



Cisco Subscriber Edge Services Manager Web Developer Guide

SESM 3.3

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

Text Part Number: OL-2068-05



THE SPECIFICATIONS AND INFORMATION REGARDING THE PRODUCTS IN THIS MANUAL ARE SUBJECT TO CHANGE WITHOUT NOTICE. ALL STATEMENTS, INFORMATION, AND RECOMMENDATIONS IN THIS MANUAL ARE BELIEVED TO BE ACCURATE BUT ARE PRESENTED WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED. USERS MUST TAKE FULL RESPONSIBILITY FOR THEIR APPLICATION OF ANY PRODUCTS.

THE SOFTWARE LICENSE AND LIMITED WARRANTY FOR THE ACCOMPANYING PRODUCT ARE SET FORTH IN THE INFORMATION PACKET THAT SHIPPED WITH THE PRODUCT AND ARE INCORPORATED HEREIN BY THIS REFERENCE. IF YOU ARE UNABLE TO LOCATE THE SOFTWARE LICENSE OR LIMITED WARRANTY, CONTACT YOUR CISCO REPRESENTATIVE FOR A COPY.

The Cisco implementation of TCP header compression is an adaptation of a program developed by the University of California, Berkeley (UCB) as part of UCB's public domain version of the UNIX operating system. All rights reserved. Copyright © 1981, Regents of the University of California.

NOTWITHSTANDING ANY OTHER WARRANTY HEREIN, ALL DOCUMENT FILES AND SOFTWARE OF THESE SUPPLIERS ARE PROVIDED "AS IS" WITH ALL FAULTS. CISCO AND THE ABOVE-NAMED SUPPLIERS DISCLAIM ALL WARRANTIES, EXPRESSED OR IMPLIED, INCLUDING, WITHOUT LIMITATION, THOSE OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT OR ARISING FROM A COURSE OF DEALING, USAGE, OR TRADE PRACTICE.

IN NO EVENT SHALL CISCO OR ITS SUPPLIERS BE LIABLE FOR ANY INDIRECT, SPECIAL, CONSEQUENTIAL, OR INCIDENTAL DAMAGES, INCLUDING, WITHOUT LIMITATION, LOST PROFITS OR LOSS OR DAMAGE TO DATA ARISING OUT OF THE USE OR INABILITY TO USE THIS MANUAL, EVEN IF CISCO OR ITS SUPPLIERS HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

CCSP, the Cisco Square Bridge logo, Follow Me Browsing, and StackWise are trademarks of Cisco Systems, Inc.; Changing the Way We Work, Live, Play, and Learn, and iQuick Study are service marks of Cisco Systems, Inc.; and Access Registrar, Aironet, ASIST, BPX, Catalyst, CCDA, CCDP, CCIE, CCIP, CCNA, CCNP, Cisco, the Cisco Certified Internetwork Expert logo, Cisco IOS, Cisco Press, Cisco Systems, Cisco Systems Capital, the Cisco Systems logo, Cisco Unity, Empowering the Internet Generation, Enterprise/Solver, EtherChannel, EtherFast, EtherSwitch, Fast Step, FormShare, GigaDrive, GigaStack, HomeLink, Internet Quotient, IOS, IP/TV, iQ Expertise, the iQ logo, iQ Net Readiness Scorecard, LightStream, Linksys, MeetingPlace, MGX, the Networkers logo, Networking Academy, Network Registrar, *Packet*, PIX, Post-Routing, Pre-Routing, ProConnect, RateMUX, ScriptShare, SlideCast, SMARTnet, StrataView Plus, SwitchProbe, TeleRouter, The Fastest Way to Increase Your Internet Quotient, TransPath, and VCO are registered trademarks of Cisco Systems, Inc. and/or its affiliates in the United States and certain other countries.

All other trademarks mentioned in this document or Website are the property of their respective owners. The use of the word partner does not imply a partnership relationship between Cisco and any other company. (0501R)

Cisco Subscriber Edge Services Manager Web Developer Guide
Copyright © 2002-2005 Cisco Systems, Inc. All rights reserved.



About This Guide	ix
Document Objectives	ix
Audience	ix
Document Organization	x
Document Conventions	x
Related Documentation	xi
Obtaining Documentation	xi
Cisco.com	xii
Documentation CD-ROM	xii
Ordering Documentation	xii
Documentation Feedback	xii
Obtaining Technical Assistance	xiii
Cisco TAC Website	xiii
Opening a TAC Case	xiii
TAC Case Priority Definitions	xiv
Cisco Developer Support Program	xiv
Program Benefits	xiv
Contacting Cisco Developer Support	xv
Obtaining Additional Publications and Information	xv

CHAPTER 1

SESM Web Development Overview	1-1
Cisco SESM System	1-1
Cisco SESM Web Portal Applications	1-3
Conventional WebSites: A Problem	1-5
SESM Technology: A Solution	1-6
Customizing a SESM Web Portal Application: An Overview	1-7
Basic SESM Customization	1-7
Advanced SESM Customization	1-8
System-Integrator Customization	1-8
Configuring a Customized SESM Web Portal	1-8
J2EE Development Platform	1-8
Changing web.xml and webdefault.xml	1-9
Importance of the Web Portal Application Name	1-9

Hardware and Software Requirements for Development 1-10
 Demo Installation and Development 1-10
 Environment Variables 1-10
 Development Tools 1-11
 Learning about SESM Web Portal Application Development 1-11

CHAPTER 2

Basic SESM Customization and Development 2-1

Changing the Look-and-Feel Elements 2-1
 Using a SESM Web Portal Application 2-2
 NWSP User Interface 2-2
 JavaServer Pages 2-3
 Banner Images and Background Colors 2-4
 Icons and Buttons 2-4
 Navigation Bar 2-4
 Style Sheets 2-4
 Dreamweaver Templates 2-5
 Customizing Attribute 18 Messages 2-5
 Customization for iPass Logon 2-6
 Developing a SESM Web Portal Application 2-6
 Defining the Business Requirements 2-7
 Designing and Implementing a SESM Web Portal Application 2-7
 SESM Class Libraries and Tag Library Descriptor Files 2-8
 Javadoc Documentation 2-10
 JSP Compilation for Web Portal Application Development (On the Fly Compilation) 2-10
 Compiling JavaServer Pages for Web Portal Applications in Production 2-12
 Demo Installation 2-12
 Dreamweaver MX Live Data Window 2-14
 Double-Byte Language Packs 2-16
 General Web Development Considerations 2-17
 Debugging a SESM Web Portal Application 2-17
 Using Test Decorators 2-18
 Using Device Simulators for WAP and WML 2-20
 Using Device Simulators for Pocket PCs 2-21
 Managing a SESM Website 2-22

CHAPTER 3**Advanced SESM Customization 3-1**

Introduction	3-1
SESM Architecture: An Overview	3-2
Model	3-2
Controls	3-3
Views	3-4
MVC Design Pattern Example	3-4
NWSP Controls, View Beans, and Views	3-6
SESM Software Concepts	3-8
User Shapes and User-Shape Decoration	3-8
Decorators	3-9
Using a Sparse-Tree Directory Structure	3-10
Website Pages Hierarchy	3-11
User-Shape Hierarchy	3-11
User-Shape Example	3-11
Location of Directory Dimensions and Order of ShapeDecorator Dimensions	3-12
Sparse-Tree Directory Structure	3-13
Implementing the Sparse-Tree Directory	3-14
Searches for a Web Resource	3-14
Example 1: Searches for a Web Resource	3-15
Example 2: Searches for a Web Resource	3-17
Example 3: Searches for a Web Resource	3-17
Decorating a User Shape	3-18
Configuring User-Shape Dimensions	3-19
ShapeDecorator Initialization Parameters	3-19
Default Dimension Decorator Values	3-21
SESM-supplied Dimension Decorators	3-21
Creating or Customizing Dimension Decorators	3-25
Modifying Dimension Decorators	3-28
Adding Dimension Decorators	3-29
Modifying SESM Web Portal Application Functionality	3-29
Modifying the Functionality of JSP-Page Views	3-30
Creating JSP-Page Decorators and Post-Decorators	3-31
Using the SESM Deployment Descriptor File	3-33
Configuration Information	3-34
Configuration Techniques	3-35
Servlet Mapping	3-36
Servlet Chaining	3-36
Mapping a Virtual File Name to an Actual File Name	3-37

preDecorate Initialization Parameter 3-38
 SESM Deployment Descriptor File Techniques Example 3-38
 Invoking a Decorator 3-40
 Decorator Invocation Locations 3-40
 Decorator Invocation Methods 3-41

CHAPTER 4

SESM Web Portal Applications 4-1
 NWSP Web Portal Application 4-2
 NWSP User Interface 4-2
 NWSP Customization Based on Subscriber Characteristics 4-2
 NWSP Functionality 4-2
 NWSP Directory Hierarchy 4-3
 NWSP JavaServer Pages and Servlets 4-6
 Modularized JSPs 4-6
 JSPs for Basic SESM Web Portal Application Functions 4-7
 JSPs for SPE Installation Functions 4-9
 JSPs for the Service List 4-10
 JSP for the Navigation Bar 4-17
 JSPs for User-Shape Decoration 4-18
 JSPs and JSPFragment Class for Include File Log Entries 4-18
 Servlets for the SESM Controls 4-19
 NWSP Templates 4-19
 Main Template 4-19
 Banner-Only Template 4-21
 PDA Web Portal Application 4-21
 PDA User Interface 4-22
 PDA Functionality 4-22
 WAP Web Portal Application 4-23
 WAP User Interface 4-23
 WAP Functionality 4-24
 Captive Portal Web Solution 4-24
 Captive Portal Solution Overview 4-25
 Captive Portal Solution Web Applications 4-25
 Captive Portal Solution Redirection Types 4-27
 Captive Portal Solution Configuration 4-27
 Captive Portal 4-28
 Captive Portal Web Application 4-28
 Content Web Applications 4-29

CHAPTER 5**SESM Internationalization and Localization 5-1**

- Localizing a Web Portal Application 5-2
- Using Resource Bundles 5-3
 - Using Properties Files 5-3
- Using Internationalized Resources 5-5
- Using the Localization Tag Library 5-6
 - context Tag 5-7
 - locale, timeZone, and language Tags 5-9
 - country Tag 5-9
 - format Tag 5-11
 - resource Tag 5-12
 - template Tag 5-13
- Setting a Default Localization Context 5-14

APPENDIX A**SESM Testing and Development A-1**

- Introduction A-1
- Using Web Portal Demo Installation A-2
 - Installing SESM in Web Portal Demo Installation A-3
 - Installation Procedure A-3
 - Choosing a Web Browser for a Demo A-3
 - Running a Web Portal as a Demo Installation A-4
 - Logon Names and Passwords for a Demo A-5
- SSG Simulator A-5
 - Installing the SSG Simulator A-6
 - Running SSG Simulator A-7
- SESM Bundled RADIUS Server A-8
 - Installing SESM with Bundled RADIUS Server A-9
 - Profile File Requirements A-10
 - Communication with Components in the Deployment A-10
 - Running Bundled RADIUS Server A-11
- Demo Profile File A-11
 - Installed Path Names of Demo Profile Files A-11
 - File Contents and Format A-12
 - Subscriber Profile Attributes A-12
 - Service Profile Attributes A-15
- Downloading International Character Sets A-17

APPENDIX B

SESM Tag Libraries B-1

- Configuring a Tag Library **B-1**
- Iterator Tag Library **B-2**
 - start Tag **B-2**
 - val Tag **B-3**
- Navigator Tag Library **B-3**
 - decorate Tag **B-4**
- Shape Tag Library **B-4**
 - file Tag **B-5**
 - path Tag **B-6**

APPENDIX C

SESM Utility Servlets Quick Reference C-1

- Introduction **C-1**
- Using the preDecorate and postDecorate Parameters **C-2**
- SESM Utility Servlet Quick Reference **C-3**

APPENDIX D

Using the Cisco Navigation Bar Extension D-1

- Introduction **D-1**
- Creating a Navigation Bar **D-1**
 - New JSP Page **D-2**
 - JSP Page with Existing JavaScript **D-3**
 - JSP Page with Included JavaScript **D-4**
- Installing the Navigation Bar Extension **D-4**

INDEX



About This Guide

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

- [Document Objectives](#)
- [Audience](#)
- [Document Organization](#)
- [Document Conventions](#)
- [Related Documentation](#)
- [Obtaining Documentation](#)
- [Obtaining Technical Assistance](#)
- [Obtaining Additional Publications and Information](#)

Document Objectives

This guide explains how to develop a Cisco Subscriber Edge Services Manager (SESM) web portal application. It also provides web development information on the Captive Portal. For both application types, the guide describes SESM web application components and techniques.

Audience

This guide is intended for web designers and web developers responsible for the look-and-feel, branding, and functionality of a SESM web portal application. 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 and appendixes:

Chapter	Title	Description
Chapter 1	SESM Web Development Overview	Provides an overview of a Cisco SESM system and a SESM web portal application.
Chapter 2	Basic SESM Customization and Development	Explains how customize the look-and-feel elements of a SESM web portal application. Also provides information on web portal application development.
Chapter 3	Advanced SESM Customization	Explains some of the advanced customization techniques that you can use with a Cisco SESM web portal application.
Chapter 4	SESM Web Portal Applications	Provides information about the SESM web portal applications and Captive Portal solution and describes how you can use and modify the web components.
Chapter 5	SESM Internationalization and Localization	Explains the SESM components and techniques that help a deployer internationalize and localize a SESM web portal application.
Appendix A	SESM Tag Libraries	Provides information on configuring a SESM tag library and describes the SESM Iterator, Navigator, and Shape tag libraries.
Appendix B	SESM Utility Servlets Quick Reference	Provides quick-reference information on the utility Java servlets that are used in a SESM web portal application.
Appendix C	Using the Cisco Navigation Bar Extension	Explains how you install and use the Cisco Navigation Bar extension to Dreamweaver.
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 covered 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

Documentation for SESM 3.3 includes:

- *Cisco Subscriber Edge Services Manager Release Notes for Release 3.3 (1)*
- *Cisco Subscriber Edge Services Manager Introduction*
- *Cisco Subscriber Edge Services Manager Installation Guide*
- *Cisco Subscriber Edge Services Manager Administration and Configuration Guide*
- *Cisco Subscriber Edge Services Manager Web Portal Guide*
- *Cisco Subscriber Edge Services Manager Profile Management Guide*
- *Cisco Subscriber Edge Services Manager Web Services Gateways Guide*
- *Cisco Subscriber Edge Services Manager Web Developer Guide*
- *Cisco Subscriber Edge Services Manager SDK Programmer Guide*
- *Cisco Subscriber Edge Services Manager Troubleshooting Guide*

Documentation for SESM is online at:

<http://www.cisco.com/univercd/cc/td/doc/solution/sesm/index.htm>

Documentation for the Cisco SSG is online at:

<http://www.cisco.com/univercd/cc/td/doc/product/software/ios122/122newft/122t/122t13/ssg/>

Information related to configuring the SSG authentication, authorization, and accounting features is included in:

- *Cisco IOS Security Configuration Guide:*
http://www.cisco.com/univercd/cc/td/doc/product/software/ios123/123cgcr/sec_vcg.htm
- *Cisco IOS Security Command Reference*

If you are including the Cisco Access Registrar (a RADIUS server) in your SESM deployment, see the documentation for Cisco Access Registrar (AR) online at:

<http://www.cisco.com/univercd/cc/td/doc/product/rtrmgmt/cnsar/index.htm>

Obtaining Documentation

Cisco provides several ways to obtain documentation, technical assistance, and other technical resources. These sections explain how to obtain technical information from Cisco Systems.

Cisco.com

You can access the most current Cisco documentation on the World Wide Web at this URL:

<http://www.cisco.com/univercd/home/home.htm>

You can access the Cisco website at this URL:

<http://www.cisco.com>

International Cisco websites can be accessed from this URL:

http://www.cisco.com/public/countries_languages.shtml

Documentation CD-ROM

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

Registered Cisco.com users can order a single Documentation CD-ROM (product number DOC-CONDOCCD=) through the Cisco Ordering tool:

http://www.cisco.com/en/US/partner/ordering/ordering_place_order_ordering_tool_launch.html

All users can order annual or quarterly subscriptions through the online Subscription Store:

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

Ordering Documentation

You can find instructions for ordering documentation at this URL:

http://www.cisco.com/univercd/cc/td/doc/es_inpkc/pdi.htm

You can order Cisco documentation in these ways:

- Registered Cisco.com users (Cisco direct customers) can order Cisco product documentation from the Networking Products MarketPlace:
<http://www.cisco.com/en/US/partner/ordering/index.shtml>
- Nonregistered Cisco.com users can order documentation through a local account representative by calling Cisco Systems Corporate Headquarters (California, USA.) at 408 526-7208 or, elsewhere in North America, by calling 800 553-NETS (6387).

Documentation Feedback

You can submit comments electronically on Cisco.com. On the Cisco Documentation home page, click **Feedback** at the top of the page.

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

You can submit comments by using the response card (if present) behind the front cover of your document or by writing to the following address:

Cisco Systems
Attn: Customer Document Ordering
170 West Tasman Drive
San Jose, CA 95134-9883

We appreciate your comments.

Obtaining Technical Assistance

For all customers, partners, resellers, and distributors who hold valid Cisco service contracts, the Cisco Technical Assistance Center (TAC) provides 24-hour, award-winning technical support services, online and over the phone. Cisco.com features the Cisco TAC website as an online starting point for technical assistance.

Cisco TAC Website

The Cisco TAC website (<http://www.cisco.com/tac>) provides online documents and tools for troubleshooting and resolving technical issues with Cisco products and technologies. The Cisco TAC website is available 24 hours a day, 365 days a year.

Accessing all the tools on the Cisco TAC website requires a Cisco.com user ID and password. If you have a valid service contract but do not have a login ID or password, register at this URL:

<http://tools.cisco.com/RPF/register/register.do>

Opening a TAC Case

The online TAC Case Open Tool (<http://www.cisco.com/tac/caseopen>) is the fastest way to open P3 and P4 cases. (Your network is minimally impaired or you require product information). After you describe your situation, the TAC Case Open Tool automatically recommends resources for an immediate solution. If your issue is not resolved using these recommendations, your case will be assigned to a Cisco TAC engineer.

For P1 or P2 cases (your production network is down or severely degraded) or if you do not have Internet access, contact Cisco TAC by telephone. Cisco TAC engineers are assigned immediately to P1 and P2 cases to help keep your business operations running smoothly.

To open a case by telephone, use one of the following numbers:

Asia-Pacific: +61 2 8446 7411 (Australia: 1 800 805 227)

EMEA: +32 2 704 55 55

USA: 1 800 553-2447

For a complete listing of Cisco TAC contacts, go to this URL:

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

TAC Case Priority Definitions

To ensure that all cases are reported in a standard format, Cisco has established case priority definitions.

Priority 1 (P1)—Your network is “down” or there is a critical impact to your business operations. You and Cisco will commit all necessary resources around the clock to resolve the situation.

Priority 2 (P2)—Operation of an existing network is severely degraded, or significant aspects of your business operation are negatively affected by inadequate performance of Cisco products. You and Cisco will commit full-time resources during normal business hours to resolve the situation.

Priority 3 (P3)—Operational performance of your network is impaired, but most business operations remain functional. You and Cisco will commit resources during normal business hours to restore service to satisfactory levels.

Priority 4 (P4)—You require information or assistance with Cisco product capabilities, installation, or configuration. There is little or no effect on your business operations.

Cisco Developer Support Program

The Developer Support Program was developed to provide formalized support for Cisco interfaces to accelerate the delivery of compatible solutions to Cisco customers. The program web site at <http://www.cisco.com/go/developersupport> provides a central resource point for all your development needs.

Program Benefits

- Product and document downloads
- Bug reports
- Sample scripts
- Frequently Asked Questions
- Access to Developer Support Engineers

Many of the product and document downloads are accessible with a Cisco.com guest level login. However, as a member of the program, you will get access to all the program benefits listed above to promote your development efforts. The subscription also provides the ability to open support cases using the same infrastructure and processes used by Cisco Technical Assistance Center (TAC).

Our Subscription membership is fee-based. The Developer Support Agreement, with the subscription fees and list of supported interfaces, is available on the Developer Support Web site.



Note

The Cisco TAC does NOT provide support for this API/interface under standard hardware or software support agreements. All technical support for this API/interface, from initial development assistance through API troubleshooting/bugs in final production applications, is provided by Cisco Developer Support and requires a separate Developer Support contract. When opening cases, a Developer Support contract number must be provided in order to receive support.

Contacting Cisco Developer Support

You can contact Cisco Developer Support using the following:

- Email: developer-support@cisco.com
- Web: <http://www.cisco.com/go/developersupport>

Obtaining Additional Publications and Information

Information about Cisco products, technologies, and network solutions is available from various online and printed sources.

- *The Cisco Product Catalog* describes the networking products offered by Cisco Systems, as well as ordering and customer support services. Access the *Cisco Product Catalog* at this URL:
http://www.cisco.com/en/US/products/products_catalog_links_launch.html
- Cisco Press publishes a wide range of networking publications. Cisco suggests these titles for new and experienced users: Internetworking Terms and Acronyms Dictionary, Internetworking Technology Handbook, Internetworking Troubleshooting Guide, and the Internetworking Design Guide. For current Cisco Press titles and other information, go to Cisco Press online at this URL:
<http://www.ciscopress.com>
- Packet magazine is the Cisco quarterly publication that provides the latest networking trends, technology breakthroughs, and Cisco products and solutions to help industry professionals get the most from their networking investment. Included are networking deployment and troubleshooting tips, configuration examples, customer case studies, tutorials and training, certification information, and links to numerous in-depth online resources. You can access Packet magazine at this URL:
<http://www.cisco.com/go/packet>
- iQ Magazine is the Cisco bimonthly publication that delivers the latest information about Internet business strategies for executives. You can access iQ Magazine at this URL:
<http://www.cisco.com/go/iqmagazine>
- Internet Protocol Journal is a quarterly journal published by Cisco Systems for engineering professionals involved in designing, developing, and operating public and private internets and intranets. You can access the Internet Protocol Journal at this URL:
http://www.cisco.com/en/US/about/ac123/ac147/about_cisco_the_internet_protocol_journal.html
- Training—Cisco offers world-class networking training. Current offerings in network training are listed at this URL:
<http://www.cisco.com/en/US/learning/index.html>



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 Portal application—the SESM Web Portal. It also provides a high-level overview of the web components and techniques used to develop a Cisco SESM Web Portal application.



Note

Most information in this guide concerns web development for the *Cisco SESM Web Portal*. In addition, the [Captive Portal Web Solution, page 4-24](#) provides web development information on the Captive Portal solution web applications. For information on documentation for other SESM applications such as the RADIUS Data Proxy (RDP) application, see the [Related Documentation, page xi](#).

Cisco SESM System

Cisco SESM is an extensible set of applications for providing on-demand services, service management, and subscriber management to the broadband, public wireless LAN, and mobile wireless markets. Cisco SESM is part of a Cisco solution that allows subscribers of DSL (Digital Subscriber Lines), cable, wireless, and dialup to simultaneously access multiple services provided by different Internet service providers, application service providers, and corporate access servers.

Cisco SESM enables you to create a customized Web Portal application that provides a branded Web Portal for individual subscribers. Through the Cisco SESM Web 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 configuration that is used, you can configure a deployed SESM Web Portal application for one of two installations:

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

A Cisco SESM Web Portal application enables each subscriber to:

- Select or deselect services to which they are subscribed.
- Subscribe to or unsubscribe from services they are authorized to access (By default, available in SESM SPE installation. However, a service provider can implement this feature in a SESM Radius installation.)

- Change account details, such as address information and passwords (By default, available in SESM SPE installation. However, a service provider can implement this feature in SESM Radius installation.)
- Create subaccounts for other family members (By default, available in SESM SPE installation. However, a service provider can implement this feature in SESM Radius installation.)
- Define a personal firewall.
- View the status of service connections.
- View system messages.

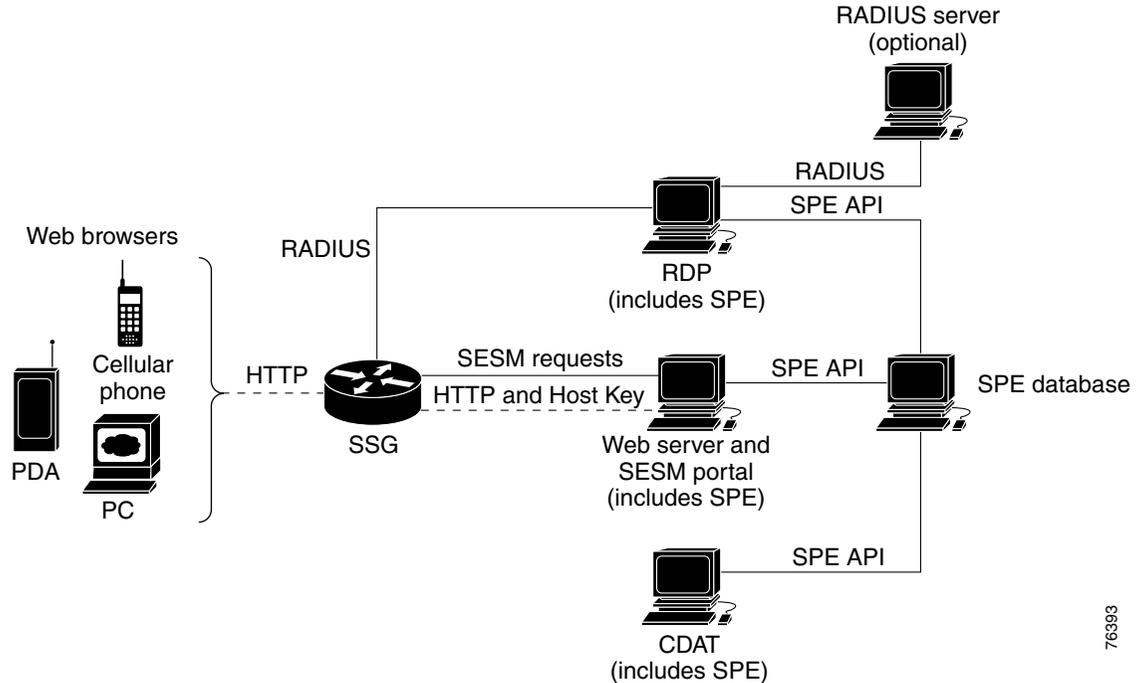
A Cisco SESM system includes one or more SESM servers running one or more Java 2 Enterprise Edition (J2EE) web servers.

The Cisco SESM system components need the Cisco Service Selection Gateway (SSG) installed and configured on the network. Cisco SSG is a Cisco IOS software feature module that is available on the Cisco 6400, 7200, and 7400 platforms. Cisco SSG works in conjunction with SESM to provide a number of features such as the prepaid billing feature. The HTTP Redirect feature of SSG works with a SESM Captive Portal web application.

Figure 1-1 shows a SESM configuration with the components for an LDAP-compliant directory implementation including:

- RDP (RADIUS Data Proxy) is a RADIUS proxy server that uses the SPE 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 set of web-based facilities that enable the service-provider administrator to perform two different sets of tasks:
 - Remote managing and monitoring of Cisco SESM applications.
 - Managing subscriber, service, and policy information used by Cisco SESM and SSG.

Figure 1-1 SESM System Components



For detailed information on SESM system components, see *Cisco Subscriber Edge Services Manager Introduction*.

Cisco SESM Web Portal Applications

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

- A business model of the relationship between subscribers and services. In SESM SPE installation, 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 a SESM Web Portal application.
- Development guidelines that allow a SESM Web Portal application to be deployed and configured in the context of a J2EE web server. The Jetty web server from Mortbay.com is installed with SESM — although a Cisco SESM Web Portal application can be deployed with any J2EE-compliant web server. However the SSG Port-Bundle Host Key (PBHK) feature requires the Jetty web server.

76393

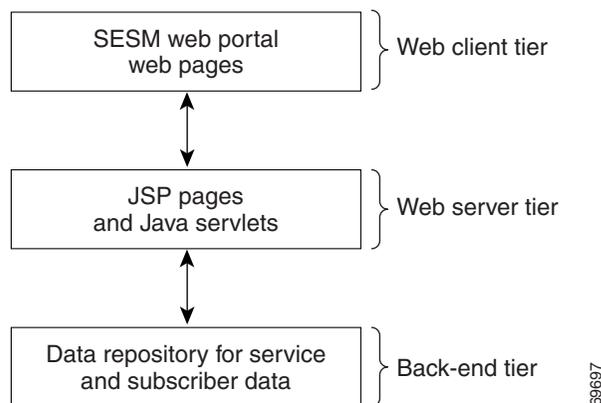
Three-Tiered Web Portal application

From a development perspective, a Cisco SESM Web Portal 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 JSPs 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 Portal Application



The interactions between a SESM Web Portal application and the back-end components are implemented with a set of specialized SESM programming interfaces. Many of these interactions are preprogrammed in the JSPs and can be configured by the SESM deployer. The functionality of the pre programmed JSPs can be modified or extended by the service-provider's developer. Because no extensive Java programming is required, SESM web development can be completed in a short period of time.

Web Development Roles

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 website.
- Web developer—Responsible for integrating the HTML or WML design with the SESM web components using JSPs.

SESM provides components and techniques for customizing and localizing the web pages. The web designer and web developer determine what customization is needed and use the SESM components and techniques to modify the JSPs and other SESM components in the required manner.

Cisco SESM Web Portal Applications

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

Three other sample Web Portal applications demonstrate how SESM might be used for service selection on specific devices:

- The PDA (personal digital assistant) Web Portal application is designed for handheld devices.
- The WAP (Wireless Application Protocol) Web Portal application is designed for mobile phones.
- The SP (Subscriber Portal) Web Portal application is designed is designed for the Public Wireless LAN (PWLAN).

SESM also includes three sample Captive Portal solution web applications:

- Captive Portal web application.
- Service portal web application.
- Message portal web application.

For information on the SESM Web Portal applications, see [Chapter 4, "SESM Web Portal Applications."](#)

Conventional WebSites: A Problem

Most conventional websites 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 websites.

The problem with conventional websites is that they cannot adapt their content and format to the variety of user characteristics that are currently on the web. 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 the same information or services must be provided to the same subscriber with a variety of applications, devices, and locations.

Customer circumstances such as the device, bandwidth, and language are outside of the control of the website. Other customer characteristics including whether it is a new or existing customer, brands with which the website is associated, the level of service the customer has selected are imposed by the website.

SESM Technology: A Solution

With SESM, 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 Portal application are designed to provide a SESM Web Portal that is customized for each user's shape.

Because a SESM Web Portal application is designed to detect and adapt to each user's shape, the Cisco SESM Web Portal can provide customer-tailored content and service offerings as well as a high degree of brand identity. For example, a SESM Web Portal 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

Strategy 1 can make web pages complex, error prone, and difficult to maintain. For example, the presentation of a web page targeted for a PDA differs 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

Strategy 2 is often used when there is a requirement to support multiple languages. The entire website is copied, and one language is replaced with another. For example, an English-language website 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 website for each language that is supported.

Strategy 2 ignores the commonality of elements on the multiple sets of web pages. For example, the company logo might be the same on the pages in each website but would need to be included in the set of resources in each web set. When you change the logo on one website, the change has to be replicated on the other websites. 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 websites.

Strategy 3

Strategy 3 is the approach to user customization that a Cisco SESM Web Portal application takes. This approach combines the most useful aspects of strategy 1 and strategy 2 while it tries to minimize the drawbacks of both techniques. Cisco SESM software provides 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 JSPs.
 - Templates that enable an entire set of web pages to be updated when a template is changed.
 - Images and icons that are shipped with the native image-editor source files so they can be quickly modified.
- SESM directory hierarchy—SESM uses a *sparse-tree directory structure*. With this directory structure, although some website 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 Portal application infrastructure—SESM includes Java servlets, JSPs, and specially designed Java classes that allow the SESM website to be customized and localized for each subscriber.

For detailed information on each of the preceding elements of SESM, see [Chapter 2, “Basic SESM Customization and Development”](#) and [Chapter 3, “Advanced SESM Customization.”](#)

Customizing a SESM Web Portal Application: An Overview

When you develop a Cisco SESM Web Portal application, three levels of customization are possible: basic, advanced, and system integrator.

Basic SESM Customization

In some Cisco SESM deployments, you can use one of the SESM Web Portal applications as a starting point and accomplish look-and-feel modifications to the JSPs without changing the preprogrammed SESM software. In this type of customization, the developer might change text, images, and the positioning of elements on a page. Basic customization might be appropriate when using SESM for Small-to-Medium-Size Enterprise (SME) wireless LAN hotspots that require simple look-and-feel changes to the Web Portal application or require some subset of SESM functionality, such as authentication. This level of customization is also used by deployments whose business requirements correspond closely to the existing structure of a SESM reference, such as NWSP. This chapter discusses basic customization. For information on basic customization techniques, see [Chapter 2, “Basic SESM Customization and Development”](#).

Advanced SESM Customization

Other SESM deployments will require use of the more advanced customization techniques. In this type of customization, you might add or remove JSPs, or move elements from one JSP to another. You might add a JSP dimension decorator for the SESM user-shape mechanism. You can accomplish some of the more advanced customizations by modifying the deployment descriptor file (web.xml) for the SESM Web Portal application. You accomplish other advanced customizations by creating new JSPs. No Java servlet programming is needed. For information on advanced customization techniques, see [Chapter 3, “Advanced SESM Customization.”](#)

System-Integrator Customization

Some service-provider deployments of SESM might need the professional services of system integrators to modify the functionality of the internal SESM software, such as the Java servlets for the SESM controls. This type of customization is beyond the scope of this guide and requires the involvement of Cisco technical assistance. For information on this type of customization, see *Cisco Subscriber Edge Services Manager SDK Programmer Guide*.

Configuring a Customized SESM Web Portal

Application developers might make changes to the delivered Web Portal applications, producing a customized implementation. Customized implementations require their own set of configuration files, although the files might be very similar to those provided for the Web Portal applications. This section describes how to configure your customized Web Portal applications.

J2EE Development Platform

SESM is a collection of components for creating specialized Java 2 Platform, Enterprise Edition (J2EE) web server applications. J2EE provides a framework for using various Java-based components to develop multi-tiered applications. The multi-tiered application (as opposed to the 2-tiered client server application) provides many opportunities for isolating and controlling functional pieces of a large application. For more information about the J2EE development platform, see the sun website.

Changing web.xml and webdefault.xml

J2EE configuration files, such as web.xml and webdefault.xml, define how the applications run in the J2EE environment. These files conform to Java specifications, as described in the Java Servlet Version 2.3 specifications from Sun Microsystems.

This manual describes application-specific parameters in the J2EE configuration files. For information about other parameters, see the Java Servlet Version 2.3 specifications. To download these specifications, go to:

<http://java.sun.com/aboutJava/communityprocess/first/jsr053>

Table 1-1 shows the J2EE configuration files used to configure SESM Web Portals.

Table 1-1 Summary of J2EE Configuration Files

Component	File Path and Name	Description
Container (Jetty)	jetty config webdefault.xml	This file sets attributes for the Jetty server's handling of HTTP requests and how they map to servlets and JSPs.
SESM application	applicationName webapp WEB-INF web.xml	This file defines J2EE application parameters, including parameters related to Java Server Pages (JSPs).
SESM application	applicationName webapp WEB-INF web-jetty.xml	This file is required for the port-bundle host key feature. See the Container Requirement for the Port-Bundle Host Key Feature section on page 3-1 in <i>Cisco Subscriber Edge Services Manager Web Portals Guide</i> for more information.

Importance of the Web Portal Application Name

The SESM application name that you use for a customized application is arbitrary, but it must match in all of the following locations:

- The name of the application-specific subdirectory under the installation directory. For example, the directory that holds all application specific information for the SP application is:

```
<installDir>
sp
```

- The application parameter inside the application startup script. In the installed scripts, the application name is hard-coded on the line that calls the generic start script. For example, for the SP application on Windows, the call line is:

```
call "%SCRIPTDIR%start.cmd" sp %PORTNO%
```

- Name of the application's configuration file in the jetty subdirectory. For example, for the SP application, the configuration filename is:

```
sp.jetty.xml
```

An application name in the startup script tells the ConfigAgent which configuration file to open. ConfigAgent is a type of JMX agent used for SESM. This is a management entity implemented in accordance with the JMX Agent Specification.

The application name is passed to ConfigAgent by the application startup scripts. The application name might also be used in other ways. For example, you can configure the parameter that defines the Jetty server log filename to incorporate the application name in the log filename.

Hardware and Software Requirements for Development

The following software must be installed and set up correctly on the development machine:

- Cisco SESM Release 3.3.1
- Java 2 SDK, Standard Edition, Version 1.4.2 or later, which is available for downloading at the Sun website.



Tip

Whenever possible, use the latest version of the Java 2 SDK as features available in the latest release might be used in future releases of SESM.

In addition to the preceding software, the other hardware and software requirements for SESM web development are the same requirements that apply to deploying a SESM Web Portal application except that no separate Java Runtime Environment (JRE) is needed because the Java 2 SDK is installed. For information on these other requirements and how to install SESM, see *Cisco Subscriber Edge Services Manager Installation Guide*.

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

Demo Installation and Development

You can install or configure SESM as a Web Portal applications to observe how the NWSP, PDA, WAP or SP Web Portal applications work. Demo installation is also useful during some phases of Web Portal application development because this installation does not require other system components such as a configured Cisco edge router and SSG. For more information on demo installation, see the [Demo Installation, page 2-12](#).

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, make sure that the environment variables shown in [Table 1-2](#) are set to the indicated locations.

Table 1-2 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:\jdk1.3.1\bin). This allows you to run the SDK executables from any directory.

Development Tools

The Cisco SESM software includes web components that were developed with:

- Dreamweaver UltraDev—a visual editor for creating and managing websites and pages.
- Fireworks 4— a web graphics design and production tool.

The current version of these products is Dreamweaver MX and Fireworks MX. Neither of these is required to develop a SESM Web Portal application. However, If you use these two state-of-the-art tools for SESM web development, you can create a customized SESM Web Portal application more quickly and easily.



Note

The Dreamweaver-related directions in this guide assume you are using the Dreamweaver 4 Workspace—a workspace layout preference you can choose in Dreamweaver MX.

Dreamweaver MX has a Live Data window feature that shows actual dynamic content that is generated by JSPs. When you are developing a Cisco SESM Web Portal application, the Live Data window allows you to make changes to the JSPs in a live-data environment.

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

For information on Dreamweaver MX and Fireworks MX, see the Macromedia website.

Both Dreamweaver MX and Fireworks MX are available at the website for a free 30-day evaluation.

Learning about SESM Web Portal Application Development

You should become familiar with the SESM components and techniques by reading this document and experimenting with the SESM Web Portal applications. [Table 1-3](#) lists some topics with which the developer should be familiar.

Table 1-3 *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”
Customizing a SESM Web Portal application’s look and feel	Changing the Look-and-Feel Elements, page 2-1
Using the web components in a sample SESM Web Portal application	Using a SESM Web Portal Application, page 2-2
Developing a SESM Web Portal application, including the steps needed to compile JSPs	Developing a SESM Web Portal Application, page 2-6
Serving SESM web pages that match the user’s shape	User Shapes and User-Shape Decoration, page 3-8 and Decorating a User Shape, page 3-18
Using advanced customization techniques, such as modifying SESM Web Portal application functionality	Chapter 3, “Advanced SESM Customization”

Table 1-3 *SESM Web Development Reading Path (continued)*

For information on this topic	Read this chapter or section
Learning about the sample SESM Web Portal applications	Chapter 4, “SESM Web Portal Applications”
Internationalizing and localizing a SESM Web Portal application	Chapter 5, “SESM Internationalization and Localization”
Configuring and using the SESM tag libraries	Appendix B, “SESM Tag Libraries”
Configuring pre-decorators and post-decorators and learning about the SESM utility servlets	Appendix C, “SESM Utility Servlets Quick Reference”
Using the Cisco Navigation Bar extension	Appendix D, “Using the Cisco Navigation Bar Extension”

Helpful Websites and Other Resources

For information on J2EE, web application development, JSP s, and other topics, the web provides many resources including the Java Developer Connection at the Sun website. [Table 1-4](#) lists some other recommended web development resources.

Table 1-4 *Web Development Resources*

Resource	Description
<i>Core Servlets and JavaServer Pages</i> (by Marty Hall, Sun Microsystems Press, 2000)	Excellent book for learning about JSPs.
<i>Java Servlet Specification Version 2.3</i> (Sun Microsystems, Inc., 2000)	Useful general information on web applications, deployment descriptor files, and other related topics. The PDF file containing the specification is at available at the Sun website.
Java servlet or JSP class library documentation	Available as Javadoc in the /doc directory at the Java 2 SDK installation location.
HTML and cascading style sheet specifications	Useful reference material is available at the World Wide Web Consortium (W3C) website.
Web application archive (WAR) file information	Available in the Web Applications section of the Java Web Services Tutorial at the Sun website.
<i>Beginning WAP, WML, and WMLScript</i> (by Wei Meng Lee, Soo Mee Foo, et al, Wrox Press Ltd., 2000)	Very good resource for learning about WAP and WML.

If you plan to use Dreamweaver MX or Fireworks MX as development tools and are not familiar with their use, those facilities have their own documentation, Help systems, and web resources. For a list of documents available on the web, go to the Dreamweaver and Fireworks support centers at the Macromedia website.



Basic SESM Customization and Development

In some Cisco SESM deployments, you can use one of the SESM Web Portal applications as a starting point and accomplish look-and-feel modifications to the JSPs without changing the preprogrammed SESM software. This chapter explains how to do basic customization of the look-and-feel elements of a SESM Web Portal application. The chapter discusses these topics:

- [Changing the Look-and-Feel Elements, page 2-1](#)
- [Using a SESM Web Portal Application, page 2-2](#)
- [Developing a SESM Web Portal Application, page 2-6](#)

The SESM software uses Java resource bundles and properties files for localization of text, labels on web-page controls, and messages. For information on localization and resource bundles, see [Chapter 5, “SESM Internationalization and Localization.”](#)

Changing the Look-and-Feel Elements

Basic SESM Web Portal application customizations involve changing the look-and-feel elements to meet the service provider’s brand requirements. The customizable JSPs 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.

To meet the service provider’s corporate standards, you can modify static markup language elements and images that appear in a template or JSP. For example, you can change the static HTML elements for text and formatting. In these elements, the developer can:

- Change the background colors and background images used in the JSPs.
- Adjust the layout of the JSPs.
- Modify or replace icons, images, and graphical elements used in the JSPs.
- Customize a style sheet.

If you are performing only basic look-and-feel customizations, you must avoid changing programmatic elements within the JSPs. Do not change directives and code in scripting elements such as JSP expressions, scriptlets, and declarations.

Customizing some of the SESM functionality that is preprogrammed into the JSPs can be accomplished by configuring the application. Other functionality changes may require modifying the code in the JSPs. For information on these types of customization, see [Chapter 3, “Advanced SESM Customization.”](#)

Using a SESM Web Portal Application

Each Cisco SESM Web Portal application, such as New World Service Provider (NWSP), includes a fully functional set of web components. Each Web Portal uses the same SESM infrastructure. This section provides an overview of the set of components in the SESM Web Portal applications and focuses on the NWSP Web Portal application.

The NWSP, PDA, WAP and SP Web Portal applications share many of the same components.

The sets of infrastructure components provided in the PDA, WAP and SP Web Portal applications are identical to the NWSP set of infrastructure components. However, because the PDA and WAP Web Portal applications are designed for specific client devices, they use JSPs that are different from the JSPs of NWSP.

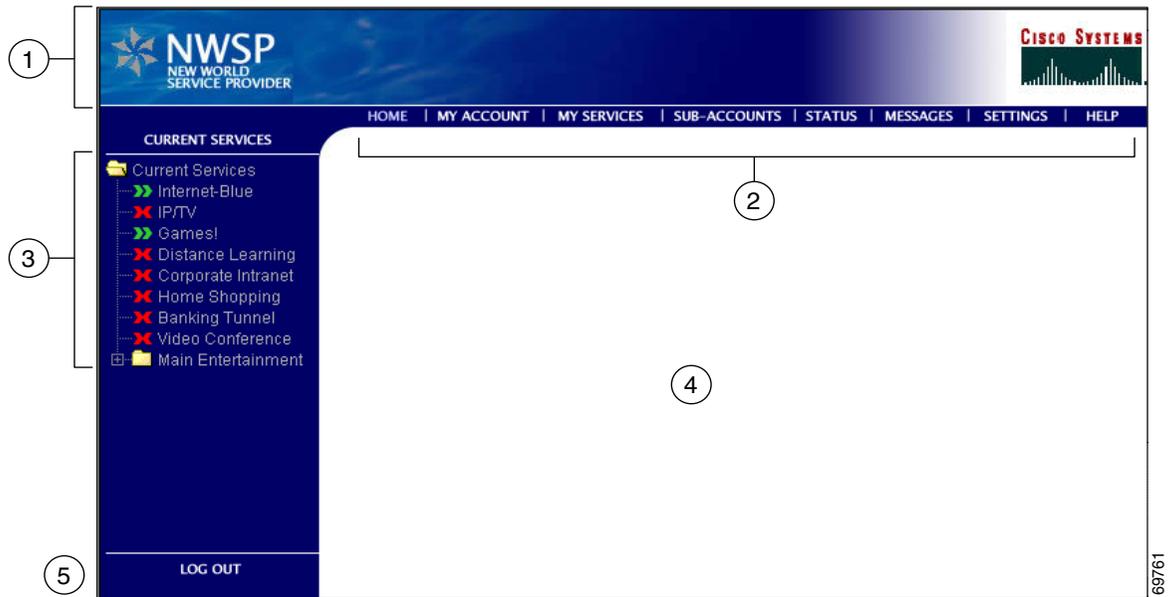
The sets of resources (Java resource bundles) used in the PDA, WAP and SP Web Portal applications are very similar to the NWSP set of resources. The deployment descriptor files used for the NWSP, PDA, WAP and SP applications are also similar.

For more detailed information on the elements that appear on the JSPs associated with a SESM Web Portal application, see [Chapter 4, “SESM Web Portal Applications.”](#)

NWSP User Interface

The NWSP user interface provides a Web Portal for network services. Subscribers use the Web Portal for subscribing to and selecting services, changing account details, creating subaccounts, defining a personal firewall, and viewing session status and messages. [Figure 2-1](#) shows the home page from the NWSP Web Portal application's user interface.

Figure 2-1 NWSP Home Page



1	Banner
2	Navigation bar
3	Service list
4	Body
5	Log Out button

The Home page and many other pages in the NWSP Web Portal application are organized into five parts:

- Banner—One or more images for product and brand identification.
- Navigation bar—A set of buttons that link to other pages where the subscriber can perform tasks such as service subscription or view information such as service status.
- Service list—A tree of icons and associated service names that the subscriber uses to select subscribed services from a tree of services and service groups.
- Body—An area for the content provided by the NWSP Web Portal application. Forms for subscriber tasks such as account management, service subscription, and subaccount creation appear in the body area. (With NWSP, network services are displayed in a new browser window.)
- Log Out button—A button to log out of the NWSP Web Portal application.

JavaServer Pages

The SESM Web Portal applications use a set of JSPs and Java servlets. To change the look and feel of a SESM Web Portal application, you need to become familiar with the elements that are present in the JSPs provided in a Web Portal application such as NWSP.

The JSPs in the NWSP Web Portal application include a complete set of customizable images, background colors, icons, buttons, a navigation bar, and style sheets. For NWSP, the Fireworks graphics design tools were used to create the icons for the service list and the buttons for the navigation bar.

Banner Images and Background Colors

In the NWSP banner (see [Figure 2-1](#)), the images and background colors are used for branding. You can replace these images with company, product, and brand logos that are appropriate for your deployment.

Icons and Buttons

For the NWSP service list and navigation bar (see [Figure 2-1](#)), the icons and buttons are provided in various formats, including GIF, JPEG, and PNG.

- The GIF and JPEG image files are incorporated into the web pages. Their small file size makes the optimized GIF and JPEG files suitable for downloading. You can also use these files to edit the images. You can use Fireworks or another graphics design tool, such as Photoshop, to customize the images and embedded text.
- The PNG files are used when the web designer edits the images. Because PNG format is the native Fireworks file format, you can use Fireworks to customize the images and embedded text. PNG files can also be read by other graphics applications, such as Photoshop.

Buttons, such as the Log Out button and Cancel button, that do not appear in the NWSP navigation bar are defined once in a single JSP (for example, `cancelButton.jsp`) or in a Dreamweaver template. You can customize each button in the one location where it is defined and have the change take effect in all JSPs that use the button.

Navigation Bar

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

When modifying a Dreamweaver navigation bar, you can use the existing NWSP buttons or create a new set of buttons. You can use Fireworks or another graphics design tool to change the text on the existing NWSP buttons. For more information on the navigation bar, see the [JSP for the Navigation Bar, page 4-17](#).

Style Sheets

The NWSP Web Portal application uses cascading style sheets for its presentation elements. The style sheets define classes for the textual elements that are used on the JSPs. The style sheets enable designers to change fonts, margins, colors, and other aspects of style that control the layout and design of the NWSP web pages. The style sheets used by NWSP are located in the `\install_dir\nwsp\webapp\styles` directory. In other SESM Web Portal applications such as PDA, there may be multiple brand-specific style sheets in different directories from which the SESM software dynamically selects one style sheet based on the subscriber's brand.

Dreamweaver Templates

The NWSP Web Portal application includes two Dreamweaver templates. These templates have the suffix *.dwt* and reside in the */nwsp/webapp/templates* directory. Dreamweaver templates can be very useful for customizing or maintaining a web application's JSPs when many pages have the same layout. By modifying a template and then updating the JSPs that use the template, you can change the look and feel of an entire set of pages very quickly.

When a JSP is derived from a template, the JSP has locked regions that cannot be edited and regions that can be edited. The purpose of locked regions is that if changes are required on a locked region, the changes are made in the template file. After you make the changes 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 website using the Dreamweaver **update** commands in the Modify > Templates menu.

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

If changes are required to individual JSP (as opposed to in a Dreamweaver template), the part of the individual JSP that you can modify appears between `BeginEditable` and `EndEditable` comments. For example, the main area that can be edited in the `mainTemplate.dwt` is empty, allowing each JSP to provide content that is appropriate for its purpose:

```
!-- #BeginEditable "main" -->
 
<!-- #EndEditable -->
```

For more information on templates, see the [NWSP Templates, page 4-19](#) and the Dreamweaver MX documentation.

Customizing Attribute 18 Messages

NWSP supports the display of RADIUS attribute 18 messages (also known as RADIUS Reply Message). An Attribute 18 message could be returned as a response to an access request. In other words, any Access-Accept, Access-Reject, or Access-Challenge response that the AAA server sends to the client, may include an attribute 18 message.

NWSP extracts the attribute 18 messages from the response and displays them to the user in the appropriate pages. Messages extracted from access reject responses are displayed on the logon page while messages extracted from access accept responses are displayed in the messages page.

The messages to be displayed can be customized using the message catalog. The message catalog is implemented as a Java resource bundle that provides internationalization and localization support. The catalog configuration within the `messages.properties` text file is held in a `<key>=<value>` fashion.

SESM provides a message catalog that has entries of attribute 18 messages as keys and associated messages as values.

NWSP retrieves the attribute 18 message and does one of two things:

- If the retrieved attribute 18 message matches a key in the catalog, NWSP displays the value that corresponds to this key.
- If the catalog file is not available or if the retrieved attribute 18 message does not match a key in the catalog, NWSP displays the attribute 18 message as is.

SESM is shipped with a default catalog file in the English language that contains only two attribute 18 related messages for the failed and successful logins, as shown below:

```
messages.properties
...
#Messages within attr 18
attr18.login.success=You have successfully logged on. Welcome to the NWSP.
attr18.login.failure=This is a message from the RADIUS server about why you failed to
login.
...
```

Deployers can add, remove, or edit these messages. The instructions provided in the catalog file as comments must be followed.

Developers might want to change the default language by:

- Rewriting the original messages.
- Support multiple languages and the local language by creating copies of the file with the appropriate suffix, such as `messages_de.properties` for German and `messages_fr.properties` for French.

For a comprehensive description of how to support multiple languages and locales see [Chapter 5, “SESM Internationalization and Localization”](#) and the explanation at the beginning of the `messages.properties` file in `sesm-base-dir/nwsp/webapp/WEB-INF/classes` directory.

Developers should be aware of the possible values the AAA server can return within attribute 18, and ensure that the `messages.properties` file is updated with the messages appropriate for these values.

Customization for iPass Logon

SESM provides an integration with iPass (For more information about iPass, see the iPass website) to allow iPass users to connect to the Internet. A subscriber can log on using iPass clients as well as the NWSP login page. To log on from the NWSP login page, the subscriber types in his username with the iPass prefix: `IPASS/username@domain`. However, the NWSP login page can be customized to concatenate the `IPASS/` prefix to the username so that the subscriber need not type in the `IPASS/` prefix.

There are several methods to customize the NWSP login page. One method is to add a check box to mark the authentication as an iPass authentication. If the check box is selected the `IPASS/` prefix will be concatenated to the username (using JavaScript, for example)

Developing a SESM Web Portal Application

Depending on the service provider’s business requirements for a SESM Web Portal application, the steps required to develop the application may vary. These steps are described in the following sections:

- [Defining the Business Requirements, page 2-7](#)
- [Designing and Implementing a SESM Web Portal Application, page 2-7](#)
- [Debugging a SESM Web Portal Application, page 2-17](#)
- [Managing a SESM Website, page 2-22](#)

**Tip**

One error that developers frequently commit when developing a SESM Web Portal application is that they do not perform the steps required *to compile* the JSPs after they have modified them. The SESM Web Portal applications are installed with *precompiled JSPs*. After changing the JSPs, you must take a few simple steps to recompile them successfully. Two options are available for compiling a modified set of JSPs. The first option is to use the JSPs directly, in which case a page is compiled by the web server the first time it is requested (on the fly) after the Web Portal application is started. For information on compiling JSPs on the fly, see [JSP Compilation for Web Portal Application Development \(On the Fly Compilation\)](#), page 2-10. The second option is to precompile the modified JSPs using the UNIX script `precompile.sh` and the instructions in the [Running the Precompiling Script](#), page 2-12.

Defining the Business Requirements

Defining the business requirements for a specific service provider's SESM Web Portal application can be organized around the following questions:

- What functionality is required?
- What look-and-feel elements (icons, images, background colors, and style sheets) must be customized?
- Will SESM web pages be rendered to match one or more subscriber characteristics, such as device or brand?
- What types of localization (if any) are required?
 - If only English-language subscribers will be using a SESM Web Portal application, the base set of components in a SESM Web Portal application might not require localization.
 - If subscribers use a single language other than English, a single website 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 might be required.
- Do any application-specific messages need to be internationalized and localized?

Designing and Implementing a SESM Web Portal Application

The SESM software includes a set of precompiled JSPs for the SESM Web Portal applications such as NWSP. In any production deployment, the default JSPs require customization by the deployer. Two options are available for compiling a modified set of JSPs.

- The first option is to use the JSPs directly, in which case a page is compiled by the web server the first time it is requested after the Web Portal application is started. This option is convenient—especially for development—but it has two disadvantages:
 - The first time a page is requested, the access time is slow. Subsequent requests are processed faster because the compiled JSP is stored.
 - The web server requires the presence of an installed JDK. This is not convenient for deployment.

This option is not recommended in production for the following reasons:

- The delay because of pages being compiled on-the-fly for the first time.
- The risk of JSPs being accidentally or deliberately altered.
- The inconvenience of having to install a JDK.



Note When compiling on the fly, if you make any changes to web.xml, you must stop and start the web server.

See [JSP Compilation for Web Portal Application Development \(On the Fly Compilation\)](#), page 2-10

- The second option is to precompile the modified JSPs using the UNIX script precompile.sh and the instructions in the “Running the Precompiling Script” section on page 2-12. The script is shipped with the SESM software in the install_dir/tools/bin directory. With precompilation, the JSPs are translated into compiled Java servlet classes, and there is no significant performance impact when a page is requested the first time.

The recommended approach is to use compilation on-the-fly for development. After development is complete, and the application is ready, precompile jsp, and install on the production server.

See [Compiling JavaServer Pages for Web Portal Applications in Production](#), page 2-12



Tip

Changes to the SESM Web Portal application deployment descriptor file (web.xml) should be carefully tracked, for example, with a change history section in the file itself. For some situations, such as migration to an updated version of the SESM software, you might need to identify and reproduce any changes you have made within the web.xml file.

SESM Class Libraries and Tag Library Descriptor Files

To successfully compile the JSPs for a SESM Web Portal application, the Java compiler must be able to find the needed SESM-related class libraries and tag library descriptor (TLD) files.

The Cisco SESM software provides several specialized Java class libraries that encapsulate the functionality required in a SESM Web Portal application. The SESM class libraries are distributed in the JAR (Java archive) files listed in [Table 2-1](#).

Table 2-1 *JAR Files for a SESM Web Portal Application*

JAR File	Description
com.cisco.sesm.appmgmt.remotemgmt.jar	Classes for remote management of SESM applications.
com.cisco.sesm.erp.jar	Classes for the Extensible Request Proxy framework, the foundation of the RADIUS Data Proxy (RDP).
com.cisco.sesm.jmx.jar	Classes for the SESM extensions to the Java Management Extensions (JMX) tools.
com.cisco.sesm.i18n110n.jar	Classes for internationalization and localization.
com.cisco.sesm.logging.jar	Classes for the SESM logging utilities.
com.cisco.sesm.model.jar	Classes for the SESM core model and associated functionality.

Table 2-1 JAR Files for a SESM Web Portal Application (continued)

JAR File	Description
com.cisco.sesm.platform.jar	Classes for the platform framework for extensions.
com.cisco.sesm.radius.jar	Classes for the RADIUS-related functionality.
com.cisco.sesm.types.jar	Classes for some SESM types.
com.cisco.sesm.util.jar	Classes for the SESM utilities.
com.cisco.sesm.webapps.jar	Classes for the SESM decorators and controllers, and tag libraries.
jsp.jar	Classes for the SESM precompiled JSPs.
dess.jar auth.jar authentication.jar gsal.jar protect.jar jakarta-regexp1.2.jar log4j-1.2.6.jar	Classes for using Security Policy Engine (SPE). These files are needed only for SESM Web Portal applications that will be deployed in SESM SPE installation.

With two exceptions, the SESM-related JAR files reside in the *install_dir/web_app_name/webapp/WEB-INF/lib* directory, where *install_dir* is the directory in which the SESM software is installed, and *web_app_name* is a directory in which a SESM Web Portal application, such as NWSP, is installed. The two exceptions are:

- com.cisco.sesm.erp.jar resides in the *install_dir/libs/erp/lib* directory.
- com.cisco.sesm.jmx.jar resides in the *install_dir/libs/jmx/lib* directory.

In addition, there are three non-SESM-related JAR files in the following locations:

- javax.servlet.jar resides in the *install_dir/jetty/lib* directory.
- org.apache.jasper.jar resides in the *install_dir/jetty/lib* directory.
- crimson.jar resides in the *install_dir/redis/jaxp/lib* directory.

To compile the class for a SESM Web Portal software component, the `CLASSPATH` environment variable must be set to the needed directory path (for example, *install_dir/web_app_name/webapp/WEB-INF/lib* to tell the Java compiler the location of the SESM class libraries).

The Cisco SESM software also includes a set of TLD files for the SESM tag libraries. Each TLD file is an XML file describing a tag library. The TLD files reside in the *install_dir/web_app_name/webapp/WEB-INF* directory and are as follows:

- iterator.tld
- localization.tld
- navigator.tld
- shape.tld

For more information on the TLD files and using a tag library, see the [Configuring a Tag Library, page B-1](#).

Javadoc Documentation

The Cisco SESM software includes a full set of online documentation in Javadoc format. The Javadoc documentation includes information on all Java classes that implement the SESM software. If you are customizing the JSPs of a SESM Web Portal application, the Java class descriptions for control servlets, decorator servlets, JavaBeans, and tag libraries are of particular interest. You can access the Javadoc by opening the file `\install_dir\docs\apidoc\index.html` in a browser.

JSP Compilation for Web Portal Application Development (On the Fly Compilation)

If the SESM server has the required Java Runtime Environment (JRE), precompiled JSPs must be used to run Web Portal applications like NWSP. However, JSPs can be compiled on the fly if the Java2 SDK is installed on the SESM server.

The JSPs in each SESM Web Portal application are precompiled and the resulting servlet classes are installed in the `\install_dir\web_app_name\webapp\WEB-INF\lib\jsp.jar` file. `install_dir` is the directory where the SESM software is installed, and `web_app_name` is a directory where a SESM Web Portal application, such as NWSP, is installed.

Because the JSPs are precompiled, a Java 2 SDK is not required to run a SESM Web Portal application. *However, to perform development on the JSPs, you must install the Java 2 SDK on the development machine.* The Java 2 SDK is necessary for recompiling changed JSPs. For information on the required Java 2 SDK, see the [Customizing a SESM Web Portal Application: An Overview, page 1-7](#).



Tip

On machines that are used for SESM Web Portal application development, we recommend that you install the Java 2 SDK before you install the SESM software. In that way, the SESM installation program uses the Java 2 SDK in the SESM Web Portal application startup scripts, rather than a JRE.

To check whether a Java 2 SDK is installed on the development machine, look for a `tools.jar` file in the `JDK_install_dir/lib` directory, where `JDK_install_dir` is the location where you think the Java 2 SDK is located. If the `tools.jar` file is present, a Java 2 SDK is installed.

If you install the Java 2 SDK *before* installing SESM, read the [Recompiling and the web.xml File \(Compiling on the Fly\), page 2-11](#).

If you need to install the Java 2 SDK *after* installing SESM, read the [Installing a Java 2 SDK After Installing SESM, page 2-10](#) and the [Recompiling and the web.xml File \(Compiling on the Fly\), page 2-11](#).

Installing a Java 2 SDK After Installing SESM

If you install the Java 2 SDK *after* installing SESM, you must do the following:

- Ensure that the `JDK_HOME` environment variable points to the directory where you installed the Java 2 SDK.
- Edit the SESM application startup script to use the Java 2 SDK.

The SESM application startup scripts are named `start.cmd` on Windows-based installations and `start.sh` on UNIX-based installations. The script is located in the `\install_dir\jetty\bin` directory.

In the startup script on a Windows-based installation, change the line that specifies the Java 2 SDK location (JDK_HOME) to point to the correct directory, and uncomment the two lines that check for a Java compiler (JAVAC). For example:

```
rem find some java or other
set JDK_HOME=D:\jdk1.3

set JAVA=%JDK_HOME%\bin\java.exe
set JAVAC=%JDK_HOME%\bin\javac.exe

rem Note that we only need this if we are developing, so for now
rem we can comment the check out. Developers should remove these comments
rem if exist "%JAVAC%" goto gotjdkhome
rem echo JDK_HOME does not point to a valid JDK. JSPs will not be compiled.
```

In the startup script on a UNIX-based installation, change the line that specifies the Java 2 SDK location (JDK_HOME) to point to the correct directory, and uncomment the four lines that check for a Java compiler (JAVAC). For example:

```
# Check we can find a suitable version of the JDK
JDK_HOME=/usr/java1.2

JAVAC=$JDK_HOME/bin/$JAVACEXE
JAVA=$JDK_HOME/bin/$JAVACEXE

# Note that we only need this if we are developing, so for now
# we can comment the check out. Developers should remove these comments
if [ ! -x $JAVAC ]
then
    echo WARNING: JDK_HOME does not point to a valid JDK. JSPs can not be compiled.
fi
```

Recompiling and the web.xml File (Compiling on the Fly)

The web.xml file in `\install_dir\web_app_name\webapp\WEB-INF` is the SESM Web Portal application deployment descriptor file. This file defines how the Jetty web server finds the JSPs. The default web.xml file for each SESM Web Portal application specifies that the web server uses the precompiled JSPs. To compile modified JSPs, use the web.recompile.xml file in place of the default web.xml file.



Note

Until you perform the following steps, changing the JSPs in `install_dir/web_app_name/webapp` has no effect on the HTML pages that the NWSP Web Portal application displays.

To recompile modified JSPs in the NWSP Web Portal application:

-
- Step 1** Stop the web server.
 - Step 2** In the `\install_dir\nwsp\webapp\WEB-INF` directory, rename the web.xml to web.xml.bak.
 - Step 3** In the `\install_dir\nwsp\webapp\WEB-INF` directory, rename web.recompile.xml to web.xml.
 - Step 4** Restart the web server.
-

Compiling JavaServer Pages for Web Portal Applications in Production

The `precompile.sh` script precompiles a full set of JSPs for the SESM Web Portal application (for example, NWSP) that you specify when you invoke the script and creates a JAR file containing the resulting compiled servlet classes. The script also makes adjustments to the SESM Web Portal application's `web.xml` file so that the Web Portal application uses the precompiled JSPs.

Running the Precompiling Script

To execute the `precompile.sh` script (located in `install_dir\tools\bin`) for precompiling a set of JSPs, perform the following steps on a UNIX workstation where the SESM software is installed:

Step 1 To make the script executable, issue the following command:

```
chmod a+x precompile.sh
```

Step 2 Run the script `precompile.sh` and wait for completion, which may take some time.



Note The comments at the beginning of the `precompile.sh` script provide more information on how to use the script.

If you are precompiling the JSPs for a SESM Web Portal application other than NWSP, you must supply an argument naming the application when you run the `precompile.sh` script. For information on specifying this argument, see the comments at the beginning of the script.

You can run the script from any directory because the paths used in the script are all full pathnames. If you run the script from the recommended directory, set the environment variable `SESM_HOME` to be the full pathname of the SESM installation directory.

Even when precompiled JSPs are used, the shape mechanism is checking which non-compiled JSPs exists in order to decide which page to use. Please do not delete the files under the `<webapp_name>` (for example NWSP) directory even when using precompiled JSPs.



Note This script cannot be converted to run on Windows, as JspC does not currently work on Wintel.

Demo Installation

You can install or configure the SESM software as a Web Portal application to observe how the NWSP Web Portal application works. In a Demo Installation, the NWSP Web Portal application can demonstrate most SESM functionality. For information on installing or configuring for a Demo Installation, see *Cisco Subscriber Edge Services Manager Installation Guide* and *Cisco Subscriber Edge Services Manager Web Portal Guide*.



Note The PDA and WAP Web Portal applications also work in Demo installation. However, because the PDA and WAP applications demonstrate only the subset of SESM functionality that is appropriate for their targeted devices, this description of Demo installation focuses on the NWSP Web Portal application.

Demo installation is also useful during some phases of Web Portal application development because this installation does not require other system components, such as a configured SSG.

- If the web designer modifies the look and feel of the NWSP Web Portal application, the changes can be viewed in Demo installation to observe the results.
- If the web developer changes programmatic pieces of a JSP, many changes in Web Portal application behavior can be initially tested in Demo installation to verify the results.

For the NWSP Web Portal application, Demo installation uses a Merit RADIUS file for subscriber, service, and service group information. By default, the Merit RADIUS file is named `aaa.properties` and resides in the `\install_dir\nwsp\config` directory. A Merit RADIUS file is an ASCII file that you can modify using a text editor. Using Demo installation, you can observe how changes to a subscriber, service, or service group profiles affect the NWSP Web Portal application.



Tip

If you make changes to the `aaa.properties` file, you will not be able to observe the changes until the web server is stopped and restarted.

Captive Portal and Demo Installation

You cannot observe the full set of behaviors of the Captive Portal solution in a Demo installation because a configured SSG and its TCP Redirect feature are required to make the solution work. For this reason, the Captive Portal software is not installed in a Demo installation of SESM.

To run Captive Portal in a Demo installation, you must install the Captive Portal software in a SESM RADIUS or SESM SPE installation, and then start the Captive Portal using the **-mode** option. For example:

```
startCAPTIVEPORTAL.sh -mode Demo
```

With Captive Portal running in a Demo installation, you can use simulated HTTP requests to view the behavior of the Captive Portal servlet and the content web applications. For example, you could send a simulated HTTP request to the message portal servlet (`MessagePortalServlet`) if you included the required query-string parameters (such as `CPURL` and `CPDURATION`) in the request. For information on the Captive Portal solution and the query-string parameters, see the [Captive Portal Web Application, page 4-28](#).

SESM RADIUS Installation Functionality

To simulate RADIUS installation functionality (such as service selection), the subscriber, service, and service-group profiles in `aaa.properties` use the standard RADIUS attributes and vendor-specific RADIUS attributes (VSAs). The attributes are described in *Cisco Subscriber Edge Services Manager Deployment Guide*. For example, the following profile from `aaa.properties` 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"
```

SESM SPE Installation Functionality

To simulate SPE-installation functionality (such as subscriber self-subscription, account management, subaccount creation, and personal firewall), the allowed subscriber profile attributes have been extended so that the NWSP Web Portal application can demonstrate SESM SPE installation 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 Portal application are read-only. However, NWSP as a Demo installation does correctly simulate SESM SPE installation functionality. For example, in a Demo installation, 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 Portal application does not modify attributes in the Merit RADIUS file.

**Note**

The extended set of subscriber profile attributes are allowed for Demo installation only. The extended set of attributes are not valid vendor-specific attributes (VSAs). They are not valid in RADIUS installation or SPE installation and are not recognized by SSG. They should not be added to the RADIUS dictionary.

In NWSP, the `aaa.properties` file contains subscriber profiles for simulating SPE installation, including `golduser`, `silveruser`, and `bronzeuser`. These subscriber profiles provide examples of the attribute extensions. For detailed information on the extended set of attributes for demonstrating SPE features, see *Cisco Subscriber Edge Services Manager Administration and Configuration Guide*.

Dreamweaver MX Live Data Window

When you are developing a Cisco SESM Web Portal application, you can use the Live Data window feature of Dreamweaver MX to make changes to your JSP s in a live data environment. Unlike the Document window, which uses placeholders for dynamic content on a JSP, the Live Data window shows the actual dynamic content that is generated by the JSP.

[Figure 2-2](#) shows how `myAccount.jsp` is displayed in the Live Data window. To display dynamic data, Dreamweaver runs the JSP on the web server before displaying it in the Live Data window. In [Figure 2-2](#), notice that the Live Data window displays services in the service list and any subscriber data (for example, first name and last name) in the subscriber profile.

Figure 2-2 Live Data Window

The screenshot shows a web browser window with the NWSP logo and navigation menu. The main content area displays a 'My Account Details' form. The form fields are as follows:

First Name	Felicity	Middle Initial		Last Name	Hopkins
Street		City		Country	
		Postal Code		State	
Home Phone		Mobile Phone		Pager	
Fax		Email		Home URL	
Date of Birth		(Pattern is "ddMMyy", for example "14/01/02")			
Gender	<input type="radio"/> Male <input checked="" type="radio"/> Female		Single Sign-On <input checked="" type="radio"/> yes <input type="radio"/> no		
Interests	<input type="checkbox"/> Cinema <input checked="" type="checkbox"/> Science <input type="checkbox"/> Internet <input checked="" type="checkbox"/> News <input type="checkbox"/> Sports <input checked="" type="checkbox"/> Travel <input type="checkbox"/> Finance <input type="checkbox"/> Community				

Buttons: OK, Cancel, Reset, Change Password

**Note**

Editing in a live data environment is available starting with Dreamweaver UltraDev 4 (the predecessor of Dreamweaver MX).

To configure and use the Live Data window for the NWSP Web Portal application, perform the following steps. The procedure assumes that the web server is running on the local machine and listening for requests for the NWSP Web Portal application on port 8080.

- Step 1** Start Dreamweaver MX.
- Step 2** In the **Site** menu, click **New Site**.
The Advanced Tab of the Site Definition dialog box is displayed.
- Step 3** In the Category list, click **Local Info** and specify the following:
Site Name: NWSP
Local Root Folder: *C:\SESM_location\nwsp\webapp* (In this procedure, *C:\SESM_location* is the location where SESM is installed.)
HTTP Address: *http://localhost:8080*
- Step 4** In the Category list, click **Remote Info** and specify the following:
Access: Local/Network
Remote Folder: *C:\SESM_location\nwsp\webapp*
- Step 5** In the Category list, click **Testing Server** and specify the following:
Server Model: JSP
Access: Local/Network
Testing Server Folder: *C:\SESM_location\nwsp\webapp* (for example)
URL Prefix: *http://localhost:8080/user=golduser/device=pc/locale=en/shape/*

In URL Prefix, the values given after the port number are test decorator values that use URL-to-servlet mapping to specify the user, device, and locale. For information on using test values, see the [Using Test Decorators, page 2-18](#).



Note To use the test decorators specified in the URL Prefix, you must modify the SESM Web Portal application's web.xml file by adding some additional servlet declarations and servlet mappings from the web.dev.xml file. The web.dev.xml file is located in the `\install_dir\nwsp\config` directory. Given the preceding URL prefix, the declarations and mappings for `/user=golduser` and `/device=pc` must be added to the web.xml file. For information on using the test decorators, see the [Using Test Decorators, page 2-18](#).

Step 6 Click **OK**.

Step 7 To use the Live Data window to view a JSP:

- a. Double-click the name of the JSP.
- b. In the Document window that displays the JSP, from the **View** menu, click **Live Data**.

Dreamweaver MX displays the actual dynamic content that is generated by the JSP. In the Live Data window, you can edit the page's layout and JSP content. For information on editing in a live see the Dreamweaver MX documentation.



Note If you use the Dreamweaver Live Window feature, the NWSP Web Portal application's home page (home.jsp) does not work correctly.

To use the Live Data window with home.jsp, comment out the following statement in home.jsp:

```
<%@ include file="/decorators/openWindow.jspf"%>
```

The commented-out statement is as follows:

```
<%-- @ include file="/decorators/openWindow.jspf" --%>
```

Before you test and deploy the Web Portal application, remove the comment delimiters.

Double-Byte Language Packs

Some double-byte languages that are supported by the NWSP Web Portal application (for example, Chinese and Japanese) require that the client PC have the needed language packs installed so that the language characters can be displayed. The language pack that you require depends on the browser that you are using:

- Internet Explorer - To install a language pack for a double-byte language, you access the Microsoft website for the country of the language. The language pack is downloaded and installed automatically. You can use the country specific webpages at the Microsoft website.
- Netscape Navigator - You can get information on the available language/region packs and international browsers from the Netscape website.

General Web Development Considerations

Some general web development considerations for a Cisco SESM Web Portal application are:

- We recommend the technique in which the Web Portal application redirects a `POST` request to a `GET` request so that `POST` requests cannot be requested with the browser history after they are sent. Redirecting to a `GET` removes the `POST` request from the HTTP client's history list. Most controls (for example, `MyAccountControl`) in the SESM Web Portal applications use this technique.
- The decoration JSPs 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 Portal application, all references to a web resource from a web page must be relative.

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

Debugging a SESM Web Portal Application

If a SESM Web Portal 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 a SESM Web Portal application, see *Cisco Subscriber Edge Services Manager Administration and Configuration Guide*.

The following hints may help with debugging a SESM Web Portal application:

- Most web servers can be set up so that the servlet code generated from JSPs is saved and available for examination. With the Jetty web server, when the JSPs are compiled, the servlet code is put in a temporary directory that is determined by the machine's Java Virtual Machine (JVM). Normally, the temporary directory is `/tmp` or `/usr/tmp` on UNIX, and `C:\TEMP` on Windows. 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.
- If you are using the Internet Explorer browser, when an error occurs in the dynamic portion of a JSP, the browser displays an alternative "friendly" (and less helpful) HTTP error message if you check the Show friendly HTTP error messages box. To view server-generated error messages, make sure this box is not checked in the Advanced Tab for Internet Options, which you access through the Tools menu.
- Some SESM Web Portal application files such as `web.xml` and the properties files for resource bundles are read once when the Web Portal application is started.



Caution

Changes to these files have no effect until the Web Portal application is stopped and restarted.

Using Test Decorators

SESM includes a number of Java servlets called test decorators that are very useful for development and testing of a SESM Web Portal application. The test decorators allow the SESM developer to set specific values for

- Username and password.
- Locale language, country, and variant.
- Other dimensions of the user shape such as brand and device.

The `web.dev.xml` file has declarations for a number of test decorators preconfigured with some common values. For NWSP, the `web.dev.xml` file is located in the `\install_dir\nwsp\config` directory. You can add to or modify the test values in `web.dev.xml` to meet your development and testing needs. For information on user-shape decoration and decorators, see the [SESM Software Concepts, page 3-8](#).

To use the test decorators in `web.dev.xml`, perform the following steps:

-
- Step 1** Add the required servlet declarations and servlet mappings from the `web.dev.xml` file to the appropriate locations in the `web.xml` file used by the SESM Web Portal application.
- a. Add the servlet declarations from `web.dev.xml` immediately after the existing servlet declarations in `web.xml`.
 - b. Add the servlet mappings from `web.dev.xml` immediately after the existing servlet mappings in `web.xml`.
- Step 2** Modify the servlet declarations from the `web.dev.xml` to have the same order of XML tags (for example, the `<load-on-startup>` tag) as is found in `web.xml`.
- Step 3** Save the modified `web.xml` file.
- Step 4** Restart the web server.
-

Each test decorator has an associated URL-to-servlet mapping that allows you to specify the test values in the URL when requesting a SESM web page. When the URL for a Web Portal application resource is preceded by URL-mapped test decorators, the corresponding test decorator servlets are invoked. Consider the following URL requesting `myAccount`—a control servlet for the My Account page:

```
http://localhost:8080/user=golduser/device=pda/locale=de/myAccount
```

When the preceding URL is used in demonstration installation, test decorator servlets are invoked and do the following:

- Authenticate the username and password for `golduser`.
- Set the device dimension of the user shape to `pda`.
- Set the locale dimension of the user shape to `de` (Germany).

The following sections provide information on the test users, locales, and device and brand dimensions that are preconfigured in the `web.xml` file. For information on how you can modify and add to the set of test values and for information on the test decorator servlets, see [Appendix C, “SESM Utility Servlets Quick Reference.”](#)

Test Users

Table 2-2 lists some of the test users, passwords, and URL mappings that are preconfigured in the web.dev.xml file. For each URL mapping, the TestUserDecorator servlet sets the username and password to the values shown in the table. TestUserDecorator then automatically attempts to authenticate with the username and password. In Demo installation, SESM authenticates against the subscriber profiles defined in the Merit RADIUS file aaa.properties. The aaa.properties file resides in the /config directory of the SESM Web Portal application.

Table 2-2 Test Users: Names and Password

Username and Password	URL Mapping
Username: user1 Password: cisco	/user=user1/*
Username: user2 Password: cisco	/user=user2/*
Username: user31 Password: cisco	/user=user31/*
Username: user42 Password: cisco	/user=user42/*
Username: user43 Password: cisco	/user=user43/*
Username: user44 Password: cisco	/user=user44/*
Username: golduser Password: cisco	/user=golduser/*
Username: subgolduser Password: cisco	/user=subgolduser/*

Test Locales

Table 2-3 lists some of the test locales, associated values, and URL mappings that are preconfigured in the web.dev.xml file. For each URL mapping, the LocaleDecorator servlet sets the locale dimension to the value shown in the table.

Table 2-3 Test Locales

Locale	Value	URL Mapping
Australia	au	/locale=au/*
France	fr	/locale=fr/*
Germany	de	/locale=de/*
Great Britain	en	/locale=en/*
	gb	/locale=gb/*
	uk	/locale=uk/*
Italy	it	/locale=it/*
Portugal	pt	/locale=pt/*
Spain	es	/locale=es/*

Table 2-3 Test Locales (continued)

Locale	Value	URL Mapping
Sweden	se	/locale=se/*
USA	us	/locale=us/*

Test Devices and Brands

Table 2-4 lists some of the test devices, brands, associated values, and URL mappings that are preconfigured in the web.dev.xml file. For each URL mapping, the `TestDimensionDecorator` servlet sets the `device` or `brand` dimension to the value shown in the table.

Table 2-4 Test Devices and Brands

Device or Brand	Value	URL Mapping
WAP device	wap	/device=wap/*
PDA device	pda	/device=pda/*
PC device	pc	/device=pc/*
XML device	xml	/device=xml/*
NWSP brand	nwsp	/brand=nwsp/*
Unstyled brand	unstyled	/brand=unstyled/*
Gold brand	gold	/brand=gold/*
Silver brand	silver	/brand=silver/*
Bronze brand	bronze	/brand=bronze/*
No brand	an empty string	/brand=/*

Using Device Simulators for WAP and WML

WAP phone simulators are very useful for development and testing a SESM Web Portal application that generates WAP content using Wireless Markup Language (WML). The WAP Web Portal application is designed for providing content to a subscriber who is using a WAP phone. WAP phone simulators enable you to view WAP content by using the HTTP protocol to access the JSPs of a SESM Web Portal application.

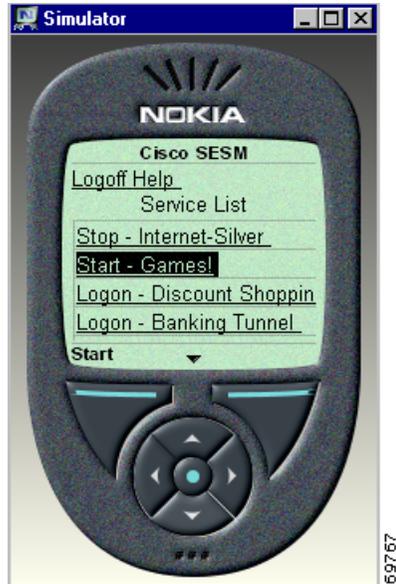


Note

The WAP Web Portal application uses WML version 1.2. You can upgrade to WML 2.0.

Figure 2-3 shows the home page and service list of the WAP Web Portal application as it appears on a Nokia simulator.

Figure 2-3 WAP Phone Simulator



When you use the HTTP protocol mode to access a SESM Web Portal application, you request pages from a SESM Web Portal application in the usual manner. For example, to log on to SESM, you specify the following URL: `http://web_server:8080`, where `web_server` is the IP address of the Jetty web server.

The NWSP and WAP Web Portal applications recognize many WAP phone simulators, including the Nokia Mobile Internet Toolkit simulators, the Openwave UP.Simulator (Version 4.0 and 4.1), and the WinWAP Pro simulator. The NWSP and WAP Web Portal applications set the `device` dimension of the user shape to `wap` when these simulators are the client device.

Nokia Mobile Internet Toolkit simulators work well with the SESM WAP Web Portal application. The toolkit contains two simulators and is available at the Nokia website.

Using Device Simulators for Pocket PCs

The Microsoft website for mobile-device application development allows you to download the Pocket PC 2002 Software Development Kit. This SDK is free and includes a Pocket PC emulator. The mobile devices site is located at:

<http://www.microsoft.com/mobile/developer/default.asp>

The Pocket PC emulator requires Microsoft eMbedded Visual Tools 3.0 in order to run. eMbedded Visual Tools 3.0 is also free and is available at the following location:

<http://www.microsoft.com/mobile/downloads/emvt30.asp>

Managing a SESM Website

Dreamweaver has a number of features that could be useful for developing a SESM Web Portal application. For example, if a Dreamweaver template is modified, you can automatically update all files in a website that use the template. To use some Dreamweaver features, you must define a website in Dreamweaver.

To define a SESM Web Portal application as a site, do the following:

-
- Step 1** From the **Dreamweaver Site** menu, choose **New Site**.
 - Step 2** In the **Local Info** dialog box, specify the `/web_app_name/webapp` directory of the SESM Web Portal application as the Local Root Folder.
 - Step 3** In the **Site Map Layout** dialog box, specify `home.jsp` as the Home Page.

After the site is defined in this manner, Dreamweaver creates a useful Site Files view. However, because `home.jsp` is designed to contain few links to other JSPs, the Site Map view is not very useful.

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

The Dreamweaver Preview in Browser feature is of limited use with a SESM Web Portal application. For example, in the NWSP Web Portal application, most JSPs require some form of input. Without the input, the web browser displays an error page. If you are using Dreamweaver MX, the Live Data window feature is an excellent alternative to the Preview in Browser feature. The Live Data window allows you to view dynamic data in a SESM Web Portal application. For information on configuring the Live Data window, see the [Dreamweaver MX Live Data Window, page 2-14](#).



Advanced SESM Customization

This chapter explains some of the advanced customization techniques that you can use with a Cisco SESM Web Portal application. The chapter contains the following:

- [Introduction, page 3-1](#)
- [SESM Architecture: An Overview, page 3-2](#)
- [SESM Software Concepts, page 3-8](#)
- [Using a Sparse-Tree Directory Structure, page 3-10](#)
- [Decorating a User Shape, page 3-18](#)
- [Modifying SESM Web Portal Application Functionality, page 3-29](#)
- [Invoking a Decorator, page 3-40](#)

Introduction

In some SESM deployments, it might be possible for you to use one of the Web Portal applications as a starting point and accomplish look-and-feel modifications to the JSPs without changing the SESM Web Portal application's preprogrammed functionality. The J2EE deployment descriptor file (web.xml) for the Cisco SESM Web Portal application allows the deployer to configure some of the Web Portal application functionality without coding changes to the JSPs and without adding to the JSPs.

Other SESM deployments will require some JSP-page coding changes or additions to meet the requirements of a specific deployment. You can accomplish some of the advanced customizations by modifying the web.xml for the SESM Web Portal application. You accomplish other advanced customizations by modifying existing JSPs or creating new ones. No Java servlet programming is needed.

The advanced SESM customization techniques fall into two general categories:

- **Decorating a user shape**—The user-shape decoration mechanisms enable you to create a SESM Web Portal application that renders the web-portal user interface based on subscriber characteristics, such as the device, brand, and locale. These sections describe the mechanisms that provide for decorating the user shape:
 - [User Shapes and User-Shape Decoration, page 3-8](#)
 - [Using a Sparse-Tree Directory Structure, page 3-10](#)
 - [Decorating a User Shape, page 3-18](#)
- **Modifying SESM Web Portal application functionality**—The SESM Web Portal applications are fully functional and ready for styling, configuration, and deployment. However, in some deployments, modifications to the functionality of a SESM Web Portal application might be required. These sections provide you with an overview of the SESM Web Portal application functional components and give you some guidance on how to use and modify components:
 - [SESM Architecture: An Overview, page 3-2](#)
 - [SESM Software Concepts, page 3-8](#)
 - [Modifying SESM Web Portal Application Functionality, page 3-29](#)

For illustration purposes, the explanations in this chapter use the NWSP Web Portal application. Though certain advanced techniques are typically used in a SESM Web Portal application, you decide what techniques to use, what techniques to modify, and what techniques not to use based on the application's presentation and business requirements.

SESM Architecture: An Overview

The architecture of a SESM Web Portal application uses the Model-View-Control (MVC) design pattern.

- **Model**—Service, subscriber, and policy information in a data repository as well as the operations that can be used to access and modify this data.
- **Views**—JSPs that generate the markup language, such as HTML or WML, which determines the user interface.
- **Control**—Java servlets that process HTTP requests and are responsible for creating any JavaBeans or other objects used by the JSPs. Each control servlet forwards to a JSP (a view).

The MVC design pattern allows the processing logic in the Java servlets to be separated from the presentation components in the JSPs.

The service-provider developer makes deployment-specific modifications, such as look-and-feel customizations and functionality modifications, to the JSPs that act as views. The SESM model and controls are preprogrammed and configurable by the deployer. No coding tasks related to the model or controls are required to develop a SESM Web Portal application.

Model

In a SESM Web Portal application, the *model* is responsible for the interactions between a number of system components, possibly including one or more Service Selection Gateways (SSGs), RADIUS Data Proxy servers, and data repositories, either a RADIUS AAA server or an LDAP-compliant directory. The model also includes the programming interfaces that a SESM Web Portal application uses to interact with these system components and access information in the data repositories.

The SESM software that implements the model is preprogrammed and configurable by the deployer. The service-provider developer is not required to modify any of the model's preprogrammed software.

Controls

A *control* is a Java servlet that prepares an HTTP request for a view. A control is responsible for creating any JavaBeans or other objects (for example, request or session attributes) used by the JSPs. SESM Web Portal application controls are subclasses of the `com.cisco.sesm.navigator.Control` abstract class.

If a control needs to give a JSP (the corresponding view) access to dynamic data, such as subscriber account data, that the control has retrieved from the model, the control creates a JavaBean and sets the values of the bean properties. The properties hold the data that the control retrieved. In general, the component-to-component flow is that, after the bean is created and its properties are set, the control forwards the request to the corresponding view. The view then uses the bean to access the retrieved data.

After a control has processed an HTTP request, it usually handles a GET or POST request as follows:

- If the request is a GET, the control forwards the request to the corresponding view.
- If the request is a POST, the control redirects to a GET request for the same control servlet. Redirecting to a GET removes the POST request from the HTTP client's history list. If someone goes back in the browser history list, the POST request and its data will not be available.

Internationalized Resources Provided by Controls

The SESM controls provide internationalized data to the views. The JavaBeans created by the SESM controls use internationalized resources, which you can adapt for various languages and regions without programming changes. A view JSP retrieves these internationalized resources from the view bean. The resource is usually text on a label or button, or a message to the subscriber. For example, the `AccountLogonControl`, a control for the subscriber logon page, uses internationalized resources for certain phrases, such as *Please enter your username*. Each text string is a key-value pair in the resource bundle for the SESM Web Portal application.

Localization by View JSP

The control internationalizes the resources by associating them with a key. The view JSP is responsible for localizing the resources. The localization performed by the view JSP formats the internationalized data (for example, a number or date) according to the current localization context (`Locale`). The view JSPs use the `format` tag of the `Localization` tag library to perform the localization. For information on how a SESM Web Portal application localizes internationalized resources, see the [Using Internationalized Resources, page 5-5](#).



Tip

The NWSP Web Portal application's JSPs have the needed code to localize the resources provided by the SESM controls. NWSP software automatically detects the language and country of the subscriber, as defined by the subscriber's, browser preferences and localizes the resources using this locale as the current localization context.

Default Localization Context

The NWSP Web Portal application has default localization context initialization parameters that the deployer can configure in the web.xml file. If the SESM Web Portal application does not support the subscriber's preferred locale as defined by the subscriber's browser preferences, the Web Portal application uses these default values. For information on the default localization context parameters, see the [Setting a Default Localization Context, page 5-14](#).

Views

When the SESM user-shape mechanisms are used, each JSP that acts as a view for a control provides content targeted for the subscriber's specific characteristics. For example, a view might be implemented in three different JSPs. Each JSP provides content using a different markup language, such as HTML, WML and XML, targeted for a different device. Ideally, a view contains little or no Java code so that no Java coding is required.

A view is given all the dynamic data it requires in a JavaBean. Each view is responsible for retrieving data from any beans or other objects that the control creates. The view JSP uses standard JSP tags to retrieve the data from the view bean's properties. The view can also collect data from the subscriber and post it to a control.

A view JSP can contain navigation buttons that link to other controls. When the subscriber clicks the button, the HTTP request is sent to the control, which processes the needed information and forwards the request to the view JSP associated with the control.

Virtual File Names for Views

Each control can have many views. For example, there is one MyAccountControl but multiple JSP files can be used for the view myAccount.jsp. Each of the JSPs for myAccount.jsp might provide content tailored for a different device (for example, WAP phone, PDA, or PC). In effect, MyAccountControl forwards the HTTP request to a logical view named in the deployment descriptor file (web.xml). The control does not know the actual file name—a URI—for the JSP. The control does not know whether there is more than one view or which view will be used.

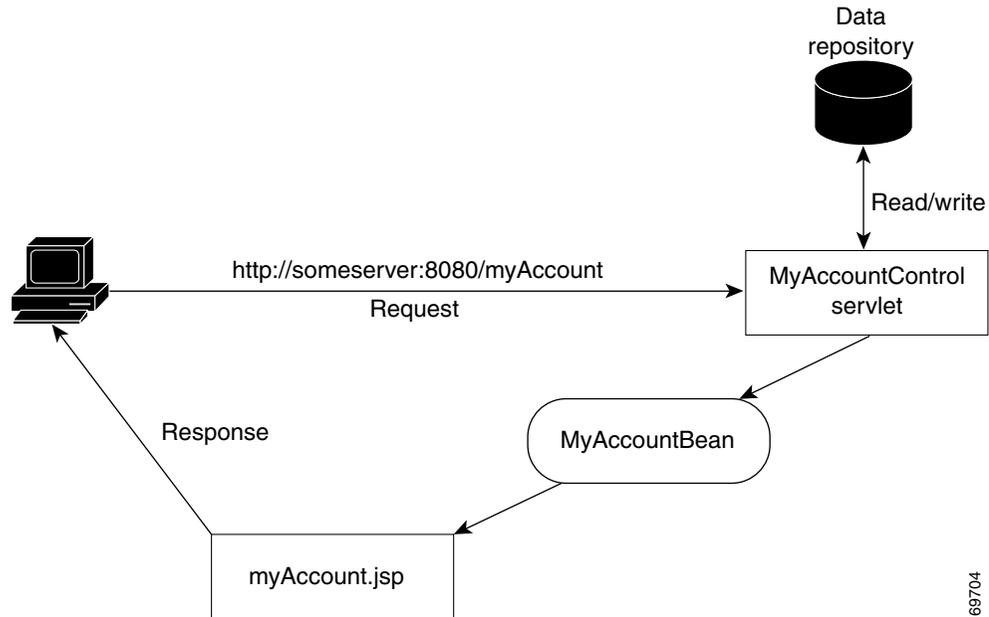
When the view MyAccountView is declared in the web.xml file, the JSP for the view is specified as a virtual file name. When the control forwards the HTTP request to the view, the SESM web-application software uses the characteristics of the subscriber and the subscriber's HTTP client to determine the specific JSP to use for myAccount.jsp.

SESM uses the VirtualFile servlet to translate the virtual file name for the view to an actual file name (URI) based on the subscriber's characteristics. For information on VirtualFile and this translation process, see the [Mapping a Virtual File Name to an Actual File Name, page 3-37](#).

MVC Design Pattern Example

[Figure 3-1](#) shows how the NWSP Web Portal application uses the MVC design pattern for its My Account page. In a SESM SPE installation, the My Account page (shown in [Figure 4-1 on page 4-3](#)) allows the subscriber to display and update account information, such as the subscriber name and address. For simplicity, [Figure 3-1](#) and the accompanying explanation do not show an SSG or explain its role in routing a request.

Figure 3-1 MVC Design for My Account Page



69704

When the subscriber clicks the My Account button in the SESM navigation bar, the following sequence of interactions occurs between the MVC components. **Control:** The following HTTP request is sent to a SESM Web Portal application:

`http://someserver:8080/myAccount`

In the NWSP deployment descriptor file (`web.xml`), the `/myAccount` URL pattern is mapped to the `MyAccount` servlet name. The `MyAccount` servlet name is associated with the `MyAccountControl` servlet class, which is the control for the My Account page. Therefore, the `MyAccountControl` servlet receives the request.

1. **Control and Model:** The `MyAccountControl` servlet does one of the following:

- If the request is a POST, `MyAccountControl` attempts to use form data to update the subscriber profile in the data repository, adds a `MyAccountBean` to the HTTP session, and redirects to a GET request for `MyAccountControl`.
- If the request is a GET, `MyAccountControl` retrieves the account information from the subscriber profile in the data repository, uses an existing or adds a new `MyAccountBean` to the HTTP request, and forwards the request to the `MyAccountView` (which is associated with the virtual filename `myAccount.jsp`).

In both cases, `MyAccountBean` contains the values of the fields (such as names and addresses) that `MyAccountControl` has retrieved from the data repository and that `myAccount.jsp` will display.

2. **View:** When the control forwards the request to the view, the `myAccount.jsp` (the view) gets user-account properties from `MyAccountBean` for each field in the form. For example, to get the value of the `givenName` field for the form, `myAccount.jsp` uses `MyAccountBean` to get the value of the `givenName` bean property.
3. **View:** After `myAccount.jsp` retrieves all user-account information from the bean, it sends the My Account page to the HTTP client, which displays the page.

NWSP Controls, View Beans, and Views

Table 3-1 lists the SESM controls, view beans, and view JSPs that the NWSP Web Portal application uses. The components are listed according to the functionality they provide. The other SESM reference implementations like PDA and WAP provide subsets of the NWSP functionality and use subsets of these controls and beans. They may also use different view JSPs.

Each control forwards to the corresponding view JSP. For example, if a control servlet is named AccountLogoffControl, it forwards to the AccountLogoffView. In the NWSP deployment descriptor file (web.xml), the servlet name AccountLogoffView is associated with the virtual filename accountLogoff.jsp. Therefore, AccountLogoffControl, in effect, forwards to accountLogoff.jsp. For information on declaring controls and views in the web.xml file, see the [Using the SESM Deployment Descriptor File, page 3-33](#).

Table 3-1 *SESM Web Portal Application Controls, View Beans, and Views*

Control	View Bean	View JSPs
Account Information		
MyAccountControl	MyAccountBean	myAccount.jsp myAccountBody.jsp
Account Logon and Logoff		
AccountLogoffControl		accountLogoff.jsp accountLogoffBody.jsp
AccountLogon3KeyControl	Authenticate3KeyBean	accountLogon3Key.jsp accountLogon3KeyBody.jsp
AccountLogonControl	AuthenticateBean	accountLogon.jsp accountLogonBody.jsp
IpassAccountLogonControl	AuthenticateBean	Not applicable. Pages are served from the IpassServlet.
Account Passwords		
AccountPasswordControl	MessagesBean	accountPassword.jsp accountPasswordBody.jsp
Locale Settings		
None	None	locale.jsp localeBody.jsp
Personal Firewalls		
FirewallControl	FirewallBean ACLBean	firewall.jsp firewallBody.jsp
AdvFirewallControl	AdvFirewallBean AdvACLBean ACLBean	advFirewall.jsp advFirewallBody.jsp
Service Subscription		
SubscriptionConfirmControl	SubscriptionManageBean ServiceBean ServiceGroupBean	subscriptionConfirm.jsp subscriptionConfirmBody.jsp

Table 3-1 *SESM Web Portal Application Controls, View Beans, and Views (continued)*

Control	View Bean	View JSPs
SubscriptionManageControl	SubscriptionManageBean ServiceBean ServiceGroupBean	subscriptionManage.jsp subscriptionManageBody.jsp
Service Selection and Logon		
ServiceListControl	ServiceListBean ServiceListServiceBean ServiceListServiceGroupBean	serviceListHead.jsp serviceList.jsp serviceListService.jsp serviceListGroup.jsp
ServiceLogonControl	ServiceAuthenticateBean	serviceLogon.jsp serviceLogonBody.jsp
ServiceStartControl		serviceStart.jsp
ServiceStopControl	None	serviceStop.jsp
Service Status		
StatusControl	StatusBean	status.jsp statusBody.jsp
SESM Model and HTTP Session Messages		
com.cisco.sesm.webapp.control.MessagesControl	MessagesBean	messages.jsp messagesBody.jsp
Subaccounts		
SubaccountConfirmControl	SubscriptionManageBean ServiceBean ServiceGroupBean	subaccountConfirm.jsp subaccountConfirmBody.jsp
SubAccountListControl	SubAccountListBean SubAccountDetailBean	subAccountList.jsp subaccountListBody.jsp
SubaccountSubscriptionsControl	SubaccountSubscriptionsBean ServiceBean ServiceGroupBean	subaccountSubscriptions.jsp subaccountSubscriptionsBody.jsp
SubaccountSubscriptionsControl	SubaccountSubscriptionsBean ServiceBean ServiceGroupBean	subaccountSubscriptions.jsp subaccountSubscriptionsBody.jsp

**Note**

In the NWSP Web Portal application, authentication with 3 keys uses `AccountLogon3KeyControl` and the `accountLogon3Key.jsp` view. To configure the NWSP Web Portal application for 3-key authentication, you must make changes to the `web.xml`. For information on how to configure authentication with 3 keys, see *Cisco Subscriber Edge Services Manager Web Portal Guide*.

For more information on each control and JavaBean, see the Javadoc for the associated class. The Javadoc for the controls and beans provides information that is useful if you need to modify the code in a view JSP. For example:

- If a JSP associated with a control displays a form that the subscriber uses to fill in information, the description of the control includes the `POST` parameters that are expected by the control.
- If a JSP retrieves information from a JavaBean that a control creates, the description of the bean lists the information that the bean provides.

When a JSP posts parameters to a control, the HTTP protocol provides no mechanism for checking that the correct parameters and appropriate values have been posted. The posting of parameters requires that the developer test to verify the results.

SESM Software Concepts

This section provides introductory information on some concepts that you need to understand before developing a Cisco SESM Web Portal application:

- [User Shapes and User-Shape Decoration](#)
- [Decorators](#)

User Shapes and User-Shape Decoration

A *user shape* is a set of characteristics that defines the web resources that a Cisco SESM Web Portal application uses for a specific subscriber. A `Shape` object encapsulates the 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 website.
- Branding for the website, such as a brand for business use and a brand for personal use.
- Language and country of the user.
- Personal characteristics, such as the interests of the subscriber.

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

When a SESM Web Portal application uses the user-shape mechanisms to customize the web resources for a specific subscriber, you perform two sets of development and deployment activities. You:

- Organize web resources into a sparse-tree directory structure.
- Implement user-shape decoration based on the Web Portal application's requirements.

Sparse-Tree Directory Structure

All web resources, such as JSPs and GIF images, that will be served to the subscriber need to be organized into a sparse-tree directory structure. A *sparse-tree directory structure* is the directory structure of a Cisco SESM Web Portal application. For example, if a SESM Web Portal application is required to accommodate subscribers whose language is English or Spanish, two sets of GIF images that contain text may be required, one set of images for each language. These two sets of images would be organized into separate English and Spanish directories in the sparse-tree directory structure. For information how you implement this type of directory structure, see the [Using a Sparse-Tree Directory Structure](#), page 3-10.

User-Shape Decoration

Based on application-specific business requirements, you need to determine and implement what the Cisco SESM Web Portal application requires for user-shape decoration. A SESM Web Portal application includes a group of JSPs and servlets that are responsible for *decoration of the user shape*: setting shape *dimensions* (characteristics) such as the device, brand, and locale for a specific subscriber.

Each dimension corresponds to a characteristic of the subscriber (for example, the browser software used by the subscriber). The value of each dimension specifies one or more directories that SESM searches—in the sparse-tree directory structure—for requested web resources for this subscriber. You determine the dimensions that the SESM Web Portal application will use and the values for each dimension. For information on how you implement user-shape decoration, see the [Decorating a User Shape, page 3-18](#).

Decorators

In a SESM Web Portal application, decorators are servlets or JSPs that implement most of the SESM control and view functionality. A set of decorators is included with SESM. A SESM decorator modifies some aspect of the current HTTP request, response, session, or application. Most decorators add an attribute to the HTTP request or session. Some decorators modify HTTP response headers or status.

For example, a decorator might add a request attribute specifying a characteristic of the user shape. Another decorator might add a status code to a response header indicating that a resource cannot be found. The SESM controls, such as the MyAccountControl and MyServicesControl, are decorators that add a JavaBean to the request.

A decorator can be implemented as a Java servlet or as a JSP.

- The service-provider developer does *not* usually create decorators that are Java servlets. For information on creating a Java servlet decorator, see *Cisco Subscriber Edge Services Manager SDK Programmer Guide*.
- The service-provider developer might need to create decorators that are JSPs. This chapter provides information on creating JSP-page decorators.

Decorator Class

Though the developer does not program decorators that are servlets, it is important to understand the role played by some of the methods that are found in a Decorator class. All decorators that are servlets are subclasses of Decorator. Three methods in the Decorator class are particularly significant:

- `isNecessary`—Tells whether or not decoration is needed. For example, if a `shape` attribute is not present in the HTTP session, the ShapeDecorator servlet, which is responsible for creating a Shape object and adding the attribute, is programmed so that `isNecessary` returns true.
- `decorateIfNecessary`—Calls the `decorate` method to perform decoration if `isNecessary` returns true.
- `decorate`—Performs decoration. For example, with ShapeDecorator, decoration adds the `shape` attribute to the HTTP session.

For information on the Decorator class, see the Javadoc documentation that is installed with SESM.

The SESM software and the NWSP Web Portal application provide a complete set of decorators. In many deployments, no additional decorators will be required. However, you can modify or extend the preprogrammed SESM software with JSP-page decorators. The JSP-page decorators created by the service-provider developer are typically dimension decorators or post-decorators.

Dimension Decorators

A dimension decorator is a decorator that detects a user-shape characteristic (such as device, brand, or locale). Dimension decorators are the SESM mechanism for detecting and setting user-shape characteristics so that the SESM Web Portal can serve customized web resources to each subscriber.

For example, consider a dimension decorator that detects the device of the subscriber and updates an HTTP session attribute *shape* that defines the user shape. The user shape in the *shape* attribute contains a Shape object that includes a device dimension. If the subscriber is using a PDA, the dimension decorator detects this using HTTP header information. When updating the device dimension, this device dimension decorator sets the directory path for the dimension to *pda*. The *pda* directory is where PDA-specific resources reside in directory hierarchy used by the application. The name of the dimension ID (in this example, device) and name of the directory (*pda*) are arbitrary and could be defined differently by the deployer.

You can add, modify, or remove dimension decorators based on deployment-specific requirements. For more information on dimension decorators, see the [Decorating a User Shape, page 3-18](#).

Post-Decorators

A post-decorator is a decorator that is invoked after a normal servlet or JSP-page decorator executes. A post-decorator is a mechanism for extending the usual, preprogrammed SESM Web Portal application functionality. You can create custom post-decorators for a variety of application-specific tasks. You configure a post-decorator in the deployment descriptor file (*web.xml*) so that the post-decorator is invoked after another decorator executes.

For example, the NWSP Web Portal application software includes a decorator called *HttpSniffDecorator* to detect HTTP client characteristics such as the browser, device, and operating system that the subscriber is using. If your deployment requires that additional information about the HTTP client be detected, you could create a JSP to act as a post-decorator.

The NWSP Web Portal application includes such a post-decorator named *httpSniff.jsp*, which sends a JavaScript probe to the HTTP client device to try to detect the characteristics of the HTTP client device. It also detects whether the HTTP client device is a WAP phone simulator. When a WAP phone simulator is detected, the NWSP Web Portal application uses WML markup language in the content served to the subscriber.

The SESM developer can add, modify, or remove post-decorators based on deployment-specific requirements. For more information on creating post-decorators, see the [Creating JSP-Page Decorators and Post-Decorators, page 3-31](#).

Using a Sparse-Tree Directory Structure

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

- Website pages hierarchy.
- User-shape hierarchy.

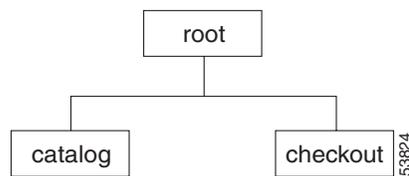
These two hierarchies are developed independently of each other. However, all web resource files are located within one Web Portal application because you combine the two hierarchies into one large hierarchy whose root is the Web Portal application. For this discussion, a web resource might be a JSP,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

For information on the configuration and programmatic details for implementing the user-shape model, see the [Decorating a User Shape, page 3-18](#).

Website Pages Hierarchy

A website pages hierarchy is the directory structure of a conventional website, which typically includes a single copy of each required resource. For example, consider a Web Portal 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 3-2](#) shows the website pages hierarchy.

Figure 3-2 Website Pages Directory Hierarchy



User-Shape Hierarchy

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

User-Shape Example

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 Portal application have the following characteristics:

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

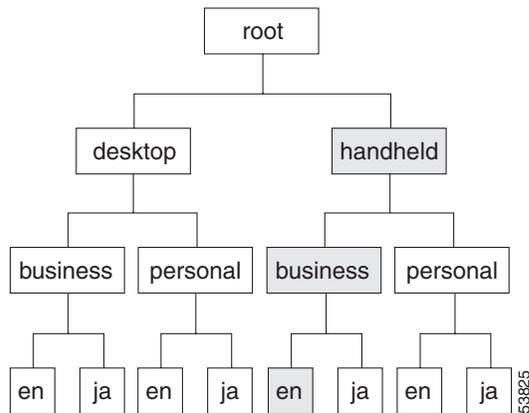
Each of these characteristics (devices, brands, and locales) is a dimension, and each dimension has one or more possible values. The user shape in this example has three dimensions and two possible values for each dimension. Each subscriber's user shape has a specific value for each dimension. In this 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 3-3](#) shows the user-shape directory hierarchy.

Figure 3-3 User-Shape Directory Hierarchy



The Cisco SESM Web Portal application detects the characteristics of the subscriber and the HTTP client and sets the values of each dimension. For each specific user shape, a SESM Web Portal 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 3-3](#), 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 dimension could be extended. For example, the third dimension (locales) could be extended to include French and Spanish. The number of dimensions is configurable in the `web.xml` file. The range of values allowed for each dimension is determined by the SESM Web Portal application.

Location of Directory Dimensions and Order of ShapeDecorator Dimensions

You should consider the following factors when deciding where a dimension appears within the user-shape directory hierarchy.

In general, the more frequently a value appears in the user-shape directory hierarchy, the further down in the hierarchy it is located. For example, in [Figure 3-3](#) 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 given in the dimensions initialization parameter of the `ShapeDecorator` determines the order in which SESM searches the directories for a web resource. In the Web Portal application `web.xml` file, the dimensions initialization parameter must specify the dimensions using the same order as is found in the user-shape directory hierarchy. For example, in [Figure 3-3](#), the dimensions in the example user-shape directory hierarchy are in this order:

- Devices.
- Brands.
- Locales.

The order given in dimensions initialization parameter of the ShapeDecorator must use the same order:

```
<servlet>
  <servlet-name>Shape</servlet-name>
  <servlet-class>com.cisco.sesm.navigator.ShapeDecorator</servlet-class>
  <init-param>
    <param-name>dimensions</param-name>
    <param-value>
      device      <!-- Order of dimensions must be the same as order -->
      brand       <!-- in the user-shape directory hierarchy -->
      locale
    </param-value>
  </init-param>
  ...

```



Note

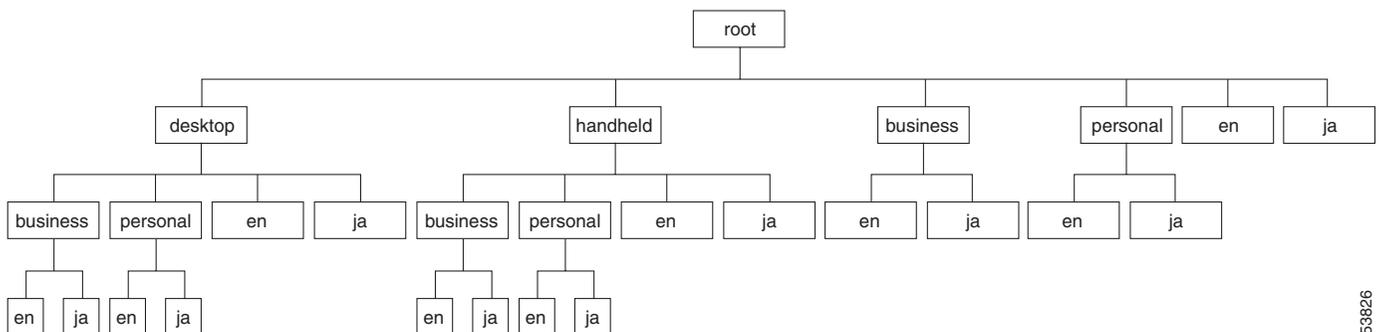
Specifying an order in the dimensions initialization parameter that does not match the order of user-shape directory hierarchy will result in a failed search for a web resource. For example, if the user-shape directory hierarchy has a location `brand_dir/device_dir/locale_dir`, but the order of dimensions in the dimensions parameter is `device, brand, locale`, SESM will never search the location `brand_dir/device_dir/locale_dir`. The order of dimensions in the dimensions parameter determines the search order. In this example, the search order is `device_dir/ brand_dir/locale_dir`.

Except for search order, the dimensions in the user-shape directory hierarchy are independent of each other. For more information on the dimensions initialization parameter and search order, see the [Searches for a Web Resource, page 3-14](#).

Sparse-Tree Directory Structure

The directory hierarchy that a Cisco SESM Web Portal 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 3-4](#) shows the user-shape directory hierarchy expanded to include all possible combinations of dimensions.

Figure 3-4 Fully Expanded User-Shape Directory Hierarchy



53826

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 3-4](#). 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 3-4](#) is not likely in a production 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. For example, in [Figure 3-3](#), 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 SESM 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 website page hierarchy at the root directory of the user-shape hierarchy. This directory structure will appear as a typical website.
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 SESM Web Portal application 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 Example, page 3-11](#). 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, you need to know these facts about the Cisco SESM Web Portal application and one of the initialization parameters of the `ShapeDecorator` servlet.

- The SESM Web Portal application sets the values for the dimensions of the user shape, specifying one or more directories for each dimension. The value of a dimension is set by a dimension decorator, which is a servlet or JSP that specifies the dimension value such as *premium* for the brand dimension or *en/GB* for the locale dimension. For information on dimension decorators, see the [SESM-supplied Dimension Decorators, page 3-21](#) and the [Creating or Customizing Dimension Decorators, page 3-25](#).
- In the Web Portal application `web.xml` file, the `ShapeDecorator` servlet's dimensions initialization parameter determines the order in which SESM searches directories for a web resource. For information on the dimensions parameter, see the [dimensions Initialization Parameter, page 3-20](#).

Order in dimensions Initialization Parameter

In the `web.xml` file, the dimensions initialization parameter of the `ShapeDecorator` servlet specifies a set of dimension IDs that SESM uses when it creates the dimensions for a subscriber shape. The order in which the dimension IDs are listed is significant because it defines the search order that SESM uses to find a web resource for a subscriber.

In the following example, the dimensions initialization parameter specifies three directory paths, one path for each dimension in the user shape. The dimensions are for devices, brands, and locales. To simplify this description, the dimensions are designated in the comments as A, B, and C.

```
<servlet>
  <servlet-name>Shape</servlet-name>
  <servlet-class>com.cisco.sesm.navigator.ShapeDecorator</servlet-class>
  <init-param>
    <param-name>dimensions</param-name>
    <param-value>
      device    <!-- dimension A -->
      brand     <!-- dimension B -->
      locale    <!-- dimension C -->
    </param-value>
  </init-param>
  ...

```

For a subscriber with the user shape *desktop*, *business*, *ja*, assume that the SESM Web Portal 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 initialization parameter, user shape, and directory paths, the Cisco SESM software searches the locations shown in [Table 3-2](#) (in the order listed) for `/images/logo.gif`. As shown in [Table 3-2](#), 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 Portal 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 3-2](#) until the Cisco SESM software finds the resource.

Table 3-2 SESM Search Algorithm: Example 1

Search Order	Directory Path (from document root)	Path Assembly
1	<code>/desktop/business/ja/images/logo.gif</code>	A/B/C
2	<code>/desktop/business/images/logo.gif</code>	A/B
3	<code>/desktop/ja/images/logo.gif</code>	A/C
4	<code>/desktop/images/logo.gif</code>	A
5	<code>/business/ja/images/logo.gif</code>	B/C
6	<code>/business/images/logo.gif</code>	B
7	<code>/ja/images/logo.gif</code>	C
8	<code>/images/logo.gif</code>	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 initialization parameter in `web.xml`. The order of the dimension IDs in the dimensions initialization parameter 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 initialization parameter. In the search, the last dimension (C) is the most persistent and is discarded last. The first dimension (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 Example, page 3-11](#). 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 initialization parameter and directory paths as in Example 1. For a user with the shape `desktop`, `business`, `ja`, the Cisco SESM Web Portal application searches the locations shown in [Table 3-3](#), in the order listed, for `/images/button.gif`.

Table 3-3 *SESM Search Algorithm: Example 2*

Search Order	Directory Path (from document root)	Path Assembly
1	<code>/desktop/business/ja/images/button.gif</code>	A/B/C
2	<code>/desktop/business/images/button.gif</code>	A/B
3	<code>/desktop/ja/images/button.gif</code>	A/C
4	<code>/desktop/images/button.gif</code>	A
5	<code>/business/ja/images/button.gif</code>	B/C
6	<code>/business/images/button.gif</code>	B
7	<code>/ja/images/button.gif</code>	C
8	<code>/images/button.gif</code>	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 dimension (C) is the most persistent, and the first dimension (A) is the least persistent.

Example 3: Searches for a Web Resource

The value of a user-shape dimension can be a list of multiple directory names. For example, the value of the locale dimension could be two directories: a language and a region such as “fr/CA” (the French language and the region Canada). This example illustrates how SESM searches for a web resource when an ordered list of multiple directory names defines a dimension.

For this example, assume that the user shapes and user-shape directory hierarchy are a little different than those described in the [User-Shape Example, page 3-11](#). In this example, the values of the device dimension and brand dimension are each defined by a single directory, but the value of the locale dimension can be defined by two directories (a language and a region):

Dimension	Directories
device	<code>device_dir</code>
brand	<code>brand_dir</code>
locale	<code>language_dir/region_dir</code>

Assume that Example 3 uses the same dimensions initialization parameter as in Example 1. In this example, assume that an `/images/banner.gif` file resides in the following locations:

- `/handheld/fr/CA/images/banner.gif`.
- `/fr/CA/images/banner.gif`.
- `/images/banner.gif`.

For a user with the shape `handheld, business, fr/CA`, the Cisco SESM Web Portal application searches the locations shown in Table 3-4, in the order listed, for `/images/banner.gif`.

Table 3-4 SESM Search Algorithm: Example 3

Search Order	Directory Path (from document root)	Path Assembly
1	<code>/handheld/business/fr/CA/images/banner.gif</code>	A/B/C (fr/CA)
2	<code>/handheld/business/fr/images/banner.gif</code>	A/B/C (fr)
3	<code>/handheld/business/images/banner.gif</code>	A/B
4	<code>/handheld/fr/CA/images/banner.gif</code>	A/C (fr/CA)
5	<code>/handheld/fr/images/banner.gif</code>	A/C (fr)
6	<code>/handheld/images/banner.gif</code>	A
7	<code>/business/fr/CA/images/banner.gif</code>	B/C (fr/CA)
8	<code>/business/fr/images/banner.gif</code>	B/C (fr)
9	<code>/business/images/banner.gif</code>	B
10	<code>/fr/CA/images/banner.gif</code>	C (fr/CA)
11	<code>/fr/images/banner.gif</code>	C (fr)
12	<code>/images/banner.gif</code>	none (document root)

In the Example 3 search, the Cisco SESM software first finds the web resource `/images/banner.gif` at `/handheld/fr/CA/images/banner.gif` and uses that resource in its response to the client. In the search, notice that the directory path for the locale dimension (C) is deconstructed each time it is encountered. Each time SESM searches `/fr/CA` first, and then searches `/fr`.

Decorating a User Shape

This section describes how a developer configures and customizes a Cisco SESM Web Portal application's components for user-shape decoration.

A SESM Web Portal application includes a group of JSP components that are responsible for decoration of the user shape: setting dimensions (characteristics) such as the device, brand, and locale for a specific subscriber.

Each SESM Web Portal application includes a set of components called dimension decorators that are responsible for setting the user-shape characteristics. You can use these SESM-supplied dimension decorators, or create one or more customized dimension decorators to meet application-specific requirements. This section includes information on these topics:

- Configuring User-Shape Dimensions.
- Customizing Dimension Decorators.


Note

Before you read this section on using the decorator components, read the [Using a Sparse-Tree Directory Structure, page 3-10](#). The techniques for user-shape decoration require that the structure of the SESM website and the techniques for user-shape decoration take a coordinated approach. The explanations in the two sections complement each other.

Configuring User-Shape Dimensions

For detailed information about shape objects see [User Shapes and User-Shape Decoration, page 3-8](#)

ShapeDecorator Initialization Parameters

In the web.xml file, the ShapeDecorator servlet has two initialization parameters that relate to the dimensions of the user shape:

- `dimensions`.
- `postDecorate`.

In the following example, the user shape consists of three dimensions: device, brand, and locale.

```
<servlet>
  <servlet-name>Shape</servlet-name>
  <servlet-class>com.cisco.sesm.navigator.ShapeDecorator</servlet-class>
  <init-param>
    <param-name>dimensions</param-name>
    <param-value>
      device
      brand
      locale
    </param-value>
  </init-param>
  <init-param>
    <param-name>postDecorate</param-name>
    <param-value>
      DeviceDimension
      BrandDimension
      LocaleDimension
    </param-value>
  </init-param>
  <load-on-startup>1</load-on-startup>
</servlet>
```

Given the preceding dimensions and postDecorate initialization parameters, the ShapeDecorator servlet, when it is invoked, creates a Shape object having three dimensions: device, brand, and locale. Each dimension is initially created with no value.

In the preceding example, three post-decorators are specified in the postDecorate parameter of ShapeDecorator. When the post-decorators execute, these servlets set the values of the three dimensions of the user shape.

The following sections provide more details on the dimensions and postDecorate initialization parameters.

dimensions Initialization Parameter

In the web.xml file, the dimensions initialization parameter specifies a set of zero or more dimension IDs that SESM uses when it creates the dimensions for a user shape. You can modify the number and order of the dimensions in the dimensions parameter to meet the needs of a specific deployment.



Note

In the Web Portal application web.xml file, the `dimensions` initialization parameter must specify the dimensions using the same order as is found in the user-shape directory hierarchy. For more information, see the [Location of Directory Dimensions and Order of ShapeDecorator Dimensions, page 3-12](#).

In the `dimensions` parameter, the dimension IDs are strings that are separated by whitespace or commas. The order in which the dimensions are listed is significant because it defines the search order that SESM uses to find a resource for a subscriber. For information on search order, see the [Searches for a Web Resource, page 3-14](#).

When you are configuring dimension IDs in the `dimensions` parameter, the dimension ID that is associated with a dimension decorator is determined as follows:

- If the dimension decorator is a JSP, the dimension ID corresponds to the first argument specified for the `Dimension` constructor when the dimension is created in the JSP. The following example shows the relevant code from the `locationDimension.jsp`.

```
Dimension dim = new Dimension("location", "CityHotel")
```

The ID that identifies this dimension decorator is *location*.

- If the dimension decorator is a servlet, the dimension ID specified in the dimensions parameter is the ID that the servlet returns in its `getId` method. In the typical case, service-provider developers do not create servlet dimension decorators.

[Table 3-5](#) lists the dimension IDs for the SESM-supplied dimension decorators that are part of the NWSP Web Portal application.

postDecorate Initialization Parameter

In the web.xml file, the postDecorate initialization parameter for ShapeDecorator specifies a set of post-decorators that define the values of the dimensions for a user shape. The number of post-decorators corresponds to the number of dimensions in the user shape. The value of each dimension is a directory name or a list of directory names that the dimension's post-decorator determines based on the characteristics of the user shape.

The post-decorators declared with the postDecorate parameter give the names of the servlets that set the dimension values for the user shape. The names given in postDecorate are the servlet names specified in the web.xml file. Given the preceding sample postDecorate servlet initialization, the DeviceDimension servlet sets the value of the device dimension, the BrandDimension servlet sets the value of the brand dimension, and so on.

When you declare the post-decorators in the postDecorate parameter of ShapeDecorator, the names are delimited by whitespace or commas. In the postDecorate parameter, the post-decorators can be servlets or JSPs declared as servlets elsewhere in the web.xml file.

Default Dimension Decorator Values

Like any other decorator, a dimension decorator JSP or servlet can use a default value for the dimension. The default value comes from the defaultValue initialization parameter for the JSP or servlet as defined in the Web Portal application web.xml file.

A dimension decorator servlet or JSP can retrieve and use the default value for the dimension. The default value is used when the dimension decorator cannot detect a characteristic for that dimension of the subscriber. One advantage to using the defaultValue initialization parameter is that the deployer can change the default value without modifying any code.

For example, the locationDimension.jsp of the NWSP Web Portal application specifies a default value for the location dimension as follows:

```
<servlet>
  <servlet-name>LocationDimension</servlet-name>
  <jsp-file>/decorators/locationDimension.jsp</jsp-file>
  <init-param>
    <param-name>defaultValue</param-name>
    <param-value>airport</param-value>
  </init-param>
  <load-on-startup>1</load-on-startup>
</servlet>
```

A JSP can retrieve the default value using the getInitParameter method. In the following example, the locationDimension.jsp retrieves the default value when creating the location dimension and no other value is available:

```
dim = new Dimension(ID, getServletConfig().getInitParameter("defaultValue"));
```

For an explanation of locationDimension.jsp, see the [Creating or Customizing Dimension Decorators](#), page 3-25.

SESM-supplied Dimension Decorators

Each SESM Web Portal application includes a set of dimension decorators that set the values of the dimensions of the user shape. You can use these SESM-supplied dimension decorators, or you can create one or more customized dimension decorators to meet application-specific requirements. [Table 3-5](#) provides information on the SESM-supplied dimension decorators that are part of the NWSP Web Portal application.



Note

To detect some characteristics of the HTTP client device, the NWSP Web Portal application uses a JavaScript facility contained in the javascriptProbe.jsp file. The subscriber's browser must have JavaScript enabled for this facility to work.

Table 3-5 **SESM-supplied Dimension Decorators**

Dimension Decorator	Dimension ID	Description
brandDimension.jsp	brand	Sets the brand dimension according to the user group to which the subscriber belongs.
ColorDimensionDecorator	color	Sets the value for the color dimension to <i>color/16</i> for all client devices.
ConnectionDimensionDecorator	connection	Not currently used.

Table 3-5 SESM-supplied Dimension Decorators (continued)

Dimension Decorator	Dimension ID	Description
DeviceDimensionDecorator	device	<p>Sets the device dimension to the category of the device running the HTTP client. Possible values are:</p> <ul style="list-style-type: none"> pc (personal computer) pda (personal digital assistant) wap (WAP phone) <p>If SESM cannot determine the client device, the value is set to <i>pc</i>.</p>
LocaleDimensionDecorator	locale	<p>Sets the locale dimension to the value of locale for the current localization (L10nContext) context. For example, if the L10nContext locale is <i>en_GB</i>, the value of the dimension is <i>en/GB</i>. This value specifies two directories (en and GB) where Web Portal application resources are located if the subscriber speaks English and is from Great Britain.</p>
locationDimension.jsp	location	<p>Sets the location dimension to the value specified in the "LOCATION" attribute of the SESM session key.</p>
MarkupDimensionDecorator	markup	<p>Sets the markup dimension to the name and version numbers of the markup language that is supported by the browser on the HTTP client device.</p> <ul style="list-style-type: none"> an empty string (unknown markup language) html html/3 html/3/4 html/3/layers wml xhtml
OSDimensionDecorator	os	<p>Sets the os dimension to the name of the operating system that is running on the HTTP client device. Possible values are:</p> <ul style="list-style-type: none"> an empty string (unknown operating system) ce (Windows CE) nt (Windows NT)

Table 3-5 *SESM-supplied Dimension Decorators (continued)*

Dimension Decorator	Dimension ID	Description
ScriptDimensionDecorator	script	<p>Sets the script dimension to the name and version numbers of the script language that is supported by the browser on the HTTP client device. Possible values are:</p> <ul style="list-style-type: none"> • an empty string (unknown or no script language) • javascript • javascript/1 • javascript/1/2 • javascript/1/2/3 • javascript/1/2/3/4 • javascript/1/2/3/4/5 • javascript/2 • wmlscript
SizeDimensionDecorator	size	Not currently used.

If you want to implement the color, connection, or size dimensions, you can obtain the values for these dimensions from the HTTP client browser by using the JavaScript probe facility. You must modify the `javascriptProbe.jsp` file to set the values for these dimensions.

For more information on the SESM-supplied dimension decorators, see the Javadoc documentation that is installed with SESM.

Creating or Customizing Dimension Decorators

For some deployments, you might be required to create or customize a dimension decorator. This section provides some guidance on using a JSP to create or customize a dimension decorator.

Like any other decorator, a decorator that defines a dimension of the user shape can be a JSP. Using a JSP to define a dimension has the advantage of not requiring the coding of a Java class or packaging in a JAR file. Each JSP-page dimension decorator performs certain fundamental tasks:

1. Provides a definition for the `jspInit` method and registers itself with the decorator pool
2. Defines a `decorate` method that:
 - a. Detects or retrieves a value for some characteristic of the user shape: device, brand, locale, browser, and so on
 - b. Based on the detected characteristic, creates and defines a `Dimension` object, setting the value of the dimension to the detected or retrieved value
 - c. Updates the user's `Shape` object with the new value of the `Dimension`
3. Invokes the `decorate` method

The following example shows the JSP `locationDimension.jsp` that defines the location of the subscriber. The location is the physical location of a mobile client device. For example, the location might be a restaurant, library, or airport. A SESM Web Portal application can use the location to serve customized resources to the subscriber. For information on location awareness and location-based branding, see *Cisco Subscriber Edge Services Manager Web Portal Guide*.

```
<%@ page import="java.io.IOException" %>
<%@ page import="com.cisco.sesm.logging.Log" %>
<%@ page import="com.cisco.sesm.shape.Shape" %>
<%@ page import="com.cisco.sesm.core.model.SESMSession" %>
<%@ page import="com.cisco.sesm.core.location.LocationFactory" %>
<%@ page import="com.cisco.sesm.shape.Dimension" %>
<%@ page import="com.cisco.sesm.navigator.DecoratorPool" %>
<%@ page import="com.cisco.sesm.navigator.ShapeDecorator" %>
<%@ page import="com.cisco.sesm.webapp.decorator.SESMSessionDecorator" %>
<%@ taglib uri="http://www.cisco.com/taglibs/navigator" prefix="nav" %>

<!-- static final String ID = "location"; %>

<!-- register with pool -->
<!-- public void jspInit() {DecoratorPool.register(this);} %>

<!-- decorate -->
<!--
public void decorate(HttpServletRequest request,
                    HttpServletResponse response)
throws ServletException, IOException
{
    Dimension dim;
    String loc;
    SESMSession sesmSession = SESMSessionDecorator.getSESMSession(request, response);
    Object location = sesmSession.getKey().getAttribute("LOCATION");
    Log.debug("SESM attribute 'LOCATION'=", location);
    if (location == null) {
        // This code also sets the location in the session.
        location = LocationFactory.getInstance().getLocation(sesmSession);
    }
    if(location != null) {
        loc = location.toString();
    }
    else {
        // If the location still isn't set, use the default.
        loc = getServletConfig().getInitParameter("defaultValue");
    }
    if (loc == null) {
        // If the location still isn't set, use an empty string.
        loc = "";
    }
    // Set the location string in the request if want a JSP bean.
    request.setAttribute(ID,loc);
    Log.debug("Set request attribute '" ,ID,"'=", loc);

    // Now construct a dimension and update the shape with this.
    if(loc != null && loc.length() != 0) {
        // Construct a dimension with a single directory
        // if there are no '/' in the location string,
        // or with multiple directories if there are any '/'.
        // For example, if the location string is "a/b/c",
        // then the search includes "a/b/c", "a/b", "a" and "".
        dim = new Dimension(ID, loc, "/" );
    } else {
        // Construct a dimension with a location set to null.

```

```

        loc = null;
        dim = new Dimension(ID, loc);
    }
    Shape shape = (Shape)request.getSession().getAttribute("shape");
    if (shape != null)
        shape.updatedimension(dim);
    else
        Log.warning("No Shape to decorate with dimension=", dim);
}
%>

<!-- logging -->
<!-- static final String fragment = "/decorators/locationDimension.jsp"; %>
<%@ include file="/logging/enterFragment.jspf" %>

<!-- Exit if invoked by DecoratorPool (to force initialization). -->
<% if (request.getParameter("DecoratorPool") != null) return; %>

<!-- Run the decorate() method and any post-decorators. -->
<nav:decorate name="<%=getServletName()%>"/>

<!-- logging -->
<%@ include file="/logging/exitFragment.jspf" %>

```

As shown in the preceding example, when a JSP is a dimension decorator, it performs the following tasks:

1. Provides a definition for the `jspInit` method, the standard JSP initialization mechanism. The `jspInit` definition registers the JSP using the `DecoratorPool.register` method. This step is required.
2. Defines the `decorate` method. The definition for the `decorate` method must match the signature of the `Decorator.decorate` method. The signature includes the method name, type, visibility, arguments, and return type. This step is required.

In this example, the subscriber location is stored in a SESM session attribute.



Note

Tasks 3 and 4 describe the method that is used to retrieve a SESM session attribute named *LOCATION*, which stores a value for the location dimension. Dimension decorators detect characteristics of the user shape in a variety of ways. In this example, the details for retrieving the value of the location dimension are specific to `locationDimension.jsp` and are not generally applicable to other dimension decorators.

3. Uses the `getSESMSession` method to get the `SESMSession`. `SESMSession` is the main access class for the model. `SESMSession` provides the functionality necessary to manipulate a session and get any data associated with that session. For more information on the `SESMSession` class, see the Javadoc that is installed with SESM.
4. Uses `sesmSession.getKey().getAttribute(LOCATION)` to get the `SESMSessionKey` for this session and to retrieve the `SESMSession` attribute named *LOCATION*.

5. Creates a new Dimension object for the location and initializes the Dimension object as follows:
 - The first argument to the Dimension constructor is the ID for the dimension. In this example, the ID is the string *location*. In the web.xml file, the ID for each dimension decorator is specified in the dimensions parameter of the ShapeDecorator servlet.
 - The second argument to the Dimension constructor is the value (a directory name) for the dimension.
 - If the SESMSession attribute named *LOCATION* (now stored in location variable) is not equal to null, the second argument is the value of that attribute.
 - If the SESMSession attribute equals null, the second argument is obtained from the defaultValue initialization parameter of the locationDimension.jsp decorator. The initialization parameter is specified in the Web Portal application web.xml file.
6. Assigns the value of the *shape* session attribute to the shape variable and tests whether shape is equal to null. The ShapeDecorator servlet, which runs when the SESM Web Portal application starts, creates a Shape object and sets the *shape* session attribute to the value of the subscriber's Shape. JSPs use the *shape* session attribute to access the Shape object.
 - If shape is not equal to null, the JSP updates the *location* dimension of the Shape object with the value (directory name) specified when the Dimension was created in Step 5.
 - If *shape* is equal to null, the JSP logs a warning message.
7. When an unregistered decorator is requested, DecoratorPool attempts to force initialization of the decorator in a separate HTTP request. In this case, the JSP returns without calling the decorate method.

The JSP-page decorator tests whether it is being executed for its intended purpose or to force initialization. The test for forced initialization is the existence of the DecoratorPool request parameter. If the DecoratorPool parameter exists, the DecoratorPool servlet is executing the JSP to force initialization.
8. Uses the decorate tag of the Navigator tag library to invoke the decorate method of locationDimension.jsp. The decorate tag provides specialized functionality that invokes any pre- and post-decorators that have been declared in the web.xml file for locationDimension.jsp. If a JSP were to call its decorate method directly (rather than using the decorate tag), pre-decorators and post-decorators are not invoked.

Modifying Dimension Decorators

The Cisco SESM Web Portal applications like NWSP contain a set of dimension decorators. [Table 3-5 on page 3-21](#) lists the dimension decorators that are found in NWSP. You can extend the functionality of the SESM-supplied dimension decorators that are servlets, and can modify the functionality of the dimension decorators that are JSPs.

Servlet Dimension Decorators

Most of the SESM-supplied dimension decorators are implemented as servlets that you cannot modify. You can use the postDecorate initialization parameter to specify a JSP-page post-decorator that extends the behavior of a dimension decorator. You can use the post-decorator mechanism to add to or modify the functionality a SESM-supplied dimension decorator that is implemented as a servlet. For information on creating a post-decorator, see the [Creating JSP-Page Decorators and Post-Decorators, page 3-31](#).

JSP-Page Dimension Decorators

Some of the SESM-supplied dimension decorators are implemented as JSPs. In the NWSP Web Portal application, the dimension decorators that are JSP s include `brandDimension.jsp` and `locationDimension.jsp`. You can directly modify the functionality of a dimension decorator that is a JSP. As an alternative, you could extend the behavior of a JSP dimension decorator by defining a `post-decorator`.

Adding Dimension Decorators

If a SESM Web Portal application requires a new dimension for the user shape, the new dimension decorator is implemented as a JSP-page. When you add a new dimension for the user shape, you must do the following:

1. Create a JSP for the dimension decorator. See the [Creating or Customizing Dimension Decorators, page 3-25](#).
2. Optionally, create a JSP for a post-decorator that will be invoked after the dimension decorator. See the [Creating JSP-Page Decorators and Post-Decorators, page 3-31](#).
3. Declare both the dimension decorator and any post-decorators as servlets in the Web Portal application's `web.xml` file.
4. Optionally, in the `web.xml` declaration of the dimension decorator, use the `postDecorate` initialization parameter to specify the post-decorator that SESM invokes after the decorator.
5. In the `web.xml` declaration of the `ShapeDecorator` servlet, use the `dimensions` and `postDecorate` initialization parameters to configure the new dimension decorator. See the [ShapeDecorator Initialization Parameters, page 3-19](#).
6. Optionally, in the `web.xml` file, use the `defaultValue` initialization parameter to define a default value for the dimension decorator, the post-decorator, or both. See the [Default Dimension Decorator Values, page 3-21](#).

Modifying SESM Web Portal Application Functionality

The preprogrammed functionality of a SESM Web Portal application can be modified in a number of ways including:

- [Modifying the Functionality of JSP-Page Views, page 3-30](#)
- [Creating JSP-Page Decorators and Post-Decorators, page 3-31](#)
- [Using the SESM Deployment Descriptor File, page 3-33](#)

This discussion of functionality changes focuses on the components and `web.xml` file for the NWSP Web Portal application. Before reading this section, become generally familiar with the parts of the NWSP Web Portal application by reading the [NWSP User Interface, page 2-2](#).

Modifying the Functionality of JSP-Page Views

The JSP-page views in the NWSP Web Portal application are preprogrammed with the functionality that most SESM Web Portals require. Most deployments will use the NWSP pages and possibly make minor changes to the functionality.



Note

All changes to the functionality of the JSP-page views in NWSP must be planned and implemented with care. The NWSP controls and views work together in a coordinated manner. Changing one piece of the Web Portal application may affect others pieces.

In NWSP, most JSP-page views have a number of functional elements. In some cases, the service-provider developer can modify the functionality of an element. In other cases, modifications are not needed. This section provides some general guidance on whether you can modify a functional element and the types of changes that you might accomplish.

Service List

The service list usually requires no functional modifications. The service list is a tree structure with the services and service groups that the subscriber can select. The service list is created dynamically by the JSP based on the service and subscriber information stored in the data repository. For information on the service list, see the [Main Template, page 4-19](#).

Navigation Bar

The navigation bar sometimes requires functional modification. The navigation bar consists of a set of buttons, such as the Services and Accounts buttons, whose display changes based on user actions.

In many deployments, the NWSP navigation-bar functionality requires no changes. Various SESM features (such as service subscription and account management) and the associated navigation-bar buttons require that subscribers have appropriate permissions. The set of buttons that appear in the NWSP navigation bar automatically varies according to the user's permissions.

In some deployments, you might want to add a button to or remove a button from the standard NWSP navigation bar. For example, in some deployments the Help button may not be needed.

- To add a button, you create a new navigation bar. If you want the same functionality as the NWSP navigation bar, you must use Dreamweaver and the Cisco Navigation Bar extension. For information on creating a navigation bar, see [Appendix D, "Using the Cisco Navigation Bar Extension."](#)
- To remove a button, you can edit navbar.jsp and delete the lines of HTML that create the button's hyperlink.

Body JSP

The body page of a JSP view sometimes requires functional modifications. Each body JSP contains the functional elements (for example, an HTML form) that appear in the `<body>` section. Subscribers use the functional elements to perform a task that is unique to the JSP. For example, the `subscriptionManageBody.jsp` contains the functional elements that the subscriber uses to subscribe to or unsubscribe from services.

In some cases, you can remove functionality from the body JSP without causing any problems. For instance, you can remove the blocks of HTML code that `statusBody.jsp` uses to display information about the status of each connected service. As an example, you could remove a field like Elapsed Time from the table of data that `statusBody.jsp` displays without creating web-application problems or confusing the subscriber.

In cases where a body JSP uses a form to post data to a NWSP control, you need to determine the effect of removing an input field from the form before making any changes.

- If the input field is not required by the SESM control or model, removing the input field will be harmless. You can make the change.
- If the input field is required by the SESM control or model, removing the input field will not be harmless. You cannot make the change.

Creating JSP-Page Decorators and Post-Decorators

You can create or modify a JSP-page decorator to modify the functionality of a SESM Web Portal application. When you implement a decorator as a JSP, no compiling or packaging is required.

One use for a new JSP-page decorator is as a replacement for an existing servlet decorator. In the `web.xml` file, you keep the same `<servlet-name>` for the decorator but specify the new JSP-page decorator for the `<servlet-class>`.

A second, more typical use for a new JSP-page decorator is as a post-decorator. A post-decorator is a special type of decorator that is invoked after a normal servlet or JSP-page decorator executes. A post-decorator is a mechanism for extending the usual, preprogrammed SESM Web Portal application functionality. In the `web.xml` file, you use the `postDecorate` initialization parameter in a servlet declaration to specify post-decorators. For information on the `postDecorate` parameter, see the [Using the `preDecorate` and `postDecorate` Parameters](#), page C-2.

The `DecoratorByJSP` class makes it possible to implement a decorator with a JSP. A decorator that is a JSP is different from a servlet decorator in a number of ways including:

- It has no `decorateIfNecessary` or `isNecessary` methods. In effect, decoration is always needed and always occurs.
- As with any JSP, it is not a Java class and does not inherit and cannot override any of the methods of the `Navigator` or `Decorator` classes.

For a JSP to be used as a decorator, the JSP must do the following:

1. Provide a definition for the `jspInit` method and register itself with the decorator pool.
2. Define a `decorate` method.
3. Use the `decorate` tag from the `Navigator` tag library to invoke the `decorate` method.

In the NWSP Web Portal application, a good example of a JSP-page decorator that you might create is `httpSniff.jsp`. This post-decorator is invoked after `HttpSniffDecorator` and provides additional *browser sniffing* capabilities for detecting characteristics of the HTTP client device that are not found in `HttpSniffDecorator`.

The following NWSP code from `httpSniff.jsp` shows what is required in a JSP-page decorator. It also shows the categories of tasks a post-decorator might perform.

```
<%@ page import="java.io.IOException" %>
<%@ page import="com.cisco.sesm.logging.Log" %>
<%@ page import="com.cisco.sesm.navigator.HttpSniffBean" %>
<%@ page import="com.cisco.sesm.navigator.DecoratorPool" %>
<%@ page import="com.cisco.sesm.navigator.Navigator" %>

<%@ taglib uri="http://www.cisco.com/taglibs/navigator" prefix="nav" %>
<nav:decorate name="NoCache" />

<!-- (1) Register with the decorator pool -->
<%! public void jspInit() {DecoratorPool.register(this);} %>

<!-- (2) Define the decorate method -->
<%!
public void decorate(HttpServletRequest request,
                    HttpServletResponse response)
throws ServletException, IOException
{
<!-- (3) Retrieve and make adjustments to the httpSniffBean -->
    HttpSniffBean httpSniffBean = (HttpSniffBean)
        request.getSession().getAttribute("httpSniffBean");
    if (httpSniffBean == null)
        throw new ServletException("HttpSniffBean expected.");

    // Sniff the HTTP headers for extra information.
    String userAgent = request.getHeader("User-Agent");
    if ((userAgent != null) &&
        (userAgent.indexOf("OWG1 UP/4.1") >= 0 ||
         userAgent.indexOf("UPG1 UP/4.0") >= 0 ||
         userAgent.indexOf("WinWAP") >= 0))
    {
        httpSniffBean.setClientDeviceName("wap");
    }

<!-- (4) If the client device is JavaScript-enabled, send out a JavascriptProbe -->
        //Does the client accept JavaScript?
        //For an accept with "text/html" assume so for PC browsers.
        //Do not assume so for PDA, as not the case for older browsers.
        String accept = request.getHeader("Accept");
        String device = httpSniffBean.getClientDeviceName();
        if (device != null && device.equals("pc") &&
            accept != null && accept.indexOf("text/html") >= 0)
            httpSniffBean.setClientScriptLanguage("javascript");
        if (accept != null && accept.indexOf("text/javascript") >= 0)
            httpSniffBean.setClientScriptLanguage("javascript");

        if ("MMHttp".equals(userAgent))
        {
            //Macromedia DreamWeaver does not support JavaScript.
            httpSniffBean.setClientScriptLanguage(null);
            httpSniffBean.setClientBrowserName("dreamweaver");
            httpSniffBean.setClientMarkupLanguage("html/3");
        }

        //Should we send a JavaScript probe?
        String script = httpSniffBean.getClientScriptLanguage();
        if (script != null && script.startsWith("javascript"))
            DecoratorPool.get("JavascriptProbe")
                .decorateIfNecessary(request, response);
    }
}
```

```

%>

<%-- This is where this JSP starts running. --%>

<%-- Always capture original URL at start of each request. --%>
<nav:decorate name="OriginalURL" />

<%-- Do not allow Http Client to cache the response. --%>
<nav:decorate name="NoCache" />

<%-- Exit if invoked by DecoratorPool (to force initialization). --%>
<% if (request.getParameter("DecoratorPool") != null) return; %>

<%-- (5) Use the decorate tag to execute the decorate method and any post-decorators. --%>
<nav:decorate name="<%=getServletName()%>" />

```

As shown in the preceding example, when a JSP acts as a post-decorator, it performs the following tasks:

- Provides a definition for the `jspInit` method, the standard JSP initialization mechanism. The `jspInit` definition registers the JSP using the `DecoratorPool.register` method. This step is required.
- Defines the `decorate` method. The definition for the `decorate` method must match the signature of the `Decorator.decorate` method. The signature includes the method name, type, visibility, arguments, and return type. This step is required.
- Performs some deployer-specific post-decoration tasks. These tasks typically include:
 - Retrieving a `JavaBean` that was created by and had properties set by another decorator (in this example, by `HttpSniffDecorator`)
 - Adjusting one or more properties in the `JavaBean` based on additional knowledge that the post-decorator has

In this example, `httpSniff.jsp` retrieves the bean `HttpSniffBean` and adjusts its `clientDeviceName` when it detects that a WAP phone simulator is the client device.

- Performs other deployer-specific post-decoration tasks. In `httpSniff.jsp`, a JavaScript probe is sent to the HTTP client device to detect some of its characteristics. It then makes adjustments to the `httpSniffBean` based on what it is able to detect. The developer could modify `httpSniff.jsp` to use third-party browser-sniffing software rather than the SESM-supplied JavaScript probe.
- Uses the `decorate` tag of the Navigator tag library to invoke `httpSniff.jsp`. The `decorate` tag provides specialized functionality that invokes any pre- and post-decorators that have been declared in the `web.xml` file for `httpSniff.jsp`. If a JSP were to call its `decorate` method directly (rather than using the `decorate` tag), the post-decorators would not be invoked.

Using the SESM Deployment Descriptor File

This section provides information on the how you can use the deployment descriptor file (`web.xml`) to customize the functionality of a SESM Web Portal application.

The `web.xml` file for the NWSP Web Portal application, which resides in the `\install_dir\nwsp\webapp\WEB-INF` directory, provides an example of the elements that you must specify in the deployment descriptor for a SESM Web Portal application. You can use this example `web.xml` file as a template for a file specifically tailored for your deployment.

The `web.xml` contains the standard J2EE elements found in most such files. For detailed information on the standard elements of a deployment descriptor file, see *Java Servlet Specification Version 2.3*.

Configuration Information

The categories of information in a web.xml file are similar for all SESM Web Portal applications. However, the specific elements that are used can differ from one SESM Web Portal application to another. This section outlines the general categories that you need to specify in the web.xml file.

In the following examples, the declarations use the elements in the NWSP Web Portal application's web.xml file.

In a SESM web.xml file, you specify the Web Portal application's elements as follows:

1. Declare each logical control as a servlet using the `<servlet>` tag. For example:

```
<servlet>
  <servlet-name>MyAccount</servlet-name>
  <servlet-class>com.cisco.sesm.webapp.control.MyAccountControl</servlet-class>
  <load-on-startup>1</load-on-startup>
</servlet>
```



Note All correctly coded decorators are automatically registered with DecoratorPool at servlet initialization. All decorators, including the SESM controls, should be loaded at startup to ensure that they are initialized and registered. The `<load-on-startup>` attribute with a value of 1 loads a servlet when the web server starts.

2. Declare each logical view as a servlet using the `<servlet>` tag. For example:

```
<servlet>
  <servlet-name>MyAccountView</servlet-name>
  <servlet-class>com.cisco.sesm.navigator.VirtualFile</servlet-class>
  <load-on-startup>1</load-on-startup>
  <init-param>
    <param-name>vfile</param-name>
    <param-value>/pages/myAccount.jsp</param-value>
  </init-param>
  <init-param>
    <param-name>preDecorate</param-name>
    <param-value>User, NoCache</param-value>
  </init-param>
</servlet>
```



Note

The logical name of the view must have the logical name of the control with *View* added. For example, the logical name for the MyAccount control must be MyAccountView.

3. Map an appropriate URL to each logical control using the `<servlet_mapping>` tag. For example:

```
<!-- servlet mapping for the control -->

<servlet-mapping>
  <servlet-name>MyAccount</servlet-name>
  <url-pattern>/myAccount</url-pattern>
</servlet-mapping>
```

4. Declare each decorator (both servlets and JSPs) and each SESM utility servlet using the `<servlet>` tag. Non-decorator servlets, such as `VirtualFile`, can be loaded on startup for better performance. For example:

```
<servlet>
  <servlet-name>VirtualFile</servlet-name>
  <servlet-class>com.cisco.sesm.navigator.VirtualFile</servlet-class>
  <load-on-startup>1</load-on-startup>
</servlet>
```

5. If a decorator or SESM utility servlet will be requested as part of a URL, map the appropriate URL to each decorator and utility servlet using the `<servlet-mapping>` tag. For example:

```
<servlet-mapping>
  <servlet-name>VirtualFile</servlet-name>
  <url-pattern>/vfile/*</url-pattern>
</servlet-mapping>
```

In addition to the preceding general configuration guidelines, you might also need to specify deployer-specific configuration information in the `web.xml` file. The deployer-specific parameters are, for the most part, servlet initialization parameters that SESM uses when the servlet is invoked. [Appendix C, “SESM Utility Servlets Quick Reference,”](#) lists the initialization parameters that SESM uses.

When testing a SESM Web Portal application, a `web.xml` file can also include declarations and servlet mappings for test decorators such as `TestDimensionDecorator`. For information on using the test decorators, see the [Using Test Decorators, page 2-18](#).

Configuration Techniques

A Cisco SESM Web Portal application’s deployment descriptor file (`web.xml`) provides a number of simple but flexible mechanisms for specifying and invoking a servlet or JSP. The JSPs of a SESM Web Portal application use some of the same techniques to invoke servlets specified in the URL for a link.

To understand the techniques that are used in the SESM deployment descriptor file, read these sections in the order listed:

- [Servlet Mapping, page 3-36](#)
- [Servlet Chaining, page 3-36](#)
- [Mapping a Virtual File Name to an Actual File Name, page 3-37](#)
- [preDecorate Initialization Parameter, page 3-38](#)

In addition, the [SESM Deployment Descriptor File Techniques Example, page 3-38](#) has an example of how these techniques can be combined to accomplish a set of related Web Portal application tasks.

Some of the deployment descriptor file techniques are standard Web Portal application mechanisms; other techniques are unique to a Cisco SESM Web Portal application. You can combine the techniques in a variety of ways to configure Web Portal application functionality and to accomplish Web Portal application tasks, such as finding a shape-specific web resource.

Servlet Mapping

Servlet mapping is a standard technique that is used in many deployment descriptor files, including the web.xml file for NWSP. Servlet mapping allows an HTTP request matching a specific URL pattern to be mapped to a particular servlet. The following two examples of servlet mapping are from the NWSP web.xml file:

```
<!-- Example 1: the pattern /cache/* maps to the servlet named Cache -->
<servlet-mapping>
  <servlet-name>Cache</servlet-name>
  <url-pattern>/cache/*</url-pattern>
</servlet-mapping>

<!-- Example 2: the pattern /myAccount maps to the servlet named MyAccount -->
<servlet-mapping>
  <servlet-name>MyAccount</servlet-name>
  <url-pattern>/myAccount</url-pattern>
</servlet-mapping>
```

The preceding two examples show common ways in which servlet mapping is used in the NWSP web.xml file.

- In example 1, the URL pattern `/cache/*` is mapped to the `Cache` servlet name, which is declared elsewhere in web.xml, to be the `CacheDecorator` servlet class. The `CacheDecorator` class is used to tell the HTTP client to cache the SESM response. The URL pattern allows `CacheDecorator` to be invoked in a servlet chain. For example:

```
/cache/pages/accountLogon.jsp
```

In the preceding servlet chain, calling `CacheDecorator` prior to invoking the JSP given in `/pages/accountLogon.jsp` causes that page to be cached by the client browser. For information on servlet chaining, see the [Servlet Chaining, page 3-36](#).

- In example 2, the URL pattern `/myAccount` is mapped to the `MyAccount` servlet name, which is declared elsewhere in web.xml, to be the `MyAccountControl` servlet class. This use of servlet mapping provides a layer of logical names that can be used in the JSP navigational links and that are independent of the name of the servlet implementation classes.

Servlet Chaining

Servlet chaining is a standard technique that is used in some deployment descriptor files and in the URLs for links in the JSPs. For some HTTP requests, the URL can specify that the Web Portal application invoke a chain of servlets in a particular order rather than just one servlet. For example:

```
/user/nocache/vfile/pages/home.jsp
```

In the preceding URL, three different servlets are invoked, in order, before the JSP given in `/pages/home.jsp` is invoked. In the NWSP web.xml file, three servlet names are mapped to these URL patterns:

- `/user/*`
- `/nocache/*`
- `/vfile/*`

With a servlet chain, the HTTP request is sent to the first servlet in the chain. The output from the last servlet in the chain (home.jsp) creates the response sent back to the browser. With a SESM Web Portal application, servlets in the chain are usually decorators that modify some aspect of the current HTTP request, response, session, or application. Except for the last servlet in the chain, a decorator in a servlet chain terminates by forwarding the request to the remainder of the URL.

For the NWSP Web Portal application, [Table 3-6](#) lists the SESM utility servlets that are sometimes invoked in servlet chains.

Table 3-6 *Servlets Invoked in Servlet Chains*

Decorator	URL Pattern	Description
VirtualFile	/vfile/*	Translates a virtual filename into an actual filename according to the current values for the dimensions of the user shape. The actual filename is a URI for a web resource. As part of the translation process, <code>VirtualFile</code> attempts to find the resource located by the actual URI.
CacheDecorator	/cache/*	Tells the HTTP client to cache the HTTP response.
NoCacheDecorator	/nocache/*	Prevents caching of the HTTP response by the HTTP client.
UserDecorator	/user/*	Ensures that the user is known (authenticated).

For more information on the SESM utility servlets, see the [SESM Utility Servlet Quick Reference, page C-3](#).

Mapping a Virtual File Name to an Actual File Name

When a Cisco SESM Web Portal application employs the user-shape mechanisms for customizing the web resources served to the subscriber, the `VirtualFile` servlet provides important and required functionality. The `VirtualFile` servlet translates a virtual file name into an actual file name according to the current values for the dimensions of the user shape. The actual file name is a URI for a web resource. `VirtualFile` also attempts to find the resource located by the actual file name and, if it finds the resource, forwards the HTTP request to the resource.

A Cisco SESM Web Portal application uses `VirtualFile` when a web resource might differ according to the user shape. For example, if a Cisco SESM Web Portal application uses different language-specific icons for a JSP based on the locale dimension of the user shape, the Web Portal application uses `VirtualFile` to find the correct language-specific icon for each subscriber. For an example of how a Cisco SESM Web Portal application uses `VirtualFile`, see the [SESM Deployment Descriptor File Techniques Example, page 3-38](#).

In the `web.xml` file for NWSP, the URL pattern `/vfile/*` is mapped to the `VirtualFile` servlet.

In a servlet chain, `VirtualFile` (`vfile`) must be located immediately before the virtual filename because `VirtualFile` forwards the request to the remainder of the URL. In the following example, the virtual file name for the web resource is `/pages/help.jsp`, and the `/vfile/*` URL pattern is specified immediately before the virtual filename:

```
/user/nocache/vfile/pages/help.jsp
```

preDecorate Initialization Parameter

The `preDecorate` initialization parameter can be specified in the declaration of `VirtualFile` to invoke a sequence of decorator servlets prior to translating a virtual file name into an actual file name. This technique is used in the NWSP `web.xml` file for the declarations of the logical views to which the SESM controls forward requests. In the following example, the logical view `MyAccountView` is declared using `VirtualFile` and its `preDecorate` parameter:

```
<servlet>
  <servlet-name>MyAccountView</servlet-name>
  <servlet-class>com.cisco.sesm.navigator.VirtualFile</servlet-class>
  <load-on-startup>1</load-on-startup>
  <init-param>
    <param-name>vfile</param-name>
    <param-value>/pages/myAccount.jsp</param-value>
  </init-param>
  <init-param>
    <param-name>preDecorate</param-name>
    <param-value>User, NoCache</param-value>
  </init-param>
</servlet>
```

With the preceding declaration, when the SESM control forwards a request to `MyAccountView`, the `VirtualFile` servlet does the following:

1. Invokes, in sequence, the decorator servlets specified in the `preDecorate` parameter: `User` and `NoCache`. Each servlet in the `preDecorate` list is invoked by calling its `decorateIfNecessary` method.
2. Translates the virtual file name (`/pages/myAccount.jsp`) given in the `vfile` parameter into an actual file name (URI) according to the values for the dimensions of the user shape.

Given the servlet mappings in the NWSP `web.xml` file, using the `preDecorate` parameter in this manner with `VirtualFile` is identical to the following servlet chain:

```
/user/nocache/vfile/pages/myAccount.jsp
```

SESM Deployment Descriptor File Techniques Example

This section uses an example to show how two deployment descriptor file techniques (which were explained in the preceding sections) can be combined to accomplish a set of related Cisco SESM Web Portal application tasks:

- Servlet mapping.
- Mapping a virtual file name to an actual file name.

Using the `web.xml` file for the NWSP Web Portal application, the following example traces the relevant component-to-component flow for this HTTP request:

```
http://someserver:8080/myAccount
```

When the subscriber clicks the My Account button and the web server receives the preceding request, the following occurs if the request is a GET:

1. The MyAccountControl servlet is invoked because, in web.xml, the URL pattern /myAccount maps to the servlet name MyAccount. Also, the servlet name declaration for MyAccount specifies the MyAccountControl servlet class.

```
<!-- Servlet mapping for /myAccount -->
<servlet-mapping>
  <servlet-name>MyAccount</servlet-name>
  <url-pattern>/myAccount</url-pattern>
</servlet-mapping>

<!-- Servlet name declaration for MyAccount -->
<servlet>
  <servlet-name>MyAccount</servlet-name>
  <servlet-class>com.cisco.sesm.webapp.control.MyAccountControl</servlet-class>
  <load-on-startup>1</load-on-startup>
</servlet>
```

2. After it finishes processing the request, MyAccountControl forwards the request to MyAccountView.

```
<servlet>
  <servlet-name>MyAccountView</servlet-name>
  <servlet-class>com.cisco.sesm.navigator.VirtualFile</servlet-class>
  <load-on-startup>1</load-on-startup>
  <init-param>
    <param-name>vfile</param-name>
    <param-value>/pages/myAccount.jsp</param-value>
  </init-param>
</servlet>
```

Notice that, in the web.xml file, the servlet name declaration for MyAccountView specifies the VirtualFile servlet class and the vfile initialization parameter specifies a URI for the view myAccount.jsp.

If the Cisco SESM Web Portal application employs user-shape decoration, VirtualFile plays a very important role. As an example of the VirtualFile functionality, assume that some content on the myAccount.jsp can differ depending on the user shape associated with the subscriber.

When the HTTP request is forwarded to VirtualFile, the servlet translates the virtual file name into an actual file name. An actual file name is a URI that has been expanded to include any additional directory paths that are currently associated with the user shape.

For this example, assume the user shape consists of three dimensions (device, brand, and locale) having, respectively, the values /pda, /gold, and /fr/CA. The VirtualFile servlet expands the virtual file name /pages/myAccount.jsp to the following actual file name:

```
/pda/gold/fr/CA/pages/myAccount.jsp
```

VirtualFile then uses the actual URI to search for the resource `/pages/myAccount.jsp` using the search algorithm described in the [Searches for a Web Resource, page 3-14](#).

- If VirtualFile finds the resource, it forwards the request to the `/pages/myAccount.jsp` located in the appropriate directory.
- If VirtualFile does not find the resource, it throws an exception and displays the following on the client browser:

```
HTTP ERROR: 503 Service Unavailable
javax.servlet.ServletException: Could not find actual file given virtual
file=/pages/myAccount.jsp
RequestURI=/pages/myAccount.jsp
```

If the `/pages/myAccount.jsp` is found by VirtualFile, the web server sends that JSP's content to the HTTP client. The content is displayed on the subscriber's browser.

Invoking a Decorator

This section provides guidance on where and how a decorator should be invoked.

Decorator Invocation Locations

The manner in which a decorator is used determines the location of the decorator invocation. Many decorators are called as predecorators because the processing of the pre-decorator is required before a second decorator can perform its function. Predecorators can be invoked from these locations:

- A servlet declaration in the `web.xml` can define a `preDecorate` parameter.
- The Java code for the SESM control class can call the `addPreDecorator` method.
- A JSP-page view can use the custom JSP tag `<nav:decorate name=servletName/>`.

All three invocation locations are equivalent. Calling a decorator more than once is slightly inefficient but harmless.

SESM follows these guidelines to determine where to invoke a decorator:

- If a SESM control knows that the decorator must run, the control adds the decorator to its list of pre-decorators using the `addPreDecorator` method. This location is required when the control would fail if the decoration does not take place.
- If a control or view does not require a decorator to run, the decorator should be specified in the `preDecorate` parameter. This parameter is specified in the `<servlet>` declaration of the control or view where the deployer can easily change the configuration.

For example, in NWSP, the `StatusControl` and the `Status` view JSP s might be capable of displaying the `Status` page with or without an authenticated user. If the deployment allows an unauthenticated user to display the `Status` page, the control and view would not invoke the `UserDecorator` servlet, which ensures that the subscriber is authenticated. However, if the deployment allows only an authenticated user to display the `Status` page, `UserDecorator` should be invoked through the `preDecorate` parameter in the `<servlet>` declaration for the view.

Decorator Invocation Methods

A SESM Web Portal application can invoke a decorator in a number of different ways. In general, invoking a decorator as a servlet is more expensive than other invocation methods. A SESM decorator can be configured so that, when it is called, it uses pre-decorators and post-decorators to extend or modify the functionality. For the examples that follow, the web.xml file entry for the UserDecorator servlet is as follows:

```
<servlet>
  <servlet-name>User</servlet-name>
  <servlet-class>
    com.cisco.sesm.webapp.decorator.UserDecorator
  </servlet-class>
  <load-on-startup>1</load-on-startup>
  <init-param>
    <param-name>preDecorate</param-name>
    <param-value>Decorator1, Decorator2</param-value>
  </init-param>
  <init-param>
    <param-name>postDecorate</param-name>
    <param-value>Decorator3, Decorator4</param-value>
  </init-param>
</servlet>
```

When a JSP invokes a decorator, such as UserDecorator, in one of the following ways, any decorators in the preDecorate and postDecorate lists are called if the decorator being declared in <servlet-class> (in this example, UserDecorator) requires decoration.

- Invoking the decorator as a servlet (for example, through an HTTP request, or by forwarding to the decorator or including it in a JSP).
- Using the decorate tag of the Navigator tag library. For example:

```
<nav:decorate name="User" />
```

- Directly invoking the decorateIfNecessary method. For example:

```
DecoratorPool.get("User").decorateIfNecessary(request, response);
```

In all cases, you invoke the pre-decorators and post-decorators by calling the decorateIfNecessary method of each.

**Note**

When a decorator is invoked directly by calling its decorate method, the decorators in the preDecorate and postDecorate lists are not called.

For information on the preDecorate and postDecorate initialization parameters, see the [Using the preDecorate and postDecorate Parameters, page C-2](#).



SESM Web Portal Applications

This chapter provides information on the SESM Web Portal applications and the Captive Portal solution and describes how you can use and modify the web components. The Cisco SESM software includes the following Web Portal applications and solutions that are currently available for use as a development base:

- [NWSP Web Portal Application, page 4-2](#)
- [PDA Web Portal Application, page 4-21](#)
- [WAP Web Portal Application, page 4-23](#)
- [Captive Portal Web Solution, page 4-24](#)

The New World Service Provider (NWSP) Web Portal application is a full-featured example of the technology that SESM provides. This chapter provides detailed information on NWSP and its components.

The PDA and WAP Web Portal applications and the Captive Portal solution are designed for specific purposes and illustrate a subset of SESM web application technology. This chapter provides overview information on the PDA and WAP applications and Captive Portal.

General Considerations for SESM Web Applications

The following general considerations apply to this chapter's descriptions of the SESM Web Portal applications and solutions.

- Before you read this chapter, read [Chapter 2, “Basic SESM Customization and Development”](#) and [Chapter 3, “Advanced SESM Customization”](#) for an explanation of SESM components and techniques that are, in general, used in all SESM Web Portal applications.
- Depending on the SESM software that is used, you can configure a deployed SESM Web Portal application for one of two installations:
 - SESM RADIUS installation—Service and subscriber information is stored in a RADIUS server.
 - SESM SPE installation—Service, subscriber, and policy information is stored in an LDAP-compliant directory, which you access with the SPE application programming interfaces. (APIs)
- The set of web components used for a SESM Web Portal application varies according to the configuration (SESM RADIUS installation or SESM SPE installation). Where it is required, the descriptions of components in this chapter indicate the installation in which each component is used.

NWSP Web Portal Application

The New World Service Provider (NWSP) Web Portal application contains all of the components required for a fully functional Web Portal for network services. You can use the NWSP web components as a starting point for designing and creating a SESM Web Portal application. This section provides information on the NWSP Web Portal application and describes how you can use and modify the NWSP web components.

NWSP User Interface

For information on the NWSP user interface, see the [Basic SESM Customization and Development, page 2-1](#).

NWSP Customization Based on Subscriber Characteristics

You can customize the look and feel of a SESM user interface for each subscriber. For information on customization and localization, see [Chapter 3, “Advanced SESM Customization,”](#) and [Chapter 5, “SESM Internationalization and Localization.”](#)

NWSP Functionality

The NWSP Web Portal application provides a set of functions that is typical of many directory-enabled SESM applications. The subscriber logs on to the Web Portal with a username, password, and for 3-key authentication, a telephone number. 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 a SESM Web Portal application that is operating as a SESM SPE installation. The NWSP Web Portal application includes the required logic to determine the permissions that were granted to a subscriber and to generate the appropriate content. For example, if a subscriber has the required permissions to create subaccounts, the NWSP Web Portal 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 that implements the functionality for the specific task or set of tasks. As an example, the My Account button is linked to myAccount.jsp, which generates the content for the account management page ([Figure 4-1](#)).

Figure 4-1 Account Management Page

NWSP
NEW WORLD SERVICE PROVIDER

CISCO SYSTEMS

HOME | MY ACCOUNT | MY SERVICES | MY FIREWALL | SUB-ACCOUNTS | STATUS | MESSAGES | SETTINGS | HELP

My Account Details

Current Services

- Gold Internet
- Corporate Intranet
- Games!
- Discount Shopping
- Banking
- News
 - CNN News
 - BBC News
 - Economist News

LOG OUT

First Name Middle Initial Last Name

Street City Country

Postal Code State

Home Phone Mobile Phone Pager

Fax Email Home URL

Date of Birth (Pattern is "M/d/yy", for example "9/19/02")

Gender Male Female Single Sign-On Yes No

Interests Cinema Science Internet News Sports Travel Finance Community

OK Cancel Reset Change Password

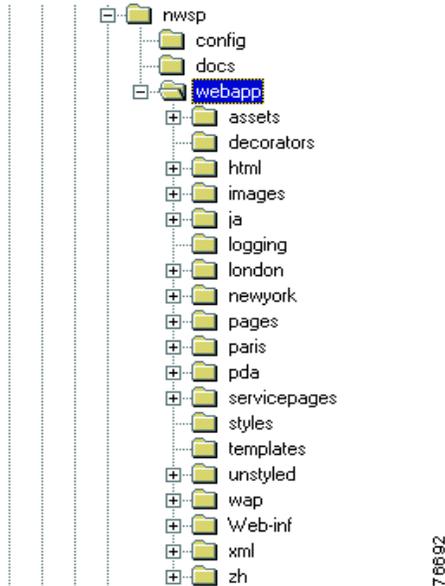
76687

NWSP Directory Hierarchy

After you install the Cisco SESM software, the NWSP Web Portal application is located in a structured hierarchy of directories. As with any Web Portal 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 Portal 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 the Cisco SESM software is installed, the NWSP Web Portal application's hierarchy of directories is located below the `\install_dir\nwsp` directory (Figure 4-2).

Figure 4-2 NWSP Directories



The NWSP directories contain the complete set of files required for the Web Portal application and include the Portable Networks Graphics (PNG) source files required to customize the Fireworks images and buttons. The NWSP directories and files are as follows.

The following directories and their files are used for application development and are not required in a deployed SESM Web Portal application: `\webapp\assets`, `\webapp\templates`, and `\docs`.

config

Contains the Web Portal application configuration file `nwsp.xml`. For information on the configuration files, see *Cisco Subscriber Edge Services Manager Web Portal Guide*.

docs

Contains the Javadoc files for the NWSP classes. For information on the Javadoc files, see the [Javadoc Documentation, page 2-10](#).

webapp

Is the Web Portal application's document base.

webapp\assets

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

webapp\decorator

Contains JSPs used for decorating the user's shape. For information on user-shape decoration, see the [Decorating a User Shape, page 3-18](#).

webapp\html

Contains files for specific versions of HTML: 3.0 and 4.0, such as a set of service-list files if the client browser supports HTML 4.

webapp\images

Contains the GIF and JPEG images for the NWSP user interface. The `\webapp\images\serviceList` directory contains the icons for service status that appear in the service list.

webapp\ja

Contains the images and navbar.jsp for the NWSP user interface when the locale dimension is ja (Japanese).

webapp\logging

Contains two files that are used when logging entries are made from a JSP include file. For information on these files, see the [JSPs and JSPFragment Class for Include File Log Entries](#), page 4-18.

webapp\london

Contains an image used to demonstrate location-based branding functionality in NWSP.

webapp\newyork

Contains an image used to demonstrate location-based branding functionality in NWSP.

webapp\pages

Contains the JSPs that contain the web-application content. The \l10n directory contains JSPs that provide information on the current localization context and the locales and timezones supported by the JRE. For information on the JSPs used in NWSP, see the [NWSP JavaServer Pages and Servlets](#), page 4-6.

webapp\paris

Contains an image used to demonstrate location-based branding functionality in NWSP.

webapp\pda

Contains the files that are used when the subscriber's device is a PDA.

webapp\servicepages

Contains files and images for some simulated "services" that the NWSP Web Portal application can use in Demo installation.

webapp\styles

Contains the cascading style sheets sesm.css and serviceList.css.

webapp\templates

Contains the Dreamweaver templates. For information on the NWSP templates, see the [NWSP Templates](#), page 4-19.

webapp\unstyled\pages

Contains files that are currently not used.

webapp\wap

Contains files that are used when the subscriber's device is a WAP phone.

webapp\WEB-INF

Contains various Java-related NWSP components:

- Tag library descriptor (.tld) files for the SESM tag libraries are in \WEB-INF. For information on the .tld files and using a tag library, see the [Configuring a Tag Library](#), page B-1.
- The Web Portal application's deployment descriptor file, web.xml, is in \WEB-INF. For information on this file, see the [Using the SESM Deployment Descriptor File](#), page 3-33.

- The `\lib` directory contains the Java archive (JAR) files for the classes used by the SESM Web Portal application. For information on the JAR files required for a SESM Web Portal application, see the [SESM Class Libraries and Tag Library Descriptor Files, page 2-8](#).
- The `\classes` directory contains the NWSP properties files. For information on localization and properties files, see [Chapter 5, “SESM Internationalization and Localization.”](#)

webapp\xml\pages

Contains files that are used for an XML-based view for the service list control.

webapp\zh

Contains the images for the NWSP user interface when the `locale` dimension is `zh` (Chinese).

NWSP JavaServer Pages and Servlets

The NWSP Web Portal application includes a set of JSP s 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 s contain the elements that you modify for the specific requirements of the service provider. No servlet programming is required.

This section provides guidance on the JSPs that generate the content for NWSP. For information on the architecture of a SESM Web Portal application and the servlets and decorator JSPs used in NWSP, see [Chapter 3, “Advanced SESM Customization.”](#)

After the subscriber logs on to the NWSP user interface, `home.jsp` is the home page for the Web Portal application. From the home page, the subscriber can control service subscription and selection and, in a SESM SPE installation, can perform account-management, self-subscription, subaccount-creation, and personal firewall functions. The navigation bar at the top of the `home.jsp` links the subscriber to these capabilities.

Modularized JSPs

In NWSP, many JSPs that generate content are modularized into two pieces: a wrapper JSP and a body JSP.

Wrapper JSPs

Each *wrapper JSP* contains standard functionality found on many JSPs such as:

- JavaScript that must appear in the `<head>` section.
- The NWSP banner.
- The navigation bar.
- The service list.

For example, `subscriptionManage.jsp` is a wrapper JSP containing JavaScript in its `<head>` section, the NWSP banner, the navigation bar, and the service list.

Most content for the wrapper JSPs comes from the Dreamweaver template `mainTemplate.dwt`. If you are not using Dreamweaver as your HTML editor or your SESM Web Portal application does not use templates, having common elements like the JavaScript and banner isolated in a separate wrapper file should make it easier to modify these elements.

Body JSPs

Each body JSP contains the functional elements (for example, an HTML form) that appear in the <body> section. The subscriber uses the functional elements to perform a task that is unique to the JSP. Each body JSP is included into the corresponding wrapper JSP. The subscriptionManageBody.jsp, which is included into the body of the wrapper page subscriptionManage.jsp, contains the functional elements that the subscriber uses to subscribe or unsubscribe from services.

User-Interface Control JSPs

In NWSP, the user-interface controls for submitting, canceling, and resetting a form are also modularized in their own JSPs. The JSP for a button is included into an HTML form on the JSPs where that user-interface control is needed. [Table 4-1](#) lists the JSPs that are used for user-interface controls.

Table 4-1 NWSP JSPs for User-Interface Controls

JSP	Use of Control
cancelButton.jsp	Canceling an action. Links to the URL specified in the request parameter <i>url</i> .
deleteButton.jsp	Deleting an item.
editButton.jsp	Editing an item.
homeButton.jsp	Returning to home. Links to the URL /home.
newButton.jsp	Creating a new item.
returnButton.jsp	Returning to the previous location.
resetButton.jsp	Resetting the enclosing form.
submitButton.jsp	Submitting the enclosing form.

The button can be a Submit button, Reset button, or Graphical submit button using the specified image. The type of the button depends on the value of the `type` attribute. For example, in `submitButton.jsp`, the `type` attribute specifies a button that users can click to submit the form's contents to the web server:

```
<input type="submit" value="<l10n:resource key='OK'"/>">
```

As shown in the preceding example, the label (the value attribute) on the submit button can be localized through the use of the resource tag and NWSP resource bundle property files. For the submit button, the key attribute of the resource tag specifies the key for which to obtain a resource from the resource bundle. For information on the use of resource bundles and the resource tag, see [Chapter 5, “SESM Internationalization and Localization,”](#) and the “resource Tag” section on page 5-12.

JSPs for Basic SESM Web Portal Application Functions

[Table 4-2](#) lists JSPs for the basic SESM Web Portal application functions that do not require a data repository that can be modified (for example, that do not require an SPE directory). The basic functions are available in both SESM RADIUS installations and SESM SPE installations. These basic functions include account logon and logoff, service selection, service status display, message display, and locale setting. These JSPs are located in the `\nwsp\webapp\pages` directory.

Table 4-2 NWSP JSPs for Basic SESM Web Portal Application Functions

JSP	Description
accountLogoff.jsp	After the subscriber clicks the Log Out button, asks the subscriber to confirm or cancel session termination.
accountLogon.jsp accountLogonBody.jsp	Allows a subscriber to log on to the NWSP Web Portal application. The subscriber's username and password are authenticated by the SESM software.
accountLogon3Key.jsp accountLogon3KeyBody.jsp	Allows a subscriber to log on to the NWSP Web Portal application when 3-key authentication is used. When 3-key authentication is used, the subscriber credentials include a username, password, and telephone number.
confirmBody.jsp	Displays a message and asks the subscriber to confirm or cancel a specific action by clicking the OK or Cancel button.
help.jsp helpBody.jsp	Displays help information. The help information is for the demonstration installation NWSP.
home.jsp	Displays the NWSP home page. See Figure 2-1 on page 2-3 .
locale.jsp localeBody.jsp	Displays locale settings from which the subscriber can choose.
messages.jsp messagesBody.jsp	Displays messages associated with the current session.
messagesText.jsp	Displays one or more messages from a <code>MessagesBean</code> .
navbar.jsp	Contains the NWSP navigation bar. For more information, see the JSP for the Navigation Bar, page 4-17 .
serviceList.jsp serviceListGroup.jsp serviceListService.jsp	Displays the service list, a tree of subscribed services and service groups. For more information, see the JSP for the Navigation Bar, page 4-17 .
serviceLogon.jsp serviceLogonBody.jsp	Allows a subscriber to enter a username and password when logging on to a service.
serviceStart.jsp serviceStop.jsp	Asks whether the subscriber wants to start a service or stop a service.
status.jsp statusBody.jsp	Displays the status of each connected service: <ul style="list-style-type: none"> • Username. • Service description from the service profile. • Service status. • Connected as—the username specified for a service that requires authentication. • Elapsed connect time. • Packets sent.

**Note**

When Web Portal applications are installed, they are configured to use the key store for test purposes. These key store files have expiration dates and may expire. However, you can generate and self-sign new certificates for test purposes. See the *Cisco Subscriber Edge Services Manager Administration and Configuration Guide* for details. These files are not to be used in a production environment.

Standard and Secure Mode

The `accountLogonBody.jsp` and `accountLogon3KeyBody.jsps` include Standard | Secure hyperlinks below the Log In button. These links enable users to choose either standard mode or secure mode. On the logon page that `accountLogon.jsp` or `accountLogon3KeyBody.jsp` displays, if the subscriber is using secure mode, only the standard-mode link is available. Similarly, if the subscriber is using standard mode, only the secure mode link is available.

When the subscriber logs on using secure mode, the SESM Web Portal 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 Portal 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 for the password.

If the service provider does not require SSL or does not have a certificate, remove the Standard | Secure elements in the `accountLogonBody.jsp` and `accountLogon3KeyBody.jsps`. In addition, you must remove 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 *Cisco Subscriber Edge Services Manager Web Portal Guide*.

JSPs for SPE Installation Functions

[Table 4-3](#) lists JSPs for the SPE installation SESM Web Portal application functions that do require a data repository (an LDAP directory) holding subscriber information that can be modified. These functions require that the SESM Web Portal application be deployed as an SPE installation. In addition to all basic SESM Web Portal application functions ([Table 4-2](#)), the SESM SPE installation functions include service subscription, account management (subscriber self-care), subaccount creation, and personal firewall protection. These JSPs are located in the `\nwsp\webapp\pages` directory.

Table 4-3 NWSP JSPs for SPE Installation Functions

JSP	Description
<code>accountPassword.jsp</code> <code>accountPasswordBody.jsp</code>	Displays a form for changing a password.
<code>advFirewall.jsp</code> <code>advFirewallBody.jsp</code>	Displays a form that allows the subscriber to create or modify Cisco IOS access control list (ACL) entries for a personal firewall.
<code>myAccount.jsp</code> <code>myAccountBody.jsp</code>	Displays a form that allows a subscriber to change account information such as an address, telephone number, email address, and so on.
<code>firewall.jsp</code> <code>firewallBody.jsp</code>	Displays a form that allows the subscriber to specify user-friendly definitions for a personal firewall.
<code>subaccountConfirm.jsp</code> <code>subaccountConfirmBody.jsp</code>	Displays a page that asks the subscriber to confirm changes to subaccounts specified in <code>subaccountSubscriptions.jsp</code> .

Table 4-3 NWSP JSPs for SPE Installation Functions (continued)

JSP	Description
subAccountList.jsp subAccountListBody.jsp	Displays information about a subaccount that is selected from a list of existing subaccounts. Allows creation of new subaccounts and modification of the services associated with subaccounts.
subaccountSubscriptions.jsp subaccountSubscriptionBody.jsp	Displays a form that allows a subscriber to modify subaccount details, such as subscribed and blocked services and, if applicable, the username and password for a service.
subscriptionConfirm.jsp subscriptionConfirmBody.jsp	Displays a page that asks the subscriber to confirm changes to the subscriptions specified in subscriptionManage.jsp.
subscriptionManage.jsp subscriptionManageBody.jsp	Displays a form that allows a subscriber to subscribe to or unsubscribe from services, and to modify attributes associated with subscriptions. For example, a subscriber can specify whether the service is an auto-connect service, or can define the username and password to access a service.

In SESM SPE installations, the permissions defined for each subscriber determine whether the subscriber can subscribe to services, manage account details, and create subaccounts. The NWSP Web Portal application obtains the subscriber's permissions using the `JavaBean permissionBean` and then generates the NWSP user interface based on the permissions. For example, if the subscriber does not have the needed permissions to create subaccounts, the NWSP Web Portal application has the needed logic to omit the Accounts button from the navigation bar.

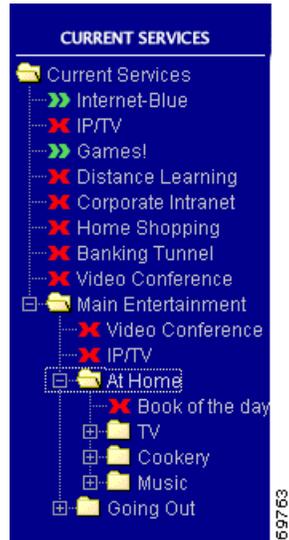
JSPs for the Service List

The *service list* (Figure 4-3) is a tree structure with the subscribed services and service groups from which the subscriber can select. The service list is dynamically created by the JSPs based on the service and subscriber information stored in the data repository. In SPE installations, the subscriber can use a SESM Web Portal 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*.

In NWSP, there are two implementations of the service list: one is used for browsers that do not support HTML 4 and the other is for browsers that do support HTML 4. For HTML 4, the service list is implemented with JavaScript (`serviceList.js`).

- The `serviceListService.jsp` for browsers that do *not* support HTML 4 is located in the `\nwsp\webapp\pages` directory.
- The `serviceListService.jsp` for browsers that support HTML 4 is located in the `\nwsp\webapp\html\3\4\pages` directory.

With NWSP, the SESM software automatically detects whether the subscriber's browser supports HTML 4 and serves the JSPs with the appropriate version of the service list.

Figure 4-3 Service List with Service-Status Icons and Text

Service-Status Icons. For each service in the service list, a service-status icon indicates the state of the service. Table 4-4 lists the service-status icons that are used. The files for the images used in the service list are located in the \nwsp\webapp\images\serviceList directory.

Table 4-4 Service-Status Icons

Icon (color)	Description	File
 (green)	Indicates that the subscriber is connected to the service.	serviceOn.gif
 (red)	Indicates that the subscriber is not connected to the service.	serviceOff.gif
 (red and green)	Indicates that the service connection was lost.	serviceLost.gif
 (blue)	Indicates a service for which a connection is not required.	servicenotRequired.gif

Services in the Service List

For each subscribed service, the service list displays these elements:

- Service page URL.
- Service-status icon.
- Alternative text.
- Text for the service.

As an example of how a SESM Web Portal application determines the preceding elements, consider this code from `/webapp/pages/serviceListService.jsp`.

```
<%@ taglib uri="http://www.cisco.com/taglibs/localization" prefix="l10n" %>
...
<jsp:useBean id="serviceBean" class="com.cisco.sesm.webapp.control.ServiceListServiceBean"
scope="request" />
...
<a href="/service<%=serviceBean.getAction()%>/home?service=<%=serviceBean.getName()%>">
' />"
alt="<l10n:format object='<%= serviceBean.getActionDesc() %>' />" border="0">
<l10n:format object='<%= serviceBean.getDescription() %>'>description</l10n:format>
</a>
```



Note

The preceding service-list code is used for browsers that do not support HTML 4. For HTML 4, the service list is implemented with JavaScript (`serviceList.js`). The `serviceListService.jsp` for HTML 4 constructs the service list elements in a manner similar to the manner described in this section. The `serviceListService.jsp` for HTML 4 is located in the `/webapp/html/3/4/pages` directory.

The service-related information used by `serviceListService.jsp` comes from the properties of the view bean `serviceBean`, which is an instance of the `ServiceListServiceBean` class. The following sections explain how the view uses these properties. For more information on the bean properties, see the Javadoc description of the `ServiceListServiceBean` class.

Service-Page URL. In `serviceListService.jsp`, the service page is the page that the service-status icon links to. The URL to the service page is defined as follows:

```
href="/service<%=serviceBean.getAction()%>/home<%=cookie%>?service=
<%=serviceBean.getName()%>">
```

A unique `serviceBean` request attribute exists for each service. The `serviceBean.getAction` method returns the subscriber's action. The action that clicking the link will perform depends on the service status.

- If the status of the service is On, the action is Stop.
- If the status of the service is Off or Lost (session lost), the action is Start.

The `serviceBean.getName` method obtains the service name, as defined in the service profile. When the subscriber's action and the service name are combined, the result is one of the following service-page URLs:

```
"/serviceStop?service=service_name"
"/serviceStart?service=service_name"
```

In the `web.xml` file for NWSP, the URLs for `/serviceStop` and `/serviceStart` are mapped, respectively, to the `ServiceStopControl` and `ServiceStartControl` servlets, which stop and start the service (`service_name`) specified in the URL's query string.

Service-Status Icon. The file name for the service-status icon (Table 4-4) is constructed from the current status of the service:

```
<img src=
"<shape:file name='<%= "/images/serviceList/service"+serviceBean.getStatus()+".gif"%>' />"
```

The `serviceBean.getStatus` method returns the current status of the service: On, Off, or Lost. When the status is added to the directory and file information, the result is one of the following URLs:

```
"/images/serviceList/serviceOn.gif"
"/images/serviceList/serviceOff.gif"
"/images/serviceList/serviceLost.gif"
```

The `file` tag from the `Shape` tag library gets the actual URI of the GIF file from the virtual file name given in its `name` attribute. For example, if the user shape included a brand dimension with a value of `nwsp`, the actual URI for `serviceOn.gif` might be:

```
"/nwsp/images/serviceList/serviceOn.gif"
```

For information on the `file` tag, see the [file Tag, page B-5](#).

Alternative Text. A text alternative to the image is obtained with the `serviceBean.getActionDesc` method:

```
alt="<l10n:format object='<%= serviceBean.getActionDesc() %>' />"
```

The `serviceBean.getActionDesc` method returns the action description for the current status of the service. When the subscriber passes the pointer over the hyperlink for the service, the browser displays the action description.

The action descriptions are internationalized resources of type `IL18NObject` that have entries in the NWSP properties files (for example, `messages.properties`). [Table 4-5](#) shows the keys and values in the NWSP `messages.properties` file for the three possible actions.

Table 4-5 Action Descriptions from messages.properties

Service Action	Action Description key	Action Description value (English)
Stop	<code>serviceStopDesc</code>	Stop
Start	<code>serviceStartDesc</code>	Start

The value for `alt` uses the `format` tag and its `object` attribute from the `Localization` tag library to localize the action description. The localization extracts a value for the action description's key from a shape-specific NWSP properties file.

Text for the Service. The text that is part of the link for the service is obtained with the `serviceBean.getDescription` method:

```
<jsp:useBean id="serviceBean" class="com.cisco.sesm.webapp.control.ServiceListServiceBean"
  scope="request" />
...
<l10n:format object='<%= serviceBean.getDescription() %>'>description</l10n:format>
```

The `serviceBean.getDescription` method returns the service description. The description of the service comes from the first of the following that the SESM software finds:

- The service description in the resource bundle.
- The service description in the service profile.
- The service name in the service profile.

If the service description is defined in a resource bundle, the key has the form:

```
service_nameDescription
```

In the preceding key, `service_name` is the service name as defined in the service profile. For more information on service descriptions in a resource bundle, see the `messages.properties` file, which is located in the `\install_dir\nwsp\webapp\WEB-INF\classes` directory.

As shown in the preceding code, the JSP uses the localization tag library's `format` tag and its `object` attribute to localize the string that `getDescription` returns. If the service description is from a resource bundle, the localization extracts a value for the key associated with the service description from a shape-specific NWSP properties file. For information on the use of the `format` tag, see the [format Tag, page 5-11](#).

Services Groups in the Service List

A *service group* is a set of one or more services. The set of services in a service group is defined in the service-group profile. In SPE installations, the subscriber can use a SESM Web Portal application to add or remove service groups from the set of subscribed groups that are displayed in the service list. The three types of service groups are:

- *Standard service group*—The subscriber can subscribe and connect to one or more services in the group.
- *Mutually exclusive subscription group*—The subscriber can subscribe to only one of the group's services at a time.
- *Mutually exclusive connection group*—The subscriber can subscribe to one or more services in the group but can connect to only one of the group's services at a time.

When the subscriber attempts to subscribe or connect to a service in a mutually exclusive subscription group or mutually exclusive connection group, the SESM software controls subscriptions and connections based on the specifications in the user, user group, service, and service groups profiles.

Service-Group Icons. For each service group in the service list, a service-group icon is used to represent the service group. [Table 4-6](#) lists the service-group icons that are used. For each folder image in the table, there is also an expanded folder image that indicates that the subscriber has clicked on the folder to display the set of services in the group. For example, `folder.gif` has an expanded folder image `folderopen.gif`. The files for the images used in the service list are located in the `\nwsp\webapp\images\serviceList` directory.

Table 4-6 Service-Group Icons

Icon	Description	File
	Indicates a standard service group.	<code>folder.gif</code>
	Indicates a mutually exclusive connection group.	<code>mutexConnectfolder.gif</code>
	Indicates a mutually exclusive subscription group.	<code>mutexSubscribefolder.gif</code>

Services in the Service List

For each subscribed service group, the service list displays these elements:

- Service-group icon.
- Text for the service.

As an example of how a SESM Web Portal application determines the preceding elements, consider this code from `/webapp/pages/serviceListGroup.jsp`.

**Note**

The following service-group code is used for browsers that do not support HTML 4. For HTML 4, the service list is implemented with JavaScript (serviceList.js). The serviceListGroup.jsp for HTML 4 constructs the service group elements in a manner similar to the manner described in this section. The serviceListService.jsp for HTML 4 is located in the /webapp/html/3/4/pages directory.

```
<jsp:useBean id="groupBean"
class="com.cisco.sesm.webapp.control.ServiceListServiceGroupBean" scope="request" />

<table border="0" cellspacing="0" cellpadding="0">
<tr>
<!-- (1) Display the icon for the service group -->
<td rowspan="999" valign="top">
    </td>

<!-- (2) Display the text for the service group -->
<td align="left" nowrap class="ServiceDescription"><l10n:format object=
    '<%=groupBean.getDescription() %>'>group description</l10n:format></td><tr>

<!-- (3) Iterate through the group's set of services -->
% Iterator serviceIter = groupBean.getServices().iterator(); %>
<% while (serviceIter.hasNext()) { %>
<% request.setAttribute("serviceBean", serviceIter.next()); %>
<tr><td>
<jsp:include page="/vfile/pages/serviceListService.jsp" flush="true"/>
</td></tr>
<% } %>

<!-- (4) Iterate through the group's set of (nested) service groups -->
<% Iterator groupIter = groupBean.getServiceGroups().iterator(); %>
<% while (groupIter.hasNext()) { %>
<% request.setAttribute("groupBean", groupIter.next()); %>
<tr><td>
<jsp:include page="/vfile/pages/serviceListGroup.jsp" flush="true"/>
</td></tr>
<% } %>
</table>
```

The service-group information used by serviceListGroup.jsp comes from properties of the view bean groupBean, which is an instance of the ServiceListServiceGroupBean class.

ServiceListServiceGroupBean class encapsulates information about a service group including:

- Description of the group.
- Type of the group (for example, mutually exclusive connection group).
- Services in the group.
- Service groups in the group.

As shown in the preceding code, the view serviceListGroup.jsp performs the following tasks:

1. Displays the icon for the service group. The file name for the image is constructed using the type of the service group retrieved from the groupBean.
2. Displays the text for the service group, which appears with the icon. The text comes from the description of the group retrieved from the groupBean.

3. Iterates through the group's set of services. The list of services is retrieved using the `getServices` method of `groupBean`. For information on the `getServices` method, see the Javadoc description of the `ServiceListServiceGroupBean` class.
4. Iterates through the group's set of service groups. In addition to services, a service group can contain nested service groups. The list of service groups is retrieved using the `getServiceGroups` method of `groupBean`. For information on the `getServiceGroups` method, see the Javadoc description of the `ServiceListServiceGroupBean` class.

The following sections provide more information on how the view constructs the file name for the service-group icon and the text for the service group that appears with the icon.

Service-Group Icon. The file name for the service-group icon (Table 4-6) is constructed from the type of the service group:

```
<td rowspan="999" valign="top">
  </td>
```

The `groupBean.getType` method returns a string with the type of the service group:

- an empty string (" ") for a standard service group.
- "mutexSubscribe" for a mutually exclusive subscription group.
- "mutexConnect" for a mutually exclusive connection group.

When the type is added to the directory and file information, the result is one of the following URLs:

```
"/images/serviceList/folderopen.gif"
"/images/serviceList/mutexSubscribefolderopen.gif"
"/images/serviceList/mutexConnectfolderopen.gif"
```

In the `web.xml` file, the `/cache` part of the URL maps to the `CacheDecorator` servlet, which tells the HTTP client to cache the response (the GIF file).

In the `web.xml` file, the `/vfile` part of the URL maps to the `VirtualFile` servlet. This servlet uses the values of the dimensions of the user shape to translate the virtual file name given in the remainder of the URL into an actual URI for the GIF file. The remainder of the URL could be, for example,

```
/images/serviceList/folderopen.gif.
```

For information on `CacheDecorator` and `VirtualFile` servlets, see [Appendix C, "SESM Utility Servlets Quick Reference."](#)

Text for the Service Group. The text that is part of the link for the service is obtained with the `serviceGroup.getDescription` method:

```
<td align="left" nowrap class="ServiceDescription"><l10n:format object=
  '<%=groupBean.getDescription() %>'>group description</l10n:format></td><tr>
```

The `serviceGroup.getDescription` method returns the service-group description. The description of the service comes from the first of the following that the SESM software finds:

- The service-group description in the resource bundle.
- The service-group description in the service profile.
- The service-group name in the service profile.

If the service-group description is defined in a resource bundle, the key has the form:

```
service_group_nameDescription
```

For more information on service-group descriptions in a resource bundle, see the `messages.properties` file, which is located in the `\install_dir\nwsp\webapp\WEB-INF\classes` directory.

As shown in the preceding code, the JSP uses the Localization tag library's format tag and its object attribute to localize the string that getDescription returns. For information on the use of the format tag, see the [format Tag, page 5-11](#).

JSP for the Navigation Bar

In the NWSP Web Portal application, the Dreamweaver-created navigation bar ([Figure 4-4](#)) 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. The navbar.jsp in the \nwsp\webapp\pages directory contains the navigation bar.

Figure 4-4 Navigation Bar with Buttons for SPE Installation



[Figure 4-4](#) shows the NWSP navigation bar with the buttons for the dynamic update features: My Account, My Services, and Sub-Accounts. These buttons are supported only when the dynamic update functionality associated with SESM SPE installation is available. The HOME, STATUS, MESSAGES, SETTINGS, and HELP buttons are displayed in both SESM RADIUS and SESM SPE installations.

In SPE installations, the set of buttons that appears in the NWSP navigation bar varies depending on the subscriber's permissions. Various SESM features requires that subscribers have appropriate permissions. For example, the ability to create subaccounts or change account information such as a password requires that the subscriber have the required permissions. The NWSP Web Portal application has the required logic to determine the permissions that were granted to the subscriber and to display the corresponding set of navigation bar buttons.

The NWSP navigation bar was created with the Cisco Navigation Bar extension to Dreamweaver. This extension changes the behavior of the Dreamweaver navigation-bar tool and uses a custom script to provide some special functionality in the NWSP navigation bar. For information on using the Cisco Navigation Bar extension, see [Appendix D, "Using the Cisco Navigation Bar Extension."](#)



Note

Because the NWSP navigation bar has conditional statements that display certain buttons based on subscriber permissions, you cannot use the Dreamweaver navigation-bar tool to modify the NWSP navigation bar. The HTML for the NWSP navigation bar is located in the file `/nwsp/webapp/pages/navbar.jsp`. *If you need to change the NWSP navigation-bar coding, you must modify it manually with an HTML code editor or a text editor.*

Each button in the NWSP navigation bar uses three images for three button states. For example, the Home button uses these images:

- home_button.gif for the up state.
- home_button_over.gif for the over state.
- home_button_down.jsp for the down state.

In a production SESM Web Portal application that uses the NWSP components, you provide any customized or localized images for each button in the navigation bar. In NWSP, GIF and JPEG files for the navigation-bar buttons are located in the `\nwsp\webapp\images` directory.

In the NWSP Web Portal application, the PNG files for the buttons in the navigation bar exist in the `\nwsp\webapp\assets` directory. The easiest way to modify a button is to open the button's PNG file, change the image or its text, export the image to GIF or JPEG, and save the PNG file. You can use Fireworks or any graphics tool to modify the button images.

JSPs for User-Shape Decoration

Another smaller set of JSPs performs some user-shape decoration functions. These JSPs reside in the `\nwsp\webapp\decorator` directory. You can modify or extend these decorator JSPs. For information on the user-shape decoration JSPs, see the [SESM Software Concepts, page 3-8](#) and the [Decorating a User Shape, page 3-18](#).

JSPs and JSPFragment Class for Include File Log Entries

NWSP uses two specialized JSPs, `enterFragment.jspf` and `exitFragment.jspf`, and the `JSPFragment` class to make log file entries from JSP include files more usable.

When one JSP includes another JSP, you will have difficulty determining the JSP include file or fragment that wrote a particular message to the application log file. JSPs are precompiled into one large Java function. When one JSP includes other JSPs, the result is one `_jspService` method that can be thousands of lines long. When you examine a stack trace or debug frame, the log file displays the name of the outermost JSP, not the name of the JSP include file.

The `JSPFragment` class represents a fragment of JSP code (for example, a JSP include file) and provides information about it by logging messages. If your SESM Web Portal application writes logging messages from a JSP include file and you want the specific include file to be identified in the message, the Web Portal application can use the `JSPFragment` class.

The SESM components contain two files that provide the code needed to use the `JSPFragment` class:

- `enterFragment.jspf`
- `exitFragment.jspf`

These two files simplify the use of the `JSPFragment` class. They reside in the `\install_dir\nwsp\webapp\logging` directory. The two files are included into any JSP file that you want identified in the application log when a message is issued during execution of the fragment. The `enterFragment.jspf` file is included into the fragment at its beginning, and `exitFragment.jspf` is included into the fragment at the end. For example:

```
<%-- debug --%>
<%! static final String fragment = "createLocationDimension.jspi"; %>
<%@ include file = "/logging/enterFragment.jspf" %>

<%-- body --%>
...
<%-- debug --%>
<%@ include file = "/logging/exitFragment.jspf" %>
```

You must declare a `String` variable named `fragment` before the `enterFragment.jspf` file is included into the JSP. The value of the string is the name that you want to appear in the log entry. This name is usually the name of the JSP. For information on the `JSPFragment` class, see the Javadoc documentation that is installed with the SESM software.

Servlets for the SESM Controls

The architecture of a SESM Web Portal application uses the Model-View-Control (MVC) design pattern. A Cisco SESM Web Portal application like NWSP includes a set of controls that are implemented as Java servlets. The compiled classes for the control servlets are packaged in a JAR file named `sesm.jar` that is located in the `\nwsp\webapp\WEB-INF\lib\` directory. You are not required to perform servlet programming. For information on the MVC design pattern and the SESM control servlets, see the [SESM Architecture: An Overview, page 3-2](#).

NWSP Templates

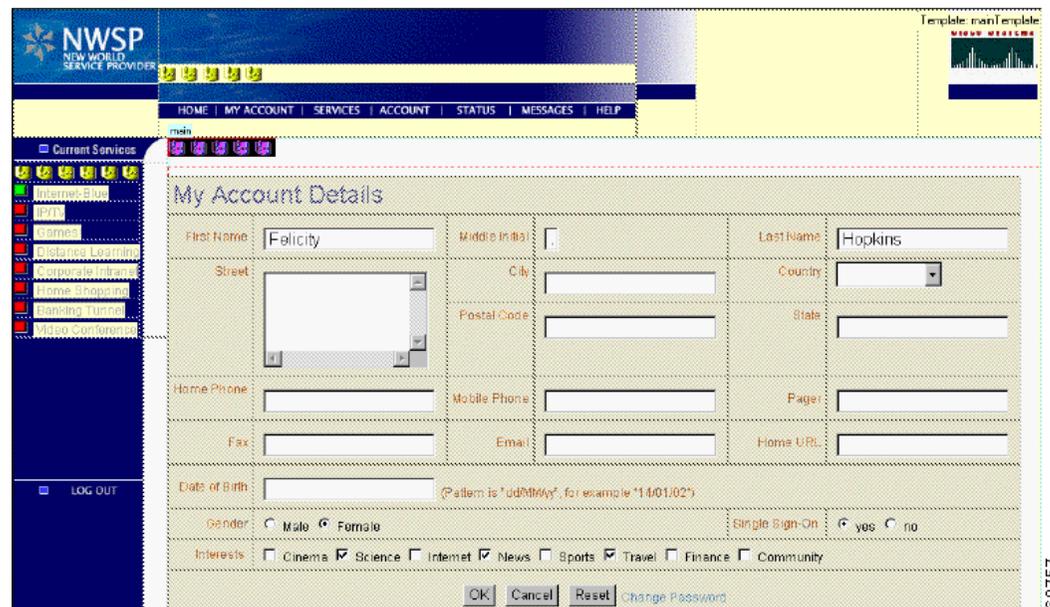
The NWSP components include two Dreamweaver templates for customizing and maintaining the JSP pages:

- `mainTemplate.dwt`.
- `bannerOnlyTemplate.dwt`.

Main Template

The template `mainTemplate.dwt` is used for NWSP pages that require a service list, navigation bar, Log Out button, and banner with brand icons. Many NWSP JSPs, including `home.jsp`, use this template. [Figure 4-5](#) shows the `home.jsp`, which is derived from `mainTemplate.dwt`, as it appears in Dreamweaver with its table borders visible.

Figure 4-5 Page That Uses `mainTemplate.dwt`

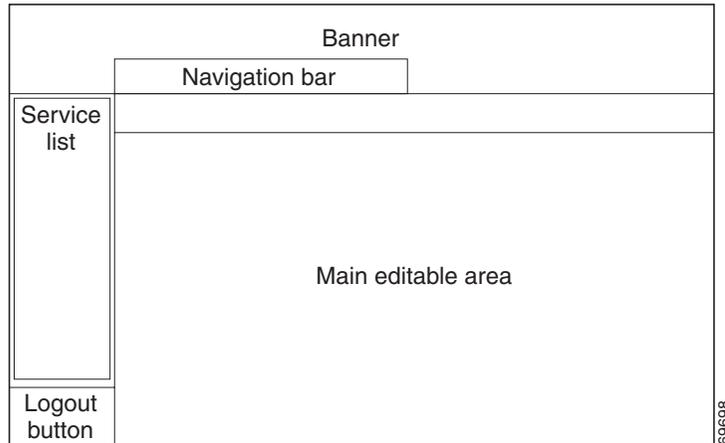


The template `mainTemplate.dwt` ([Figure 4-6](#)) is a set of tables that structure the NWSP page content into the following parts:

- Banner.
- Navigation bar.

- Service list.
- Main editable area.
- Log Out button.

Figure 4-6 Structure of the Template *mainTemplate.dwt*



In *mainTemplate.dwt*, the NWSP banner contains a set of images for branding.

In the template, the NWSP navigation bar is implemented with the Cisco Navigation Bar extension to Dreamweaver and a custom JavaScript, *navbar.js*. For information on the navigation bar, see the [JSP for the Navigation Bar, page 4-17](#).

The *service list* is a tree structure with the subscribed services and service groups from which the subscriber can select. For information on the service list, see the [JSPs for the Service List, page 4-10](#).

In the template, there are two editable areas: main and head. The JSPs that are derived from *mainTemplate.dwt* define the content of these editable areas.

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

The head editable area (not shown in [Figure 4-6](#)) has boilerplate code that the individual JSPs, which are derived from the template, modify with page-specific information:

- A page title defines the text in the title bar of the browser window.
- To improve performance, the `pageOnLoad` function can preload one or more files when the JSP is loaded. The files usually contain images for buttons used on the JSP. For example:

```
function pageOnLoad() {
MM_preloadImages(
'<shape:file name='/images/logout_button_over.gif'/>',
'<shape:file name='/images/home_button_over.gif'/>',
'<shape:file name='/images/my_account_button_over.gif'/>',
...
);
}
```

**Tip**

Preloading images can cause a page to load more slowly. Because the client browser caches these image files, files that have been loaded once do not need to be reloaded by the other JSPs. The general rule is: if possible, preload a file once only.

In `mainTemplate.dwt`, the Log Out button is implemented with two GIF files (`logout_button.gif` and `logout_button_over.gif`). The Log Out button uses two JavaScript functions from `navbar.js` to swap images when an `onMouseOver` or `onMouseOut` event occurs.

Banner-Only Template

The `bannerOnlyTemplate.dwt` file (Figure 4-7) is used for NWSP pages that do not require a service list or navigation buttons but do require a banner with brand icons.

Figure 4-7 Structure of the Template `bannerOnlyTemplate.dwt`



Most JSPs that use the template `bannerOnlyTemplate.dwt` contain message or help text: `help.jsp`, and `message.jsp`. Other JSPs that use the template have a simple form or button or both: `accountLogon.jsp`, `accountLogon3Key`, and `accountLogoff.jsp`.

PDA Web Portal Application

The Personal Digital Assistant (PDA) Web Portal application provides a subset of the SESM functionality for subscribers who are using a personal digital assistant or other handheld devices. This application demonstrates how you might use SESM on a handheld device for service selection. The PDA Web Portal application also shows how to provide a customized look and feel based on a brand.

The files for the PDA Web Portal application are located in the `\install_dir\pda` directory. For Demo installation, the Merit RADIUS file `aaa.properties` file is located in the `\install_dir\pda\config` directory. To determine the subscriber and service profiles that are used for PDA application in Demo installation, see `aaa.properties`.

Branding Based on User Group

The PDA Web Portal application is an example of how a SESM Web Portal application can create a look and feel customized for a specific subscriber based on a brand. With the PDA application, the user group to which a subscriber belongs determines the brand. In the PDA Web Portal application, there are three user groups: gold, silver, and bronze. The appearance of the PDA Web Portal application is different for each user group. For example, the navigation buttons for the gold user group are gold in color, for the silver user group they are silver in color, and so on.

PDA User Interface

The PDA user interface provides a Web Portal for network services as well as a suite of tools such as email, text messaging, and an address organizer. The subscriber uses the Web Portal for selecting services and logging on to services and for using the tools. [Figure 4-8](#) shows the home page from the PDA Web Portal application's user interface.

Figure 4-8 PDA Home Page



Designed for the small screen of a handheld device, the home page is organized into three parts:

- **Tools bar**—Buttons for the email, text messaging, and address organizer tools.
- **Service list**—A set of service names that the subscriber uses to log on to, start, and stop subscribed services.
- **Home, Logout, and Help Buttons**—Buttons that link to pages where the subscriber can log off from the SESM session or view Help information.



Note

The tools linked to by the Tools bar buttons are intended for Demo installation only and are not fully functional.

PDA Functionality

The SESM-related functionality of the PDA Web Portal application enables the subscriber to:

- Log on to and log off from a SESM session.
- Connect to or disconnect from services that are subscribed.

- Log on to a service.
- View help text.

The service list in the PDA Web Portal application includes services but no service groups. Having no service groups in the service list is not an inherent limitation of the SESM technology but a conscious design choice dictated by the size of a handheld-device screen.

The PDA Web Portal application employs the SESM user-shape mechanisms to detect the subscriber's brand, set the `brand` dimension, and serve brand-specific resources. For information on the user-shape mechanisms, see the [User Shapes and User-Shape Decoration](#), page 3-8.

So that the PDA Web Portal application can be viewed on a standard-sized PC or workstation, the application assumes that the HTTP client device is always a PC or workstation with a standard browser.

WAP Web Portal Application

The Wireless Application Protocol (WAP) Web Portal application provides a subset of the SESM functionality for WAP subscribers. This application demonstrates how SESM can be used for service selection on a WAP device.

The WAP Web Portal application is an example of how the view JSPs can serve dynamic WAP content in markup language other than HTML. The JSPs of the WAP Web Portal application are coded in Wireless Markup Language (WML).



Note

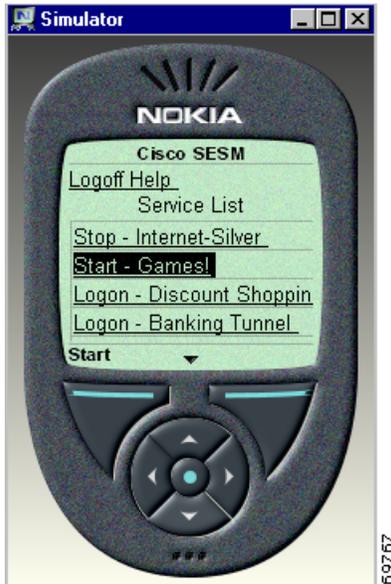
A WAP phone or WAP phone simulator is needed to view the WAP Web Portal application. The freely available Nokia Mobile Internet Toolkit contains two simulators and tools for developing a Web Portal application that uses WML. For information on using a WAP phone simulator, see the [Using Device Simulators for WAP and WML](#), page 2-20.

The files for the WAP Web Portal application are located in the `\install_dir\wap` directory. For Demo installation, the Merit RADIUS file `aaa.properties` file is located in the `\install_dir\wap\config` directory. To determine the subscriber and service profiles that used for WAP application in Demo installation, see `aaa.properties`.

WAP User Interface

The WAP user interface provides a Web Portal for network services. The subscriber uses the Web Portal for selecting services and logging onto the service. [Figure 4-9](#) shows the home page from the WAP Web Portal application's user interface.

Figure 4-9 WAP Home Page



Designed for the small screen and low bandwidth of a mobile phone, the home page is organized into two parts:

- **Logoff and Help Buttons**—Buttons that link to other pages where the subscriber can log off from the SESM session or view Help information.
- **Service list**—A set of service names that the subscriber uses to log on to, start, and stop subscribed services.

WAP Functionality

The WAP Web Portal application enables the subscriber to:

- Log on to and log off from a SESM session.
- Connect to or disconnect from services that are subscribed.
- Log on to a service.
- View help text.

The service list in the WAP Web Portal application includes services but no service groups. Having no service groups in the service list is not an inherent limitation of the SESM technology but a conscious design choice dictated by the size of a mobile-phone screen.

Captive Portal Web Solution

This section provides a brief overview of the SESM Captive Portal solution and describes how to use and modify the SESM Web Portal applications that are associated with the Cisco SESM Captive Portal solution.

For a comprehensive overview of the SESM captive solution and for information on configuring and deploying the Cisco SESM Captive Portal solution and TCP Redirect, see these documents on Cisco Connection Online at www.cisco.com:

- *Cisco Subscriber Edge Services Manager Introduction.*
- *Cisco Subscriber Edge Services Manager Administration and Configuration Guide.*
- SSG Features in Release 12.2(4)B.

**Note**

The explanations of the SESM Captive Portal solution and TCP Redirect feature in this guide assume that SESM Web Portal applications provide the Captive Portal functionality.

Captive Portal Solution Overview

The Cisco SESM Captive Portal solution works with the TCP Redirect for Services feature of the Service Selection Gateway (SSG) to provide TCP packet redirection and Captive Portal functionality. On SSG, TCP Redirect is configured so that, in certain defined situations, TCP packets from a subscriber HTTP request are redirected by SSG to a SESM Captive Portal.

The SESM Captive Portal is a servlet that determines the reason for the redirection (for example, an unauthenticated subscriber). The SESM Captive Portal then redirects the HTTP request to a SESM Web Portal application—the “content application”—that provides service content or message content through a JSP.

- Service content might be a logon page for an unauthenticated subscriber, or a service subscription page for subscribing to a service.
- Message content might include a greetings page after logon, or an advertising page for new services that is displayed at a specified interval.

Captive Portal Solution Web Applications

A typical Cisco SESM Captive Portal solution consists of two types of web applications:

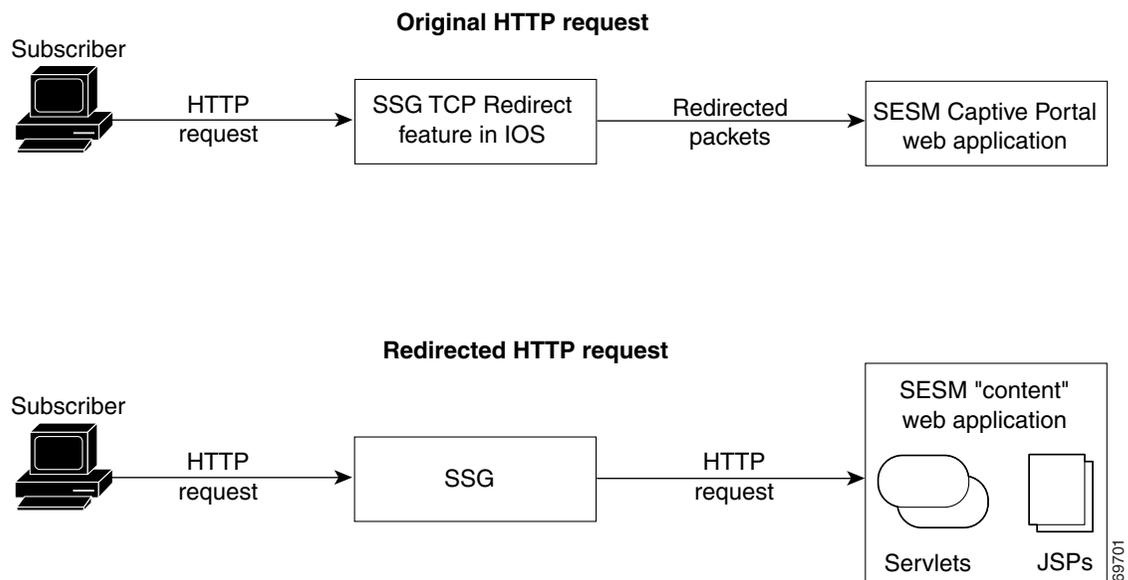
- A SESM Captive Portal web application.
- One or more SESM Web Portal applications (“content applications”) that provide service content or message content or both.

TCP Redirect and the SESM Captive Portal work together to redirect an HTTP request to a designated SESM content web application that provides the required service content or message content. In some cases, after the content is sent to the subscriber and displayed for a defined interval, the content web application forwards the subscriber to the URL that was originally requested. For example, after a greeting page is displayed for a number of seconds, the subscriber is forwarded to the originally requested URL.

For example, one type of redirection that the SESM Captive Portal solution and TCP Redirect provides is unauthenticated user redirection. If a subscriber is not logged in and sends an HTTP request to one of a configurable group of TCP Redirect ports on an SSG, the following interactions occur between three components: TCP Redirect on an SSG, a SESM Captive Portal web application, and a SESM Web Portal application, which is responsible for servicing unauthenticated subscribers.

- **TCP Redirect on an SSG:** The subscriber's browser sends a request to SSG, which redirects the packets to a specified SESM Captive Portal web application. (See Original HTTP Request in Figure 4-10.)
- **SESM Captive Portal web application:** The SESM Captive Portal web application determines that the subscriber is not authenticated and sends a redirect response to the subscriber's browser. The redirect response specifies the SESM content web application responsible for servicing unauthenticated subscribers. The SESM Captive Portal web application captures the original URL in the subscriber's original request and sends the URL in the query-string of the redirect response.
- **SESM content web application:** The subscriber's browser is redirected to the SESM content web application responsible for servicing unauthenticated users. (See Redirected HTTP Request in Figure 4-10.) The SESM content application responds to the subscriber with a logon page customized for the user's shape: device, brand, locale, and so on. After the subscriber sends a username and credentials, the SESM content web application authenticates the subscriber and redirects the subscriber's browser to the originally requested URL. The originally requested URL might be for a home page setting or for a specific service.

Figure 4-10 TCP Redirect, Captive Portal, and Content Web Application



Captive Portal Solution Redirection Types

The SESM Captive Portal solution and TCP Redirect feature support the four redirection types listed in [Table 4-7](#).

Table 4-7 **Redirection Types**

Redirection Type	Description
Unauthenticated user redirection	Handles attempted access to services by subscribers who have not yet authenticated to SESM. This type of redirection provides a logon page so the subscriber can authenticate and also redirects the originally requested URL so the user does not need to reenter the URL. In some configuration scenarios, such as with a Point-to-Point protocol (PPP) client with single sign-on enabled, the authentication is performed transparently to the subscriber.
Unauthorized service redirection	Handles unauthorized attempts to access a service. <ul style="list-style-type: none"> • If the access was rejected because the subscriber is not authenticated to the service, this feature provides the opportunity for the subscriber to authenticate. In some cases, the authentication can be transparent to the subscriber. • If the access was rejected because the subscriber is not authorized for the service, this feature provides the opportunity for the subscriber to become authorized. For example, it can access a billing server that allows the subscriber to make a payment on an overdue account. If the Cisco SESM is configured for SPE installation, a service self-subscription page could be displayed.
Initial logon redirection	Displays advertising or other messages to the subscriber upon login, for a specified length of time.
Advertising redirection	Displays advertising messages to the subscriber at timed intervals during an active SESM session.
Prepaid redirection	Handles attempted access to services by subscribers whose prepaid quota is exhausted. A Web Portal application like NWSP will be redirected to the NWSP recharge page if the prepaid limit for the requested service is reached. This feature is valid only if the SSG Prepaid feature is implemented.

Captive Portal Solution Configuration

SSG and SESM configurations define the needed network and device information:

- The SSG configuration defines the TCP ports on which SSG will redirect incoming TCP packets.
- The SSG configuration also specifies the SESM web server and ports where the SESM Captive Portal listens for incoming requests.
- The SESM configuration defines a corresponding set of Captive Portal web application port numbers.
- The SESM configuration also indicates the URLs for the content web applications. The URLs for content applications can indicate the different Web Portal applications, or different pages within the same Web Portal application.

For information on configuring SSG and SESM Captive Portal solution, see *Cisco Subscriber Edge Services Manager Captive Portal Guide* and *SSG Features in Release 12.2(4)B*.

Captive Portal

The SESM Captive Portal included with SESM consist of three web applications:

- Captive Portal web application—This application used for all redirection types.
- Service portal web application—The content application used for unauthenticated user redirection and unauthorized service redirection.
- Message portal web application—Used for login redirection and advertising redirection.

This section provides information on the SESM Captive Portal Web Portal application and the service and message portal web applications that are used in the Captive Portal solution.

Captive Portal Web Application

The Captive Portal web application that is included with SESM handles all redirection types. The Captive Portal web application performs the following functions:

- Determines the reason for the redirection (unauthenticated user, unauthorized service redirection, initial logon redirection, or advertising redirection).
- Captures information from the subscriber's original HTTP request.
- Redirects the HTTP request to a SESM service portal web application or message portal web application that provides the appropriate content. The HTTP redirect contains information about the subscriber's request in query-string parameters appended to the redirection URL.

The SSG and SESM configurations define the port numbers where the SESM Captive Portal application listens for incoming requests. The SESM captive-portal files configuration also provides hostnames or addresses, port numbers, and the URLs for the service portal and message portal web applications. The URLs for the servicing and messages content can indicate different web applications or different pages within the same web application. The SESM Captive Portal web application is preprogrammed and requires no modifications.

The parameters in [Table 4-8](#) are defined in the captiveportal.xml file, located in the `\install_dir\captiveportal\config` directory. These are the names of the parameters that the SESM Captive Portal web application sends in the query-string of the redirect response to the service portal web application or message portal web application.

Table 4-8 Parameters Appended by Captive Portal Web Application

Redirection Type	Parameter Name	Description
Unauthenticated user redirection	CPURL	The URL in the subscriber's original HTTP request.
Unauthorized service redirection	serviceURL	The URL of the service that was requested in the subscriber's HTTP request.
	service	The name of the service that was requested in the subscriber's HTTP request.
	username	The username from the subscriber profile.

Table 4-8 Parameters Appended by Captive Portal Web Application (continued)

Redirection Type	Parameter Name	Description
Initial logon redirection and Advertising redirection	CPURL	The URL in the subscriber's original HTTP request.
	CPDURATION	The message duration defined in the relevant captiveportal MBean attribute: <ul style="list-style-type: none"> initialCaptiveDuration for initial logon redirection advertisingCaptiveDuration for advertising redirection
	CPSUBSCRIBER	The username from the subscriber profile.

The parameters in [Table 4-8](#) can be configured in the captiveportal.xml file:

- You can change the names of the parameters that are appended to the redirected URL by modifying their names.
- You can indicate that the SESM Captive Portal web application should not append a parameter by setting the parameter name to an empty string. For example, to omit the unauthorized service redirection parameter serviceRedirectSubscriberParam, delete service from the following:

```
<Set name="serviceRedirectServiceParam">service</Set>
```

For information on configuring the SESM Captive Portal solution, see *Cisco Subscriber Edge Services Manager Administration and Configuration Guide*.

Content Web Applications

The content web applications (Web Portal applications) provide content to the subscriber. You should include the following basic functions in the content applications:

- For unauthenticated user redirections—The content application should present a SESM logon page so the subscriber can authenticate.
- For unauthorized service redirections—The content application should accept parameters from the Captive Portal application identifying the originally requested service and perform some appropriate action such as:
 - Coordinating with SSG to authenticate to the service and then connect to the service.
 - Presenting subscription information if the subscriber is not subscribed. For example, in a SPE SPE installation, the content web application can present a self-subscription page.
 - Presenting a payment page if the subscriber account requires additional funding.
- For initial logon redirection—The content web application should send pages containing general messages or advertising to the subscriber. Configuration parameters define how long the messages are displayed on either a global or individual subscriber basis. The content web application should accept a parameter CPURL from the Captive Portal identifying the subscriber's originally requested URL. Using the original URL, the application can perform another redirection when it finishes displaying messaging.
- For advertising redirection— The content application should send pages containing advertising messages to the subscriber. This application should perform the same functions as described above for the initial logon redirection, except that it sends advertising content rather than messages.

The service portal Web Portal application and message portal Web Portal application provide this basic functionality.

Service Portal Web Application

The service portal reference implementation is the NWSP Web Portal application. NWSP provides the content application for unauthenticated user redirection and unauthorized service redirection. The NWSP Web Portal application, which includes a set of servlets and JSPs, handles these types of redirection as follows:

- For the unauthenticated user redirection, the default Captive Portal configuration parameters cause the user to be redirected to /home in the NWSP Web Portal application. For an unauthenticated user, /home displays the SESM logon window.
- For unauthorized service redirection, the default Captive Portal configuration parameters cause the user to be redirected to /serviceRedirect, the ServiceRedirectControl servlet in the NWSP Web Portal application. For information on this servlet, see the Javadoc documentation for the ServiceRedirectControl class.

Message Portal Web Application

The message portal Web Portal application is the content application for initial logon redirection and advertising redirection. The message portal web application includes a servlet (MessagePortalServlet) and a set of JSPs. MessagePortalServlet sets the value of two HTTP request attributes so that the message JSPs can use them:

- url—The URL in the subscriber's HTTP request. This URL comes from the CPURL parameter in the query-string sent by the Captive Portal web application.
- duration—The length of time that the web application displays the message on the subscriber's browser. The duration comes from the CPDURATION parameter in the query-string sent by the Captive Portal web application.

For initial logon redirection and advertising redirection, MessagePortalServlet forwards the redirected request to the appropriate message JSP based on the subscriber's interests as defined in the subscriber profile.

The message JSP displays its content for duration, which is defined in the captiveportal.xml file. The JSP then forwards the subscriber to the originally requested URL.

Before redirecting to the message JSP, the message portal Web Portal application determines the device that the subscriber is using so that the appropriate JSP serves device-specific content is used.

- If the subscriber's device is a WAP phone, the message web application looks for message JSPs in the \wap\pages directory.
- If the subscriber's device is a PDA, the message web application looks for message JSPs in the \pda\pages directory.
- If the subscriber's device is not a WAP phone or PDA, the message web application looks for message JSPs in the \pages directory.

Beneath these directories, the JSPs for initial logon redirection are in an \initial directory, and the JSPs for advertising redirection are in an \advertising directory.

To determine what JSP to use for each interest, the message portal web application uses information in the `messageportal.xml` file, located in the `\install_dir\messageportal\config` directory. The message portal application uses the JSP that is defined for the *first interest* that the subscriber has selected from a list of possible interests on the My Account page. In a deployer-created message portal web application, the algorithm for determining what message or advertising to display is based on the requirements of the application.

For more information on `MessagePortalServlet`, see the Javadoc that is installed with SESM.



SESM Internationalization and Localization

The Cisco SESM software includes a set of components that can be used for internationalization and localization. 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 a SESM Web Portal application. This chapter discusses these topics:

- [Localizing a Web Portal Application, page 5-2](#)
- [Using Resource Bundles, page 5-3](#)
- [Using Internationalized Resources, page 5-5](#)
- [Using the Localization Tag Library, page 5-6](#)
- [Setting a Default Localization Context, page 5-14](#)

For information on configuring a tag library, see the [Configuring a Tag Library, page B-1](#).

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 Portal 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, localization involves translating text messages to another language and, where required, providing locale-specific images.

SESM Components for Internationalization and Localization

The Cisco SESM web components include a set of Java classes and JSP tag libraries that help the deployer to internationalize and localize a SESM Web Portal 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 Portal application to change the locale and time zone of the subscriber's session.

A Cisco SESM Web Portal application is not required to use the extended classes and techniques that are provided with SESM. A SESM Web Portal 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 Portal application.

Localizing a Web Portal Application

SESM Web Portal applications such as NWSP use the SESM classes and techniques for internationalization and localization. The NWSP Web Portal application is implemented so that it dynamically detects each subscriber's language and country and generates SESM pages with resources appropriate to the language and country. For more information on these mechanisms and techniques, see the [User Shapes and User-Shape Decoration, page 3-8](#) and the [Decorating a User Shape, page 3-18](#).

SESM Web Portal application web components, such as buttons and icons, are provided for various languages, including English, Chinese, and Japanese. You can create a SESM website for another language with two sets of modifications:

- Changing text, icons, and images.
- Creating additional properties files.

The first set of modifications involves changing SESM Web Portal application components so that they meet the needs of the language. All text in icons and images must be translated into the required language. For example, the current services image in the NWSP Web Portal application contains the English language text "Current Services." If a SESM Web Portal 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 5-1](#).

Figure 5-1 Localizing *currentservices.gif*



SESM web components include many 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 `\nwsp\webapp\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 `\web_app_name\webapp\images` directory, or a location in a sparse-tree directory structure where locale-specific images reside (for example, `\web_app_name\webapp\de\images` for a set of German-language images).

The second set of modifications for localization is that message text and other items in resource bundles must be translated, and an additional properties file must be created for each new language. The NWSP Web Portal application includes the logic to determine and set the subscriber's locale so that it uses the appropriate properties file. For information on translating a properties file into another language, see the explanation at the beginning of the `messages.properties` file in the `\nwsp\webapp\WEB-INF\classes` directory.

For information on resource bundles, see the [Using Resource Bundles, page 5-3](#).

Using 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 Portal application.

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

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

The following sections provide some general information on resource bundles and properties files and their use with a Cisco SESM Web Portal application. For detailed information on resource bundles, see the descriptions of these classes at the Java 2 platform area of the java.sun.com website:

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



Tip

The Java Internationalization and Localization Toolkit 2.0 from Sun Microsystems, Inc. can be a useful tool for translating messages in a resource bundle file into the target locale language, for merging and comparing resource bundle files, and for other internationalization and localization tasks. The toolkit is free and can be downloaded at: <http://java.sun.com/products/jilkit/>

Using Properties Files

You can implement a resource bundle with a set of one or more properties files. A *properties file* is a plain-text file that contains key–value pairs for each item that can be localized. For example, the English version of a Web Portal application 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 Portal 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 Web Portal, the properties files (for example, `messages_en.properties`) reside in the `\webapp\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 `<`.

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` website.
- For information on using the localization tag library to retrieve values from a resource bundle, see the [resource Tag, page 5-12](#) and the [template Tag, page 5-13](#).

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 5-1](#) shows five examples of properties file names.

Table 5-1 *Names of Properties Files in a Resource Bundle*

Properties Filename	Description
<code>SESMResources.properties</code>	base name—The file is the default version.
<code>SESMResources_en.properties</code>	base name and language name—The file is the English version.
<code>SESMResources_fr.properties</code>	base name and language name—The file is the French version.
<code>SESMResources_fr_CA.properties</code>	base name and language name and country name—The file is the French version used for Canada.
<code>SESMResources_fr_CA_WIN.properties</code>	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 `SESMResources_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 a SESM Web Portal 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 Portal application. For information on the benefits of using the `L10nContext` class and the localization tag library, see the [Using the Localization Tag Library, page 5-6](#).

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` website.

Using Internationalized Resources

The controls in a SESM Web Portal application provide internationalized data to the views. The SESM controls do this by using internationalized resources in the view JavaBeans. These resources are provided in a resource bundle and, therefore, you can adapt them for various languages and regions without any changes to the programme.

Many objects used by the SESM controls (for example, `MyAccountControl`) are already internationalized and do not require any special treatment. For example, a Java date or number is an internationalized object. For objects that do need internationalization, each SESM control internationalizes the resources that it uses by associating them with a key in the Web Portal application's resource bundle. The controls use the `I18nResource` class to create these internationalized resources.

In a SESM Web Portal application, each view JSP retrieves internationalized resources from the view bean. The resource is usually text on a label or button, or a message to the subscriber. For example, the `AccountLogonControl`, a control for subscriber logon page, uses internationalized resources for certain phrases, such as "Please enter your username". Each internationalized resource is associated with a key-value pair in the resource bundle for the SESM Web Portal application.



Tip

The Javadoc description of each SESM control provides specific information about the control's use of internationalized resources and about the keys that the control uses.

Web Application Tasks for Localizing Internationalized Resources



Tip

If you use the web components of the NWSP Web Portal application, the internationalization and localization coding already exists in the controls and view JSPs of NWSP.

A SESM Web Portal application is responsible for localizing the internationalized resources that the SESM controls provide. For the Web Portal application, the localization tasks are as follows:

1. The SESM Web Portal application specifies a current localization context (`L10nContext`) for the subscriber. When retrieving resources, SESM gets the locale and the resource bundle base name specified by the current localization content to determine which properties file to use. For example, the NWSP Web Portal application sets the current localization context for the subscriber's HTTP session.
2. Each view JSP formats any internationalized resources that the SESM control provides. Use the `object` attribute of the `format` tag in the localization tag library for this purpose. For example, in `statusBody.jsp`, the description and status of each service is an internationalized object localized with the `format` tag.

```
<%@ taglib uri="http://www.cisco.com/taglibs/localization" prefix="l10n" %>
...
<jsp:useBean id="statusBean" class="com.cisco.sesm.webapp.control.StatusBean"
scope="request" />
...
<it:start over="<%=statusBean.getStatus()%>" name="status"
type="com.cisco.sesm.webapp.control.StatusBean.State">
<tr>
```

```

<td class="PageText"><l10n:format object="%= status.getDescription()
%>">description</l10n:format></td>
<td class="PageText"><l10n:format object="%= status.getStatus()
%>">status</l10n:format></td>
...</tr>
</it:start>

```

The `getDescription` and `getStatus` methods return values of the type `I18nObject`. These are internationalized resources that can use key–value pairs in the properties files of the resource bundle. For example, the keys associated with service status include `connected`, `disconnected`, and `connection_lost`. The `object` attribute of the `format` tag, in effect, retrieves the value associated with the key from the properties file for the current localization context (`L10nContext`).

Other Deployer Tasks for Localizing Internationalized Resources

You are responsible for performing the following localization-related tasks:

- Setting the default localization context for the SESM Web Portal application. For information, see [Setting a Default Localization Context, page 5-14](#).
- Providing the resource bundle (a set of one or more properties files) that the SESM Web Portal application uses. For more information, see the [Using Resource Bundles, page 5-3](#).

If you use the NWSP Web Portal application web components as the base for developing a SESM Web Portal application and keep the internationalization and localization coding in place, the tasks required for localizing internationalized resources are limited to the two tasks.

Using the Localization Tag Library

The Cisco SESM software includes a localization tag library that helps reduce the complexity of localizing a SESM Web Portal 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`—Gets 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. Also, formats an internationalized object.
- `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. You can use the `template` tag with the `resource` tag for token replacement in a resource.

The localization tag library is specifically designed for localizing Web Portal applications. 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 are 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 Portal application's default `L10nContext` object.

context Tag

You can use the context tag 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 Portal 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, and resource bundle base name from the first of the following localization contexts that exist:

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 neither of these exist, the values come from the default localization context, which it obtains with the `L10nContext.getDefault` method.

A Web Portal 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 Portal 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.

You can specify the current localization context, including all its characteristics, 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 location 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 5-2](#) lists the attributes of the context tag.

Table 5-2 Context Tag Attributes

Attribute	Description	Required	Runtime Expression
variable	Declares a variable that has the type <code>L10nContext</code> and the specified name. You can use the named variable as a scripting variable within the tag body. The value assigned to variable is the declared <code>L10nContext</code> object's name. See the example following this table.	No	No
context	Specifies the current localization context explicitly. For example: <pre><l10n:context context = "<%= someL10nContextObject %>" /></pre>	No	Yes
resourceBundleName	Specifies the resource bundle base name.	No	Yes
locale	Specifies the location of the current localization context. For example: <pre><l10n:context locale = "<%= Locale.GERMANY%>" /></pre>	No	Yes
timeZone	Specifies the time zone of the current localization context.	No	Yes
preferredLocales	Specifies the preferred locations of the current localization context. This attribute requires that you specify another location with the otherwise attribute.	No	Yes
otherwise	Specifies the location to use if no resource bundle can be found for any of the preferred locations. This attribute requires that you specify a preferred location with the preferred attribute.	No	Yes
scope	Specifies the scope of the current localization context. Allowed values for scope are: <ul style="list-style-type: none"> • <code>PageContext.APPLICATION_SCOPE</code> • <code>PageContext.SESSION_SCOPE</code> • <code>PageContext.REQUEST_SCOPE</code> • <code>PageContext.PAGE_SCOPE</code> For the meaning of each scope, see the documentation for the Java <code>PageContext</code> class in the Java 2 platform area of the java.sun.com website.	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 location.
- 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 location 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 you do not use an attribute, 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 5-3](#) lists the attributes of the country tag.

Table 5-3 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

Table 5-3 Country Tag Attributes (continued)

Attribute	Description	Required	Runtime Expression
locale	Gets the text description for the country of the specified location.	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

You can use the format tag to convert currency, number, date, time, or internationalized object values into text according to the formatting conventions associated with the Java class `java.text.MessageFormat` and the current `L10nContext` localization context.

Table 5-4 lists the attributes of the `format` tag.

Table 5-4 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 <i>Date</i> value to be converted into a time.	No	Yes
dateTime	Specifies a <i>Date</i> value to be converted using date-time format.	No	Yes
shortDate	Specifies a <i>Date</i> value to be converted using short date format.	No	Yes
shortTime	Specifies a <i>Date</i> value to be converted using short time format.	No	Yes
shortDateTime	Specifies a <i>Date</i> value to be converted using short date-time format.	No	Yes
mediumDate	Specifies a <i>Date</i> value to be converted using medium date format.	No	Yes
mediumTime	Specifies a <i>Date</i> value to be converted using medium time format.	No	Yes
mediumDateTime	Specifies a <i>Date</i> value to be converted using medium date-time format.	No	Yes
longDate	Specifies a <i>Date</i> value to be converted using long date format.	No	Yes
longTime	Specifies a <i>Date</i> value to be converted using long time format.	No	Yes
longDateTime	Specifies a <i>Date</i> value to be converted using long date-time format.	No	Yes
fullDate	Specifies a <i>Date</i> value to be converted using full date format.	No	Yes
fullTime	Specifies a <i>Date</i> value to be converted using full time format.	No	Yes
fullDateTime	Specifies a <i>Date</i> value to be converted using full date-time format.	No	Yes
object	Formats an internationalized object of type <i>I18nObject</i> .	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 5-5](#) lists the attributes of the resource tag.

Table 5-5 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 `PleaseAuthenticate` from the `messages.properties` resource bundle of the NWSP Web Portal 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

You can use the template tag with the resource tag to replace a token 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 location-sensitive information such as dates, messages, and numbers using the conventions of the current localization context (`LI0nContext`). [Table 5-6](#) lists the attributes of the template tag.

Table 5-6 *Template Tag Attributes*

Attribute	Description	Required	Runtime Expression
param	Specifies a single parameter value to substitute for a token. Use the param attribute 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 <code>Object[]</code> 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 braces (`{ }`) 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.
```

You can use the template tag 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. For 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).
```

Setting a Default Localization Context

You can use the initialization parameters for the `L10nContextDecorator` servlet to set the values of the Web Portal application default localization context. The `L10nContextDecorator` servlet creates an `L10nContext` object and adds it as an attribute that has session scope. In the `web.xml` file of each SESM Web Portal application, the initialization parameters for the `L10nContextDecorator` servlet can be used to specify a default localization context.

In SESM Web Portal applications, the value of a localization context (`L10nContext`) is determined as follows:

1. The values for the localization context come from the preferred location of the subscriber HTTP client machine if the SESM Web Portal application supports the preferred location. For example, a subscriber can set the preferred location for a Windows client machine with the Regional Settings tool in Control Panel.
2. If the SESM Web Portal application does not support the preferred location, the values for the localization context come from the Web Portal application default localization context, which you can specify with the `L10nContextDecorator` initialization parameters.

[Table 5-7](#) lists the `L10nContextDecorator` initialization parameters that you can specify in the `web.xml` file for a SESM Web Portal application.

Table 5-7 *L10nContextDecorator Default Initialization Parameters*

Parameter	Description
<code>defaultResourceBundle</code>	A base name for a resource bundle properties file. (for example, <code>SESMResources</code>).
<code>defaultLanguage</code>	A language code (for example, <i>it</i>). These codes are the lowercase, two-letter codes defined in ISO-639. You can find a full list of these codes at: http://www.ics.uci.edu/pub/ietf/http/related/iso639.txt
<code>defaultCountry</code>	A country code (for example, <i>IT</i>). These codes are the uppercase, two-letter codes defined in ISO-3166. You can find a full list of these codes at: http://www.chemie.fu-berlin.de/diverse/doc/ISO_3166.html
<code>defaultVariant</code>	A variant code (for example, <i>EURO</i>). Variant codes are vendor and browser specific.
<code>defaultTimeZone</code>	A time zone ID (for example, <code>Europe/Stockholm</code>) or the value <i>default</i> . <ul style="list-style-type: none"> • For information on time-zone IDs, see the description of the <code>TimeZone</code> class in the Java 2 platform area of the java.com site. • For information on the value <i>default</i>, see the explanation that follows this table.

When you use the value *default* for the `defaultTimeZone` parameter, a time-zone map associates a time zone with a specific location. The `L10nContext` class uses the time-zone mapping for the Web Portal application default localization context. If the subscriber later changes the preferred location, the `L10nContext` software selects a new time zone that matches the new location.

Two `L10nContextDecorator` servlet initialization parameters are used to define a time-zone map.

- `timeZoneMapCountries` is a set of country codes that define the countries that are mapped to a time zone.
- `timeZoneMapTimeZones` is a set of time zone IDs that define the time zones that are associated with the countries given in the `timeZoneMapCountries` parameter.

For both of these parameters, you can separate the values by whitespace or commas.

The following example shows how to use the `timeZoneMapCountries` and `timeZoneMapTimeZones` parameters.

```
<servlet>
  <servlet-name>L10nContext</servlet-name>
  <servlet-class>com.cisco.sesm.navigator.L10nContextDecorator</servlet-class>
  <!-- Specify associations from Country -> TimeZone. -->
  <init-param>
    <param-name>timeZoneMapCountries</param-name>
    <param-value>AU,NZ,SE,PT</param-value>
  </init-param>
  <init-param>
    <param-name>timeZoneMapTimeZones</param-name>
    <param-value>
      Australia/Sydney
      Pacific/Auckland
      Europe/Stockholm
      Europe/Lisbon
    </param-value>
  </init-param>

  <!-- The value "default" indicates that a time-zone map is used -->
  <init-param>
    <param-name>defaultTimeZone</param-name>
    <param-value>default</param-value>
  </init-param>
  <load-on-startup>1</load-on-startup>
</servlet>
```

Given the preceding time-zone map, if the location of the current localization context (`L10nContext`) is for Australia (country code `AU`), `L10nContextDecorator` uses the corresponding time-zone ID (`Australia/Sydney`) to determine a time zone. For each country specified in `timeZoneMapCountries`, there should be a corresponding time-zone ID given in the `timeZoneMapTimeZones` parameter.

For the default localization context values used for a SESM Web Portal application like `NWSP`, see the `web.xml` for the application.



SESM Testing and Development

This appendix describes the tools for establishing a testing environment for Cisco Subscriber Edge Services Manager (SESM) deployments. A testing environment can help with evaluating SESM features, developing customized portals, and testing deployment options. This appendix contains the following sections:

- [Introduction, page A-1](#)
- [Using Web Portal Demo Installation, page A-2](#)
- [SSG Simulator, page A-5](#)
- [SESM Bundled RADIUS Server, page A-8](#)
- [Demo Profile File, page A-11](#)
- [Downloading International Character Sets, page A-17](#)

Introduction

SESM includes features for demonstrating the Web Portal user interface and simulating the Service Selection Gateway (SSG). It also provides a bundled RADIUS server for test and evaluation purposes. An overview of the SESM test features is provided in [Table A-1](#).

Table A-1 Overview of SESM Test Features

Test Feature	Description
Web Portal Demo installation	Used for demonstrating the Web Portal user interface. See Using Web Portal Demo Installation, page A-2 .
SSG simulator	Used for simulating SESM behavior with an SSG. See SSG Simulator, page A-5 .
Bundled RADIUS Server	Used as a quick way to establish an actual SESM deployment to test, instead of relying on the Demo installation. See SSG Simulator, page A-5 .

SESM uses the Demo Profile file to provide example profiles and attributes. For details about the Demo Profile file, see [Demo Profile File, page A-11](#).

For details of how SESM handles international character sets, see [Downloading International Character Sets, page A-17](#).

**Note**

The bundled RADIUS server is a specific configuration of the RADIUS Data Proxy (RDP) application. It is included as a separate script and configuration file as it is much simpler to configure than the standard RDP application. The bundled RADIUS server operates in the same way and shares the same performance characteristics as the RDP application.

Using Web Portal Demo Installation

SESM Demo installation allows a portal to run in a simulated network, without access to other solution components, such as the SSG, a RADIUS server, or an LDAP directory. Typically you use Demo installation for the following purposes:

- To demonstrate the capabilities of SESM when other required network components are not available. In Demo installation, you can demonstrate the features of both RADIUS and SPE deployments on a laptop.
- To test customizations to JSPs in a SESM portal application. Refer to [Chapter 2, “Demo Installation”](#) for information about using Demo installation during application development.

Standalone Demo installation is intended *only* for the above purposes. Demo installation is not in any way representative of SESM performance in an end-to-end solution with actual network components.

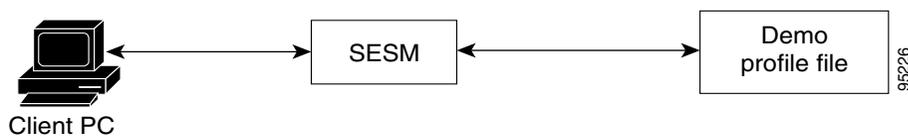
You can run any SESM portal, including your own customized portals, as a Demo installation.

**Note**

For any SESM portal running as a Demo installation:

The portal reads profiles in Merit flat file format. The file path name, <application name>/config/aaa.properties is configured in the SESMDemoMode MBean. The portal does not alter the contents of the Demo Profile file.

Figure A-1 SESM in Standalone Web Portal Demo Installation



Installing SESM in Web Portal Demo Installation

This section describes how to install the SESM portals in order to use Demo installation. The Demo installation is quick and requires the entry of only a few parameters.

If your installation is a Demo installation, you must perform another installation before trying to run an application in RADIUS or SPE installations. You cannot switch from Demo installation to SPE or RADIUS installations at runtime for the following reasons:

- The MBean configuration files are not set up to support the switch to those other installations. Several manual changes are required in the files.
- The Demo installation may not install all of the components required by the other installations. For example, a Demo installation does not install the SPE component, which is required for running an SPE installation.

Installation Procedure

To install SESM as Demo installation, use the procedures described in *Cisco Subscriber Edge Services Manager Installation Guide*. During installation, select the license type you require: either *Evaluation-RADIUS* installation or *Evaluation-SPE* installation.

Choosing a Web Browser for a Demo

You can use the following browsers to demonstrate the NWSP application:

- Netscape 7.0 and later. SESM uses Unicode Transformation Format Version 8 (UTF-8) character representations. UTF-8 supports both 1-byte and double-byte character sets. To demonstrate support for double-byte character sets on a Netscape browser, use Netscape 7.0 or later.
- Internet Explorer version 5.x and later.

When you develop SESM applications for deployment, you should consider the end users of your deployed application and design the application to accommodate the devices or browsers that they commonly use.

Unpredictable results could occur when you run the demonstrations if you do not consider your browser limitations.

Running a Web Portal as a Demo Installation

To start a Web Portal as a Demo installation (for example NWSP), perform the following procedure:

- Step 1** Execute the appropriate startup script as shown in [Table A-2](#).

Table A-2 Starting the Demo

Platform	SESM Installed Mode	Demo Startup Command
Solaris and Linux	Demo installation	jetty/bin/startNWSP.sh
	Any (Switches to Demo installation)	startNWSP.sh -mode Demo
Windows	Demo installation	jetty\bin\startNWSP.cmd
	Any (Switches to Demo installation)	startNWSP.cmd Demo

- Step 2** Open a web browser.

- Step 3** Go to the NWSP URL:

`http://host:port`

For example:

`http://localhost:8080`

where:

host is the IP address or host name of the computer on which you installed the NWSP application. You can enter the value `localhost`, or the IP address, to indicate the local computer.

port is the NWSP port number that you specified during the installation.

- Step 4** On the SESM portal logon page, use any username whose profile is defined in the Demo profile file.

Logon Names and Passwords for a Demo

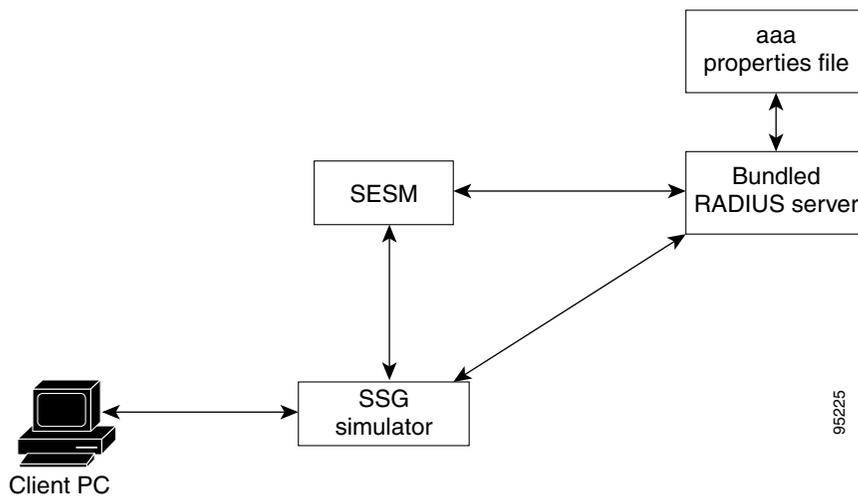
Table A-3 shows the user IDs and passwords in the profiles in the installed Demo profile file, “demo.txt”.

Table A-3 Logon Names and Passwords in demo.txt

To Demonstrate RADIUS Installation Features	To Demonstrate SPE Installation Features	To Demonstrate Branding Based on User Groups
User ID: radiususer Password: cisco Other valid users for RADIUS installation demos are user1, user2 up to user45.	User ID: golduser Password: cisco User ID: subgolduser Password: cisco User ID: silveruser Password: cisco User ID: bronzeuser Password: cisco	User ID: bronzeuser Password: cisco User ID: silveruser Password: cisco User ID: golduser Password: cisco

SSG Simulator

Figure A-2 SESM SSG Simulator



The SESM SSG simulator is installed by default in both RADIUS and SPE installations. This simulator provides a way to test a SESM deployment with actual authentication and service connection services, when a Cisco device that can host a real SSG is not available. You can configure the SSG Simulator in SESM RADIUS or SPE deployments.

None of the SESM installation parameters affects the default configuration of the SESM bundled RADIUS server. However, you can edit the “ssgsim.xml” file in the tools directory to change the installed configuration.

The files that support the SSG simulator are in the tools directory, under the SESM installation directory:

```
tools
  config
    erp.xml
    ssgsim.xml
```

Installing the SSG Simulator

To use the SSG simulator you must perform a standard SESM RADIUS installation. To do this, proceed as follows:

Tip Use the **show run** command on the SSG platform to determine how the SSG is configured.

- Step 1** Install SESM using the procedures described in *Cisco Subscriber Edge Services Manager Installation Guide*, using the following attributes:

Table A-4 Required Attributes for Installation

Component	Attribute	Value
SSG	Port number	Enter required value.
	Shared secret	Enter required value.
	Port-bundle host key bundle length	0
SSG-to-client mapping	IP address of the SSG	IP address of machine running SSG simulator; for example, 1.2.3.5
	Client subnet	0.0.0.0
	Subnet mask	0.0.0.0
RADIUS server	Primary AAA server IP address or hostname	Select required values for your RADIUS server.
	Primary AAA server port	
	Secondary AAA server IP address or hostname	
	Secondary AAA server port	
	Shared secret	
Profile passwords	Service password	Use passwords required for your RADIUS server.
	Service group password	

- Step 2** Edit the SSG simulator configuration file `ssgsim.xml`, in the `tools/config` directory as shown in Table A-5:

Table A-5 Required Attributes for Installation

Component	Attribute	Value
SSG	Port number	Enter required value.
	Shared secret	Enter required value.
	Port-bundle host key bundle length	0
RADIUS server	Primary AAA server IP address or hostname	Enter values for the RADIUS server you are using with SESM.
	Primary AAA server port	
	Secondary AAA server IP address or host name	
	Secondary AAA server port	
	Shared secret	
Profile passwords	Service password	Use passwords required for your RADIUS server.
	Service group password	

- Step 3** Run SESM with the SSG simulator.

Running SSG Simulator

To run the SSG simulator, navigate to the `tools/bin` directory and then enter one of the commands, shown in Table A-6:

Table A-6 Running SSG Simulator

Platform	Command
Windows	<code>startSSGSim.cmd</code>
Linux, Unix and Solaris	<code>startSSGSim.sh</code>



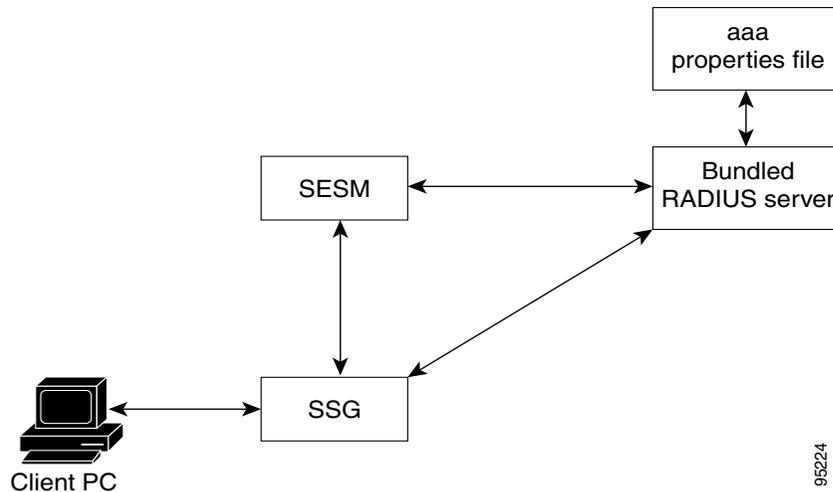
Note

The SSG simulator will remove sessions from its cache once the cache is full. As a result in a development environment some operations may fail as the session might have been removed from the cache.

SESM Bundled RADIUS Server

The SESM bundled RADIUS server is installed by default in both RADIUS and SPE installations. This server provides a quick way to establish an actual SESM deployment to test, without having to configure a RADIUS server or use Demo installation.

Figure A-3 SESM Bundled RADIUS Server



The SESM bundled RADIUS server is ready to run immediately after installation. It uses the following configuration:

- port—1813, on the localhost.
- secret—cisco.

None of the SESM installation parameters affect the default configuration of the SESM bundled RADIUS server. However, you can edit the *aaa.xml* file in the tools directory to change the installed configuration. *Cisco Subscriber Edge Services Manager Deployment Guide* contains more information about the JMX MBeans in the *aaa.xml* file.

The installed location of configuration files and startup scripts that support the SESM bundled RADIUS server is the tools directory, under the SESM installation directory:

```

tools
  bin
    startAAA.sh/.cmd
  config
    aaa.xml
    erp.xml
    aaa.properties
  
```

The *aaa.xml* and *erp.xml* files are MBean configuration files for the SESM bundled RADIUS server. The *aaa.properties* file is a sample profile file in Merit format.

Installing SESM with Bundled RADIUS Server

To use the bundled RADIUS server, you must perform a standard SESM RADIUS installation. To do this, proceed as follows:

Tip Use the **show run** command on the SSG platform to determine how the SSG is configured.

- Step 1** Install SESM using the procedures described in the *Cisco Subscriber Edge Services Manager Installation Guide* using the attributes shown in Table A-7.

Table A-7 Required Attributes for Installation

Component	Attribute	Value
SSG	Port number	Enter required value.
	Shared secret	Enter required value.
	Port-bundle host key bundle length	0
SSG-to-client mapping	IP address of the SSG	For example: 1.2.3.5
	Client subnet	0.0.0.0
	Subnet mask	0.0.0.0
RADIUS server	Primary AAA server IP address or host name	localhost
	Primary AAA server port	1813
	Secondary AAA server IP address or host name	localhost
	Secondary AAA server port	1813
	Shared secret	cisco
Profile passwords	Service password	servicecisco
	Service group password	servicegroupcisco

- Step 2** Configure the SSG to point to the bundled RADIUS server using the **show run** command to change the settings shown in Table A-8.

Table A-8 Required Attributes for Installation

Component	Attribute	Value
RADIUS server	Primary AAA server IP address or host name	localhost
	Primary AAA server port	1813
	Secondary AAA server IP address or host name	localhost
	Secondary AAA server port	1813
	Shared secret	cisco
Profile passwords	Service password	servicecisco
	Service group password	servicegroupcisco

Profile File Requirements

The SESM bundled RADIUS server requires a profile file in Merit format.

The default configuration points to the *aaa.properties* file, a sample Merit file installed with the bundled RADIUS server in the RDP/config directory.

The bundled SESM RADIUS server loads the contents of the profile file during startup. You must restart the RADIUS server if you:

- Change the `aaaFilename` attribute to point to a different file.
- Make any changes to the profiles in the referenced file.

Communication with Components in the Deployment

Follow the instructions in the *Cisco Subscriber Edge Services Manager Deployment Guide* to configure other components in the deployment to communicate with the RADIUS server.

- In a RADIUS installation deployment, the following components must communicate with the RADIUS server:
 - SESM Web Portal (NWSP)
 - SSG
- In an SPE installation deployment, the bundled RADIUS server is configured to use the SPE.

Running Bundled RADIUS Server

To run SESM using the bundled RADIUS server, navigate to the tools/bin directory and then enter one of the commands, shown in Table A-9.

Table A-9 Running Bundled RADIUS Server

Platform	Command
Windows	startAAA.cmd
Linux, Solaris, and Unix	startAAA.sh

Demo Profile File

The Demo Profile file contains sample profiles to support the SESM portals running as Demo installation or the bundled RADIUS server.

The Demo Profile file enables you to:

- See the services and features associated with each Demo user ID.
- See examples of the vendor-specific attributes (VSAs) that SESM and SSG require in a RADIUS database.
- Add new profiles or change existing ones to enhance your demonstration.
- Use the profiles in the Demo Profile files as test data for SESM deployed as a RADIUS installation.



Note

If you change the name or location of the Demo Profile file, you must reflect this change in the demoDataFile attribute in the SESMDemoMode MBean in the portal's XML file.

Installed Path Names of Demo Profile Files

SESM provides a different Demo Profile file for each sample portal application. Each Demo Profile file contains profiles that illustrate specific features of the sample application. The installed Demo Profile files are listed in Table A-10.

Table A-10 Demo Profile File Installed Path Names

SESM Portal	Demo Profile File
NWSP	nwsp/config/aaa.properties
WAP	wap/config/aaa.properties
PDA	pda/config/aaa.properties
RDP	rdp/config/aaa.properties

File Contents and Format

The Demo Profile files contain example subscriber profiles, service profiles, and service group profiles that support the SESM sample applications when they are running as a Demo installation. The file is in Merit RADIUS flat-file format and includes profiles that use the following types of attributes:

- RADIUS standard attributes.
- SSG vendor-specific attributes.
- SESM demonstration attributes (attributes reserved for SESM use to simulate features available only in SPE installations).

Subscriber profile attributes are described in [Subscriber Profile Attributes, page A-12](#).

Service profile attributes are described in [Service Profile Attributes, page A-15](#).

For details of subscriber and service profiles, refer to the Demo Profile file `aaa.properties` in the `tools/config` directory.

Subscriber Profile Attributes



Note

For details of subscriber profiles, refer to the Demo Profile file `aaa.properties` in the `tools/config` directory.

Table A-11 Subscriber Profile Attributes

Attribute	Details	Examples
\$PE	Permission.	Account-Info = "\$PESelf Manage", Account-Info = "\$PESubaccount Manage", Account-Info = "\$PEService Subscription", Account-Info = "\$PEFirewall Manage",
\$SA	Service available for subscription.	Account-Info = "\$SAbanking",
\$GA	Service group available for subscription.	Account-Info = "\$GAnewsgroup",
\$AA	Account attribute.	Account-Info = "\$AAinitials;V;{A}", Account-Info = "\$AAgender;S;female", Account-Info = "\$AAsurname;S;Goodbody", Account-Info = "\$AAtitle;S;Miss", Account-Info = "\$AAgivenName;S;Felicity", Account-Info = "\$AAhobbies;V;{science;news;travel}",
\$UG	User group.	Account-Info = "\$UGgold",
\$FA	Parent account.	Account-Info = "\$FAGolduser"
\$SB	Service unavailable for subscription (blocked).	Account-Info = "\$SBinternet_gold"
\$GB	Service group unavailable for subscription (blocked).	Account-Info = "\$GBnews"
\$SL	Subaccount creation limit.	Account-Info = "\$SL0"

See [Table A-12](#) for examples of how to use these attributes.

Table A-12 Example SSG Account-Info Attributes

Sub Attribute	Example	Description
A	Ainternet_gold	Specifies the name of the service that the user is automatically logged on into after an Account-Logon. This is configured in the user profile and is present in Access-Accept packets and can appear multiple times.
N	Ninternet_gold NSSG-Example-Service	Specifies the name of the service that a host is subscribed to and can appear multiple times.
RI	RIInitial;30	Used to specify the TCP-redirection configuration for the user. It has three sub-attributes, one for SMTP redirection (RS), one for initial captivation (RI) and one for periodic advertising captivation (RA). Each can appear at most once. <i>Advert</i> and <i>Initial</i> refer to the server groups defined in the SSG configuration to which the packets will be redirected (initial captivation).
RI	Account-Info = "RIInitial;30;Internet "	Begin initial captivation when user logs on to service <i>Internet</i> as opposed to starting it on account-logon.
RA	Account-Info = "RAAdvert;15;600", or Account-Info = "RAAdvert;15;600;Inter net"	Perform advertisement redirection only when user is logged on to service <i>Internet</i> . Advertisement redirection does not happen if user is not logged on to that service.
RS	Account-Info = "RS",	The SMTP forwarding attribute is meaningful only when the CLI <i>redirect smtp to <group> user</i> is enabled on SSG. See <i>ssgconfig.txt</i> for examples.
QU	Account-Info = "QU;640000;16000;32000 ;D;640000;16000;32000" ,	This attribute is used to specify the Quality of Service (QoS) parameters for the host in both the upstream (U) and downstream (D) directions. This is configured in the user profile and is present in Access-Accept messages and can appear only once.
H	Account-Info = "Hhttp://www.spiderbai t.com",	This attribute is the default URL that is requested after a subscriber successfully authenticates in a Web Portal.

Table A-12 Example SSG Account-Info Attributes (continued)

Sub Attribute	Example	Description
av-pair	av-pair = "ip:inacl#101=permit tcp any any eq www",	Personal firewalls standard Cisco IOS. This is a list of all entries that will be applied to the users who logged onto the SSG to permit or exclude certain types of traffic.
	av-pair = "ip:outacl#101=permit tcp any any established",	
	av-pair = "ip:inacl#196=deny ip any any",	
	av-pair = "ip:outacl#196=deny ip any any",	
	av-pair = "ip:outacl#129=permit tcp any any established",	
	av-pair = "ip:inacl#138=permit tcp any any eq 80",	
	av-pair = "ip:inacl#138=permit tcp any any eq 443",	
	av-pair = "ip:addr-pool=ppp-gree n"	

Service Profile Attributes


Note

For details of service profiles, please refer to the Demo Profile file *aaa.properties* in the *tools/config* directory.

Table A-13 Example SSG Service-Info Attributes

Attribute	Example	Description
I	Service-Info = "IComplete Example",	This is a service description attribute used by SESM to display a suitable name to users on the Web Portal.
M	Service-Info = "MC",	This attribute is used to specify the mode of access to a service. If the mode is sequential (MS), a user who is already logged on to another service cannot access this service. If the user is logged on to a sequential service, then no other service can be accessed. This attribute can appear only once. If this attribute is not configured in the service profile, then the default mode for the service is concurrent.
R	Service-Info = "R199.199.199.0;255.255.255.0", Service-Info = "R199.199.199.115;255.255.255.0;E",	This attribute is used to specify the networks that belong to a service. The network is either an include network or an exclude network. Users are not allowed to access exclude networks. This is configured in the service profile and must appear at least once.
O	Service-Info = "Ocisco.com",	This attribute is used to specify the domains that are part of the service. If a user is connected to this service, all Domain Name System (DNS) queries to this domain are redirected to the DNS server for this service. This attribute is configured in the service profile and can appear multiple times.
D	Service-Info = "D64.103.101.184;144.254.71.184",	This attribute is used to set the DNS server IP address for the service. Two DNS servers, primary and secondary, can be specified using this attribute. This is configured in the service profile and can appear only once.
Q	Service-Info = "QU;640000;16000;32000;D;640000;16000;32000",	This attribute is used to set the upstream and downstream QoS parameters for a connection to the service. This attribute is configured in the service profile and can only appear once.
S	Service-Info = "S10.0.101.1;1645;1646;cisco",	This attribute is used to specify the RADIUS server to be used for authentication for the service. This is used only for proxy services. Multiple instances of this attribute can be used to configure multiple servers.

Table A-13 Example SSG Service-Info Attributes (continued)

Attribute	Example	Description
T	Service-Info = "TP",	This attribute is used to specify the type of the service. A service can be of <i>Proxy</i> (X), <i>Passthrough</i> (P) or <i>Tunnel</i> (T) type. The default type of a service is <i>Passthrough</i> if this attribute is not set in the service profile.
B	Service-Info = "B100",	This attribute is used to specify the maximum transmission unit (MTU) for a Layer Two Tunneling Protocol (L2TP) tunnel service. This is configured in the tunnel service profile and can appear only once.
G	Service-Info = "Ga_next_hop_table",	This attribute is used to set the next-hop gateway for the SSG service. This attribute is configured in the service profile and can appear only once. The string specified in this attribute is used to key off a next-hop table on the SSG to find the next-hop gateway IP address. If this attribute is not configured, the service name is used as the key to find the next-hop IP address.
L	Service-Info = "L10",	This attribute is used to set the accounting interval for interim accounting for connections to this service. This attribute can be present only once. If this attribute is not configured in the service profile, then the global SSG accounting interval configured in the SSG is used.
U	Service-Info = "Ucisco",	This attribute is used to specify the username in connection Accounting requests. The Accounting requests to the local AAA server contain the host's username, and the Accounting requests to the remote AAA server for proxy services contain the username that the user used to log on to the service.
V	Service-Info = "Va_cookie_string",	This attribute is used to specify a cookie string for a service. This string is sent in all Access-Requests for authentication for a connection and also in all Accounting-Requests for the connections to this service. This attribute is configured in the service profile and can appear only once.
X	Service-Info = "X",	If this attribute is set for a service, then the service name is appended to the username during authentication to the service as <i>username@servicename</i> . This attribute is configured in a service profile and can appear only once.
H	Service-Info = "Hhttp://www.cnn.com",	This attribute is used to specify a URL that will be requested when this service is activated via a Web Portal.

Table A-13 Example SSG Service-Info Attributes (continued)

Attribute	Example	Description
R	Service-Info = "\$RSPprimaryServiceName",	This is a SESM-specific attribute to indicate that this service should be activated only after the specified required service activation. If there are several of these attributes, then any one of the required activations is sufficient. If the attribute specifies no service, then the attribute indicates that this service requires activation on the edge device. The equivalent Account-Info attribute may be used for a service group, but a required service for activation has to be specified. Only used by UI to control, for example, service list and subscription pages.
av-pair	av-pair = "ip:inacl#101=permit tcp any any eq www", av-pair = "ip:outacl#101=permit tcp any any established"	av pairs applied to the service accurately define what a user connected to this service can access. In the example, the avpair is permitting access to the HTTP port for that service so you can browse, but you cannot Telnet, FTP or send an email.

Downloading International Character Sets

To support localization, SESM uses Unicode Transformation Format Version 8 (UTF-8) character representations. UTF-8 supports both single-byte and double-byte character sets. If your browser does not display the characters for the language that you have chosen on the NWSP Settings page, you must download the character set from the browser vendor's Internet site. For example, to download the Japanese character set, go to one of the following websites:

- For the Internet Explorer browser, go to the Microsoft website.
- For the Netscape browser, go to the Netscape website.

For instructions on localizing a SESM portal, including how to construct translated resource bundles and images for buttons, see [Chapter 5, "SESM Internationalization and Localization"](#).



SESM Tag Libraries

This appendix provides information on configuring a SESM tag library and describes the SESM tag libraries other than the Localization tag library:

- [Iterator Tag Library, page B-2](#)
- [Navigator Tag Library, page B-3](#)
- [Shape Tag Library, page B-4](#)

For information on the Localization tag library, see the [Using the Localization Tag Library, page 5-6](#).

Configuring a Tag Library

If you are using a SESM tag library in a web application, you must perform the following steps to configure the tag library.

For a SESM Web Portal application, you do not need to perform this procedure. The SESM tag libraries are already defined in the web.xml file of each SESM Web Portal application. The tag library descriptor files and the com.cisco.sesm.contextlib.jar file already exist in, respectively, the \WEB-INF and \WEB-INF\lib directories.

-
- Step 1** Copy the tag library's descriptor file from `\install_dir\nwsp\webapp\WEB-INF` to the \WEB-INF directory of your Web Portal application. [Table B-1](#) lists the tag library descriptor files.

Table B-1 *SESM Tag Library Descriptor Files*

SESM Tag Library	Tag Library Descriptor File
Iterator	iterator.tld
Localization	localization.tld
Navigator	navigator.tld
Shape	shape.tld

- Step 2** Copy the com.cisco.sesm.contextlib.jar from `\install_dir\nwsp\webapp\WEB-INF\lib` directory to the \WEB-INF\lib directory of your Web Portal application.

- Step 3** Add the appropriate <taglib> element to your Web Portal application deployment descriptor in \WEB-INF\web.xml. You can cut and paste the <taglib> elements for the libraries from the web.xml file for NWSP. 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" %>
```

Iterator Tag Library

The Iterator tag library provides tags that allow a JSP page to iterate over a Java collection. The Iterator tag library implements a java.util.Iterator and includes these tags:

- start—Marks the start and end of an iteration.
- val—Obtains the value of the current element of an iteration loop.

start Tag

Table B-2 lists the attributes of the start tag. The start tag uses either the iteration instance given in over, or the Iterator instance given in value. You must supply either the over or value attribute.

Table B-2 Start Tag Attributes

Attribute	Description	Required	Runtime Expression
type	Specifies the class of the variable given in name.	Yes	No
name	Specifies a variable name for accessing the value of the current element of the iteration.	Yes	No
over	Provides an instance of an iteration. The start tag invocation must include either over or value.	No	Yes
value	Provides an instance of an Iterator. The start tag invocation must include either over or value.	No	Yes

The following example uses the start tag to iterate through a collection of messages.

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

<it:start over="<%=messagesBean.getMessage()%" name="message"
type="com.cisco.sesm.i18n.I18nObject">
  <tr>
  <td><l10n:format shortDateTime="<%= new Date()%">timestamp</l10n:format></td>
  <td><l10n:format object="<%= message %">message</l10n:format></td>
  </tr>
</it:start>
```

In the preceding example, the `messages` variable is used to access the current value of the iteration. Each message is an internationalized object of the type `com.cisco.sesm.i18n.I18nObject`.

val Tag

[Table B-3](#) lists the attributes of the `val` tag. If you specify the `val` tag without the `value` attribute, the tag obtains the value of the current element of an iteration loop.

Table B-3 Val Tag Attributes

Attribute	Description	Required	Runtime Expression
<code>value</code>	Specifies Java code to evaluate for each iteration through the collection. The <code>value</code> attribute is optional. When a <code>val</code> tag is specified but no <code>value</code> attribute is given, the value of this iteration is the value of the current element.	No	Yes

The following example uses the `start` and `val` tags to iterate through a collection of titles.

```
<html>
  <%@ page import="java.util.*" %>
  <%@ taglib uri="http://www.cisco.com/taglibs/iterator" prefix="it" %>

  <head>
    <title>Iterator Example</title>
  </head>
  <body>
    <%
      Vector titles = ... A vector of titles of type java.lang.String
    %>
    <table>
      <!-- Define a header row for the table -->
      <tr>
        <th>Title</th>
        <th>Description</th>
      </tr>

      <!-- Iterate over titles. The current value will be in a variable called title -->
      <it:start over="<%=titles%>" name="title" type="java.lang.String">
        <tr>
          <td><it:val>a title</it:val></td>
          <td><it:val value="<%=getDescription(title)%>">a
description</it:val></td>
        </tr>
      </it:start>

    </table>
  </body>
</html>
```

Navigator Tag Library

The Navigator tag library and its `decorate` tag enable a JSP page to invoke a decorator if decoration is needed.

decorate Tag

The decorate tag invokes a decorator if decoration is needed. The decorate tag is an efficient way to invoke a decorator because the tag determines whether decoration is needed by calling the decorator's `decorateIfNecessary` method. For information on decorators and `decorateIfNecessary`, see the description of the `Decorator` class in the Javadoc documentation for SESM.

The decorator specified in the tag's name attribute must be declared as a servlet in the Web Portal application's `web.xml` file. For example, the servlet name `Cookie` is declared in the following entry to be associated with the class `com.cisco.sesm.navigator.CookieDecorator`.

```
<servlet>
  <servlet-name>Cookie</servlet-name>
  <servlet-class>com.cisco.sesm.navigator.CookieDecorator</servlet-class>
  <load-on-startup>1</load-on-startup>
</servlet>
```

The decorator specified in the name attribute must be registered with the `DecoratorPool` method. If it is not registered, the decorate tag throws a `JspException`.

Table B-4 lists the attributes of the decorate tag.

Table B-4 *decorate Tag Attributes*

Attribute	Description	Required	Runtime Expression
name	Specifies the decorator to invoke if decoration is needed.	Yes	Yes

The following example uses the decorate tag to invoke the decorator having the servlet name `Cookie`.

```
<%@ taglib uri="http://www.cisco.com/taglibs/navigator" prefix="nav" %>
...
<nav:decorate name="Cookie" />
```

The effect of the preceding decorate tag is to call the `CookieDecorator.decorateIfNecessary` method. Any pre-decorators or post-decorators specified for `CookieDecorator` in the `web.xml` file are also invoked. For information on specifying pre-decorators and post-decorators, see the [Using the preDecorate and postDecorate Parameters, page C-2](#).

Shape Tag Library

The Shape tag library provides a set of tags that translate a virtual filename into an actual filename according to the current values for the dimensions of the user shape. The JSPs of a SESM Web Portal application use the tags of the Shape tag library when retrieving a shape-specific web resource. For information on the user shapes and finding an actual file, see the [User Shapes and User-Shape Decoration, page 3-8](#) and the [Mapping a Virtual File Name to an Actual File Name, page 3-37](#).

The Shape tag library includes these tags:

- `file`—Translates a virtual file name into an actual file name according to the current values for the dimensions of the user shape.
- `path`—Translates a virtual file name into an actual file name according to the current values for the dimensions of the user shape.

The path tag provides some functionality that is not provided by the file tag and that is useful in Dreamweaver Design View (as opposed to Dreamweaver Live Data View).

**Tip**

Use the file tag, in most situations, rather than the path tag. The file tag does not evaluate the tag body and, therefore, is more efficient. The exception is that you should use the path tag when linking a JSP page to a style sheet. The file tag does not work for linking to a style sheet.

file Tag

The file tag translates a virtual file name into an actual file name according to the current values for the dimensions of the user shape. The actual file name is a Uniform Resource Identifier (URI) identifying a web resource. A Cisco SESM Web Portal application uses the user-shape mechanism for customizing the web resources served to a subscriber. In its JSPs, the Web Portal application uses the file tag to specify each web resource that can differ according to the user's shape.

For example, if the banner.jpg file can be different depending on the user's shape, a SESM Web Portal application uses the file tag in a JSP page when specifying that resource:

```

...
<td></td>
```

In the example, the file tag translates the path for the virtual file `/images/nwsp_mid_banner.jpg` into an actual file name (for example, `/gold/de/images/nwsp_mid_banner.jpg`) based on the values of the dimensions of the user's shape.

path Tag



Note

Use the file tag, in most situations, instead of the path tag. The file tag does not evaluate the tag body and, therefore, is more efficient. The exception is that, in a production SESM Web Portal application, you should use the path tag when linking a JSP page to a style sheet. The file tag does not work for linking to a style sheet.

The `path` tag translates a virtual file name into an actual file name according to the current values for the dimensions of the user shape. The actual file name is a URI that identifies an existing web resource. The path for the virtual file is given in an HTML tag attribute, such as `src` or `href`.

In the following example, the `path` tag evaluates its tag body and replaces the virtual file name for the style sheet specified for the `href` attribute with an actual file name:

```
<shape:path attrib="href">
  <link rel="stylesheet" href="/styles/sesm.css" type="text/css">
</shape:path>
```

For example, if you determine the user shape by the `brand` dimension, the `path` tag might locate the style sheet for a gold-brand user at `/gold/styles/sesm.css`. In this case, the preceding example generates the following:

```
<link rel="stylesheet" href="/gold/styles/sesm.css" type="text/css">
```

The `path` tag can be useful in the following situation:

- If you use Dreamweaver's Design View mode (as opposed to Live Data View) to edit a JSP page.
- If the runtime value for a web resource will be determined dynamically and differ from the value specified for an HTML tag attribute such as `src`, `href`, and `background`.

In Design View, the value specified for an attribute such as `src` can cause Dreamweaver to display a broken link if the file does not exist at the specified location because the actual file name (URI) is determined at runtime.

If you specify a file attribute, the `path` tag evaluates its tag body at runtime and replaces the specified value of the `src` attribute with the value specified in the file attribute. Consider the following example:

```
<shape:path file="<%=getCurrentImageName(request, name)%>"
  
</shape:path>
```

At runtime, the preceding example produces the following virtual file name, which the `path` tag then translates into an actual file name (URI) based on the user shape.

```

```

In Design View mode, Dreamweaver uses the tag body, and the preceding example produces this result:

```

```

[Table B-6](#) lists the attributes of the `path` tag.

**Note**

If you use the `id` attribute when the body of the path tag has dynamic content, different `id` values are needed to reflect the possible states of the dynamic content. With dynamic content, if the `id` attribute is used with the same value for all states, you must clear the cache in order to see a change to the dynamic content.

Table B-6 Path Tag Attributes

Attribute	Description	Required	Runtime Expression
<code>attrib</code>	Defines the HTML tag attribute (for example, <code>src</code> , <code>href</code> , or <code>background</code>) that appears in the path tag body and whose value will be translated into an actual file name (a URI) at runtime. If no <code>attrib</code> attribute is given, the HTML tag attribute whose value is translated is <code>src</code> .	No	Yes
<code>file</code>	Specifies a replacement path for the file that will provide the web resource. At runtime, the replacement path given for <code>file</code> is the virtual file name that the SESM software translates into an actual file name (a URI) based on the user shape.	No	Yes
<code>id</code>	<p>Provides caching of the web resource. If you do not use the <code>id</code> attribute in an invocation of the path tag, a new instance of the path tag object is created every time the JSP page with the invocation is requested. The <code>id</code> value specifies a unique identifier for the path tag. For example:</p> <pre><shape:path id="UP1" </shape:path></pre> <p>When you specify an <code>id</code> attribute, the path tag reuses the results of processing the initial instance of the tag if an identical use of the path tag is encountered. With the <code>id</code> attribute, tag instances are considered identical if the user shape in the first tag invocation matches the user shape in the second tag invocation.</p> <p>The value given for <code>id</code> must be globally unique within the scope of the Web Portal application.</p>	No	Yes
<code>idFile</code>	<p>Provides caching of the web resource. The <code>idFile</code> value specifies a static replacement path for the file that will provide the web resource. At runtime, the replacement path given for <code>idFile</code> is the virtual file name that the SESM software translates into an actual file name (a URI) based on the user shape. The <code>idFile</code> attribute can be used instead of the <code>id</code> attribute when a runtime computation is not used for the replacement path.</p> <p>If an <code>idFile</code> attribute is specified, the path tag reuses the results of processing the initial instance of the tag if an identical use of the path tag is encountered. With the <code>idFile</code> attribute, tag instances are considered identical if the user shape in the first tag invocation matches the user shape in the second tag invocation.</p> <p>The use of an <code>id</code> or <code>file</code> attribute takes precedence over an <code>idFile</code> attribute.</p>	No	Yes
<code>shape</code>	Specifies a user shape object explicitly rather than using the shape stored in the <code>shape</code> attribute of the current HTML session. The <code>shape</code> attribute of the HTML session is overridden if the <code>shape</code> attribute is specified in this tag or in a previous <code>shape</code> attribute that appears on the same JSP page. The most recent definition for the tag's <code>shape</code> attribute has precedence.	No	Yes



SESM Utility Servlets Quick Reference

This appendix provides brief descriptions of the utility Java servlets that are used in a SESM Web Portal application. This appendix includes the following:

- [Introduction, page C-1](#)
- [Using the preDecorate and postDecorate Parameters, page C-2](#)
- [SESM Utility Servlet Quick Reference, page C-3](#)

Introduction

SESM components include a set of utility Java servlets that the Web Portal application uses to perform a variety of common tasks, such as:

- Ensuring that a session ID is present when an HTTP client does not support cookies.
- Finding a resource customized for the user shape.
- Redirecting a request to an insecure (HTTP) or secure (HTTPS) version of the same request.
- Ensuring that the HTTP client does or does not cache an HTTP response.

Most SESM utility servlets are decorators that extend the Decorator abstract class and are subclasses of the Navigator class. The SESM utility servlets that are decorators have Decorator in their names.

The SESM utility servlets are preprogrammed with specific functionality. They do not require customization, but because they are decorators, you can extend a utility servlet with a post-decorator.

This appendix does not describe two specialized sets of servlets: SESM controls and dimension decorators. You can find information on these servlets in the following locations:

- For overview information on the control servlets, see the [Controls, page 3-3](#).
- For information on the dimension decorators, see the [Decorating a User Shape, page 3-18](#).

All SESM servlet classes including those for dimension decorators and controls are documented in the Javadoc documentation that is installed with SESM.



Tip

A few utility servlets are very helpful for testing and debugging a SESM Web Portal application: Snoop, TestDimensionDecorator, LocaleDecorator, and TestUserDecorator. For information on debugging, see the [Debugging a SESM Web Portal Application, page 2-17](#).

Using the preDecorate and postDecorate Parameters

A SESM utility servlet that is a subclass of either Navigator or Decorator can have preDecorate and postDecorate initialization parameters:

Initialization Parameter	Description
preDecorate	Specifies a list of names for other decorator servlets that are invoked, in the order listed, before the decorator declared in the <servlet-class> attribute is invoked.
postDecorate	Specifies a list of names for other decorator servlets that are invoked, in the order listed, after the decorator declared in the <servlet-class> attribute is invoked.

Depending on whether a servlet being declared in <servlet-class> is a subclass of Navigator or Decorator, the behavior of the preDecorate and postDecorate initialization parameters is different:

- If the servlet named in <servlet-class> is a subclass of Navigator, each decorator in the preDecorate list and the postDecorate list is always invoked. Each decorator is called by invoking its decorateIfNecessary method.
- If the servlet named in <servlet-class> is a subclass of Decorator and if the decoration by this servlet is needed, you invoke each decorator in the preDecorate list and the postDecorate list by calling its decorateIfNecessary method.

If a pre-decorator or post-decorator throws a ServletException, all decoration stops. No further pre-decorators or post-decorators are invoked. The servlet declared in the <servlet-class> attribute is not invoked if it has not already been called. If the servlet declared in the <servlet-class> attribute throws a ServletException, no post-decorators are invoked.

In the web.xml file, you configure the preDecorate and postDecorate initialization parameters in the servlet declaration. Consider the following declaration for MyAccountView that specifies the VirtualFile servlet:

```
<servlet>
  <servlet-name>MyAccountView</servlet-name>
  <servlet-class>com.cisco.sesm.navigator.VirtualFile</servlet-class>
  <load-on-startup>1</load-on-startup>
  ...
  <init-param>
    <param-name>preDecorate</param-name>
    <param-value>User, NoCache</param-value>
  </init-param>
</servlet>
```

In the preceding example, the pre-decorators User and NoCache are invoked by calling their decorateIfNecessary methods. With subclasses of Navigator (such as VirtualFile), the pre-decorators and post-decorators *are always invoked*. You call them by invoking their decorateIfNecessary methods.

In the next example, the `UserDecorator` servlet is a subclass of `Decorator`.

```
<servlet>
  <servlet-name>User</servlet-name>
  <servlet-class>
    com.cisco.sesm.webapp.decorator.UserDecorator
  </servlet-class>
  <load-on-startup>1</load-on-startup>
  <init-param>
    <param-name>postDecorate</param-name>
    <param-value>InitUser, RemovePermission, RefreshShape</param-value>
  </init-param>
</servlet>
```

In the preceding example, if decoration by `UserDecorator` is needed, you invoke the post-decorators `InitUser`, `RemovePermission`, and `NoCache` by calling their `decorateIfNecessary` methods. With subclasses of `Decorator` (such as `UserDecorator`), the pre-decorators and post-decorators are invoked only if decoration is needed by the decorator being declared in `<servlet-class>` (in this example, `UserDecorator`).

For more information on the `Navigator` and `Decorator` classes, see the Javadoc documentation for these classes.

SESM Utility Servlet Quick Reference

The following descriptions briefly explain the SESM utility servlets. For more information on a servlet, see the Javadoc for the servlet class. In each description, the servlet mapping entry is the URL-to-servlet mapping that is defined in the NWSP `web.xml` file. You can add a servlet mapping for any servlet or modify an existing mapping.

Alias Servlet

```
Servlet Mapping: None
Class: com.cisco.sesm.navigator.Alias
```

This maps a servlet name to a servlet chain, allowing servlets to be invoked in sequence. The `Alias` servlet's initialization parameters include the following:

Initialization Parameter	Description
<code>to</code>	Specifies the URI to which <code>Alias</code> forwards the request.

In the following example, `Alias` is used with the servlet name `StatusView`:

```
<servlet>
  <servlet-name>StatusView</servlet-name>
  <servlet-class>com.cisco.sesm.navigator.Alias</servlet-class>

  <load-on-startup>1</load-on-startup>
  <init-param>
    <param-name>to</param-name>
    <param-value>/user/nocache/vfile/pages/status.jsp</param-value>
  </init-param>
</servlet>
```

When the servlet named `StatusView` is requested, the `Alias` servlet forwards the HTTP request to the servlet chain `/user/nocache/vfile/pages/status.jsp`. The `to` initialization parameter for the `Alias` servlet specifies the URI to which the request is forwarded. For information on servlet chains, see the [Servlet Chaining, page 3-36](#).

BuildVersion Servlet

Servlet Mapping: `None`
Class: `com.cisco.sesm.webapp.decorator.BuildVersionDecorator`

This discovers the build version and adds a session scope attribute named "buildVersion". The build version is obtained from the configuration files and identifies the version of this software that is running.

The attribute is an internationalized object (I18nObject). The possible values of the internationalized object include resources with keys `versionNotAvailable` and `versionError`.

CacheDecorator Servlet

Servlet Mapping: `/cache/*`
Class: `com.cisco.sesm.navigator.CacheDecorator`

Tells the HTTP client to cache the HTTP response. The response is cached by the HTTP client until the shape of the subscriber changes. The `CacheDecorator` servlet writes HTTP headers into the response indicating that the response is to be cached.

If `CacheDecorator` can determine that the HTTP response content is not required because it is already cached, the request is quickly terminated with an empty response. In this case, `CacheDecorator` does not forward to any other decorator or servlet. The HTTP client will use the response that it has already cached.

For information on preventing caching of the HTTP response, see the `NoCacheDecorator` servlet.

CookieDecorator Servlet

Servlet Mapping: `None`
Class: `com.cisco.sesm.navigator.CookieDecorator`

Adds an attribute named "cookie" to the current HTTP session. The value of the attribute represents the HTTP session ID.

Each SESM JSP page is responsible for inserting the cookie into each URL just before the place where a question mark is located at the beginning of the query-string. If there is no question mark, it is inserted just before the place where a question mark would be located.



Tip

You do not have to modify the NWSP Web Portal application for browsers that do not support cookies. The needed code is already present in the NWSP. If the client browser *does support* cookies, the web server automatically adds a cookie with the HTTP session ID to the HTTP headers. If the client browser *does not support* cookies, the NWSP Web Portal application has the code, where it is needed, to add a session ID to the URLs that require them. The NWSP does not require any additional URL rewriting.

If the HTTP client does not support cookies (for example, because the subscriber has disabled cookies), a SESM Web Portal application uses the `cookie` attribute to write the session ID into every URL that is returned to the client.

If `CookieDecorator` can determine that the HTTP client supports cookies, the `cookie` attribute is set to the empty string. The `cookie` attribute is never null once the Web Portal application invokes `CookieDecorator`.

If this is the first HTTP response for this HTTP session, `CookieDecorator` does not know whether the HTTP client supports cookies in the HTTP headers. In this case, the `cookie` attribute is set to the session ID.

The following example shows a typical use of `CookieDecorator`. (In the NWSP `web.xml` file, `CookieDecorator` is declared with the servlet name `cookie`.) In the example, the JSP-page code does the following:

- Uses the `decorate` tag of the Navigator tag library to call the `CookieDecorator.decorateIfNecessary` method to decorate (if needed) the HTTP session with a `cookie` attribute.
- Declares the `cookie` attribute as a JavaBean instance so that it is accessed as a scripting variable.
- Embeds the `cookie` variable into the URL as `/myAccount<%=cookie%>`.

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

<nav:decorate name="Cookie" />
<jsp:useBean id="cookie" class="java.lang.String" scope="session" />
...
<th><a href="/myAccount<%=cookie%>">My-Account</a></th>
```

Notice that the JSP-page code inserts `<%=cookie%>` into the URL just before the place where a question mark would be located at the beginning of the query-string.

For information on the `decorate` tag, see the [decorate Tag, page B-4](#).

DecoratorPool Servlet

```
Servlet Mapping: /pool/*
Class: com.cisco.sesm.navigator.DecoratorPool
```

Maps from a decorator name to a `Decorator` instance. When a JSP page is used as a decorator, it must register itself in the `jspInit` method by calling `DecoratorPool.register(JspPage)`. For information on JSP-page decorators, see the [Creating or Customizing Dimension Decorators, page 3-25](#).

EndSessionDecorator Servlet

```
Servlet Mapping: /end/*
Class: com.cisco.sesm.navigator.EndSessionDecorator
```

Ends the HTTP session by calling the `HttpSession.invalidate` method.

HttpSniffDecorator Servlet

```
Servlet Mapping: None
Class: com.cisco.sesm.navigator.HttpSniffDecorator
```

Adds an `HttpSniffBean` attribute named `httpSniffBean` to the HTTP session. `HttpSniffBean` contains information about the HTTP client device. `HttpSniffBean` obtains the information from the HTTP headers. For example, `HttpSniffBean` sets the bean property `clientOSName` to the name of the operating system that is running on the HTTP client device. `HttpSniffBean` properties are as follows:

- `clientBrowserName`
- `clientColor`
- `clientDeviceName`
- `clientMarkupLanguage`
- `clientOSName`
- `clientScriptLanguage`

- clientSize
- connection

For information on all HttpSniffBean properties and the possible values of each, see the Javadoc description of the get* methods in the HttpSniffBean class. The Javadoc documentation is installed with SESM.

**Tip**

Many of the HttpSniffBean properties are used to set the values of the dimensions of the user shape. For information on the user-shape dimensions that are set by SESM, see the [SESM-supplied Dimension Decorators, page 3-21](#).

In the NWSP web.xml file, the HttpSniffDecorator servlet is configured with a deployer-customizable post-decorator (httpSniff.jsp) that provides additional *browser sniffing* capabilities. Using this additional information about the client device, httpSniff.jsp and some related JSPs modify HttpSniffBean properties based on the characteristics that it detects. The intention is that the service-provider developer, who has knowledge of the client devices to expect, can modify httpSniff.jsp to provide better information on these devices. For example, the developer could modify httpSniff.jsp to use third-party browser-sniffing software.

In NWSP, httpSniff.jsp uses a JavaScript probe to detect client-device characteristics and sets HttpSniffBean properties to the appropriate values. If the subscriber's client device supports HTML 4, NWSP uses a service list implemented with JavaScript. The httpSniff.jsp also performs some additional checks and sets the relevant bean properties. For example, it sets the clientDeviceName property to wap if it detects a WAP phone simulator.

InitServlet Servlet

Servlet Mapping: None
Class: com.cisco.sesm.core.http.InitServlet

Initializes the SESM model.

InsecureDecorator Servlet

Servlet Mapping: None
Class: com.cisco.sesm.navigator.InsecureDecorator

If the current HTTP request uses Secure Sockets Layer (SSL) encryption, InsecureDecorator redirects the request to an insecure (HTTP) version of the same request.

In the redirect, the new URL is based upon the current request not the original request. The current request is different from the original request when the original request invoked a servlet that has forwarded to the current servlet. The query string in the redirect is the same as the query string in the current request.

If the InsecureDecorator servlet is successful in redirecting the request, it sends the HTTP client an SC_MOVED_TEMPORARILY (302) status code in the response. In addition, InsecureDecorator terminates the current request by throwing a ResponseCompleteNotice, which is wrapped inside a ServletException.

For information redirecting a request with SSL encryption, see the SecureDecorator servlet.

L10nContextDecorator Servlet

Servlet Mapping: /l10n/*
Class: com.cisco.sesm.navigator.L10nContextDecorator

Sets the value of the Web Portal application default localization (L10nContext) context. In the web.xml file, the initialization parameters for L10nContextDecorator define the default values for the current localization context. For information on the L0nContextDecorator servlet initialization parameters, see the [Setting a Default Localization Context, page 5-14](#).

LocaleDecorator Servlet

Servlet Mapping: Varies
Class: com.cisco.sesm.navigator.LocaleDecorator

Sets the language, country, and (if used) variant of two HTTP session attributes:

- The current localization context (L10nContext) attribute—com.cisco.aggbu.l10n.context
- The locale dimension of the user shape attribute—shape

The LocaleDecorator servlet's initialization parameters include the following:

Initialization Parameter	Description
language	Specifies the language.
country	Specifies the country.
variant	Specifies the variant (if any).

In the web.xml file, the initialization parameters in a servlet declaration for a locale decorator define the locale. In the NWSP web.xml file, the servlet declaration for France is:

```
<servlet>
  <servlet-name>France</servlet-name>
  <servlet-class>com.cisco.sesm.navigator.LocaleDecorator</servlet-class>
  <load-on-startup>1</load-on-startup>
  <init-param>
    <param-name>language</param-name>
    <param-value>fr</param-value>
  </init-param>
  <init-param>
    <param-name>country</param-name>
    <param-value>FR</param-value>
  </init-param>
  ...
</servlet>
```

In the web.xml file, LocaleDecorator is used with a servlet mapping, which indicates the locale value that will be tested. For example:

```
<servlet-mapping>
  <servlet-name>France</servlet-name>
  <url-pattern>/locale=fr/*</url-pattern>
</servlet-mapping>
```

Given the preceding servlet declaration and servlet mapping for France, the following URL would invoke LocaleDecorator and set the language and country of the current localization context (L10nContext) to fr and FR, respectively. It also sets the locale dimension of the user shape to fr/FR.

http://someserver:8080/locale=fr/home

MemoryCheckDecorator Servlet

Servlet Mapping: None
Class: `com.cisco.sesm.webapp.decorator.MemoryCheckDecorator`

Determines whether there is sufficient memory on the web server to create a `SESMSession`. If there is sufficient memory, `MemoryCheckDecorator` sets the request attribute `MemoryOK` to `Boolean.TRUE`.

If there is not sufficient memory, `MemoryCheckDecorator` first attempts to free memory on the web server. If there is still not sufficient memory to create a `SESMSession`, the servlet does the following:

- Forwards the request to the `busyURL`
- Sets the request attribute `MemoryOK` to `Boolean.FALSE`
- Throws a `ResponseCompleteNotice`

In the `web.xml` file, `busyURL` is an initialization parameter of `MemoryCheckDecorator`. By default, `busyURL` has the value `/pages/serverBusy.html`.

MessageDecorator Servlet

Servlet Mapping: None
Class: `com.cisco.sesm.navigator.MessageDecorator`

Adds a message to the list of messages for the HTTP session by calling `MessagesControl.addMessage(javax.servlet.http.HttpServletRequest, I18nObject)`.

The content of the message is derived from the following request attributes, which are defined by the *Java Servlet Specification Version 2.3*.

- `javax.servlet.error.message`
- `javax.servlet.error.status_code`
- `javax.servlet.error.exception_type`
- `javax.servlet.error.exception`
- `javax.servlet.error.request_uri`

If the response is not committed, `MessageDecorator` forwards to the URL `/messages` and throws a `ResponseCompleteNotice` to prevent further processing.



Note In the `web.xml` file, the deployer must map the URL `/messages` to a JSP page that displays the messages of the HTTP session.

If the response is committed, `MessageDecorator` allows the request to continue. The error message is not displayed until a new request displays all the messages of the HTTP session.

NoCacheDecorator Servlet

Servlet Mapping: `/cache/*`
Class: `com.cisco.sesm.navigator.NoCacheDecorator`

Prevents caching of the HTTP response by the HTTP client. The `NoCacheDecorator` servlet writes HTTP headers into the response indicating that the response is not to be cached.

For information on caching the HTTP response, see the `CacheDecorator` servlet.

OriginalURLDecorator Servlet

Servlet Mapping: None
Class: com.cisco.sesm.navigator.OriginalURLDecorator

Decorates the current HTTP request with an attribute named *originalURL*. The attribute value is the original URL that was sent from the HTTP client to the HTTP server. The original URL is the URL of the current HTTP request before any servlet forwarding takes place.

The *originalURL* attribute contains the complete URL of the request, including any parameters. The parameters are included as a query string regardless of whether the method is GET or POST.

To capture the original URL for later use, a JSP page can use OriginalURLDecorator (whose servlet name in the web.xml file is OriginalURL). For example:

```
<!-- Always capture original URL at start of each request. --%>
<nav:decorate name="OriginalURL" />
<jsp:useBean id="originalURL" type="java.lang.String" scope="request"/>
...
<jsp:param name = "okPage" value = "<%=originalURL%>" />
```

PermissionCheckDecorator Servlet

Servlet Mapping: None
Class: com.cisco.sesm.webapp.decorator.PermissionCheckDecorator

Determines whether the subscriber has a specified permission. The PermissionCheckDecorator servlet's initialization parameters include the following:

Initialization Parameter	Description
permission	Specifies the permission for which to check.

The permissions that can be specified in the permission initialization parameter are:

- *firewallManage*—Permission for managing a personal firewall.
- *selfManage*—Permission needed for modifying account information, such as names, addresses, and so on.
- *serviceSelection*—Permission for service selection.
- *serviceSubscription*—Permission for service subscription.
- *subAccountManage*—Permission for creating, deleting, and managing subaccounts.

PermissionCheckDecorator uses the corresponding permission attribute of permissionBean to check whether the subscriber has the permission. This attribute, in turn, has been set depending on whether the user account has the needed privileges.

For example, if the permission parameter is *selfManage*, the permissionBean.isSelfManage method is called to check the permission.

- If the user has the needed permission, the corresponding session attribute is set to Boolean.TRUE. The session attribute that corresponds to the permission value given in the initialization parameter permission is named permission_value + Permission. For example, the attribute that corresponds to the permission value *firewallManage* is *firewallManagePermission*.
- If the user does not have the needed permission, a ServletException is thrown, which wraps a PermissionFailedException. The resource key used for this exception is PermissionFailedMessage.

PermissionDecorator Servlet

Servlet Mapping: None
 Class: `class com.cisco.sesm.webapp.decorator.PermissionDecorator`

Adds a `permissionBean` to the HTTP session. The bean is added as an HTTP session attribute named *permissionBean*. The `permissionBean` contains Boolean variables indicating whether the subscriber has permission to perform certain Cisco SESM Web Portal tasks, such as creating a subaccount.

The `PermissionDecorator` servlet is invoked on any JSP page where the page needs a `permissionBean` to determine the permissions that the subscriber has. For example, many JSPs in NWSP need knowledge of subscriber permissions to determine which buttons (for example, the My Account and Accounts buttons) to display in the navigation bar.

Table C-1 provides information on the `permissionBean` properties.

Table C-1 *permissionBean Properties*

Bean Property	Description
<code>isFirewallManage</code>	The value <code>true</code> indicates that the subscriber has the permissions needed to manage a personal firewall. The value <code>false</code> indicates that the subscriber does not have the needed permissions.
<code>isSelfManage</code>	The value <code>true</code> indicates that the subscriber has the permissions needed to modify account information, such names, addresses, and so on. The value <code>false</code> indicates that the subscriber does not have the needed permissions.
<code>isServiceSelection</code>	The value <code>true</code> indicates that the subscriber has the permissions needed for selecting services. The value <code>false</code> indicates that the subscriber does not have the needed permissions.
<code>isServiceSubscription</code>	The value <code>true</code> indicates that the subscriber has the permissions needed to subscribe to services. The value <code>false</code> indicates that the subscriber does not have the needed permissions.
<code>isSubAccountManage</code>	The value <code>true</code> indicates that the subscriber has the permissions needed for creating, deleting, and managing subaccounts. The value <code>false</code> indicates that the subscriber does not have the needed permissions.

RedirectRemainder Servlet

Servlet Mapping: `/redirect/*`
 Class: `com.cisco.sesm.navigator.RedirectRemainder`

Sends a redirect response to the HTTP client. The redirection is from the current URL to a new URL, where the new URL is the part of the current URL after the current servlet name.

For example, because `RedirectRemainder` is mapped to the URL pattern `/redirect/*` in the NWSP `web.xml` file, the URL `/redirect/next` redirects to `/next`.

RemoveAttributeDecorator Servlet

Servlet Mapping: None

Class: com.cisco.sesm.navigator.RemoveAttributeDecorator

Removes an attribute from one of the following scopes: request, session, or application. The RemoveAttributeDecorator servlet's initialization parameters include the following:

Initialization Parameter	Description
name	Specifies the name of the attribute.
scope	Specifies the scope of the attribute: request, session, or application.

You can use this decorator to undo the decoration of other decorators. Given the following web.xml file declaration, when RefreshShape is requested, RemoveAttributeDecorator removes the *shape* attribute, which has session scope.

```
<servlet>
  <servlet-name>RefreshShape</servlet-name>
  <servlet-class>
    com.cisco.sesm.navigator.RemoveAttributeDecorator
  </servlet-class>
  <load-on-startup>1</load-on-startup>
  <init-param>
    <param-name>name</param-name>
    <param-value>shape</param-value>
  </init-param>
  <init-param>
    <param-name>scope</param-name>
    <param-value>session</param-value>
  </init-param>
  <...
</servlet>
```

SecureDecorator Servlet

Servlet Mapping: None

Class: com.cisco.sesm.navigator.SecureDecorator

If the current HTTP request does not use Secure Sockets Layer (SSL) encryption, SecureDecorator redirects the request to a secure (HTTPS) version of the same request. As an example, if the request were:

```
http://someserver:8080/home
```

The SecureDecorator servlet redirects the request to:

```
https://someserver:8080/home
```

In the redirect, the new URL is based upon the current request, not the original request. The current request is different from the original request when the original request invoked a servlet that has forwarded to the current servlet. The query string in the redirect is the same as the query string in the current request.

If the SecureDecorator servlet is successful in redirecting the request, the servlet sends the HTTP client an SC_MOVED_TEMPORARILY (302) status code in the response. In addition, SecureDecorator servlet terminates the current request by throwing a ResponseCompleteNotice, which is wrapped inside a ServletException.

For information on redirecting a request without using SSL encryption, see the `InsecureDecorator` servlet.

ServiceDisplayDecorator Servlet

Servlet Mapping: None
Class: `com.cisco.sesm.webapp.decorator.ServiceDisplayDecorator`

Sets the session *attribute* `openWindowName` to the service name, and the session attribute `openWindowURL` to the service URL if it finds a service URL in the profile for the service with the name given by request parameter `service`. When the service starts, the NWSP Web Portal application uses these session attributes to open a popup window with the specified `openWindowURL` in the window HTTP address field and `openWindowName` as the window name.

If an optional request parameter `serviceURL` is present, it will override any URL retrieved from the service profile.

A URL is considered valid only if its length is greater than one.

ShapeDecorator Servlet

Servlet Mapping: `/shape/*`
Class: `com.cisco.sesm.navigator.ShapeDecorator`

Creates a Shape object for the user and ensures that there is a *shape* attribute with a session scope that holds the Shape object. The Shape servlet's initialization parameters include the following:

Initialization Parameter	Description
dimensions	Specifies a set of zero or more dimension IDs that SESM uses when it creates the dimensions for a user shape.

A Shape object encapsulates the set of characteristics that define the web resources available for a specific subscriber. A Shape object consists of one or more dimensions. Each dimension corresponds to a characteristic of the subscriber (for example, the subscriber's browser software). The value of each dimension specifies one or more directories that SESM searches for requested resources for this subscriber.

For information on the ShapeDecorator servlet, see the [Configuring User-Shape Dimensions, page 3-19](#).

Snoop Servlet

Servlet Mapping: `/snoop/*`
Class: `com.cisco.sesm.navigator.Snoop`

For testing purposes, displays information about the HTTP session and request. The content type of the response that Snoop sends is `text/plain` so that you can view the information on different types of devices (for example, HTML browsers and WAP phones).

Snoop is used for debugging a SESM Web Portal application and is not used in production applications.

SESMSessionDecorator Servlet

Servlet Mapping: `/session/*`
Class: `com.cisco.sesm.webapp.decorator.SESMSessionDecorator`

Adds an attribute named `sesmSession` to the current HTTP request. The value of the attribute represents the active SESM session, which has been synchronized with SSG.

If a new *SESMSession* attribute is successfully created and authenticated, `initUser.jsp` executes. The `initUser.jsp` page is a deployer-customizable decorator that you can modify to perform any subscriber-related initialization tasks that need to be accomplished after it is known that the user is authenticated. For example, `initUser.jsp` could be used to set the current localization (`L10nContext`) to the location defined in the subscriber profile.

TestDimensionDecorator Servlet

Servlet Mapping: Varies

Class: `com.cisco.sesm.navigator.TestDimensionDecorator`

For testing purposes, sets a dimension of the user shape to a specified value. The `TestDimensionDecorator` servlet's initialization parameters include the following:

Initialization Parameter	Description
<code>id</code>	Specifies the dimension ID.
<code>value</code>	Specifies a value for the dimension identified by <code>id</code> .

In the `web.xml` file, the initialization parameters in a servlet declaration for the test dimension decorator define the dimension ID and the value. In the NWSP `web.xml` file, the servlet declaration for `WAPDevice` is:

```
<servlet>
  <servlet-name>WAPDevice</servlet-name>
  <servlet-class>com.cisco.sesm.navigator.TestDimensionDecorator</servlet-class>
  <init-param>
    <param-name>id</param-name>
    <param-value>device</param-value>
  </init-param>
  <init-param>
    <param-name>value</param-name>
    <param-value>wap</param-value>
  </init-param>
  ...
</servlet>
```

In the `web.xml` file, `TestDimensionDecorator` is used with a servlet mapping, which indicates the dimension value that will be tested. For example:

```
<servlet-mapping>
  <servlet-name>WAPDevice</servlet-name>
  <url-pattern>/device=wap/*</url-pattern>
</servlet-mapping>
```

Given the preceding servlet declaration and servlet mapping for `WAPDevice`, the following URL would invoke `TestDimensionDecorator` and set the device dimension to `wap`.

n

TestUserDecorator Servlet

Servlet Mapping: Varies

Class: `com.cisco.sesm.navigator.TestUserDecorator`

For testing purposes, authenticates a username and password. The usernames and passwords that can be authenticated with TestUserDecorator are those defined in the aaa.properties file, which is used for demonstration installation. The TestDimensionDecorator servlet's initialization parameters include the following:

Initialization Parameter	Description
username	Specifies a username given in the aaa.properties file.
password	Specifies the password for the user identified in username.

In the web.xml file, the initialization parameters in a servlet declaration for a test user decorator define the username and the password. In the NWSP web.xml file, the servlet declaration for golduser is:

```
<servlet>
  <servlet-name>golduser</servlet-name>
  <servlet-class>com.cisco.sesm.webapp.decorator.TestUserDecorator</servlet-class>
  <init-param>
    <param-name>username</param-name>
    <param-value>golduser</param-value>
  </init-param>
  <init-param>
    <param-name>password</param-name>
    <param-value>cisco</param-value>
  </init-param>
  ...
</servlet>
```

In the web.xml file, TestUserDecorator is used with a servlet mapping, which indicates the username and password value that will be tested. For example:

```
<servlet-mapping>
  <servlet-name>golduser</servlet-name>
  <url-pattern>/user=golduser/*</url-pattern>
</servlet-mapping>
```

Given the preceding servlet declaration and servlet mapping for golduser, the following URL would invoke TestUserDecorator and try to authenticate the username golduser with the password cisco.

`http://someserver:8080/user=golduser/home`

UserDecorator Servlet

```
Servlet Mapping: /user/*
Class: com.cisco.sesm.webapp.decorator.UserDecorator
```

Ensures that the user is known. UserDecorator adds the username as an attribute to the HTTP session. This might require the user to authenticate. If the user is not known, the `getNewUser` method is called.



Tip

Many SESM JSPs use UserDecorator as a pre-decorator to ensure that subscribers cannot view a web page until they are authenticated.

With UserDecorator, decoration is necessary until the user is known and authenticated. SESM/SSG authentication can be lost without notification, leaving the Web Portal application with a username but no SESM authentication. To ensure that there is an authenticated user, invoke UserDecorator. For example, a JSP-page view might not care what the username is, but by invoking UserDecorator, it ensures that the user is authenticated and that authentication takes place.

In NWSP, if the user is unknown, UserDecorator forwards to AccountLogonControl.

VirtualFile Servlet

Servlet Mapping: /vfile/*
Class: com.cisco.sesm.navigator.VirtualFile

Translates a virtual file name into an actual file name according to the current values for the dimensions of the user shape. The actual file name is a URI for a web resource. VirtualFile also tries to find the resource located by the actual file name and, if it finds the resource, forwards the HTTP request to the resource. The VirtualFile servlet's initialization parameters include the following:

Initialization Parameter	Description
vfile	Specifies the path name for a virtual file.

You can specify the virtual file name in either of two ways:

- In the web.xml file, as the value of the vfile initialization parameter to VirtualFile.
- In a JSP page of a SESM Web Portal application, as the remainder of the URL in a servlet chain.

In the web.xml file, you can specify the virtual file name as the value of the vfile initialization parameter to VirtualFile. Many of the NWSP views use this method to invoke VirtualFile. For example:

```
<servlet>
  <servlet-name>AccountLogonView</servlet-name>
  <servlet-class>com.cisco.sesm.navigator.VirtualFile</servlet-class>
  <init-param>
    <param-name>vfile</param-name>
    <param-value>/pages/accountLogon.jsp</param-value>
  </init-param>
  ...
</servlet>
```

In a JSP page of a SESM Web Portal application, the VirtualFile servlet can be invoked when the Web Portal application needs to find a resource that differs according to the user shape. The virtual file name is the remainder of the URL in a servlet chain. In the following example, a URL for /pages/help.jsp contains a servlet chain that invokes VirtualFile (which is mapped to the URL /vfile/*).

```
/user/nocache/vfile/pages/help.jsp
```

In a servlet chain, VirtualFile (vfile) must be located immediately before the virtual file name for the web resource—in the preceding example, immediately before /pages/help.jsp. For more information on the VirtualFile servlet, see [Mapping a Virtual File Name to an Actual File Name, page 3-37](#).



Using the Cisco Navigation Bar Extension

This appendix explains how to install and use the Cisco Navigation Bar (NavBar) extension to Dreamweaver. This appendix includes the following:

- [Introduction, page D-1](#)
- [Creating a Navigation Bar, page D-1](#)
- [Installing the Navigation Bar Extension, page D-4](#)

Introduction

The Cisco Navigation Bar extension changes the behavior of the standard Dreamweaver navigation bar. With the extension, the current web page always displays the appropriate down-state button in the navigation bar. To accomplish this change, a JavaScript function `MM_nbIsActive(name)` determines if the passed image name should be shown in the down state.

The Cisco Navigation Bar extension is named `navbar.mxp` and is located in the `\install_dir\nwsp\webapp\assets` directory.

Creating a Navigation Bar



Note

If your Dreamweaver installation has not been extended for the Cisco Navigation Bar extension, you may need to install the extension for some of the following procedures. For installation directions, see the [Installing the Navigation Bar Extension, page D-4](#).

The manner in which you insert a navigation bar that uses the Cisco Navigation Bar extension varies depending on whether:

- It is a new JSP page.
- The JavaScript for the navigation bar is already on the JSP page itself.
- The JavaScript for the navigation bar is incorporated on the JSP page through an include directive specifying `navbar.js`.

The directions given in the following sections also apply if you want to add the navigation bar to a Dreamweaver template.

**Note**

After you create and insert a navigation bar that uses the Cisco Navigation Bar extension and edit the JavaScript, you cannot use Dreamweaver to modify the navigation bar (by clicking Navigation Bar from the Modify menu). Trying to modify a navigation in this way will corrupt the JavaScript code in the navigation bar's anchor tags (<a>).

New JSP Page

If the JSP page where you will insert the navigation bar is newly created (not an existing JSP page from a SESM Web Portal application), you can add a new navigation bar that uses the Cisco Navigation Bar extension by performing the following steps:

Step 1 Insert the navigation bar in the normal manner with Dreamweaver by selecting **Insert > Interactive Images > Navigation Bar**.

Step 2 In the created JavaScript, ensure that the MM_nbIsActive function is defined as follows:

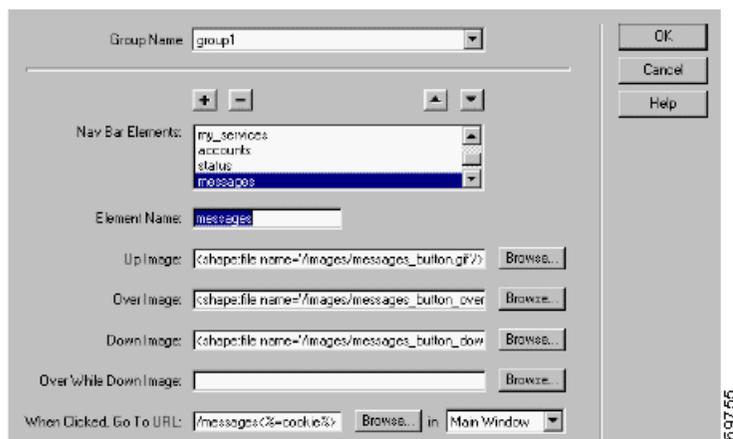
```
function MM_nbIsActive(name) { //v3.0
    return (name == '<%= selectedNavbarButton %>');
}
```

Step 3 Insert the following code above the <head> section of the JSP page where the navigation bar is used. The the navigation bar software uses the code to determine which button should be in the down state on a given page.

```
<% String selectedNavbarButton = "this_element"; %>
```

In the preceding code, *this_element* is the name of an element within the Dreamweaver navigation bar. Each button in the navigation bar has an element name. For example, in the Dreamweaver **Insert Navigation Bar** dialog box shown in [Figure D-1](#), *messages* is the element name.

Figure D-1 Navigation Bar Element Names



The string for `this_element` is different for each JSP page and corresponds to the element name for the page's button in the navigation bar. For example, in the NWSP Web Portal application, the element name of the button that links to the Messages web page is `messages`. In `messages.jsp`, the code would be:

```
<% String selectedNavbarButton = "messages"; %>
```

JSP Page with Existing JavaScript

If an existing JSP page in which you will insert the navigation bar already contains JavaScript for a navigation bar, Dreamweaver will not insert JavaScript for a new navigation bar. In this case, to add a new navigation bar that uses the Cisco Navigation Bar extension, perform the following steps:

-
- Step 1** Copy `navbar.js` from the `\install_dir\nwsp\webapp\decorators` directory into the document tree of the SESM Web Portal application. The `navbar.js` file contains the custom JavaScript functions that the navigation bar requires.
 - Step 2** In the JSP page, delete the JavaScript for the existing navigation bar.
 - Step 3** Insert the following code into the `<head>` section of the JSP page where the navigation bar is used. (If a JSP page is derived from a template that has the code, the code is automatically copied to the JSP page when the page is created.)

```
<script language="JavaScript">  
<%@ include file="/location/navbar.js" %>  
</script>
```

In the preceding code, `location` is the directory where `navbar.js` is located within the document tree of the Web Portal application. For example: `/decorators/navbar.js`.

- Step 4** In the `navbar.js` file, ensure that the `MM_nbIsActive` function is defined as follows:

```
function MM_nbIsActive(name) { //v3.0  
    return (name == '<%= selectedNavbarButton %>');  
}
```

- Step 5** Insert the following code above the `<head>` section of the JSP page where the navigation bar appears. The navigation bar software uses the code to determine which button should be in the down state on a given page.

```
<% String selectedNavbarButton = "this_element"; %>
```

For more information on step 5, see the [New JSP Page, page D-2](#).

JSP Page with Included JavaScript

If an existing JSP page where you will insert the navigation bar already has included navbar.js through the use of an include directive, Dreamweaver inserts the JavaScript for the new navigation bar. In this case, to add a new navigation bar that uses the Cisco Navigation Bar extension, perform the following steps:

Step 1 Delete the JavaScript that Dreamweaver added for the new navigation bar. (The same JavaScript code is contained in the navbar.js file.)

Step 2 Ensure that the MM_nbIsActive function in navbar.js is defined as follows:

```
function MM_nbIsActive(name) { //v3.0
    return (name == '<%= selectedNavbarButton %>');
}
```

Step 3 Ensure that the following code is inserted above the <head> section of the JSP page where the navigation bar appears. The navigation bar software uses the code to determine the button that should be in the down state on a given page.

```
<% String selectedNavbarButton = "this_element"; %>
```

For more information on step 3, see the [New JSP Page, page D-2](#).

Installing the Navigation Bar Extension

When you install the Cisco Navigation Bar extension, you will overwrite a number of Dreamweaver files that are used for creating a standard Dreamweaver navigation bar. Installing the Cisco Navigation Bar extension overwrites the following files:

- \Configuration\Behaviors\Actions\Set Nav Bar Image.js.
- \Configuration\Commands\Insert Nav Bar.htm.
- \Configuration\Commands\Modify Nav Bar.htm.
- \Configuration\Commands\NavigationBar.htm.
- \Configuration\Objects\Common\NavigationBar.js.
- \Configuration\Configuration\Shared\MM\Scripts\navBar.js.

The files are located below the directory where Dreamweaver is located (for example, the C:\Program Files\Macromedia\Dreamweaver MX directory).



Note

If you want to be able to restore an installation of Dreamweaver so that a standard navigation bar can be created, make backup copies of the preceding files under a name like *filename.ORIGINAL* before you install the Cisco Navigation Bar extension.

To install the Cisco Navigation Bar extension, do the following:

-
- Step 1** If you have not already done so, install the Macromedia Extension Manager software, which can be downloaded from the Macromedia website: www.macromedia.com.
- Step 2** Copy the Cisco Navigation Bar extension package file (navbar.mxp) from the `\install_dir\nwsp\webapp\assets` to the Extensions directory, which is located below the directory where Dreamweaver is located. For example:
- C:\Program Files\Macromedia\Dreamweaver MX\Extensions
- Step 3** Start Macromedia Extension Manager, select **Extension Manager File > Install Extension** from the , and install the Cisco Navigation Bar extension (Cisco NavBar modification).
-



Numerics

3-key authentication [3-7](#)

A

aaa.properties file [2-13](#), [A-10](#)

aaa.xml [A-8](#)

accountLogoff.jsp [3-6](#), [4-8](#)

AccountLogoffControl [3-6](#)

AccountLogoffView [3-6](#)

accountLogon.jsp [4-8](#)

accountLogon3Key.jsp [3-7](#), [4-8](#)

accountLogon3KeyBody.jsp [4-8](#), [4-9](#)

AccountLogon3KeyControl [3-7](#)

accountLogonBody.jsp [4-8](#), [4-9](#)

account management [2-14](#), [4-2](#), [4-9](#)

action descriptions for services [4-13](#)

actual file names [3-4](#), [3-37](#), [3-39](#), [B-4](#), [C-15](#)

addPreDecorator method [3-40](#)

advanced customizations [3-1](#)

advertising redirection [4-27](#), [4-29](#)

Alias servlet [C-3](#)

assets directory [4-4](#)

attrib attribute [B-7](#)

attribute 18 messages

 customizing [2-5 to 2-6](#)

attributes for requests, sessions, or applications [C-11](#)

attributes in a RADIUS file [2-13](#)

auth.jar file [2-9](#)

authentication [C-15](#)

 3 keys [3-7](#)

authentication.jar file [2-9](#)

B

back-end tier [1-4](#)

background colors [2-1](#)

bannerOnlyTemplate.dwt template [4-21](#)

banners [2-3](#), [4-20](#)

base name of a properties file [5-4](#)

basic customizations

 SESM web portal applications [2-1](#)

body JSP pages [4-7](#)

brand dimension [3-11](#), [3-15](#)

 testing [2-20](#)

brandDimension.jsp page [3-21](#), [3-29](#)

branding [4-22](#)

 by location [3-26](#)

 requirements [1-7](#), [2-1](#)

browsers [A-3](#)

 determining characteristics [C-6](#)

BuildVersion servlet [C-4](#)

Bundled RADIUS server [A-8](#)

bundled RADIUS server [A-8](#)

 installing [A-9](#)

 running [A-11](#)

buttons [2-4](#), [4-7](#)

C

CacheDecorator servlet [4-16](#), [C-4](#)

caching HTTP responses [C-4](#), [C-8](#)

cancelButton.jsp [4-7](#)

Captive Portal

 configuring [4-27](#)

 demo installation [2-13](#)

- redirection types [4-27](#)
 - solution [4-24](#)
 - web application [1-5, 4-25, 4-26, 4-28](#)
 - captiveportal.xml file [4-28, 4-29](#)
 - captive portal solution
 - configuration files [1-9](#)
 - Cascading Style Sheets [1-12, 2-4, 4-5](#)
 - certificates for security [4-9](#)
 - chaining servlets [3-36](#)
 - Chinese language [2-16, 4-6](#)
 - Cisco Distributed Administration Tool (CDAT) [1-2](#)
 - Cisco Navigation Bar extension [D-1](#)
 - creating a navigation bar [D-1](#)
 - installing [D-4](#)
 - class libraries [2-8](#)
 - class loaders [5-6](#)
 - CLASSPATH variable [5-4](#)
 - code attribute [5-9](#)
 - collections [B-2](#)
 - ColorDimensionDecorator servlet [3-21](#)
 - com.cisco.aggbu.contextlib.jar file [B-1](#)
 - com.cisco.sesm JAR files [2-8](#)
 - compiling classes [2-9](#)
 - compiling JSP pages [2-7, 2-10](#)
 - config directory [4-4](#)
 - configuration files
 - See also J2EE; MBeans
 - configuration files, customizing [1-8](#)
 - configuring
 - customized SESM web portal [1-8](#)
 - configuring a tag library [B-1](#)
 - confirmBody.jsp [4-8](#)
 - connecting to services [4-2, 4-22, 4-24](#)
 - ConnectionDimensionDecorator servlet [3-21](#)
 - content web applications for Captive Portal [4-25, 4-29](#)
 - context attribute [5-8, 5-10](#)
 - context tag [5-7](#)
 - Control class [3-3](#)
 - controls [3-3, 3-6, 3-8, 3-31](#)
 - internationalized resources [5-5](#)
 - invoking decorators [3-40](#)
 - logical [3-34](#)
 - converting values with format tags [5-11](#)
 - CookieDecorator servlet [C-4](#)
 - cookies [C-4](#)
 - countries [5-9](#)
 - country name of a properties file [5-4](#)
 - country parameter [C-7](#)
 - country tag [5-9](#)
 - CPDURATION parameter [4-29, 4-30](#)
 - CPSUBSCRIBER parameter [4-29](#)
 - CPURL parameter [4-28, 4-29, 4-30](#)
 - currencies [5-11](#)
 - currency attribute [5-11](#)
 - customization
 - iPass logon [2-6](#)
 - customized SESM web portal
 - configuring [1-8](#)
 - customizing
 - attribute 18 messages [2-5 to 2-6](#)
 - configuration files [1-8, 1-9](#)
 - web portals [1-8](#)
 - customizing SESM web portal applications [1-6, 3-1](#)
 - designer and developer [1-4](#)
 - levels of customization [1-7](#)
-
- ## D
- date attribute [5-11](#)
 - dates [5-11](#)
 - dateTime attribute [5-11](#)
 - debugging [2-17](#)
 - decorateIfNecessary method [3-9, 3-31, 3-41, C-2](#)
 - decorate method [3-9, 3-25, 3-28, 3-31, 3-33](#)
 - decorate tag [3-28, 3-31, 3-33, 3-40, 3-41, B-4](#)
 - decoration of the user shape [3-2, 3-9, 3-18](#)
 - decoration with decorate tag [B-4](#)
 - DecoratorByJSP class [3-31](#)

- Decorator class [3-9, C-1, C-2](#)
- decorator components [3-9, 3-18](#)
- decorator directory [4-4](#)
- decorator names [C-5](#)
- DecoratorPool.register method [3-27, 3-33](#)
- DecoratorPool servlet [3-28, 3-34, B-4, C-5](#)
- decorators [3-9](#)
 - creating [3-31, 3-32](#)
 - declaring [3-35](#)
 - invoking [3-40, 3-41](#)
- decryption and Secure Sockets Layer [4-9](#)
- defaultCountry parameter [5-14](#)
- defaultLanguage parameter [5-14](#)
- default localization contexts [3-4, 5-14, 5-15](#)
- defaultResourceBundle parameter [5-14](#)
- defaultValue parameter [3-21](#)
- defaultVariant parameter [5-14](#)
- demoDataFile attribute [A-11](#)
- demo installation
 - description [A-2](#)
 - installing [A-3](#)
 - install options [A-3](#)
 - logging on [A-5](#)
 - quick start [A-3](#)
 - user IDs [A-5](#)
 - web portals [A-3](#)
- Demo Profile File [A-11](#)
- demo profile file
 - contents and format [A-12](#)
- Demo profile files
 - changing [A-11](#)
 - description [A-11](#)
 - format [A-12](#)
 - pathnames [A-11](#)
- demo profile files
 - path names [A-11](#)
- deployment descriptor files [3-33, 3-34, 3-38, 4-5](#)
 - configuring [3-35](#)
 - tracking changes [2-8](#)
- description of a service [4-13](#)
- descriptor files for tag libraries [B-1](#)
- DESS
 - See *Directory Enabled Service Selection (DESS)*
- dess.jar file [2-9](#)
- developing
 - SESM web portal applications [1-11](#)
- developing SESM applications
 - development tools [1-11](#)
 - hardware and software requirements [1-10](#)
- developing SESM web portal applications [1-10, 2-6](#)
- development tools [1-12](#)
- device dimension [3-11, 3-15, 3-19, C-13](#)
 - testing [2-20](#)
- DeviceDimensionDecorator servlet [3-22, 3-23, 3-24](#)
- Dimension class [3-28](#)
- dimension decorators [3-10, 3-21](#)
 - adding [3-29](#)
 - customizing [3-25](#)
 - JSP pages [3-29](#)
 - modifying [3-28](#)
 - servlets [3-28](#)
 - setting dimension values [3-15](#)
- dimension IDs [3-20, 3-28](#)
 - ordering [3-13](#)
- Dimension object [3-25, 3-28](#)
- dimensions in user shapes [3-9, 3-11, 3-12, 3-15, 3-18, 3-20, C-13, C-15](#)
 - default values [3-21](#)
 - multiple directories [3-17](#)
 - setting dimension values [3-27](#)
- dimensions parameter [3-12 to 3-20, 3-28, C-12](#)
- Directory Enabled Service Selection (DESS)
 - software [1-1](#)
 - See also *Security Policy Engine (SPE)*
- directory hierarchies [1-7, 3-10, 4-3](#)
- docs directory [4-4](#)
- documentation for Java classes [2-10](#)
- double-byte characters [A-3](#)

double-byte character sets [A-17](#)
 double-byte languages [2-16](#)
 Dreamweaver
 Design View [B-6](#)
 live data [2-14](#)
 navigation bars [2-4, 4-17, D-1](#)
 templates [2-5, 4-5, 4-19, 4-21](#)
 web site management [2-22](#)
 Dreamweaver MX [1-11, 1-12](#)
 duration attribute [4-30](#)

E

editable areas [4-20](#)
 eMbedded Visual Tools 3.0 [2-21](#)
 emulators for mobile devices [2-21](#)
 encryption [C-6, C-11](#)
 Secure Sockets Layer [4-9](#)
 EndSessionDecorator servlet [C-5](#)
 enterFragment.jspf page [4-18](#)
 environment variables [1-10](#)
 erp.xml [A-8](#)
 exitFragment.jspf page [4-18](#)

F

file attribute [B-7](#)
 files
 aaa.properties [A-10](#)
 aaa.xml [A-8](#)
 demo profile files [A-11](#)
 erp.xml [A-8](#)
 J2EE configuration [1-9](#)
 web.xml [1-9](#)
 webdefault.xml [1-9](#)
 webdefaults.xml [1-9](#)
 web-jetty.xml [1-9](#)
 file tag [B-5](#)

firewallManage permission [C-9](#)
 firewalls [4-9](#)
 Fireworks
 buttons [2-4](#)
 Fireworks MX [1-12](#)
 folder.gif file [4-14](#)
 format tag [4-14, 4-17, 5-6, 5-11](#)
 fullDate attribute [5-11](#)
 fullDateTime attribute [5-11](#)
 fullTime attribute [5-11](#)
 functionality of SESM web portal applications [3-30](#)

G

getDescription method [4-13, 4-16](#)
 GIF images [2-4, 4-4](#)
 groupBean [4-15](#)

H

hardware requirements [1-10](#)
 help.jsp [4-8](#)
 helpBody.jsp [4-8](#)
 home.jsp [4-8, 4-19](#)
 HTML 4 [4-10, 4-12, 4-15](#)
 html directory [4-4](#)
 HTTP clients [3-12](#)
 HTTP Redirect feature [1-2](#)
 HTTP requests [3-9](#)
 GET [3-3](#)
 information on [C-12](#)
 POST [3-3](#)
 HTTP responses [3-9](#)
 HTTP sessions [3-9](#)
 ending [C-5](#)
 information on [C-12](#)
 secure mode [4-9](#)
 httpSniff.jsp [3-10, 3-33, C-6](#)

HttpSniffBean class [3-32, C-5](#)
 HttpSniffDecorator servlet [3-31, 3-33, C-5](#)

I

I18nObject class [5-6](#)
 I18nResource class [5-5](#)
 icons [2-4](#)
 service group [4-14, 4-16](#)
 service status [4-11, 4-12](#)
 id attribute [B-7](#)
 idFile attribute [B-7](#)
 id parameter [C-13](#)
 images [2-1](#)
 service group [4-14, 4-16](#)
 service status [4-11, 4-12](#)
 images directory [4-4](#)
 include files [4-18](#)
 initialization parameters
 country [C-7](#)
 defaultCountry [5-14](#)
 defaultLanguage [5-14](#)
 defaultResourceBundle [5-14](#)
 defaultValue [3-21](#)
 defaultVariant [5-14](#)
 dimensions [3-12, 3-13, 3-15, 3-16, 3-17, 3-18, 3-20, 3-28, C-12](#)
 id [C-13](#)
 language [C-7](#)
 name [C-11](#)
 password [C-14](#)
 permission [C-9](#)
 postDecorate [3-20, C-2](#)
 preDecorate [3-38, 3-40, C-2](#)
 scope [C-11](#)
 timeZoneMapCountries [5-15](#)
 timeZoneMapTimeZones [5-15](#)
 to [C-3](#)
 username [C-14, C-15](#)
 value [C-13](#)

 variant [C-7](#)
 vfile [C-15](#)
 initial logon redirection [4-27, 4-29](#)
 InitServlet servlet [C-6](#)
 InsecureDecorator servlet [C-6](#)
 installing
 bundled RADIUS server [A-9](#)
 demo installation [A-3](#)
 SPE [A-3](#)
 international character sets
 downloading [A-17](#)
 internationalization [5-1, 5-2, A-17](#)
 internationalized resources [4-13, 5-5, 5-11](#)
 localized by views [3-3, 5-5](#)
 provided by controls [3-3](#)
 Internet Explorer [A-3](#)
 Internet Explorer browser [2-17](#)
 IP addresses
 RADIUS server [A-6, A-7, A-9, A-10](#)
 iPass logon
 customization [2-6](#)
 isNecessary method [3-9, 3-31](#)
 iterator.tld file [B-1](#)
 Iterator tag library [B-2](#)

J

J2EE platform [1-8](#)
 ja directory [4-5](#)
 ja locale [4-5](#)
 Japanese language [2-16, 4-5](#)
 JAR files [2-8, 2-9, 4-6](#)
 java.text.MessageFormat class [5-13](#)
 Java 2, Enterprise Edition [1-12](#)
 web servers [1-3, 1-10](#)
 web server tier [1-4](#)
 Java 2 SDK [1-10, 2-10](#)
 JavaBeans [3-3, 3-6, 3-8, 3-33](#)
 Javadoc documentation [2-10, 3-8](#)

- JavaScript probe [3-21, 3-33](#)
 - Java servlets [1-4, 1-12](#)
 - utilities [C-1](#)
 - JDK_HOME variable [1-10, 2-10](#)
 - Jetty web servers [1-3, 1-10, 2-17](#)
 - jsp.jar file [2-9](#)
 - JSPFragment class [4-18](#)
 - jspInit method [3-25, 3-33](#)
 - JSP pages [1-3, 1-4, 1-7, 1-12, 4-6](#)
 - account management [4-9](#)
 - body [3-30, 4-7](#)
 - buttons [4-7](#)
 - compiling [2-11](#)
 - decorators [3-31](#)
 - include files [4-18](#)
 - NWSP [4-5](#)
 - precompiling [2-7](#)
 - service selection [4-7](#)
 - service subscription [4-9](#)
 - SESM RADIUS installation [4-7](#)
 - SESM SPE installation [4-7, 4-9](#)
 - user-shape decoration [4-4, 4-18](#)
 - wrapper [4-6](#)
 - JSP tag libraries [5-6, B-2, B-4](#)
-
- K**
- key attribute [5-12](#)
 - key-value pairs in a properties file [5-3, 5-12](#)
-
- L**
- L10nContext.getDefault method [5-4](#)
 - L10nContext class [5-6, 5-7, 5-8, 5-9](#)
 - L10nContextDecorator servlet [5-14, C-7](#)
 - l10n directory [4-5](#)
 - language name for a properties file [5-4](#)
 - language packs [2-16](#)
 - language parameter [C-7](#)
 - languages [5-9](#)
 - language tag [5-9](#)
 - LDAP-compliant directories [1-1, 4-1](#)
 - Live Data window feature [1-11, 2-14](#)
 - configuring [2-15](#)
 - locale attribute [5-8, 5-10](#)
 - LocaleDecorator servlet [C-7](#)
 - locale dimension [3-11, 3-15, 3-19](#)
 - LocaleDimensionDecorator servlet [3-22, 3-23, 3-24](#)
 - locales [5-8, 5-9, 5-10, 5-15, C-7](#)
 - browser preferences [3-4](#)
 - setting [4-7](#)
 - testing [2-19](#)
 - locale settings [3-6](#)
 - locale tag [5-9](#)
 - localization [3-3, 5-1, 5-2, A-17](#)
 - localization.tld file [B-1](#)
 - localization contexts [5-7, 5-8, 5-9, 5-10, C-7](#)
 - default [3-4, 5-6, 5-14](#)
 - internationalized resources [5-5](#)
 - Localization tag library [5-6, B-2](#)
 - location-based branding [3-26](#)
 - locationDimension.jsp page [3-21, 3-22, 3-23, 3-24, 3-26, 3-29](#)
 - logging directory [4-5](#)
 - logging messages [4-18](#)
 - logging on
 - to portal applications [A-4](#)
 - user IDs for demo [A-5](#)
 - logout [3-6, 4-7](#)
 - logon [3-6, 4-7](#)
 - Log Out button [2-3](#)
 - longDate attribute [5-11](#)
 - longTime attribute [5-11](#)
 - look-and-feel
 - elements [2-1](#)
 - requirements [1-7](#)

M

maintaining web portal applications [2-5](#)

mainTemplate.dwt template [4-6, 4-19](#)

mapping servlets [3-36](#)

MarkupDimensionDecorator servlet [3-22, 3-23, 3-24](#)

mediumDate attribute [5-11](#)

mediumDateTime attribute [5-11](#)

mediumTime attribute [5-11](#)

MemoryCheckDecorator servlet [C-8](#)

Merit RADIUS files [2-13](#)

MessageDecorator servlet [C-8](#)

MessagePortalServlet [4-30](#)

message portal web application [1-5, 4-28, 4-30](#)

messages [3-7](#)

- displaying [4-7](#)
- HTTP session [C-8](#)

messages_en.properties file [5-4](#)

messages for advertising [4-27](#)

MM_nbIsActive function [D-1](#)

mobile devices [2-21](#)

Mobile Internet Toolkit simulator [2-21](#)

mobile wireless [1-1](#)

models [3-2, 3-31](#)

Model-View-Control design [3-2, 3-4, 4-19](#)

mutexConnect [4-16](#)

mutexConnectfolder.gif file [4-14](#)

mutexSubscribe [4-16](#)

mutexSubscribefolder.gif file [4-14](#)

mutually exclusive connection groups [4-14, 4-16](#)

mutually exclusive subscription groups [4-14, 4-16](#)

myAccount.jsp page [4-2](#)

MyAccountBean [3-5](#)

MyAccountControl [3-4, 3-5](#)

MyAccountView [3-4, 3-5](#)

N

name attribute [B-2, B-4, B-5](#)

name parameter [C-11](#)

names, of portal applications [1-9](#)

navbar.jsp [4-17](#)

navigation bars [2-3, 2-4, 3-30, 4-17, 4-20](#)

- modifying [3-30](#)
- privileges [4-2](#)

navigator.tld file [B-1](#)

Navigator class [C-1, C-2](#)

Navigator tag library [B-3](#)

Netscape, supported version [A-3](#)

New World Service Provider (NWSP)

- See *NWSP web portal application*

NoCacheDecorator servlet [C-8](#)

Nokia Mobile Internet Toolkit [4-23](#)

number attribute [5-11](#)

numbers [5-11](#)

NWSP

- logging on [A-4](#)
- user IDs for demo [A-5](#)
- web portal application [1-10, 2-12](#)

nwsp.xml file [4-4](#)

NWSP directory [4-3](#)

NWSP web portal application [1-5, 2-2, 4-1, 4-2](#)

- adding services [4-10](#)
- configuring [3-34](#)
- controls [3-6, 4-19](#)
- cookies [C-4](#)
- customizing [2-2, 3-2, 4-2, 4-22, 4-23](#)
- directory hierarchies [4-3](#)
- functionality [4-2](#)
- GIF images [4-4](#)
- JavaBeans [3-6](#)
- JSP pages [4-6](#)
- navigation bar [4-17](#)
- PNG files [4-4](#)
- properties files [4-6](#)

removing services [4-10](#)
 service list [4-10](#)
 service portal [4-30](#)
 servlets [4-6](#)
 SESM RADIUS installation [1-1](#)
 SESM SPE installation [1-1, 4-2](#)
 style sheet [4-5](#)
 templates [4-5, 4-19](#)
 user interface [2-2, 4-2, 4-22, 4-23](#)
 views [3-6](#)
 web.xml file [3-33](#)

O

object attribute [4-14, 4-17, 5-11](#)
 Openwave UP.Simulator [2-21](#)
 openWindowName attribute [C-12](#)
 openWindowURL attribute [C-12](#)
 OriginalURLDecorator servlet [C-9](#)
 OSDimensionDecorator servlet [3-22, 3-23, 3-24](#)
 otherwise attribute [5-8](#)
 over attribute [B-2](#)

P

pages directory [4-5](#)
 param attribute [5-13](#)
 params attribute [5-13](#)
 password parameter [C-14](#)
 passwords [3-6, 4-2](#)

- reference implementation testing [2-19, C-14](#)
- service [A-6, A-7, A-9, A-10](#)
- service group [A-6, A-7, A-9, A-10](#)

 path tag [B-6](#)
 PATH variable [1-10](#)
 PDA

- web portal application [1-10](#)

 pda directory [4-5](#)

PDA web portal application [1-5, 2-2, 4-1, 4-21](#)

- branding [4-22](#)
- functionality [4-22](#)
- user interface [4-22](#)

 permissionBean [C-9, C-10](#)
 PermissionCheckDecorator servlet [C-9](#)
 PermissionDecorator servlet [C-10](#)
 PermissionFailedException [C-9](#)
 permission parameter [C-9](#)
 permissions [4-10](#)

- checking [C-9](#)

 personal firewalls [1-2, 2-14, 3-6, 4-9](#)
 Pocket PC 2002 Software Development Kit [2-21](#)
 Pocket PC emulators [2-21](#)
 Portable Network Graphics (PNG) files [1-11, 4-4, 4-18, 5-2](#)
 portals

- names [1-9](#)

 ports

- RADIUS server [A-6, A-7, A-9, A-10](#)

 postDecorate parameter [3-20, C-2](#)
 post-decorators [3-10, 3-20, 3-29](#)

- creating [3-31](#)

 POST parameters [3-8](#)
 precompile.sh script [2-12](#)
 precompiled JSP pages [2-7, 2-9](#)
 precompiling JSP pages [2-7](#)
 preDecorate parameter [3-38, 3-40, C-2](#)
 preferredLocales attribute [5-8](#)
 preloading images [4-21](#)
 Preview in Browser feature [2-22](#)
 privileges for navigation bar buttons [4-2](#)
 problems with conventional web sites [1-5](#)
 properties files [2-17, 4-6, 5-3](#)

- retrieving values [5-12](#)

 protect.jar file [2-9](#)

Q

quick start, demo installation [A-3](#)

R

RADIUS Data Proxy (RDP) [1-2](#)

RADIUS files [2-13](#)

RADIUS server

- primary [A-6, A-7, A-9, A-10](#)
- secondary [A-6, A-7, A-9, A-10](#)

See also ports

RADIUS servers [1-1, 4-1](#)

RADIUS shared secret

- with portals [A-6, A-7, A-9, A-10](#)

recompiling JSP pages [2-11](#)

redirection types for Captive Portal [4-27](#)

RedirectRemainder servlet [C-10](#)

registering with DecoratorPool [C-5](#)

RemoveAttributeDecorator servlet [C-11](#)

ResourceBundle class [5-4](#)

resourceBundleName attribute [5-8](#)

resource bundles [3-3, 5-2, 5-3, 5-5, 5-6](#)

- base name [5-8](#)
- retrieving values [5-12](#)
- search algorithm [5-4](#)
- service-group descriptions [4-13, 4-16](#)

resource tag [5-12](#)

ResponseCompleteNotice exception [C-6, C-11](#)

S

scope attribute [5-8](#)

scope parameter [C-11](#)

ScriptDimensionDecorator servlet [3-25](#)

search algorithm for a resource bundle [5-5](#)

searches for a web resource [3-14, 3-16](#)

SecureDecorator servlet [C-11](#)

secure mode [4-9](#)

Secure Sockets Layer (SSL) [4-9, C-6, C-11](#)

security [4-9](#)

Security Policy Engine (SPE)

- classes for [2-9](#)
- class libraries [1-1](#)
- SESM SPE installation [1-1](#)

selfManage permission [C-9](#)

self-subscription [2-14, 4-10](#)

serviceBean bean [4-13](#)

service connections [4-2](#)

service descriptions [4-13](#)

ServiceDisplayDecorator servlet [C-12](#)

serviceGroup bean [4-16](#)

service group profiles [2-13](#)

service groups [4-14](#)

- descriptions [4-16](#)
- types [4-16](#)

service groups, password [A-6, A-7, A-9, A-10](#)

service list [2-3, 3-30, 4-10, 4-20](#)

serviceList.jsp [4-6](#)

serviceList.js script [4-10, 4-12, 4-15](#)

serviceListGroup.jsp [4-14](#)

serviceListService.jsp [4-12, 4-15](#)

serviceListService.jsp page [4-12](#)

ServiceListServiceBean class [4-12](#)

ServiceListServiceGroupBean [4-15](#)

serviceLost.gif file [4-11](#)

servicenotRequired.gif file [4-11](#)

serviceOff.gif file [4-11](#)

serviceOn.gif file [4-11](#)

servicepages directory [4-5](#)

service-page URL [4-12](#)

service parameter [4-28](#)

service portal web application [1-5, 4-28, 4-30](#)

service profiles [2-13](#)

service request parameter [C-12](#)

services

- adding and removing [4-10](#)
- connecting to [4-2, 4-22, 4-24](#)
- selecting [4-7](#)
- status [3-7](#)
- subscribing [3-6, 4-2, 4-9, 4-10](#)
- unauthorized [4-27](#)

- Service Selection Gateway (SSG) [1-2, 4-25, 4-28](#)
- serviceSelection permission [C-9](#)
- service-status icon [4-12](#)
- service subscription [4-9](#)
- serviceSubscription permission [C-9](#)
- serviceURL parameter for Captive Portal [4-28](#)
- service URLs [C-12](#)
- servlet code for JSP pages [2-17](#)
- servlet mapping
 - test decorators [2-18](#)
- servlets [1-4, 4-6](#)
 - chaining [3-36](#)
 - mapping [3-35, 3-36, 3-39](#)
- sesm.css file [4-5](#)
- SESM development tools
 - Dreamweaver UltraDev [1-11](#)
- SESM RADIUS installation [1-1, 2-13, 4-1, 4-17](#)
- SESMSession class [3-27](#)
- SESMSessionDecorator servlet [C-12](#)
- SESM sessions [C-12](#)
- SESM SPE installation [1-1, 2-14, 4-1, 4-2, 4-10, 4-17](#)
 - functions [4-9](#)
 - permissions [4-10](#)
- SESM utility servlets
 - declaring [3-35](#)
- SESM web applications
 - Captive Portal [4-25](#)
 - developing [1-10, 1-12](#)
 - hardware requirements [1-10](#)
 - SESM RADIUS installation [4-1](#)
 - SESM SPE installation [4-1](#)
 - software requirements [1-10](#)
 - virtual file names [B-4](#)
- SESM web portal applications [1-1, 1-3, 2-12, 4-1](#)
 - architecture [3-2](#)
 - basic functions [4-7](#)
 - class libraries [2-8](#)
 - compiling [2-10](#)
 - components [2-1, 3-2](#)
 - concepts [3-8](#)
 - controls [3-3](#)
 - customizing [1-7 to 1-8, 2-1, 3-1](#)
 - debugging [2-17](#)
 - decoration of user shape [3-18](#)
 - decorators [3-9](#)
 - developing [1-11, 2-6](#)
 - dimension decorators [3-10, 3-21](#)
 - directory hierarchies [1-7, 3-10](#)
 - infrastructure [1-7](#)
 - internationalization [3-3, 5-1, 5-5](#)
 - internationalized resources [5-5](#)
 - JavaBeans [3-3](#)
 - localization [5-1, 5-2](#)
 - log files [4-18](#)
 - model [3-2](#)
 - modifying functionality [3-2, 3-29](#)
 - post-decorators [3-10](#)
 - searches for web resources [3-14, 3-16, 3-17](#)
 - security [4-9](#)
 - SESM SPE installation [4-9](#)
 - startup script [2-10](#)
 - style sheets [2-4](#)
 - views [3-4](#)
 - virtual file names [3-37](#)
- shape.tld file [B-1](#)
- shape attribute [B-7](#)
- ShapeDecorator servlet [3-12, 3-13, 3-19, 3-20, 3-28, 3-29, C-12](#)
 - ordering dimensions [3-12](#)
- Shape objects [C-12](#)
- shapes [3-8, 3-9](#)
- "shape" session attribute [3-28](#)
- Shape tag library [B-4](#)
- shared secret
 - RADIUS and portals [A-6, A-7, A-9, A-10](#)
- shortDate attribute [5-11](#)
- shortDateTime attribute [5-11](#)
- shortTime attribute [5-11](#)
- simulators for mobile devices [2-21](#)

- simulators for WAP phones [2-21](#)
 - SizeDimensionDecorator servlet [3-25](#)
 - Snoop servlet [C-12](#)
 - software requirements [1-10](#)
 - sparse-tree directory structure [1-7, 3-8, 3-9, 3-13, 3-14](#)
 - SPE
 - installing [A-3](#)
 - SSG simulator [A-5](#)
 - running [A-7](#)
 - standard mode [4-9](#)
 - start tag [B-2](#)
 - startup scripts
 - application names in [1-9](#)
 - customizing [1-8](#)
 - status of services [3-7, 4-7, 4-13](#)
 - styles directory [4-5](#)
 - style sheets [2-1, 2-4](#)
 - subAccountManage permission [C-9](#)
 - subaccounts [3-7, 4-2](#)
 - creating [2-14, 4-9](#)
 - Subscriber Edge Services Manager (SESM)
 - servers [1-2](#)
 - software versions [1-10](#)
 - system components [1-2](#)
 - See also *SESM web applications*
 - subscriber profiles [2-13](#)
 - demo installation [A-4, A-5](#)
 - for SESM RADIUS installation simulation [2-13](#)
 - for SESM SPE installation simulation [2-14](#)
 - subscribers [1-1](#)
 - subscriber self-care [4-9](#)
 - subscriptions to services [3-6, 4-2, 4-9](#)
 - system integrators [1-8](#)
 - system messages [4-2](#)
-
- T**
- taglib directive [B-2](#)
 - tag libraries [5-6, B-2, B-4](#)
 - configuring [B-1](#)
 - descriptor files [2-8, 4-5, B-1](#)
 - tags
 - context [5-7](#)
 - country [5-9](#)
 - decorate [B-4](#)
 - file [B-5](#)
 - format [5-11](#)
 - language [5-9](#)
 - locale [5-9](#)
 - path [B-6](#)
 - resource [5-12](#)
 - start [B-2](#)
 - timeZone [5-9](#)
 - TCP Redirect for Services feature [4-25](#)
 - techniques for SESM development [3-2](#)
 - templates [2-5, 4-5, 4-19, 4-21](#)
 - templates directory [4-5](#)
 - template tag [5-13](#)
 - test decorators [2-18](#)
 - TestDimensionDecorator servlet [C-13](#)
 - test features
 - web portals [A-2](#)
 - TestUserDecorator servlet [C-13](#)
 - time attribute [5-11](#)
 - times [5-11](#)
 - timeZone attribute [5-8](#)
 - timeZoneMapCountries parameter [5-15](#)
 - timeZoneMapTimeZones parameter [5-15](#)
 - time zones [5-8, 5-9, 5-15](#)
 - timeZone tag [5-9](#)
 - .tld files [4-5](#)
 - token replacement in a resource [5-13](#)
 - to parameter [C-3](#)
 - translating text for localization [5-2](#)
 - translating virtual file names [3-38, C-15](#)
 - type attribute [B-2](#)

U

unauthenticated user redirection [4-27, 4-28](#)
 unauthorized service redirection [4-27, 4-28](#)
 unstyled directory [4-5](#)
 URIs [3-4, 3-37, C-15](#)
 url attribute [4-30](#)
 URLs
 Captive Portal [4-26](#)
 original [C-9](#)
 redirecting [C-10](#)
 service [C-12](#)
 servlet mapping [3-36](#)
 UserDecorator servlet [C-14](#)
 user ID
 demo logons [A-4, A-5](#)
 user-interface controls [4-7](#)
 username parameter [4-28, C-14, C-15](#)
 usernames
 ensuring authenticated [C-14](#)
 reference implementation testing [2-19, C-14](#)
 users
 unauthenticated [4-27](#)
 user-shape directory hierarchy [3-11, 3-12](#)
 order of directories [3-13](#)
 user shapes [1-6, 3-8, 3-9, 3-10, 3-18, B-5, C-12, C-15](#)
 UTF-8 [A-3, A-17](#)

V

val tag [B-3](#)
 value attribute [B-2, B-3](#)
 value parameter [C-13](#)
 variable attribute [5-8, 5-9](#)
 variant name of a properties file [5-4](#)
 variant parameter [C-7](#)
 vendor-specific RADIUS attributes (VSAs) [2-13](#)
 versionError key [C-4](#)
 versionNotAvailable key [C-4](#)

vfile parameter [C-15](#)
 views [3-4, 3-6](#)
 internationalized resources [3-3](#)
 invoking decorators [3-40](#)
 logical [3-34](#)
 modifying functionality [3-30](#)
 virtual file names [3-4](#)
 virtual file names [3-4, 3-37, 3-39, B-4, C-15](#)
 VirtualFile servlet [3-4, 3-37, 3-39, 4-16, C-2, C-15](#)

W

WAP
 web portal application [1-10](#)
 wap directory [4-5](#)
 WAP phone simulators [4-23](#)
 WAP web portal application [1-5, 2-2, 4-1, 4-23](#)
 functionality [4-24](#)
 user interface [4-23](#)
 web.dev.xml file [2-18](#)
 web.recompile.xml [2-11](#)
 web.xml [1-9](#)
 web.xml file [2-11, 2-17, 3-19, 3-20, 3-33, 3-38, 4-5, 5-14](#)
 compiling JSP pages [2-11](#)
 configuring [3-34, 3-35](#)
 tracking changes [2-8](#)
 webapp directory [4-4](#)
 web application
 Captive Portal [1-5](#)
 web application archive (WAR) files [1-12](#)
 web applications
 localization contexts [5-7, C-7](#)
 web browser
 choosing for a Demo [A-3](#)
 web client tier [1-4](#)
 web components [2-2, 3-2, 4-2](#)
 webdefault.xml [1-9](#)
 web designers [1-4](#)
 web developers [1-4](#)

WEB-INF directory [4-3, 4-5](#)

web-jetty.xml file [1-9](#)

web portal

- installing
 - in demo installation [A-3](#)
 - running as demo installation [A-4](#)

web portal applications [1-1](#)

See also *NWSP web portal application*, *PDA web portal application*, *SESM web portal applications*, and *WAP web portal application* [4-2](#)

web portals [1-1](#)

- test features [A-2](#)

web resources [3-10](#)

- customizing for user shape [3-8, 3-14](#)
- searches for [3-14, C-15](#)

web servers [1-2, 1-10, 2-17](#)

web server tier [1-4](#)

web site management [2-22](#)

web site pages hierarchy [3-11](#)

WinWAP Pro simulator [2-21](#)

Wireless Application Protocol (WAP) [1-12](#)

wireless LAN [1-1, 1-7](#)

Wireless Markup Language (WML) [1-12](#)

wrapper JSP pages [4-6](#)

X

xml directory [4-6](#)

Z

zh directory [4-6](#)

zh locale [4-6](#)

