CHAPTER 2

SESM Components and Techniques

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

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

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

Using SESM Web Components

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

JavaServer Pages and Servlets

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

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

JSP Tag Libraries

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

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

Resource Bundles

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

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

Templates

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

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

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

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

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

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

Library Items

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

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

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

Images, Buttons, and Navigation Bars

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

Images

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

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

Buttons

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

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

Navigation Bars

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

Figure 2-1 Navigation Bar

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

Customizing an SESM Web Application

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

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

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

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

Changing the Look-and-Feel Elements

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

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

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

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

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

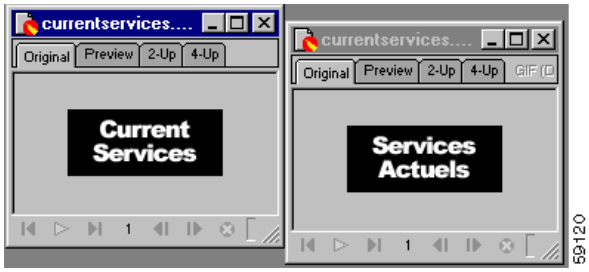
Localizing a Web Application

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

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

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

Figure 2-2 Localizing currentservices.gif



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

The second set of modifications for the simplest form of localization is that message text and other items in resource bundles must be translated, and an additional properties file must be created for the new language. The `createLocaleDimension.jspi` file includes the logic to determine and set the subscriber's locale. For information on resource bundles and setting the locale for a specific subscriber, see the "Resource Bundles" section on page 4-1 and "createLocaleDimension.jspi" section on page 2-16.

Using a Sparse Tree Directory Structure

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

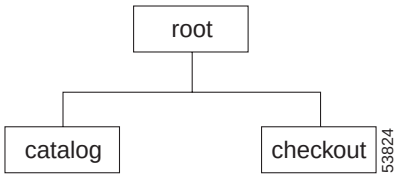
- Web site pages hierarchy
- User shape hierarchy

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

Web Site Pages Hierarchy

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

Figure 2-3 Web Site Pages Directory Hierarchy



User Shape Hierarchy

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

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

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

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

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

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

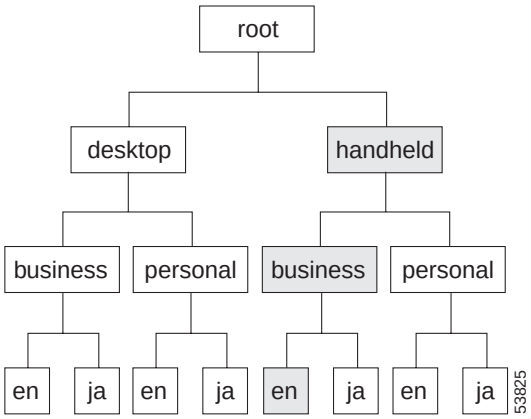
- Devices: desktop and handheld
- Brands: business and personal
- Locales (languages): English (en) and Japanese (ja)

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

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

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

Figure 2-4 User Shape Directory Hierarchy



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

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

Location of Directory Dimensions

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

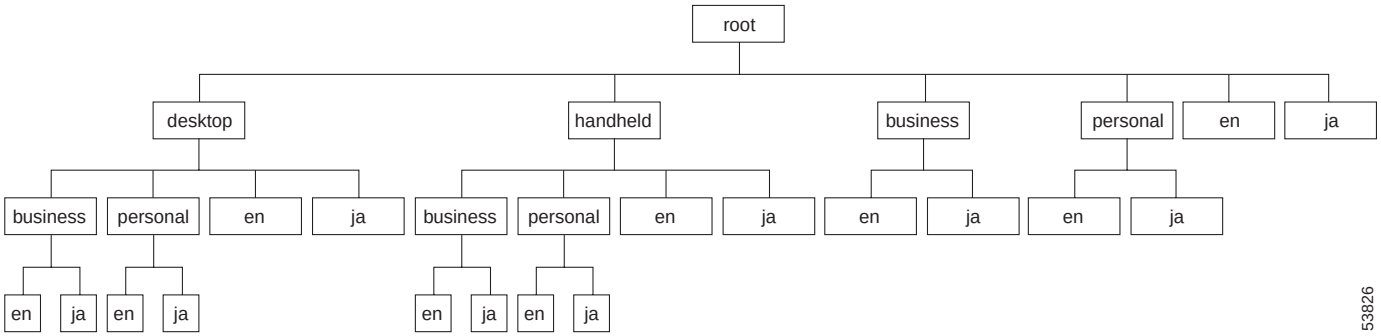
- In general, the more frequently a value appears in the hierarchy, the further down in the hierarchy it is located. As an example, in Figure 2-4 the values of the first dimension (desktop and handheld) appear only once in the hierarchy, but the values of the last dimension (en and ja) appear many times.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

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

Sparse Tree Directory Structure

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

Figure 2-5 Fully Expanded User Shape Directory Hierarchy



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

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

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

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

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

Implementing the Sparse Tree Directory

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

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

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

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

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

Searches for a Web Resource

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

Example 1: Searches for a Web Resource

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

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

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

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

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

Initialization of dimensions Array

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

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

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

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

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

Order of Paths Searched

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

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

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

Table 2-1 SESM Search Algorithm: Example 1

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

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

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

Example 2: Searches for a Web Resource

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

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

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

Table 2-2 SESM Search Algorithm: Example 2

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

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

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

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

Using the Decorator Components

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

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

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

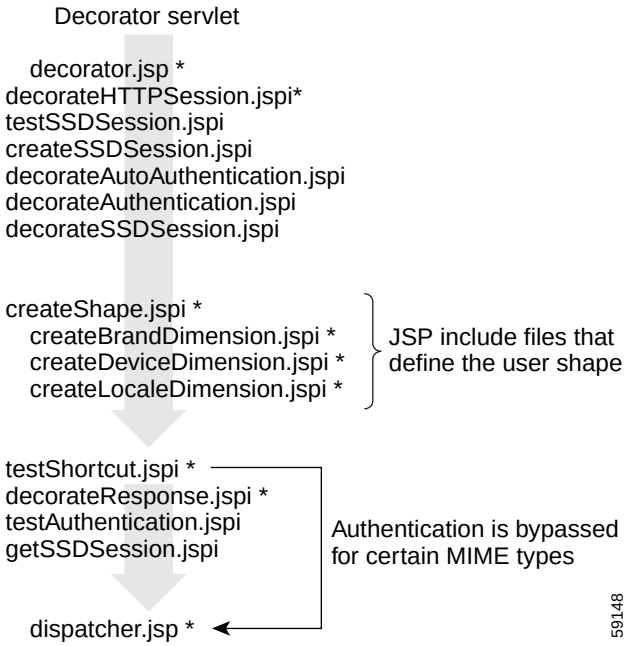
The JSP web components in /docroot/decorator for HTTP session handling and subscriber authentication work without modification. This section focuses on the use of the user-shape decoration components. However, the service-provider developer can modify any web component in the /docroot/decorator except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

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

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

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

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



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

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

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

Scripting Variables

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

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

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

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

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

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

Decorator Servlet

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

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

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

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

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

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

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

Decoration JSP Pages and Include Files

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

decorator.jsp

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

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

decorateHTTPSession.jspi

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

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

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

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

Setting the Dimensions of the User Shape

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

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

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

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

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

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

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

createDeviceDimension.jspi and createBrandDimension.jspi

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

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

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

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

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

Table 2-3 Directory List Searches within a Single Dimension

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

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

createLocaleDimension.jspi

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

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

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

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

...

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

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

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

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

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

In the second `l10n:context` tag, the `otherwise` attribute is set to the locale that will be used if the SESM software cannot find a resource bundle for any of the preferred locales.

3. After getting a string for the current locale with the `getLocale` method of the `L10nContext` class, `createLocaleDimension.jspi` sets the `"shape.locale"` request attribute to the value of the locale. The `"shape.locale"` request attribute is set to the value of a `Dimension` object, which is initialized using a directory list (in this case, a locale and country) and the _ character as the delimiter for the elements of the directory list. For example, if fr_CA were the locale, the two elements of the list are fr and CA.

createShape.jspi

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

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

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

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

testShortcut.jspi

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

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

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

```
...

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

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

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

decorateResponse.jspi

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

Rewriting URLs When Cookies Are Not Supported

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

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

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

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

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

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

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

Redirecting to the Same URL

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

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

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

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

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

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

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

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

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

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

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

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

dispatcher.jsp

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

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

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

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

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

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

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

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

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

Developing an SESM Web Application

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

Defining the Business Requirements

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

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

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

Designing and Implementing an SESM Web Application

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

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

SESM Class Libraries

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

Table 2-4 JAR Files for an SESM Web Application

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

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

JSP Compilation

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

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

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

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

Step 1 Stop the web server.

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

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

Step 4 Restart the web server.

Demo Mode

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

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

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

RADIUS Mode Functionality

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

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

DESS Mode Functionality

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

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

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

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

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

General Web Development Considerations

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

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

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

Debugging an SESM Web Application

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

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

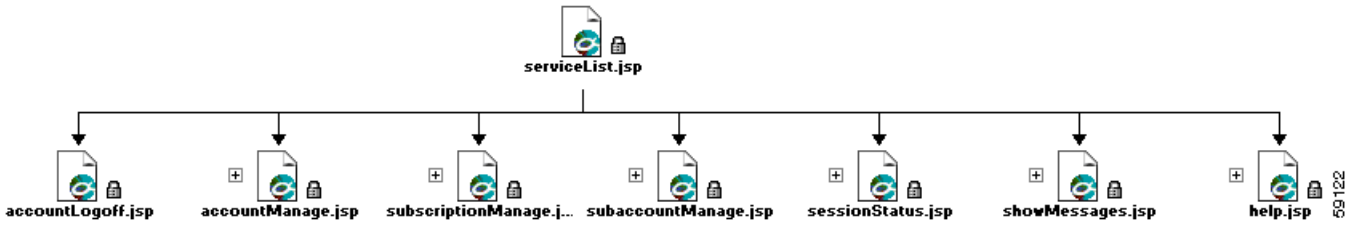
- Most web servers can be set up so that the servlet code generated from JSP pages is saved and available for examination. With the Jetty web server, when the JSP pages are compiled, the servlet code is put in a temporary directory that is determined by the machine's JVM. Normally, the temporary directory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

Managing an SESM Web Site

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

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

Figure 2-7 NWSP Site Map



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

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