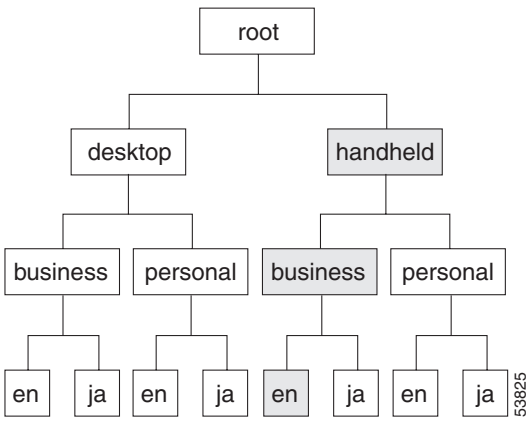
- Devices: desktop and handheld
- Brands: business and personal
- Locales (languages): English (en) and Japanese (ja)

Each of these general characteristics (devices, brands, and locales) of the user shape is a *dimension*, and each dimension has one or more *values*. The user shape in this example has three dimensions and two values for each dimension. Each subscriber's user shape has a specific value for each dimension. In this example, the eight possible user shapes are:

- desktop, business, English
- desktop, business, Japanese
- handheld, business, English
- handheld, business, Japanese
- desktop, personal, English
- desktop, personal, Japanese
- handheld, personal, English
- handheld, personal, Japanese

When the directory hierarchy is implemented, the value for each dimension is used for a directory name. In the following example, one or more directories exist named desktop, handheld, business, personal, en, and ja. Figure 2-4 shows the user shape directory hierarchy.

Figure 2-4 User Shape Directory Hierarchy



For each specific user shape, an SESM web application specifies the directories in the user shape hierarchy that the Cisco SESM software uses to produce the path for locating a web resource. In Figure 2-4, each of the eight leaves in the directory tree represents one possible user shape. For example, the shaded branch and leaf identifies the user shape that has the values "handheld, business, and English."

Each of the dimensions could be extended. For example, the third dimension (locales) could be extended to include French and Spanish. The number of dimensions and the range of values allowed for each dimension is, in theory, unlimited.

Location of Directory Dimensions

Two factors should be considered when you decide where a dimension appears within the user shape directory hierarchy.

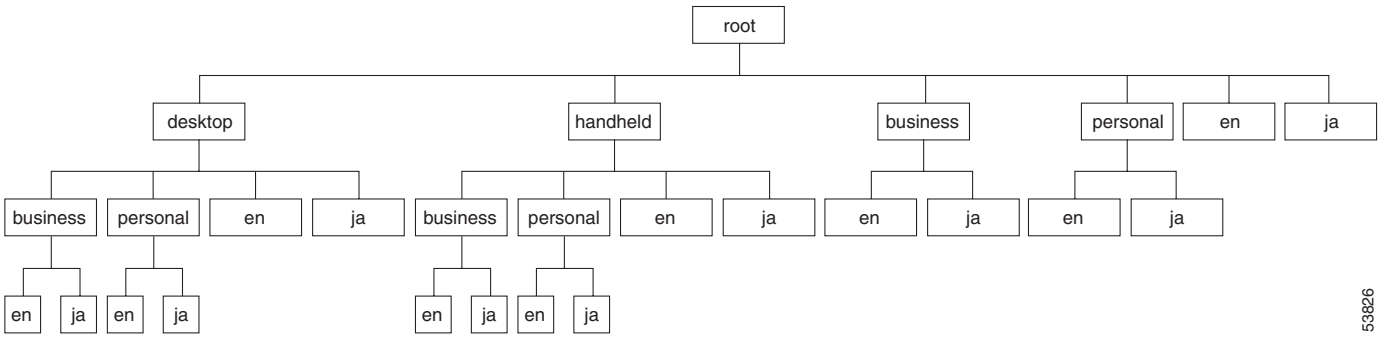
- In general, the more frequently a value appears in the hierarchy, the further down in the hierarchy it is located. As an example, in Figure 2-4 the values of the first dimension (desktop and handheld) appear only once in the hierarchy, but the values of the last dimension (en and ja) appear many times.
- The order of dimensions in the hierarchy determines the order in which directories are searched for a web page. For more information on the search order, see the "Searches for a Web Resource" section on page 2-9.

Except for search order, the dimensions in the hierarchy are independent of each other.

Sparse Tree Directory Structure

The directory hierarchy that a Cisco SESM web application uses is a combination of the web page hierarchy and the user shape hierarchy. An instance of each resource of the web page hierarchy can reside in every node of the user shape hierarchy. Figure 2-5 shows the user shape directory hierarchy expanded to include all possible combinations of dimensions.

Figure 2-5 Fully Expanded User Shape Directory Hierarchy



An instance of each web resource is not required for each node, but an instance can exist in each node. For example, the resource catalog.html can—but is not likely to—reside in all of the directories shown in Figure 2-5. For example, the catalog.html file, which is located in a /catalog directory, could reside in:

- /catalog/catalog.html
- /handheld/catalog/catalog.html
- /personal/catalog/catalog.html
- /desktop/personal/ja/catalog/catalog.html

The fully expanded hierarchy shown in Figure 2-5 is not likely in a real-life deployment. The typical deployment makes use of a *sparse tree directory structure* because some directories can be omitted. The reasons to omit a directory include:

- If the web resources are identical for all values of a dimension, that dimension can be eliminated. As an example, in Figure 2-4, if the web resources for the devices (desktop and handheld) dimension are identical, that dimension can be omitted. If the web resources for the brands (business and personal) dimension are identical, that dimension can be omitted.
- An empty directory or a directory that contains only empty directories can be pruned from the directory tree. A directory is empty if no user shape will exist for that set of values. For example, if there are no Japanese business users, the /business/ja directory can be omitted.

The web resources that the SESM software finds for a particular user shape can be located in different directories in the directory hierarchy. No one directory in the hierarchy is likely to contain all the resources for a particular user shape.

Implementing the Sparse Tree Directory

The service provider's web developer might implement the sparse tree directory in the following manner:

1. Locate the entire web site page hierarchy at the root directory of the user shape hierarchy. This directory structure will appear as a typical web site.
2. Where required, create a specialized version of a web resource and copy it to the appropriate subdirectory of the user-shape hierarchy.

For example, assume that a logo image for a desktop PC is of high resolution and 32 x 32 pixels, and the logo image for a handheld PC is of a lower resolution and 16 x 16 pixels. The same image is used for business and personal use and by English and Japanese users. The two images, both named /images/logo.gif, are copied into these locations:

- /desktop/images/logo.gif
- /handheld/images/logo.gif

When the web resource /images/logo.gif is requested, the Cisco SESM software uses one of the preceding in the response depending on the value specified for the devices dimension (desktop or handheld) of the subscriber's user shape.

Searches for a Web Resource

When the Cisco SESM web application software searches the sparse tree directory for a web resource, it uses the algorithm described in this section's examples.

Example 1: Searches for a Web Resource

For Example 1, assume the set of user shapes described in the "User Shape Hierarchy" section on page 2-6. In addition, assume that unique /images/logo.gif files are required for Japanese users—one logo for desktop and one for handheld. The logo.gif resources for Japanese-language users reside in:

- **/desktop/ja**/images/logo.gif
- **/handheld/ja**/images/logo.gif

Two additional logo.gif resources for non-Japanese users reside in:

- **/desktop**/images/logo.gif
- **/handheld**/images/logo.gif

To understand the search algorithm, it is necessary to be familiar with one of the programmatic pieces of a Cisco SESM web application. The web application's `createShape.jspi` file uses information gathered about the subscriber and the session to specify the directory paths that will be used for each dimension in the subscriber's user shape.

Initialization of dimensions Array

The order in which the Cisco SESM software searches for a resource is controlled by the order in which the web application's `createShape.jspi` file initializes one of its data objects: the `dimensions` array. In this example, the `dimensions` array specifies three directory paths, one path for each dimension in the user shape. The dimensions are devices, brands, and locales. The following code in `createShape.jspi` initializes the `dimensions` array with three elements, each of which is a request attribute containing a directory path. To simplify this description, the dimensions are designated as A, B, and C.

```
Dimension[] dimensions =
{
      (Dimension)request.getAttribute("shape.device"),    // dimension A
      (Dimension)request.getAttribute("shape.brand"),     // dimension B
      (Dimension)request.getAttribute("shape.locale"),    // dimension C
};
```

The "createShape.jspi" section on page 2-18 provides detailed information about the `dimensions` array and its use.

For a subscriber with the user shape "desktop, business, ja", assume that the SESM web application specifies that the directory paths associated with the dimensions A, B, and C are as follows:

Dimension	Directory Path
A (devices dimension)	/desktop
B (brands dimension)	/business
C (locales dimension)	/ja

Order of Paths Searched

Given the preceding `dimensions` array initialization, user shape, and directory paths, the Cisco SESM software searches the locations shown in Table 2-1 (in the order listed) for /images/logo.gif. As shown in Table 2-1, the search is carried out as follows:

- If the web resource is found at path 1 (**/desktop/business/ja/**images/logo.gif), the Cisco SESM web application uses that resource in its response to the client.
- If the web resource is not found at path 1, the Cisco SESM continues the search at path 2 (**/desktop/business/**images/logo.gif).

The search for /images/logo.gif continues in the order shown in Table 2-1 until the Cisco SESM software finds the resource.

Table 2-1 SESM Search Algorithm: Example 1

Search Order	Directory Path (from document root)	Path Assembly
1	**/desktop/business/ja/**images/logo.gif	A/B/C
2	**/desktop/business/**images/logo.gif	A/B
3	**/desktop/ja/**images/logo.gif	A/C
4	**/desktop/**images/logo.gif	A
5	**/business/ja/**images/logo.gif	B/C
6	**/business/**images/logo.gi	B
7	**/ja/**images/logo.gif	C
8	/images/logo.gif	none (document root)

In the Example 1 search, the Cisco SESM software first finds the web resource /images/logo.gif at /desktop/ja/images/logo.gif and uses that resource in its response to the client. Notice the following about the example:

- The manner in which the directory paths are assembled is determined by the `dimensions` array initialization in `createShape.jspi`. The order of the elements in the `dimensions` array initializer is A, B, and C. The assembly of the directory paths for the search mirrors this order.
- The manner in which directory paths are searched is also determined by the `dimensions` array initialization in `createShape.jspi`. In the search, the array's last element (C) is the most persistent and is discarded last. The array's first element (A) is the least persistent and is discarded first.

Example 2: Searches for a Web Resource

For this example, assume that the user shapes are as described in the "User Shape Hierarchy" section on page 2-6. However, now assume that an /images/button.gif file resides in the following locations:

- **/desktop/**images/button.gif
- **/business/**images/button.gif
- **/ja/**images/button.gif

Assume that Example 2 uses the same `dimensions` array initialization and directory paths as in Example 1. For a user with the shape "desktop, business, ja", the Cisco SESM web application searches the locations shown in Table 2-2, in the order listed, for /images/button.gif.

Table 2-2 SESM Search Algorithm: Example 2

Search Order	Directory Path (from document root)	Path Assembly
1	**/desktop/business/ja/**images/button.gif	A/B/C
2	**/desktop/business/**images/button.gif	A/B
3	**/desktop/ja/**images/button.gif	A/C
4	**/desktop/**images/button.gif	A
5	**/business/ja/**images/button.gif	B/C

Table 2-2 SESM Search Algorithm: Example 2 (continued)

Search Order	Directory Path (from document root)	Path Assembly
6	**/business**/images/button.gi	B
7	**/ja**/images/button.gif	C
8	/images/button.gif	none (document root)

In the Example 2 search, the Cisco SESM software first finds the web resource /images/button.gif at /desktop/images/button.gif and uses that resource in its response to the client. In the search, notice that the last element in the array (C) is the most persistent, and the first element in the array (A) is the least persistent.

Using the Decorator Components

This section describes how a developer customizes an SESM web application's components for user-shape decoration.

An SESM web application includes a group of JSP components that are responsible for *decoration* of the user shape: the setting of characteristics such as the device, brand, and locale for a specific subscriber. The components that the developer uses for decoration of the user shape reside in the /docroot/decorator directory and perform these tasks:

- SESM session handling
- Subscriber authentication
- User-shape decoration

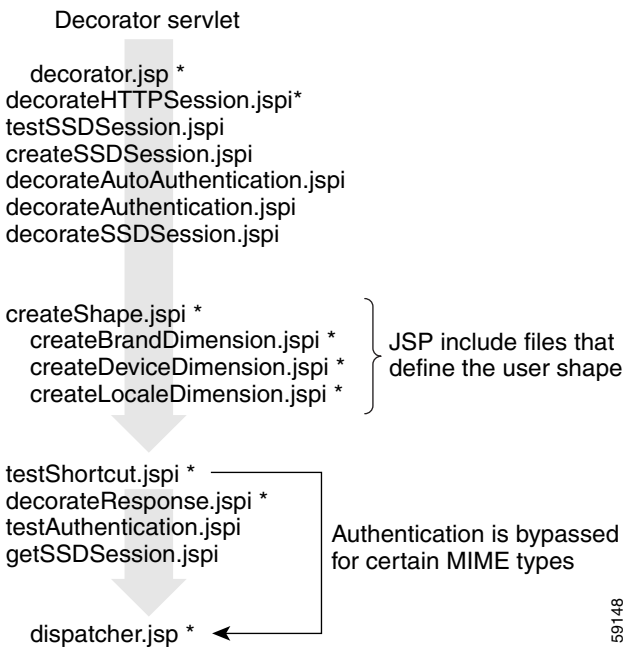
The JSP web components in /docroot/decorator for HTTP session handling and subscriber authentication work without modification. This section focuses on the use of the user-shape decoration components. However, the service-provider developer can modify any web component in the /docroot/decorator except the classes in the SESM JAR files.

Note Before you read this section on using the decorator components, read the "Using a Sparse Tree Directory Structure" section on page 2-6. The techniques for user-shape decoration require that the structure of the SESM web site and the programming techniques for user-shape decoration take a coordinated approach. The explanations in the two sections complement each other.

After an SESM web application receives a request for a web resource from a subscriber, the page-to-page flow that occurs between the SESM components in the /docroot/decorator directory is shown in Figure 2-6. All HTTP requests go first to the `Decorator` servlet and, unless an error occurs, end with the `dispatcher.jsp` page. The `dispatcher.jsp` constructs the URI for the requested web resource.

In Figure 2-6, the customizable web components that perform user-shape decoration tasks are marked with an asterisk. Only three of the `create*Dimension.jspi` files are included in Figure 2-6. The explanations in this chapter focus on the three `create*Dimension.jspi` files that are shown in the figure: `createBrandDimension.jspi`, `createDeviceDimension.jspi`, and `createLocaleDimension.jspi`.

Figure 2-6 JSPs Pages and Include Files for User-Shape Decoration



An SESM web application uses these web components for user-shape decoration:

- Decorator servlet
- Two JSP pages: decorator.jsp and dispatcher.jsp
- A set of JSP include files

The *JSP include files* have the extension .jspi and are included into decorator.jsp or into another .jspi file. The JSP include files modularize the code in the decorator JSP pages. Unlike JSP pages, the .jspi files cannot be directly compiled or requested.

Scripting Variables

With JSP pages, a *scripting variable* is a Java object created by a `jsp:useBean` action and accessible in a JSP scriptlet. The Decorator servlet and the JSP pages and include files for user-shape decoration make use of three scripting variables:

- uri—When an HTTP request is received, the `Decorator` servlet stores a partial Uniform Resource Identifier (URI) for the requested resource in the uri variable. For more information on this partial URI, see the "Decorator Servlet" section on page 2-14.
- decorator—When the Decorator servlet is invoked, the servlet stores the path name of decorator JSP page (in NWSP, decorator.jsp) in the decorator variable.
- dispatcher—When the Decorator servlet is invoked, the servlet stores the path name of the dispatcher JSP page (in NWSP, dispatcher.jsp) in the dispatcher variable.

The developer uses the decorator and dispatcher initialization parameters in the web application's web.xml file to specify the paths for the decorator and dispatcher variables. The paths specified for the decorator and dispatcher parameters are relative to */docroot*. The `Decorator` servlet uses these parameters to set the values of the two corresponding scripting variables.

The three scripting variables are of the type java.lang.String and have request scope. They can be accessed in a JSP page by declaring them as Java bean instances of the same name. The following jsp:useBean action, in effect, is a declaration for the uri scripting variable:

```
<jsp:useBean id="uri" class="java.lang.String" scope="request" />
```

When the preceding bean specification appears on a JSP page, the scripting variable `uri` can be accessed in scriptlet code on that page or any included page. For an example of the use of a scripting variable, see the "dispatcher.jsp" section on page 2-21.

Decorator Servlet

The `Decorator` servlet performs some of the session-management and user-shape decoration functions for an SESM web application. The `Decorator` servlet is *not* customizable by the deployer. In the NWSP sample web application, the `Decorator` servlet is mapped to the name `decorate`, but it can be mapped to another name. The servlet name is specified in the web.xml file.

All HTTP requests to an SESM web application go first to the Decorator servlet. All HTTP requests to Decorator must include extra path information. The extra path information is a partial URI for the resource (usually a JSP) that the subscriber is requesting. The Decorator servlet copies the extra path information into the uri variable so that it can be accessed by dispatcher.jsp. In the following example, where webapp is the web application prefix, and the Decorator.java servlet class is mapped to the name decorate. The Decorator servlet receives the following HTTP request:

```
http://www.someserver.com/webapp/decorate/pages/accountLogoff.jsp?name=Smith
```

The `Decorator` servlet adds the extra path information to the HTTP request using the `uri` scripting variable. The `uri` variable is of the type `java.lang.String` and for this example has the value:

```
"/pages/accountLogoff.jsp?name=Smith"
```

If the HTTP request contains no extra path information, the `uri` scripting variable is not set.

When the HTTP request is forwarded to `dispatcher.jsp`, that JSP constructs the complete URI using additional directory path information contained in the subscriber's `Shape` object. For information on the construction of the complete URI by `dispatcher.jsp`, see the "dispatcher.jsp" section on page 2-21.

Decoration JSP Pages and Include Files

This section describes how a developer modifies the JSP pages and include files that are used for user-shape decoration. Some modifications to the JSP pages and include files are required because these components implement the business and presentation requirements of an SESM web application. Other modifications are optional and are made at the discretion of the developer.

decorator.jsp

The `Decorator` servlet forwards each HTTP request to the `decorator.jsp` page. Though `decorator.jsp` is a deployer-customizable JSP page, it usually needs little or no modification. The `decorator.jsp` page does the following:

- Starts SESM session handling, subscriber authentication, and user-shape decoration
- Specifies the `include` directives that control top-level, page-to-page flow for the JSP include files
- In the normal case, ends decoration by forwarding the HTTP request to `dispatcher.jsp`

decorateHTTPSession.jspi

The `decorateHTTPSession.jspi` file is executed once per HTTP session. In an SESM web application, the developer can use `decorateHTTPSession.jspi` to perform application-specific decoration tasks prior to SESM session handling or subscriber authentication. For example, `decorateHTTPSession.jspi` can be used to:

- Initialize application-specific, session attributes
- Set a default user shape

If a default user shape is set in `decorateHTTPSession.jspi`, a shape customized for the subscriber can be defined after the user is authenticated.

In the NWSP web application, `decorateHTTPSession.jspi` sets a number of session attributes for characteristics such as the subscriber's operating system and browser software. NWSP uses the session attributes later in JSP include files such as `createOSDimension.jspi` and `createBrowser.jspi` to specify dimensions of the subscriber's shape.

Setting the Dimensions of the User Shape

When the user-shape decoration mechanism is used, an SESM web application uses information gathered about the subscriber and the session to specify the directory paths that are used for each dimension in the subscriber's user shape. The sample NWSP web application uses many dimensions when defining the user shape of each subscriber: a device dimension, a brand dimension, a locale dimension, a browser dimension, an operating system dimension, and so on. The number of dimensions to use and the characteristics associated with each dimension are application-specific.

An SESM web application determines the values to use for each dimension for a particular subscriber through a variety of mechanisms:

- The browser and device can be determined by the HTTP request header field User-Agent. With Windows-CE devices, screen size and color are indicated by the request header fields UA-pixels and UA-color.
- The brand can be determined by application-specific mechanisms.
- The locale can be determined by the HTTP request header fields Accept-Charset and Accept-Language.
- The location can be determined by implication from the SSG configured location, whose attributes are specified in the nwsp.xml file.

For information on creating specific dimensions of a user shape, examine the code and comments in the `create*Dimension.jspi` files like `createDeviceDimension.jspi` in the /nwsp/docroot/decorator directory.

Note For this discussion, assume that the user shape is limited to three dimensions to correspond to the sample user shape hierarchy discussed in the "User Shape Hierarchy" section on page 2-6. The sample web applications like NWSP may include more than three dimensions.

An SESM web application uses one or more JSP include files to set the values for the dimensions of the user's shape. Each value is the directory path that the SESM software uses to find a web resource for the specific subscriber. The sample NWSP web application defines a number of dimensions for the subscriber's shape. This description of NWSP focuses on three JSP include files that are used for this purpose: `createDeviceDimension.jspi`, `createBrandDimension.jspi`, and `createLocaleDimension.jsp`.

In the sample NWSP web application, the user's shape is set on the first request. On subsequent requests, the web application determines whether the shape has been set and, if it has been set, does not reset the shape. In a production SESM web application, the web application might set a default shape for unauthenticated users in `decorateHTTPSession.jspi`. When the user is authenticated, the SESM web application can then access stored user-account information and can use that information and other information to reset the user shape so that it is customized for the user. The stored user-account can include information on age, gender, and interests that the web application might consider when defining the user shape.

createDeviceDimension.jspi and createBrandDimension.jspi

The `createDeviceDimension.jspi` and `createBrandDimension.jspi` files are similar. In a production SESM web application, after the device or brand is determined, these files set the `device` and `brand` request attributes to a single directory name or list of directory names that defines, respectively, the device and brand of the user's shape.

As an example, assume the relevant scriptlet code from `createDeviceDimension.jspi`, which sets the value of the `device` request attribute to a list of directory names, is as follows:

```
request.setAttribute("shape.device", new Dimension("color/desktop/IE5", "/"));
```

In the preceding example, the `Dimension` constructor takes two arguments: the first argument is a directory path (a concatenation of directory names), and the second argument is a delimiter (in the example, the / character) that separates the names. The result is an *ordered list of directory names* that the SESM software uses to search for the web resource.

Given the preceding example, if only the devices dimension were specified as the user's shape, the search for a web resource proceeds as shown in Table 2-3. The search ends as soon as the Cisco SESM software finds the web resource. With the user-shape decoration mechanism, the search for a web resource usually involves more than one dimension. For more information on the search algorithm that the SESM software uses to find a web resource, see the "Searches for a Web Resource" section on page 2-9.

Table 2-3 Directory List Searches within a Single Dimension

Search Order	Directory Path (from document root)
1	**/color/desktop/IE5/***resource*
2	**/color/desktop/***resource*
3	**/color/***resource*
4	/*resource*

If a Cisco SESM web application uses a JSP include file such as `createDeviceDimension.jspi` to set the directory path for a dimension, that JSP include file is an appropriate location to also retrieve the specific subscriber characteristics (for example, the subscriber's browser) that will determine the dimension's value. In NWSP, the sample `createDeviceDimension.jspi` and `createBrandDimension.jspi` pages require deployer-specific modifications to retrieve the subscriber's characteristics.

createLocaleDimension.jspi

The `L10nContext` class is a special library that combines the locale-related characteristics of the subscriber, including language, country, time zone, and resource bundle base name. The `createLocaleDimension.jspi` file uses JSP tags from the Localization tag library to define attributes of an `L10nContext` object:

- The resource bundle base name for an SESM web application
- The subscriber's preferred locales
- A default (`otherwise`) locale to use if a resource bundle for the preferred locales cannot be found

In the NWSP web application, `l10n` is the prefix for the Localization library. The `l10n:context` tag is used in `createLocaleDimension.jspi` as follows:

```
<%@ taglib uri="http://www.cisco.com/taglibs/localization" prefix="l10n" %>

...

<%-- Set resource bundle name --%>
<l10n:context
     resourceBundleName="messages"
     scope="<%= pageContext.SESSION_SCOPE %>"
/>

<%-- Set locale --%>
<%
     String preferredLocales = request.getHeader("Accept-Language");
     Log.debug("createLocaleDimension.jspi, preferredLocales=", preferredLocales);
     if (preferredLocales != null)
     {
          //Use the locales preferred by the HTTP client.
          //Set this for the entire HTTP session.
          //If the user changes their preferences, this scriptlet will
          //need to be run again.
          //At the time of writing it is run every time a new SESM Session
          //is decorated and every time a new client authentication is
          //decorated.
          %>
          <l10n:context
               preferredLocales="<%= preferredLocales %>"
               otherwise="<%= Locale.UK %>"
               scope="<%= pageContext.SESSION_SCOPE %>"
          />
          <%
     }
%>

<%-- Set the "locale" Dimension of the client Shape. --%>
<l10n:context variable="l10nContext">
<%
     String locale = l10nContext.getLocale().toString();
     request.setAttribute("shape.locale", new Dimension(locale, "_"));
%>
</l10n:context>
```

As shown in the preceding example, `createShape.jspi` is responsible for performing the following tasks in the order shown:

1. In the first use of the `l10n:context` tag, the `resourceBundleName` attribute specifies the bundle's base name and is set to `"messages"`. In the sample NWSP web application, the properties files reside in the /docroot/Web-inf/classes directory and have the base name `messages` (for example, `messages_en.properties`). The scope of the localization context is set to `pageContext.SESSION_SCOPE` so that it persists for the entire session. For information on localization and properties files, see Chapter 4, "SESM Internationalization and Localization."
2. In the second use of the `l10n:context` tag, the `preferredLocales` attribute is set to the subscriber's preferred locale. The client browser sends the subscriber's preferred locale in the `Accept-Language` request header, which `createLocaleDimension.jspi` obtains using the `request.getHeader` method.

   In the second `l10n:context` tag, the `otherwise` attribute is set to the locale that will be used if the SESM software cannot find a resource bundle for any of the preferred locales.
3. After getting a string for the current locale with the `getLocale` method of the `L10nContext` class, `createLocaleDimension.jspi` sets the `"shape.locale"` request attribute to the value of the locale. The `"shape.locale"` request attribute is set to the value of a `Dimension` object, which is initialized using a directory list (in this case, a locale and country) and the _ character as the delimiter for the elements of the directory list. For example, if fr_CA were the locale, the two elements of the list are fr and CA.

createShape.jspi

After the NWSP web application defines the device, brand, locale, and other dimensions of the user shape, the `createShape.jspi` file creates a `Shape` object composed of the three dimensions. In NWSP, the relevant code in the sample `createShape.jspi` is as follows:

```
<%-- Create a Shape from the dimensions.             --%>
<%-- The order of the Dimensions can be re-arranged. --%>
<%
     Dimension[] dimensions =
     {
           ...
          (Dimension)request.getAttribute("shape.device"),
          (Dimension)request.getAttribute("shape.brand"),
          (Dimension)request.getAttribute("shape.locale"),
     };
     Shape shape = new Shape(dimensions);
     session.setAttribute("shape", shape);
```

As shown in the preceding example, `createShape.jspi` is responsible for performing the following tasks in the order shown:

1. Creates and initializes the `dimensions` array, which contains the values (directory paths) for the subscriber's device, brand, and locale. Each of the directory paths is stored in a request attribute with a name that corresponds to the dimension.
2. Creates a `Shape` object for the subscriber. The `dimensions` array is specified as the argument of the `Shape` constructor.
3. Assigns the value of the `shape` object to the session attribute `shape`.

testShortcut.jspi

The `testShortcut.jspi` file provides a mechanism for bypassing the subscriber authentication tests and the remainder of the decoration phase when retrieving certain categories of web resources. The bypass mechanism is necessary so that images, style sheets, JavaScripts, and possibly other resources can be obtained for the web application's JSP pages before the subscriber logs on. Otherwise, the HTML content of the logon pages would not have the proper web resources. Additionally, after the subscriber is logged on, testing for an authenticated subscriber creates unnecessary overhead when retrieving certain web resources.

In NWSP, the relevant code from the sample `testShortcut.jspi` is as follows:

```
<%
String uriMime = application.getMimeType(uri);
boolean shortcut = false;
shortcut = shortcut || uriMime != null && uriMime.startsWith("image/");
shortcut = shortcut || uriMime != null && uriMime.equals("text/css");
shortcut = shortcut || uriMime != null && uriMime.equals("application/x-javascript");

...

if (shortcut)
{
     //Go straight to dispatcher.
     %>
     <jsp:forward page="<%= dispatcher %>" />
     <%
}
%>
```

The `testShortcut.jspi` file uses MIME types to determine when it is appropriate to bypass authentication testing. The developer can modify the code and add to the list of MIME types that trigger the bypass mechanism. When a request for a web resource is received, the `Decorator` servlet stores the name of the requested web resource in the `uri` scripting variable.

- If a requested web resource (`uri`) is one of the specified MIME types, the requested URI is forwarded directly to `dispatcher.jspi`.
- If a requested web resource is not one of the specified MIME types, response decoration and authentication testing are performed with `decorateResponse.jspi` and `testAuthentication.jspi`.

decorateResponse.jspi

The `decorateResponse.jspi` file sets the locale and encoding of the response. This file obtains the current preferred locale and sets the locale of the appropriate response headers to the value of the preferred locale. Though `decorateResponse.jspi` is a deployer-customizable JSP page, it usually requires no modification unless the subscriber's browser does not support cookies.

Rewriting URLs When Cookies Are Not Supported

If the subscriber's browser supports cookies, the NWSP web application has the necessary code for using cookies for the subscriber's session ID. The service-provider developer does not need to perform further action.

If the browser does not accept cookies (for example, because the subscriber has disabled cookies), the SESM web application is responsible for encoding the session ID into all URLs that are in the content of this response. There are three exception cases in which the SESM web application does *not* need to encode the session ID into the URL:

- If the URL is to a static resource (for example, an image)
- If the tag `<jsp:forward>` forwards the request
- If the tag `<jsp:include>` is used

In most cases, the web application can embed the session ID in the URL by calling the `response.sendRedirect` method as follows:

```
response.sendRedirect(response.encodeRedirectURL(redirectURL));
```

In the preceding example, the code adds the cookie containing the session ID to the URL and redirects the current request to the new URL.

Note The preceding method for embedding the session ID works *except* when the web application is redirecting to the same URL. Redirecting to the same URL is common when redirecting from a GET or POST.

Redirecting to the Same URL

When the web application is redirecting to the same URL, the SESM software provides a cookie request attribute so that the application can easily access the session ID when rewriting the URL.

- If the browser does not support cookies, the SESM web application does need to encode the session ID in the URL. The SESM software sets the request attribute `"cookie"` to a string containing the session ID.
- If the browser supports cookies, the SESM web application does not need to encode the session ID in the URL. The SESM software sets the request attribute `"cookie"` to the empty string.

Two examples of typical values for the cookie request attribute are `";JSESSIONID=5f41pnewmr"` or `""` (an empty string in the case where the browser supports cookies).

If the request is for an HTML form where the `<FORM>` tag does not specify the `ACTION` attribute explicitly, the form is submitted to the same URL that produced the form. If this URL has the session ID embedded, the HTTP session is maintained. If this URL does not have the session ID embedded (and the browser does not support cookies), the form is processed inside a different HTTP session.

An SESM web application can call `Decorator.getResourceURL(pageContext, resource)` to obtain a value that is suitable for the `ACTION` attribute of an HTML form. To submit the form to the same JSP that generated it, an SESM web application can use the value of the `uri` variable as the value for `resource` parameter in the `getResourceURL` call.

The following examples illustrate different situations in which an SESM web application can use the `"cookie"` request attribute to embed the session ID in a URL. In two examples, no session ID embedding is required.

```
<jsp:useBean id="cookie" class="java.lang.String" scope="request" />

<A HREF="http://server/some.jsp<%= cookie %>>link</A>

<A HREF="http://server/some.jsp<%= cookie %>?param=x">link with ?</A>

<IMG SRC="http://server/logo.gif> <!-- static image so no session ID is needed -->

<IMG SRC="http://server/createChart.jsp<%= cookie %>> <!-- dynamic image           -->
                                                      <!-- so session ID is needed -->

<jsp:forward page="next.jsp" /> <!-- next.jsp uses the same HTTP session -->
                                <!-- so no session ID is needed          -->
```

dispatcher.jsp

The `dispatcher.jsp` page uses the extra path information that the `Decorator` servlet has stored in the `uri` scripting variable and directory paths from the subscriber's `Shape` object to construct a complete URI for the requested resource. The `dispatcher.jsp` page also finds the resource located by the complete URI and forwards the resource. If the resource is not specified or is not found, it is the responsibility of `dispatcher.jsp` to handle the error. For information on the extra path information that the `Decorator` servlet stores in `uri`, see the "Decorator Servlet" section on page 2-14.

In the NWSP web application, the relevant code from `dispatcher.jsp` is as follows:

```
<jsp:useBean id="uri" class="java.lang.String" scope="request" />
<jsp:useBean id="shape" class="com.cisco.aggbu.shape.Shape" scope="session" />
...
<%-- body --%>
<%
if (uri.length() == 0)
{
        response.sendError(HttpServletResponse.SC_NOT_FOUND,
                           "dispatcher.jsp, no URI specified");
}
else
{
        Log.debug("dispatcher.jsp, URI requested=", uri);

        Log.debug("dispatcher.jsp, pre find");
        String path = shape.getPath(uri, application);
        Log.debug("dispatcher.jsp, post find, found ", path);
        if (path != null)
        {
                %>
                <jsp:forward page="<%= path %>" />
                <%
        }
        else
        {
                response.sendError(HttpServletResponse.SC_NOT_FOUND,
                                   "dispatcher.jsp, uri=" + uri);
        }
}
%>
```

As shown in the preceding example, `dispatcher.jsp` is responsible for performing the following tasks in the order shown:

1. The `dispatcher.jsp` page tests the length of the `uri` variable:
   - If the length of `uri` equals 0, the `Decorator` servlet has not found extra path information in the HTTP request. Therefore, `dispatcher.jsp` sends `SC_NOT_FOUND` error in the response.
   - If the length of `uri` does not equal 0, the `Decorator` servlet has set the value of the `uri` variable to the extra path information that it found in the HTTP request.
2. Declares a variable `path` and sets its value to the URI returned by the `shape.getPath` method. The `shape.getPath` method takes two arguments:
   - `uri`—The scripting variable whose value was set in `Decorator` servlet
   - `application`—A predefined JSP variable of the type `ServletContext`

   The `getPath` method constructs a URI that the SESM web application uses to locate the subscriber-specific resource. To construct the URI, `getPath` combines the extra path information in the `uri` variable and the user shape in the `Shape` object. For example, given the extra path information `/pages/serviceLogon.jsp` and the user shape `/desktop/personal/en`, the URI is:

   ```
   /desktop/personal/en/pages/serviceLogon.jsp
   ```

   The `getPath` method conducts a search for the resource starting at the URI. For information on how the Cisco SESM software searches for a resource, see the "Searches for a Web Resource" section on page 2-9.
3. Tests the value of the path variable:
   - If `path` does not equal `null`, `getPath` found the resource. The `dispatcher.jsp` page forwards the HTTP request to the page specified by `path`, whose value is the URI that was just constructed.
   - If `path` equals `null`, `getpath` could not find the resource. The `dispatcher.jsp` page sends `SC_NOT_FOUND` error in the response to the subscriber.

Developing an SESM Web Application

Depending on the service provider's business requirements for an SESM web application, the steps required to develop the application may vary. This section provides some guidance on the steps involved with developing an SESM web application:

Defining the Business Requirements

The process of defining the business requirements for a specific service provider's SESM web application can be organized around the following questions:

- What look-and-feel elements (icons, images, background colors, and style sheets) must be customized?
- Will SESM web pages be rendered to match a one or more subscriber characteristics, such as device or brand?

- What types of localization (if any) are required?
  - If only English language subscribers will use an SESM web application, the base set of components in a sample SESM web application may not require localization.
  - If subscribers use a single language other than English, a single web site localized for this language may be sufficient.
  - If subscribers use many languages, rendering SESM web pages localized for each subscriber's language and character set may be required.

Designing and Implementing an SESM Web Application

The design and implementation phases may involve one or more of the following tasks:

- Customizing the look and feel of web elements such as icons, images, background colors, and style sheets)
- Localizing web elements
- Creating additional locale-specific properties files
- Implementing a sparse tree directory
- Coding the JSPs that implement decoration based on the user shape

SESM Class Libraries

The Cisco SESM software provides several specialized Java class libraries that encapsulate the functionality required in an SESM web application. The SESM class libraries are distributed in the JAR (Java archive) files listed in Table 2-4.

Table 2-4 JAR Files for an SESM Web Application

JAR File	Description
install_dir/lib/lib/ com.cisco.aggbu.lib.jar	Classes for Java Management Extensions (JMX) used with an SESM web application
install_dir/nwsp/docroot/Web-inf/lib/ com.cisco.aggbu.contextlib.jar	Classes for SESM internationalization and localization
install_dir/nwsp/docroot/Web-inf/lib/ jsp.jar	Classes for SESM localization and other JSP-related functionality
install_dir/nwsp/docroot/Web-inf/lib/ ssd.jar	Classes for SESM service selection and subscription

The CLASSPATH environment variable must be set to tell the Java compiler the location of the com.cisco.aggbu.lib.jar file. In NWSP, the environment variable is set through the start script (start.sh on UNIX and start.cmd on Windows), which is executed when the web application is started. The com.cisco.aggbu.contextlib.jar, jsp.jar, and ssd.jar files can be found by the Java compiler if they reside in the web application's /Web-inf/lib directory.

JSP Compilation

The JSP pages in the NWSP sample web application are precompiled and the resulting servlet classes are installed in the *install_dir*/nwsp/docroot/Web-inf/lib/jsp.jar file. The use of precompiled JSP pages allows an SESM server that has the required Java Runtime Environment (JRE) to run the NWSP web application. A Java 2 SDK is not required to run the web application. However, to perform development on the JSP pages, you must install the Java 2 SDK on the development machine. For information on the required Java 2 SDK, see the "Hardware and Software Requirements for Development" section on page 1-5.

The web.xml file in *install_dir*/nwsp/docroot/Web-inf is the NWSP deployment descriptor file. This file defines how the Jetty web server finds the JSP pages. The default web.xml file for NWSP specifies that the web server uses the precompiled JSP pages. To compile modified JSP pages, use the web.recompile.xml file in place of the default web.xml file.

Note Until you perform the following steps, changing the JSP pages in *install_dir*/nwsp/docroot has no effect on the HTML pages that the NWSP web application displays.

To recompile modified JSP pages in the NWSP web application, you must do the following:

Step 1 Stop the web server.

Step 2 In the *install_dir/*nwsp/docroot/Web-inf directory, rename the web.xml to web.xml.bak.

Step 3 In the *install_dir/*nwsp/docroot/Web-inf directory, rename web.recompile.xml to web.xml.

Step 4 Restart the web server.

Demo Mode

You can install or configure the SESM software for demonstration mode (Demo mode) to observe how the NWSP web application works. For information on installing or configuring for Demo mode, see the *Cisco Subscriber Edge Services Manager and Subscriber Policy Engine Installation and Configuration Guide*.

Demo mode is also useful during some phases of web-application development because this mode does not require other system components, such as a configured Cisco 6400 UAC and SSG. When the web designer modifies the look-and-feel of the NWSP web application, the changes can be viewed in Demo mode to observe the results. If the web developer makes changes to the logic of the decorator JSP pages, several changes in web-application behavior can be initially tested in Demo mode to verify the results.

For the NWSP sample application, Demo mode uses a Merit RADIUS file for subscriber, service, and service group information. By default, the Merit RADIUS file is named demo.txt and resides in the *install_dir*/nwsp/config directory. A Merit RADIUS file is an ACSII file that you can modify using a text editor. In this way, you can observe how changes to a subscriber, service, or service group profiles affect the NWSP web application. Changes to the demo.txt file do not take effect until the web server is stopped and restarted.

RADIUS Mode Functionality

To simulate RADIUS mode functionality (such as service selection), the subscriber, service, and service-group profiles in demo.txt use the standard RADIUS attributes and vendor-specific RADIUS attributes (VSAs). The attributes are described in the *Cisco Subscriber Edge Services Manager and Subscriber Policy Engine Installation and Configuration Guide*. For example, the following profile from demo.txt is for the subscriber radiususer:

```
radiususer Password = "cisco"
        Service-Type = Framed-User,
        Account-Info = "Ainternet-blue",
        Account-Info = "Ninternet-blue",
        Account-Info = "Niptv",
        Account-Info = "Ngames",
        Account-Info = "Ndistlearn",
        Account-Info = "Ncorporate",
        Account-Info = "Nshop",
        Account-Info = "Nbanking",
        Account-Info = "Nvidconf"
        Account-Info = "Hhttp://www.spiderbait.com"
```

DESS Mode Functionality

To simulate DESS mode functionality (such as subscriber self-subscription, account management, and subaccount creation), the allowed subscriber profile attributes have been extended so that the NWSP web application can demonstrate DESS mode features:

- Different types of subscriber privileges (for service selection, service subscription, account management, and subaccount creation)
- Account attributes with X.500 information (title, given name, surname, and so on)
- Services and service groups that are available for subscription but not yet subscribed
- Parent account names for subaccounts

The extended set of attributes like other RADIUS attributes used by a SESM web application are read-only. However, NWSP in Demo mode does correctly simulate DESS mode functionality. For example, in Demo mode, if a subscriber adds or deletes a subscription, the list of subscribed services on the My Services page and in the service list are dynamically updated. The changes persist until the web server is stopped. The NWSP web application does not modify attributes in the Merit RADIUS file.

Note The extended set of subscriber profile attributes are allowed for Demo mode only. The extended set of attributes are not valid VSAs. They are not valid in RADIUS mode or DESS mode and are not recognized by the SSG. They should not be added to the RADIUS dictionary.

In NWSP, the demo.txt file contains two subscriber profiles for simulating DESS mode: dessuser1 and dessuser2. These subscriber profiles provide examples of the attribute extensions. For detailed information on the extended set of attributes for demonstrating DESS features, see the *Cisco Subscriber Edge Services Manager and Subscriber Policy Engine Installation and Configuration Guide*.

General Web Development Considerations

Some general web development considerations for a Cisco SESM web application include:

- We recommend the technique in which the web application redirects a `POST` request to a `GET` request. The Cisco SESM software has no mechanisms that support or prevent this technique.
- The decoration JSP pages and include files must not write and flush any characters to the `ServletOutputStream`. If either does, the forwarding HTTP request throws an `IllegalStateException`. This restriction is imposed by the servlet container.
- As is usual with any web application, all references to a web resource from a web page must be relative.

A web application can be deployed with a prefix, which is specified in `web.xml`. An absolute reference to another web resource within the same web application must include the web application prefix. However, the web application has no knowledge of the web application prefix. Therefore, references to web resources must be relative.

Debugging an SESM Web Application

If an SESM web application is not operating as expected, you can use the SESM logging utility to change the details of the information reported in the log files to diagnose a problem. For detailed information on logging and debugging mechanisms available for an SESM web application, see the *Cisco Subscriber Edge Services Manager and Subscriber Policy Engine Installation and Configuration Guide*.

The following hints may help with debugging an SESM web application.

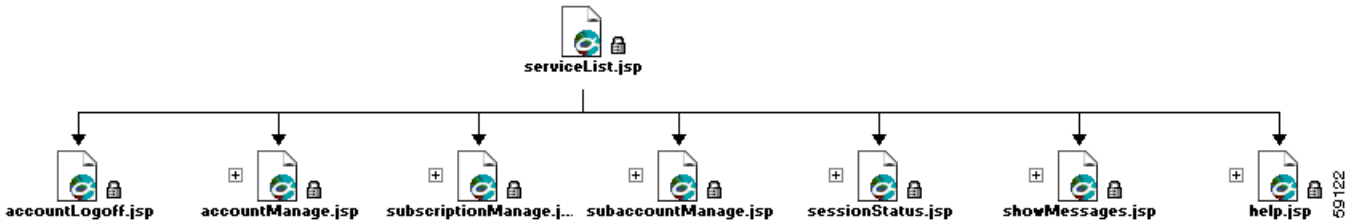
- Most web servers can be set up so that the servlet code generated from JSP pages is saved and available for examination. With the Jetty web server, when the JSP pages are compiled, the servlet code is put in a temporary directory that is determined by the machine's JVM. Normally, the temporary directory is /tmp or /usr/tmp on UNIX, and C:\TEMP on Windows NT or Windows 2000. The servlet code can be examined in the temporary directory until it is deleted by the operating system's normal cleanup mechanism. For information on where servlet code is located with other web servers, refer to the documentation from your web server vendor.
- With the Internet Explorer browser, when an error occurs in the dynamic portion of a JSP page, IE displays an alternative "friendly" (and less helpful) HTTP error message if the Show friendly HTTP error messages box is checked. To view server-generated error messages, make sure this box is not checked in the Advanced Tab for Internet Options, which is accessed through the Tools menu.
- Some SESM web application files such as web.xml and the properties files for resource bundles are read only when the web application is started. Changes to these files have no effect until the web application is stopped and restarted.

Managing an SESM Web Site

Dreamweaver has a number of features that may be useful for the developer of an SESM web application. For example, if a Dreamweaver template or library item is modified, the developer can automatically update all files in a web site that use the template. To use some Dreamweaver features, a web site must be defined in Dreamweaver.

To define an SESM web application as a site, in the Dreamweaver Site menu, you choose New Site and specify the required information. In the Local Info dialog box, you specify the */webapp/docroot* directory of the SESM web application as the Local Root Folder. In the Site Map Layout dialog box, you specify `serviceList.jsp` as the Home Page. After the site is defined in this manner, Figure 2-7 shows the resulting (partially expanded) site map for the NWSP web application.

Figure 2-7 NWSP Site Map



The Dreamweaver Preview in Browser feature is of limited use with an SESM web application. For example, in the NWSP web application, most JSP pages require some form of input. Without the input, the web browser displays an error page.

For detailed information on setting up and managing a site, refer to the Dreamweaver documentation.

CHAPTER 3

New World Service Provider Web Application

The sample New World Service Provider (NWSP) web application contains all of the components required for a fully functional web-based user interface. The developer can use the NWSP web components as a starting point for designing and creating an SESM web application. This chapter provides information on how a developer can use and modify the NWSP web components.

The following general considerations apply to this chapter's descriptions of the NWSP web components.

Before you read this chapter, read Chapter 2, "SESM Components and Techniques," for an explanation of SESM components and techniques that are, in general, used in all SESM sample web applications.

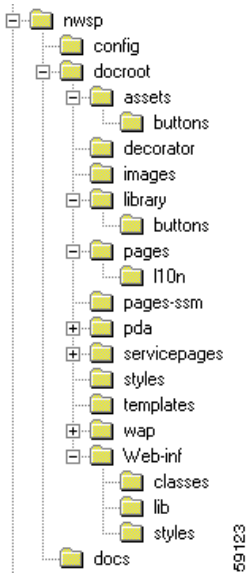
- Depending on the SESM software that is used, a deployed SESM web application can be configured for one of two modes:
  - *RADIUS mode*—Service and subscriber information is stored in a RADIUS server.
  - *DESS mode*—Service, subscriber, and policy information is stored in an LDAP-compliant directory, which is accessed with the DESS application programming interfaces.
- The set of web components used for a NWSP web application varies depending on the configuration (RADIUS mode or DESS mode). Where it is required, the descriptions of components in this chapter indicate the mode in which each component is used.
- For simplicity, the NWSP web components and the explanations in this chapter assume that English is the only language used by the subscriber. If an SESM web application must support multiple languages, the images, icons, and message text in the NWSP web components would have to be modified for the other languages.

NWSP Directory Hierarchy

After the Cisco SESM software is installed, the NWSP web application is located in a structured hierarchy of directories. As with any web application, the root of this hierarchy is the document root for serving the NWSP web pages to the subscriber. In this directory hierarchy, the \Web-inf directory contains items related to the web application that are not in the document root. That is, the files and directories in \Web-inf are not part of the public document tree from which files can be directly served to the client.

When Cisco SESM software is installed, the NWSP web application's hierarchy of directories is located below the *install_dir*\nwsp directory (see Figure 3-1). The \pda and \wap directories are currently not used. The following directories and their files are not required in a deployed SESM web application: \assets, \library, \templates, and \docs.

Figure 3-1 NWSP Directories



The NWSP directories contain the complete set of files required for the web application and include the PNG source files required to customize the Fireworks images and buttons. The NWSP directories and files are:

config

Contains the web application configuration file `nwsp.xml`. For information on the configuration files, see the *Cisco Subscriber Edge Services Manager and Subscriber Policy Engine Installation and Configuration Guide*.

docroot

Is the web application's document base.

docroot\assets

Contains the PNG source files from which the GIF images and Fireworks buttons were made. You can customize the images in the PNG files using Fireworks or another graphics tool.

docroot\decorator

Contains the JSP pages used for rendering the content according to the user's shape.

docroot\images

Contains the GIF images for the NWSP user interface.

docroot\library

Contains Dreamweaver library items. Library items contain images, text, and other objects that are used frequently throughout NWSP. For information on the NWSP library items, see the "NWSP Library Items" section on page 3-13.

docroot\pages

Contains JSP pages used for both DESS mode and RADIUS mode. The \l10n directory contains JSP pages for localization; they are currently not used. For information on the JSP pages used in NWSP, see the "NWSP JavaServer Pages and Servlets" section on page 3-5.

docroot\pages-ssm

Contains additional JSP pages used for DESS mode. For information on the JSP pages used in NWSP, see the "NWSP JavaServer Pages and Servlets" section on page 3-5.

docroot\servicepages

Contains images for services that the NWSP web application uses in Demo mode.

docroot\styles

Contains the Cascading Style Sheet `nwsp_styles.css`.

docroot\templates

Contains the Dreamweaver templates. For information on the NWSP templates, see the "NWSP Templates" section on page 3-8.

docroot\Web-inf

Contains various Java-related NWSP components:

- Tag library descriptor (`.tld`) files for the NWSP tag libraries are in \Web-inf. For information on the .tld files and using a tag library, see the "Configuring a Tag Library" section on page 4-4.
- The web application's deployment descriptor file, `web.xml`, is in \Web-inf. For information on this file, see the *Cisco Subscriber Edge Services Manager and Subscriber Policy Engine Installation and Configuration Guide*.
- The \lib directory contains the Java archive (JAR) files for some SESM classes. For information on the JAR files required for an SESM web application, see the "SESM Class Libraries" section on page 2-23.
- The \classes directory contains the NWSP properties files. For information on localization and properties files, see Chapter 4, "SESM Internationalization and Localization."

docs

The Javadoc files for the NWSP classes.

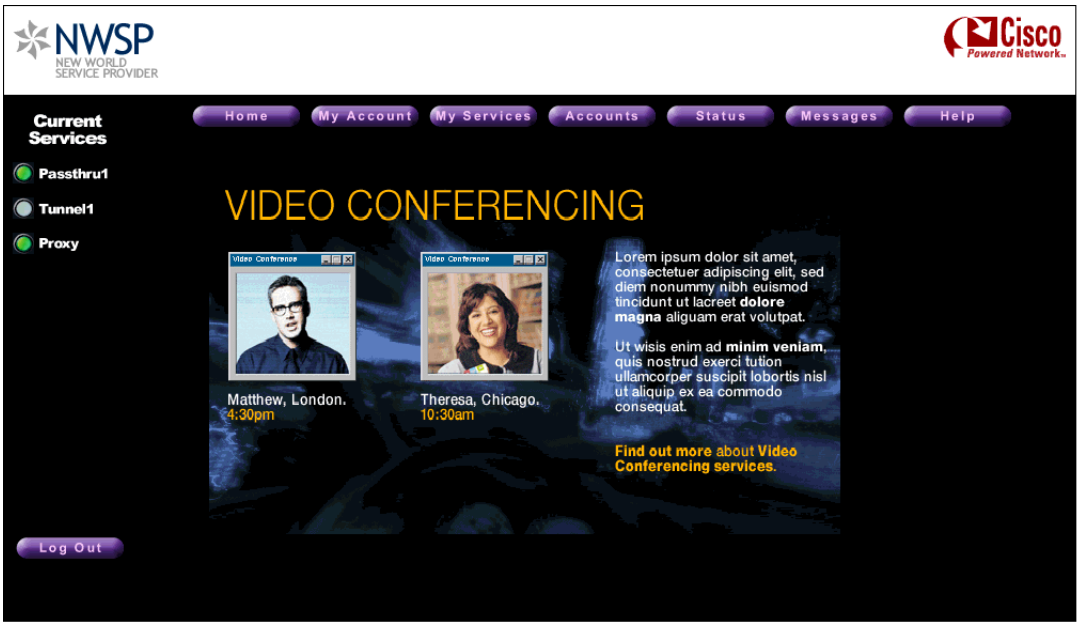
NWSP User Interfaces

From the perspective of the subscriber, the Cisco SESM is a web-based user interface that the subscriber uses for subscribing to and selecting services, changing account details, creating subaccounts, and viewing session status and messages. The look-and-feel of the user interface is determined by the service provider. The user interface can be customized for a particular subscriber based on a number of factors, such as:

- Devices and browsers used to connect to an SESM web site
- Branding for the SESM web site so the look-and-feel can be associated with, for example, a retail ISP or a corporation
- Language and country of the subscriber
- Personalization of the SESM web site to include a subscriber-specific set of services

Figure 3-2 shows the service list page from the NWSP web application's user interface.

Figure 3-2 NWSP Service List Page



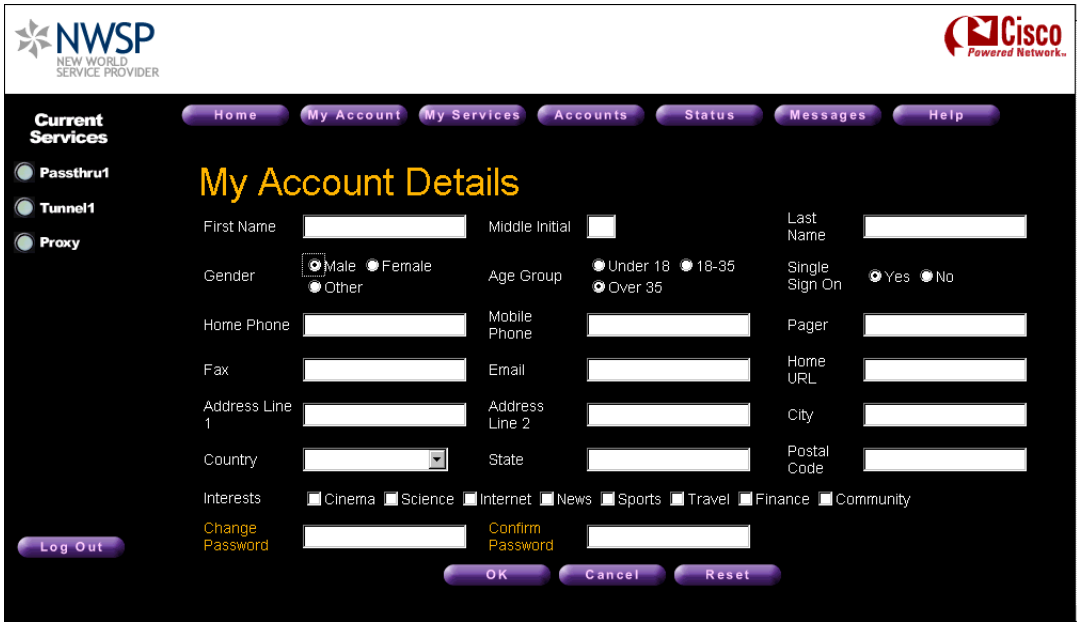
The sample NWSP web application provides a set of functionality that is typical of many directory-enabled SESM applications. The subscriber is able to logon to the user interface with a user name and password. The subscriber can then do the following:

- Subscribe to or unsubscribe from network services that are authorized
- Connect to or disconnect from services that are subscribed
- Change account details, such as address information and passwords
- Create subaccounts for other family members
- View the status of service connections
- View system messages

The features related to account management are possible only with an SESM web application that is operating in DESS mode. The NWSP web application includes the required logic to determine the privileges that were granted to a subscriber and to generate the appropriate content. For example, if a subscriber includes the required privileges to create subaccounts, the NWSP web application displays the Accounts button in the navigation bar, and the subscriber can create subaccounts.

Each button at the top of the user interface is linked to a JSP page that implements the functionality for the specific task or set of tasks. As an example, the My Account button is linked to `accountManage.jsp`, which generates the content for the account management page (Figure 3-3).

Figure 3-3 Account Management Page



The look-and-feel of an SESM user interface can be customized for each subscriber. When the HTTP request is received, the SESM web application has an organization and an infrastructure that allows the page returned in the response to be tailored for the individual subscriber. For example, if the subscriber's browser language is French and receiving device is a desktop PC, the response can be rendered in French using HTML. If another subscriber's browser language is Spanish and the receiving device is a wireless handheld computing device, the response can be rendered in Spanish using Wireless Markup Language (WML).

For information on customization and localization, see Chapter 2, "SESM Components and Techniques," and Chapter 4, "SESM Internationalization and Localization."

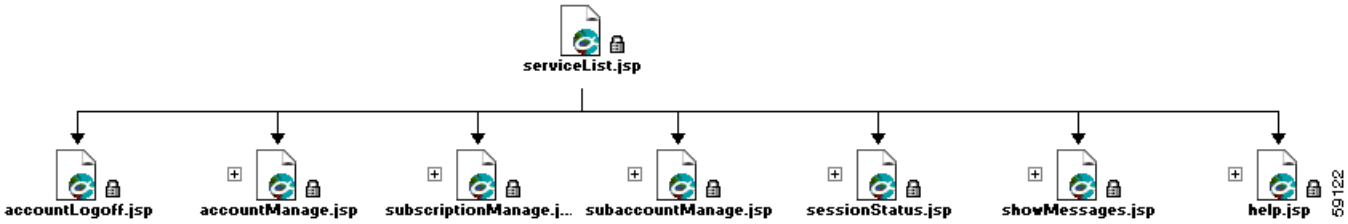
NWSP JavaServer Pages and Servlets

The NWSP web application includes a set of JSP pages and servlets that generate content for the web pages and perform other tasks, such as authentication, SESM session handling, and service selection and subscription. The JSP pages contain the elements that the developer modifies for the specific requirements of the service-provider. No servlet programming is required.

After the subscriber logs on to the NWSP user interface, `serviceList.jsp` is the home page for the web application.

Figure 3-4 shows the top levels of the NWSP site map with `serviceList.jsp` as the home page.

Figure 3-4 NWSP Site Map



From the `serviceList.jsp` home page, the subscriber can control service subscription and selection and, in DESS mode, can perform account-management, self-subscription, and subaccount functions. The navigation bar at the top of the `serviceList.jsp` links the subscriber to these capabilities.

JSP Pages for Service Selection

Table 3-1 lists JSP pages for service selection. In NWSP, the JSP pages for service selection are located in the \nwsp\pages directory. The service selection pages are used for both RADIUS mode and DESS mode.

Table 3-1 NWSP JSP Pages for Service Selection

Component	Description
`accountLogon.jsp`	Allows a subscriber to log on to the NWSP web application. The subscriber's user name and password are authenticated.
`accountLogoff.jsp`	Asks a subscriber to confirm that the Log Out button has been clicked and that the session is to be terminated.
`confirm.jsp`	Asks a subscriber to confirm a specific action by clicking the OK or Cancel button.
`error.jsp`	Displays an error message and asks the subscriber to click OK.
`groupAccess.jsp`	After the subscriber clicks a service group icon in the service list, this page processes the request for the group and redirects the user so that the service list displays the contents of the requested service group.
`help.jsp`	Allows a subscriber to view deployer-created help information.
`home.jsp`	Decides the SESM home page name. Other pages can forward to this page without needing to know the home page name.
`message.jsp`	Displays a message and asks the subscriber to click OK.
`redirecting.jsp`	Displays a message that the HTTP client was redirected to a different URL.
`serviceList.jsp`	After a subscriber is logged on to the SESM web interface, this is the SESM home page from which the subscriber can access other functions, such as service subscription and account management. These functions vary depending on whether the web application uses DESS mode or RADIUS mode.

Table 3-1 NWSP JSP Pages for Service Selection (continued)

Component	Description
serviceListDisplay.jspi	Displays the service list, consisting of services and service groups. This page is included in any page using the serviceandsettingsTemplate.dwt template.
serviceLogoff.jsp	After a subscriber clicks a connected service in serviceList.jsp, this page allows the subscriber to disconnect from the service.
serviceLogon.jsp	After a subscriber clicks a disconnected service in serviceList.jsp, this page allows the subscriber to connect to the service.
serviceWindow.jsp	Provides the pop-up window for a service after the user logs on to the service.
sessionStatus.jsp	Displays the status of each connected session: IP address and connect time for each service that the subscriber is or was connected to.
sessionStatus.jsp	Displays information about the current session: user name, IP address, services connected, and elapsed connect times.
showMessages.jsp	Displays messages associated with the current session.

The accountLogon.jsp page includes Standard | Secure hyperlinks below the Log In button. These links let the user choose either standard mode or secure mode. On the page on which accountLogon.jsp appears, the link for the mode that the subscriber is currently not using is available, but the link for the mode that the subscriber is currently using is not available.

When the subscriber logs in using secure mode, the SESM web application uses Secure Sockets Layer (SSL) encryption. The password that the subscriber enters is encrypted by the HTTP client before it is sent to the HTTP server where the SESM web application resides. The HTTP server decrypts the password. The encryption and decryption occurs for all content that passes between the client and the server, not just to the password. With secure mode, pages take longer to download.

If the service provider does not require SSL or does not have a certificate, the developer removes the Standard | Secure elements in the accountLogon.jsp page. In addition, the deployer removes the Jetty web server's SSL listener, which is configured in the \nwsp\config\nwsp.xml file. For more information on SSL and security, see the *Cisco Subscriber Edge Services Manager and Subscriber Policy Engine Installation and Configuration Guide*.

JSP Pages for Service Subscription and Account Management

Table 3-2 lists the JSP pages for service subscription and account management (subscriber self-care and subaccount creation). The service subscription and account management pages are used only with DESS mode. The JSP pages for service subscription and account management are located in the \nwsp\pages-smm directory.

Table 3-2 NWSP JSP Pages for Service Subscription and Account Management

Component	Description
accountManage.jsp	Allows a subscriber to change account information such as an address, telephone number, e-mail address, and so on.
subaccountManage.jsp	Displays and processes the form that allows a subscriber to create and delete subaccounts and modify subaccount settings: subaccount passwords, enabling single sign-on, and permissions (for example, service subscription).
subaccountProcess.jsp	Displays and asks the subscriber to confirm changes to subaccounts specified in subaccountSubscriptions.jsp.
subaccountSubscriptions.jsp	Displays the form from which a subscriber can modify subaccount subscription attributes.
subscriptionManage.jsp	Allows a subscriber to modify attributes associated with subscriptions. For example, a subscriber can specify whether the service is subscribed or is an auto-connect service.
subscriptionProcess.jsp	Displays and asks the subscriber to confirm changes to the subscriptions specified in subscriptionManage.jsp.

In Table 3-2, the page-to-page flow for account subscription management is from subscriptionManage.jsp to subscriptionProcess.jsp. The flow for subaccount subscription management is from subaccountManage.jsp to subaccountSubscriptions.jsp to subaccountProcess.jsp. All other subaccount management is processed by subaccountManage.jsp.

JSP Pages for Rendering Content

The NWSP web application also includes a set of JSP pages and a servlet that performs these tasks:

- SESM session handling
- Subscriber authentication
- Rendering content based on the user's shape

These components are located in the \docroot\decorator directory and are used in both DESS mode and RADIUS mode configurations. For information on rendering content based on the user's shape, see the "Using the Decorator Components" section on page 2-12.

NWSP Templates

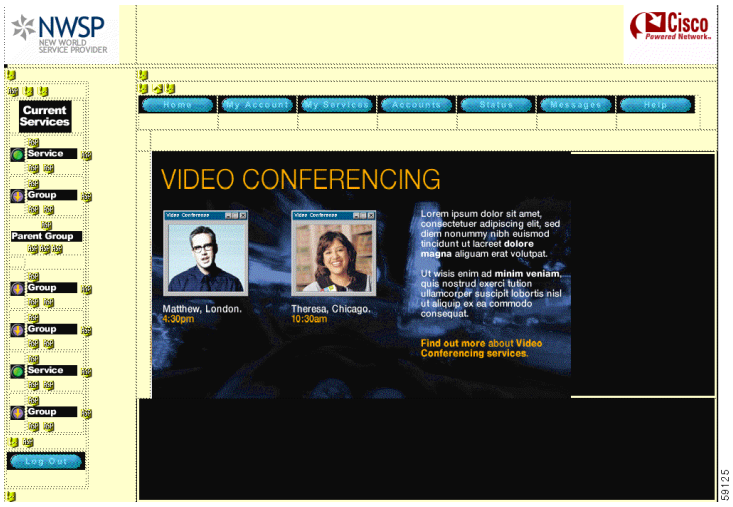
The NWSP components include two Dreamweaver templates for customizing and maintaining the JSP pages:

- serviceandsettingsTemplate.dwt
- headerOnlyTemplate.dwt

Service and Settings Template

The template `serviceandsettingsTemplate.dwt` is used for NWSP pages that require a service list and navigation buttons as well as a banner with brand icons. Most NWSP JSP pages, including `serviceList.jsp`, use this template. Figure 3-5 shows the `serviceList.jsp` page as its appears in Dreamweaver with its table borders visible.

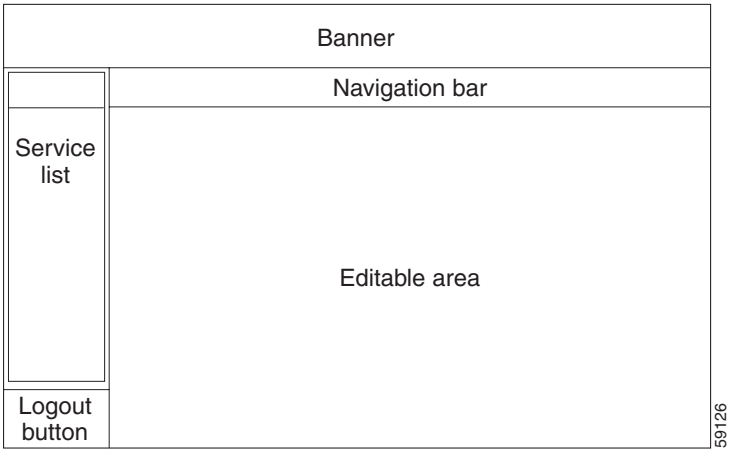
Figure 3-5 Page That Uses the Template serviceandsettingsTemplate.dwt



The template `serviceandsettingsTemplate.dwt` (Figure 3-6) is a set of tables and rows that structure the NWSP page content into the following pieces:

- banner
- navigation bar
- editable area
- service list
- logout button

Figure 3-6 Structure of the Template serviceandsettingsTemplate.dwt



The banner contains two images for branding.

The content in the editable area varies depending on the purpose of the page. For example, if the JSP page allows the subscriber to change account information, the editable area contains the appropriate form.

The logout button is similar to the buttons that are described in the "Navigation Bar" section on page 3-12.

Service List

The service list (Figure 3-7) that appears on the left side of the window is dynamically created by the JSP pages based on the service and subscriber information stored in the data repository. In DESS mode, the subscriber can use an SESM web application to add or remove services from the set of subscribed services that are displayed in the service list. This feature is called *self-subscription*.

Figure 3-7 Service List with Buttons and Text

For each service in the service list, a traffic-light icon indicates the state of the service. A gray button indicates that the subscriber is not currently connected to the service, a green button indicates the subscriber is connected, and a red button indicates the session was lost.

Within the service list, the `serviceListDisplay.lbi` library item determines which of the following buttons to display based on the current state of the service:

- `service_loggedoff.gif`
- `service_loggedon.gif`
- `service_sessionlost.gif`

Each button has a corresponding GIF image for the mouseOver state (for example, `service_loggedoff_over.gif`).

Each button is linked to the appropriate "service page." In `serviceList.jsp`, the SESM software uses the `servicePage` variable to store the JSP page to which each service button is linked as determined by the current state of the service. If the current state of the service is logged on, `servicePage` equals `serviceLogoff.jsp`. If the current state of the service is logged off or the session is lost, `servicePage` equals `serviceLogon.jsp`.

When a service group appears in the NWSP service list, the icons used for the service group are as follows:

- `group_access.gif` (a down arrow) indicates access to an unexpanded group.
- `group_up.gif` (an up arrow) indicates movement up one level in the group hierarchy.
- `group_top.gif` (an up arrow) indicates the top level in the group hierarchy.

Similar to the service buttons, each group button has a corresponding GIF image for the mouseOver state (for example, `group_access_over.gif`).

To the right of each traffic-light icon or group icon, the service or group is designated by either text or an icon. The `serviceListDisplay.lbi` library item determines whether text or an icon is used to indicate each service or group by examining the `useIcons` variable.

- If `useIcons` is set to true, each service or group is shown with an icon, whose file name is *`service`*`.gif` or `group.gif`, where *`service`* and *`group`* are, respectively the service name or group name in the service or service group profile. If the icon is not found, the service list uses the service or group description from the profile. If there is no description, the service page uses the service or group name from the profile.
- If `useIcons` is not set to true, each service is shown with a text name. The text name is the service or group description from the profile. If there is no description, the service page uses the service or group name from the profile.

For information on configuring the `useIcons` parameter, see the *Cisco Subscriber Edge Services Manager and Subscriber Policy Engine Installation and Configuration Guide*.

In the NWSP sample application, the PNG files for the various service and group icons are in the \nwsp\docroot\assets\buttons directory. In a production SESM web application that uses icons (as opposed to text) to designate a service or group, the developer provides an icon for each service and group. In NWSP, the GIF files for the icons are located in the \nwsp\docroot\images directory. The name of the icon file must be *`service`*`.gif` or *`group`*`.gif`, as explained in the preceding discussion.

The easiest way for the developer to create the service and group icons is to use Fireworks to modify an existing `.png` image, save the modified `.png` file with the appropriate name, and export the corresponding GIF image to the *docroot*\images directory. The developer is not restricted to using Fireworks. The images could be modified with any graphics tool.

Navigation Bar

The Dreamweaver-created navigation bar (Figure 3-8) that appears below the banner consists of a set of buttons whose display changes based on the actions of the user. For example, one image for a button in a navigation bar is used when the pointer is rolled over the button, and another image for the button is used when the button is clicked.

Figure 3-8 Navigation Bar with Buttons for DESS Mode

Figure 3-8 shows the NWSP navigation bar with the buttons for the dynamic update features: My Account, My Services, and Accounts. These buttons are displayed only when the dynamic update functionality associated with DESS mode is available. The Home, Status, Messages, and Help buttons are displayed in both RADIUS and DESS modes.

In DESS mode, the set of buttons that appear in the NWSP navigation bar varies depending on the user's privileges. Various SESM features require that subscriber have appropriate privileges. For example, the ability to create subaccounts or change account information such as a password require that the subscriber have the required privileges. The NWSP web application has the required logic to determine the privileges that were granted to the subscriber and to display the corresponding set of navigation bar buttons.

The navigation bar buttons are Fireworks 4-state buttons. The NWSP sample application includes the required Fireworks buttons in English. In a production SESM web application that uses the NWSP components, the developer provides any customized or localized buttons for each of these functions in the navigation bar. In NWSP, the GIF files for the navigation-bar buttons are located in the \nwsp\docroot\images directory.

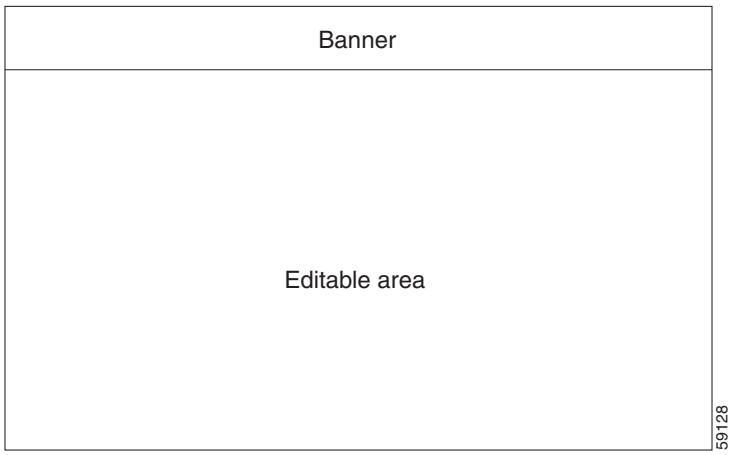
In the NWSP sample application, the PNG files for the buttons in the navigation bar exist in the \nwsp\docroot\assets\buttons directory. The easiest way for the developer to create the customized or localized buttons is to use Fireworks to modify a library element in an existing `.png` file, save the modified `.png` file with the appropriate name, and export the corresponding GIF images to the *docroot*\images directory.

A developer is not restricted to using Fireworks. The button images could be directly modified using any graphics tool. The buttons generated by Fireworks consist of a set of GIF images and are delimited in the JSP pages by `BeginLibraryItem` and `EndLibraryItem` comments.

Header-Only Template

The `headerOnlyTemplate.dwt` file (Figure 3-9) is used for those NWSP pages that do not require a service list or navigation buttons but that do require a banner with brand icons.

Figure 3-9 Structure of the Template headerOnlyTemplate.dwt



Most JSP pages that use the template `headerOnlyTemplate.dwt` contain message or help text: `error.jsp`, `help.jsp`, and `message.jsp`. Other JSP pages that use the template have a simple form or button or both: `accountLogon.jsp` and `confirm.jsp`.

NWSP Library Items

The NWSP library items contain `BODY` elements such as images, text, and other objects that are reused throughout the JSP pages. Library items have the file extension `.lbi` and are located in the \nwsp\docroot\library directory. Table 3-3 shows the library items that are included with the NWSP web application software.

Table 3-3 NWSP Library Items

Library Item	Description
`accountDetailsForm.lbi`	The content of the `accountManage.jsp` page's editable area from which a subscriber can change account information such as an address, telephone number, e-mail address, and so on. This library item is included as an example of how to provide forms as library items.
`confirmationLineDisplay.lbi`	A line of information for changes to services that the user has made, such as changes to subscriptions, the auto-logon option, user name, and password.
`serviceListDisplay.lbi`	A list of services and groups available to the user indicating the status of each service: logged on, logged off, or session lost. The image for each state is linked to the `serviceLogoff.jsp` if status is logged on, or to the `serviceLogon.jsp` if status is logged off or session lost.

For information on `serviceListDisplay.lbi`, see the "Service List" section on page 3-10.

NWSP Images and Buttons

The NWSP web components include a set of customizable images and icons in two formats: PNG and GIF. If you use the NWSP components as the starting point for an SESM web application, the PNG-formatted files in the \docroot\assets directory can be used to modify the images, icons, and buttons.

The NWSP web components also include Fireworks buttons for the navigation bar. The PNG-formatted files for the Fireworks buttons are located in the \docroot\assets\buttons directory.

CHAPTER 4

SESM Internationalization and Localization

The Cisco SESM software includes a set of components that can be used for internationalization and localization:

- Resource bundles
- Localization tag library

This chapter provides some general information on internationalization and localization. It also explains the Cisco SESM components and techniques that help a deployer internationalize and localize an SESM web application. It is beyond the scope of this chapter to provide comprehensive information on the subject of Java internationalization and localization.

Internationalization is the process of designing an application so that it can be adapted for various languages and regions without programming changes. The term *i18n* is sometimes used as an abbreviation for internationalization because that word begins with i, ends with n and contains 18 characters in between. Text in status and error messages that varies depending on the culture needs to be internationalized. Other data that vary by culture include labels on web application buttons and fields, and the formats of dates, times, numbers, and currencies.

Localization is the process of adapting an application for a specific language or region by adding locale-specific components and text. The term *L10n* is sometimes used as an abbreviation for localization because that word begins with L, ends with n, and contains 10 characters in between. Typically, the localization process involves translating text messages to another language and, where required, providing locale-specific images.

The Cisco SESM web components include a set of Java classes and JSP tag libraries that help the deployer to internationalize and localize an SESM web application. These Java classes and techniques extend the classes and techniques that are part of the J2EE classes. The Localization tag library provides methods for setting and changing the localization context associated with the subscriber's HTTP session. For example, the Localization tag library allows a web application to change the locale and time zone of the subscriber's session.

It is not required that a Cisco SESM web application use the extended classes and techniques that are provided with the SESM software. An SESM web application can use conventional Java techniques for internationalization and localization. However, the classes and tags provided with the Cisco SESM software are specifically designed for use with a web application.

Resource Bundles

Resource bundles contain locale-specific data that varies depending on the user's language and region, such as translatable text for status and error messages and for labels on GUI elements. A resource bundle allows a Cisco SESM deployer to separate localizable elements from the rest of the web application.

The localizable elements are stored in a set of properties files, one for each language-region combination. If an SESM web application uses resource bundles, the web application determines the subscriber's locale and then loads the appropriate resource bundle. If the subscriber switches locales, the web application can load a different resource bundle.

Resource bundles allow you to design and write an SESM web application that can be easily localized for the subscriber's language and region. An SESM web application can add additional resource bundles if a new locale is required.

The following sections provide some general information on resource bundles, property files, and their use with a Cisco SESM web application. For detailed information on resources bundles, see the description of these classes at the Java 2 platform area of the java.sun.com web site:

- java.util.ResourceBundle
- java.util.Locale
- java.util.TimeZone
- java.text.MessageFormat

Using Properties Files

A resource bundle can be implemented with a set of one or more properties files. A *properties file* is a plain-text file that contains key-value pairs for each localizable item. For example, the English of a sample properties file that contains message text is:

```
# English version of properties file

AMGreeting = Good Morning
PMGreeting = Good Evening
NotValid = Invalid Value
```

The French version of the properties file is:

```
# French version of properties file

AMGreeting = Bonjour
PMGreeting = Bonsoir
NotValid = Valeur Incorrecte
```

The keys and values in a properties file must be string values. A Cisco SESM web application specifies the key when it retrieves the message text from a resource bundle. The key is case-sensitive. The value associated with the key is the localized text. Properties files are not part of the Java source code. Properties files must be located in a directory specified by the `CLASSPATH` variable or in some location where the Java compiler finds web application classes. In the NWSP sample web application, the properties files (for example, `messages_en.properties`) reside in the \docroot\Web-inf\classes directory.

The value part of the key-value pair can include HTML markup. The value is passed straight through to the HTML page with no changes. Because the `L10nContext` class and the Localization tag library do not process special HTML characters when retrieving a value from a properties file, you can use HTML markup in the value. Special HTML characters that appear in a value and that are not part of HTML markup must use ISO 8859-1 character encodings. For example, if you want the less-than character (<) to appear in a value as something other than HTML markup, it must be coded as &lt.

You can find information on how to write and retrieve the entries in a properties file from these sources:

- For information on the required syntax for the key-value pairs, see the description of the `java.util.Properties.load` method in the Java 2 platform area of the java.sun.com web site.
- For information on using the Localization tag library to retrieve values from a resource bundle, see the "resource Tag" section on page 4-10 and the "template Tag" section on page 4-10.

The filename of each properties file has a base name and an optional locale identifier. The optional locale identifier can include a language name, a country name, and a variant name. Elements are separated from each other by the underscore(_) character. All properties files must have the `.properties` extension. As an example, for the resource bundle `SESMResources`, Table 4-1 shows four possible properties file names.

Table 4-1 Names of Properties Files in a Resource Bundle

Property File Name	Description
`SESMResources.properties`	base name—The file is the default version.
`SESMResources_en.properties`	base name and language name—The file is the English version.
`SESMResources_fr.properties`	base name and language name—The file is the French version.
`SESMResource_fr_CA.properties`	base name and language name and country name—The file is the French version used for Canada.
`SESMResource_fr_CA_WIN.properties`	base name and language name and country name and variant name—The file is the French version used for Canada on the Windows operating system.

Properties files within the same bundle share the same base name and have the same key-value pairs. The `ResourceBundle` class associates a parent with each bundle. For example, `SESMResources_fr` is the parent of `SESMResource_fr_CA`. If the `ResourceBundle` class looks for the file `SESMResources_fr_CA.properties` and cannot find the file, it uses the parent file `SESMResources_fr.properties` or the file having the base name if it cannot find a parent file.

Note If an SESM web application uses the `L10nContext` class or the Localization tag library, the algorithm that determines the default resource bundle calls the `L10nContext.getDefault` method (not the `Locale.getDefault` method) to get the default that is associated with the SESM web application. For information on the benefits of using the `L10nContext` class and the Localization tag library, see the "Localization Tag Library" section on page 4-4.

For detailed information on resource bundle properties files and the search algorithm used by the `ResourceBundle` class, see the class description in the Java 2 platform area of the `java.sun.com` web site.

Localization Tag Library

The Cisco SESM software includes a Localization tag library that helps reduce the complexity of localizing an SESM web application. The Localization tag library uses a special SESM class (`L10nContext`) that improves upon the standard Java locale-related classes for use in a web application. The Localization tag library includes these tags:

- `context`—Sets the characteristics of the current localization context (`L10nContext`).
- `locale`, `timeZone`, and `language`—Get a string describing the specified characteristic of the current localization context.
- `country`—Gets a string for a country specified by the attribute used with the tag.
- `format`—Converts currency, number, date, and time values according the formatting conventions associated with the Java class `java.text.MessageFormat` and the current `L10nContext` localization context.
- `resource`—Obtains a resource from the resource bundle specified for the current localization context and for a specified key.
- `template`—Replaces tokens in a template with parameter values. The `template` tag can be used with the `resource` tag for token replacement in a resource.

The Localization tag library is specifically designed for web application localization. The standard Java `Locale.getDefault` method gets the current value of the default locale for this instance of the Java Virtual Machine (JVM). This default locale is shared across all applications running on that JVM.

In contrast, if the `L10nContext` class and the Localization tag library is used, the class and tag library get the current value of the default locale for the class loader. If each application running on the JVM has its own class loader, then each application has its own localization-context object created using the Cisco SESM `L10nContext` class. Thus, the benefit to using the Localization tag library and the SESM `L10nContext` class is that another application running on the JVM can change the default locale associated with the `Locale` object, but it cannot change a Cisco SESM web application's default `L10nContext` object.

Configuring a Tag Library

If you are using an SESM tag library in a web application, you must perform the following steps to configure the tag library. The NWSP sample web application is preconfigured so that you can use the Localization tag library.

Step 1 Copy the tag library descriptor file (for example, `localization.tld`) from *install_dir*\NWSP\docroot\Web-inf to the \Web-inf directory of your web application.

Step 2 Copy the `com.cisco.aggbu.contextlib.jar` from *install_dir*\NWSP\docroot\Web-inf\lib directory to the \Web-inf\lib directory of your web application.

Step 3 If necessary, add the appropriate `<taglib>` element to your web application deployment descriptor in \Web-inf\web.xml. For example, the `<taglib>` element for the Localization tag library is:

```
<taglib>
    <taglib-uri>http://www.cisco.com/taglibs/localization</taglib-uri>
    <taglib-location>localization.tld</taglib-location>
</taglib>
```

Step 4 To use the tag libraries on a JSP page, add the necessary `taglib` directive at the top of each JSP page. For example:

```
<%@ taglib uri="http://www.cisco.com/taglibs/localization" prefix="l10n" %>
```

context Tag

The `context` tag can be used to create a scripting variable of the type `L10nContext`, specify the localization context explicitly, and set the characteristics of the current localization context (`L10nContext`). A localization context combines a locale, time zone, resource bundle base name, preferred locales, otherwise locale, and scope.

When a web application uses a `context` tag, the tag implicitly declares a localization context. The `context` tag's current localization context inherits values for locale, time zone, resource bundle base name from the first of the following localization contexts that exists:

1. A parent `context` tag
2. If no parent `context` tag is used, the values come from a localization context stored in an `L10nContext` object. The `context` tag software searches for a localization context having (in this order) page, request, session, or application scope. It uses the first context found.
3. If none of the preceding exist, the values come from the default localization context, which it obtains with the `L10nContext.getDefault` method.

A web application can override the current localization context's inherited values with the `context` tag attributes such as `locale`, `preferredLocales`, `timeZone`, and `resourceBundleName`. For example, a web application can set the locale of a user's localization context to Germany for the duration of an HTTP session as follows:

```
<l10n:context locale = "<%= Locale.GERMANY%>" scope = PageContext.SESSION_SCOPE />
```

A localization context object can exist for each of four scopes: page, request, session, and application, As shown in the preceding example, the `scope` attribute defines the scope of a localization context and can be specified with any other `context` tag attribute.

The current localization context, including all its characteristics, can be specified by setting the context using the `context` attribute and an existing `L10nContext` object.For example:

```
<l10n:context context = "<%= someL10nContextObject %>" />
```

If a tag in the Localization tag library is used *inside* the tag body of a `context` tag, the tag's functionality reflects the localization context. In the following example, the currency amount is formatted according to German conventions (`99,99 DM`) because the formatting occurs within the `context` tag body where the localization context's locale is Germany.

```
<% double amount = 99.99; %>

<!-- Set the locale of the current L10nContext localization context -->
<l10n:context locale = "<%= Locale.GERMANY%>" >
Amount formatted as currency:  <l10n:format currency="<%= amount %>" /> <BR>
</l10n:context>
```

If a tag in the Localization tag library is used *outside* the tag body of a `context` tag, the localization context is the same as having a parent context tag with no attributes set.

Table 4-2 lists the attributes of the context tag.

Table 4-2 Context Tag Attributes

Attribute	Description	Required	Runtime Expression
`variable`	Declares a variable that has the type `L10nContext` and the specified name. The named variable can be used as a scripting variable within the tag body. The value assigned to `variable` is the declared `L10nContext` object's name. See the example following this table.	no	no

`context`	Specifies the current localization context explicitly. For example:
	`<l10n:context`
	`context = "<%= someL10nContextObject %>" />`
		no	yes

`resourceBundleName`	Specifies the resource bundle base name.	no	yes
`locale`	Specifies the locale of the current localization context. For example: `<l10n:context locale = "<%= Locale.GERMANY%>"/>`	no	yes
`timeZone`	Specifies the time zone of the current localization context.	no	yes
`preferredLocales`	Specifies the preferred locales of the current localization context. This attribute requires that an otherwise locale be specified with the `otherwise` attribute.	no	yes
`otherwise`	Specifies the locale to use if no resource bundle can be found for any of the preferred locales. This attribute requires that a preferred locale be specified with the `preferred` attribute.	no	yes
`scope`	Specifies the scope of the current localization context. Allowed values for scope are: • `PageContext.APPLICATION_SCOPE` • `PageContext.SESSION_SCOPE` • `PageContext.REQUEST_SCOPE` • `PageContext.PAGE_SCOPE` For the meaning of each scope, see the documentation for the Java `PageContext` class in the Java 2 platform area of the `java.sun.com` web site.	no	yes

The following example shows how to declare and use the scripting variable that is created with the `context` tag's `variable` attribute. In the example, the scripting variable is used to access the `getLocale` method of the `L10nContext` class.

```
<%@ taglib uri="http://www.cisco.com/taglibs/localization" prefix="l10n" %>
...
<%-- Set the locale and encoding of the response --%>
<%-- to the current locale of the L10nContext.   --%>
<l10n:context variable="l10nContext">
<%
response.setLocale(l10nContext.getLocale());
Log.debug("decorateResponse.jspi, locale=", response.getLocale());
%>
</l10n:context>
```

locale, timeZone, and language Tags

The `locale`, `timeZone`, and `language` tags get a text description of the specified characteristic for the current localization context (`L10nContext`):

- `locale`—Gets a text description of the locale.
- `timeZone`—Gets a text description of the time zone.
- `language`—Gets a text description of the language.

The following example uses the `locale` and `timeZone` tags to obtain a text description for the locale and time zone of the current localization context.

```
<%@ taglib uri="http://www.cisco.com/taglibs/localization" prefix="l10n" %>
...
The current locale is: <l10n:locale /> <BR>
The current timeZone is: <l10n:timeZone /> <BR>
```

The generated HTML page displays the following:

```
The current locale is: English (United States)
The current timeZone is: America/New_York
```

country Tag

The `country` tag gets a text description for a country. The attribute supplied with the `country` tag specifies the country. If no attribute is used, the description is for the country of the current localization context (`L10nContext`). The text description is always in the language of the current localization context. Table 4-3 lists the attributes of the `country` tag.

Table 4-3 Country Tag Attributes

Attribute	Description	Required	Runtime Expression
`code`	Gets the text description for the country specified by a two-character ISO 3166 country code, such as “JP” for Japan. For a list of country codes, see: `http://www.chemie.fu-berlin.de/diverse/doc/ISO_3166.html`	no	yes
`locale`	Gets the text description for the country of the specified locale.	no	yes
`context`	Gets the text description for the country of the specified localization context.	no	yes

The following example uses the `country` tag and its various attributes to obtain text descriptions for some countries.

```
<%@ taglib uri="http://www.cisco.com/taglibs/localization" prefix="l10n" %>
...
<!-- No attribute specified -->
The country of the current localization context is: <l10n:country /> <BR>

<!-- Country code specified -->
The country associated with the country code "CH" is: <l10n:country code = "CH" /><BR>
<BR>

<!-- For the swissFrench object, set language to French and the country to Switzerland -->
<% Locale swissFrench = new Locale("fr", "CH"); %>

<!-- Set the locale of the current L10nContext localization context -->
<l10n:context locale = "<%= swissFrench %>" >

After changing the locale of the current localization context, <BR>
the country of the localization context <BR>
(in the language of that localization context) is: <l10n:country /> <BR>

</l10n:context>

<BR>

<!-- Set the locale of the current L10nContext localization context -->
<l10n:context locale = "<%= Locale.GERMANY%>" >

After changing the locale of the current localization context, <BR>
the country of the localization context <BR>
(in the language of that localization context) is: <l10n:country /> <BR>

</l10n:context>
```

The generated HTML page displays the following:

```
The country of the current localization context is: United States
The country associated with the country code "CH" is: Switzerland

After changing the locale of the current localization context,
the country of the localization context
(in the language of that localization context) is: Suisse

After changing the locale of the current localization context,
the country of the localization context
(in the language of that localization context) is: Deutschland
```

format Tag

The `format` tag can be used to convert currency, number, date, and time values into text according to the formatting conventions associated with the Java class `java.text.MessageFormat` and the current `L10nContext` localization context.

Table 4-4 lists the attributes of the `format` tag.

Table 4-4 Format Tag Attributes

Attribute	Description	Required	Runtime Expression
`currency`	Specifies a `Number`, `double`, or `long` value to be converted into a currency.	no	yes
`number`	Specifies a `Number`, `double`, or `long` value to be converted into a number.	no	yes
`date`	Specifies a `Date` value to be converted into a date.	no	yes
`time`	Specifies a `Date` value to be converted into a time.	no	yes
`dateTime`	Specifies a `Date` value to be converted using date-time format.	no	yes
`shortDate`	Specifies a `Date` value to be converted using short date format.	no	yes
`shortTime`	Specifies a `Date` value to be converted using short time format.	no	yes
`shortDateTime`	Specifies a `Date` value to be converted using short date-time format.	no	yes
`mediumDate`	Specifies a `Date` value to be converted using medium date format.	no	yes
`mediumTime`	Specifies a `Date` value to be converted using medium time format.	no	yes
`mediumDateTime`	Specifies a `Date` value to be converted using medium date-time format.	no	yes
`longDate`	Specifies a `Date` value to be converted using long date format.	no	yes
`longTime`	Specifies a `Date` value to be converted using long time format.	no	yes
`longDateTime`	Specifies a `Date` value to be converted using long date-time format.	no	yes
`fullDate`	Specifies a `Date` value to be converted using full date format.	no	yes
`fullTime`	Specifies a `Date` value to be converted using full time format.	no	yes
`fullDateTime`	Specifies a `Date` value to be converted using full date-time format.	no	yes

The following example uses the `format` tag to format:

- A `double` value into a currency amount
- A `Date` value into a time
- A `Date` value into a long time

```
<%@ taglib uri="http://www.cisco.com/taglibs/localization" prefix="l10n" %>
...
<%
double amount = 99.99;
GregorianCalendar calendar = new GregorianCalendar();
Date curDateTime = calendar.getTime();
%>

The current locale is:  <l10n:locale /> <BR>
Amount formatted as currency:  <l10n:format currency="<%= amount %>" /> <BR>
Date value formatted as time:  <l10n:format time="<%= curDateTime %>" /> <BR>
Date value formatted as long time:  <l10n:format longTime="<%= curDateTime %>" /> <BR>
```

The generated HTML page displayed the following:

```
The current locale is: English (United States)
Amount formatted as currency: $99.99
Date value formatted as time: 7:59:18 PM
Date value formatted as long time: 7:59:18 PM EDT
```

resource Tag

The `resource` tag obtains a resource from the resource bundle of the current localization context (`L10nContext`) and for a specified key. A resource obtained is the value part of the key-value pair in a properties file. Table 4-5 lists the attributes of the `resource` tag.

Table 4-5 Resource Tag Attributes

Attribute	Description	Required	Runtime Expression
`key`	Specifies the key for which to obtain a resource.	yes	yes

The following example uses the `resource` tag to obtain the resource associated with the key `EnterUserName` from the `message_en.properties` file of the NWSP sample web application.

```
<%@ taglib uri="http://www.cisco.com/taglibs/localization" prefix="l10n" %>
...
<l10n:context resourceBundleName="messages.properties" />

The resource for the key "PleaseAuthenticate" is: <l10n:resource key="PleaseAuthenticate">
Text in the tag body appears in Dreamweaver but not in the generated HTML.
</l10n:resource> <BR>
```

The generated HTML page displays the following:

```
The resource for the key "PleaseAuthenticate" is: Please log in
```

template Tag

The `template` tag can be used with the `resource` tag for token replacement in a resource. For a template (a message format) that appears in its tag body, the `template` tag replaces tokens in the template with the values specified with the `param` or `params` attributes. The syntax that you use for a token is specified by the class `java.text.MessageFormat`. The `template` tag formats locale-sensitive information such as dates, messages, and numbers using the conventions of the current localization context (`L10nContext`).

Table 4-6 lists the attributes of the `template` tag.

Table 4-6 Template Tag Attributes

Attribute	Description	Required	Runtime Expression
`param`	Specifies a single parameter value to substitute for a token. The `param` attribute is used when an array of parameter values is not required. The type of the parameter can be `Object` or any of the primitive types.	no	yes
`params`	Specifies an array of type `Object[]` containing parameter values to substitute for one or more tokens.	no	yes

In the following example, tokens in a message format in the `template` tag body are replaced by the values specified in `nameAndAge`:

```
<%! Object[] nameAndAge = new Object[] {"John", new Long(5)}; %>
<l10n:template params = "<%= nameAndAge %>" >
My name is {0}, I am {1,number,integer} years old.
</l10n:template >
```

The text in brackets (`{ }`) is a token. The `template` tag replaces tokens with the values specified with the `params` attribute. In the preceding example, token `{0}` is replaced by element 0 of the `nameAndAge` array, and token `{1, number, integer}` is replaced by element 1 of the `nameAndAge` array. The generated HTML page displays the following:

```
My name is John, I am 5 years old.
```

The `template` tag can be used with the `resource` tag to get a resource from a resource bundle and to replace tokens in the resource with specified parameter values. In a properties file, the value part of a key-value pair is a template, a message format, that can contain one or more tokens. As an example, assume the following key-value pair in a properties file:

```
message=Error is {0} ({1,number,integer}).
```

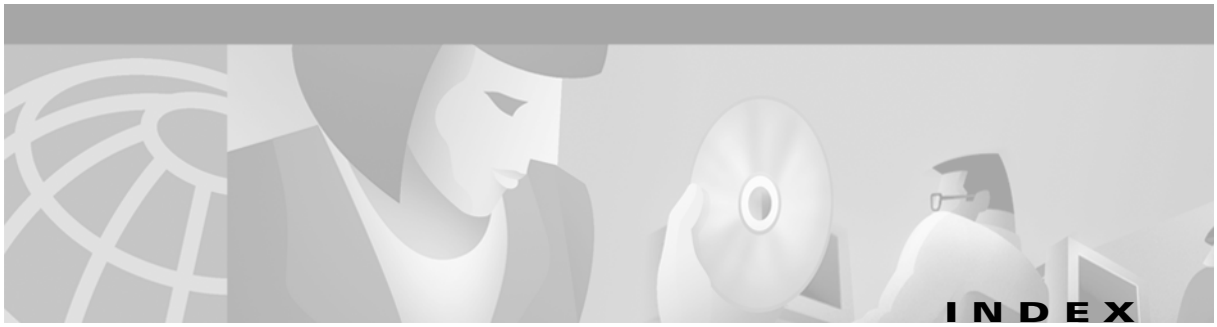
The following `template` tag uses the `resource` tag in its body to retrieve the value from and replace tokens in the preceding properties file entry:

```
<%! Object[] errorInfo = new Object[] {"Not Found", new Long(403) };  %>

<l10n:template params = "<%= errorInfo %>" >
<l10n:resource key = "message" />
</l10n:template>
```

For the preceding example, the generated HTML page displays the following:

```
Error is Not Found (403).
```



INDEX

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

R

S

T

U

V

W