CHAPTER 4

# Sample SESM Web Applications

This chapter provides information on the sample SESM web applications and solutions and describes how a developer can use and modify the sample web components. The Cisco SESM software includes the following sample web applications and solutions:

- New World Service Provider Web Application, page 4-2
- PDA Web Application, page 4-21
- WAP Web Application, page 4-23
- Captive Portal Web Solution, page 4-24

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

The PDA and WAP web 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 Sample SESM Web Applications

The following general considerations apply to this chapter’s descriptions of the sample SESM web 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 sample web applications.
- Depending on the SESM software that is used, a deployed SESM web application can be configured for one of two modes:
  - *RADIUS mode*—Service and subscriber information is stored in a RADIUS server.
  - *LDAP mode*—Service, subscriber, and policy information is stored in an LDAP-compliant directory, which is accessed with the DESS application programming interfaces.
- The set of web components used for an SESM web application varies depending on the configuration (RADIUS mode or LDAP mode). Where it is required, the descriptions of components in this chapter indicate the mode in which each component is used.

# New World Service Provider Web Application

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

## NWSP User Interface

For information on the NWSP user interface, see the "NWSP User Interface" section on page 2-3.

## NWSP Customization Based on Subscriber Characteristics

The look and feel of an SESM user interface can be customized for each subscriber. When the HTTP request is received, the SESM web application has an organization (the sparse-tree directory structure) and an infrastructure (the user-shape mechanisms) that allow the page returned in the response to be tailored for the individual subscriber. For example, if the subscriber's browser language is English and receiving device is a desktop PC, the response can be rendered in English using HTML. If another subscriber's browser language is German and the receiving device is a WAP phone, the response can be rendered in German using Wireless Markup Language (WML).

For information on customization and localization, see Chapter 3, "Advanced SESM Customization," and Chapter 5, "SESM Internationalization and Localization."

## NWSP Functionality

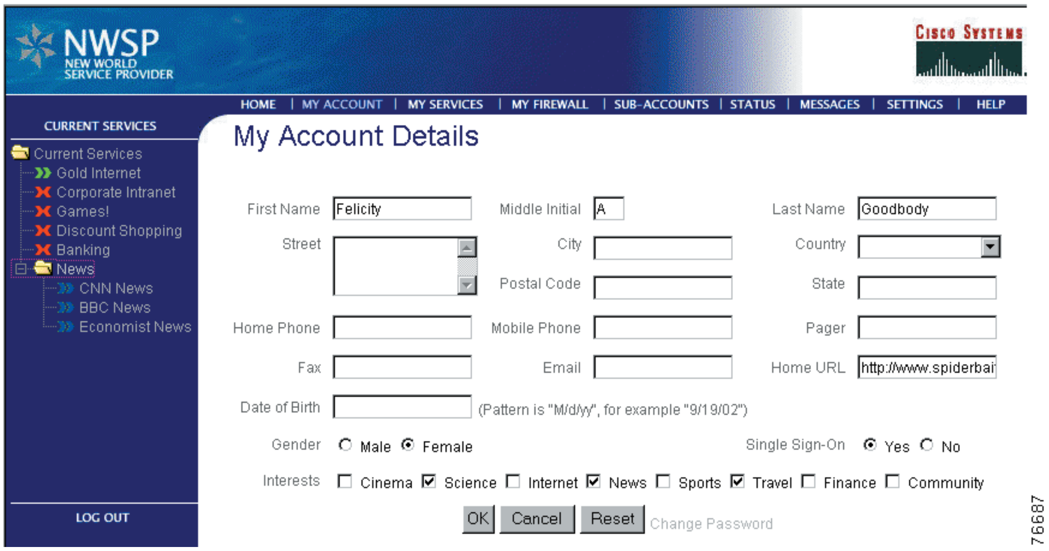
The sample NWSP web application provides a set of functionality that is typical of many directory-enabled SESM applications. The subscriber logs on to the web portal with a user name, password, and for 3-key authentication, a telephone number. The subscriber can then do the following:

- 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 an SESM web application that is operating in LDAP mode. The NWSP web 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 application displays the Accounts button in the navigation bar, and the subscriber can create subaccounts.

Each button at the top of the user interface is linked to a JSP page that implements the functionality for the specific task or set of tasks. As an example, the My Account button is linked to myAccount.jsp, which generates the content for the account management page (Figure 4-1).

*Figure 4-1 Account Management Page*



My Account Details

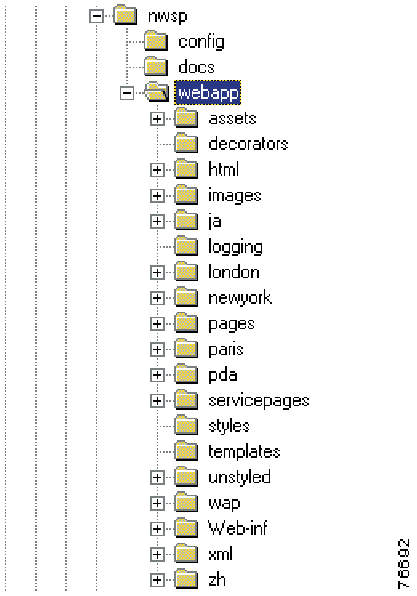
First Name: Felicity
Middle Initial: A
Last Name: Goodbody
Street:
City:
Country:
Postal Code:
State:
Home Phone:
Mobile Phone:
Pager:
Fax:
Email:
Home URL: http://www.spiderbai
Date of Birth: (Pattern is "M/d/yy", for example "9/19/02")
Gender: ○ Male ● Female
Single Sign-On: ● Yes ○ No
Interests: □ Cinema ☑ Science □ Internet ☑ News □ Sports ☑ Travel □ Finance □ Community

# NWSP Directory Hierarchy

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

When the Cisco SESM software is installed, the NWSP web application's hierarchy of directories is located below the *install_dir*\nwsp directory (see Figure 4-2).

*Figure 4-2 NWSP Directories*



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

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

### config

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

### docs

Contains the Javadoc files for the NWSP classes. For information on the Javadoc files, see the "Javadoc Documentation" section on page 2-8.

### webapp

Is the web 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 JSP pages used for decorating the user's shape. For information on user-shape decoration, see the "Decorating a User Shape" section on 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 page include file. For information on these files, see the "JSP Pages and JSPFragment Class for Include File Log Entries" section on page 4-17.

### webapp\pages

Contains the JSP pages that contain the web-application content. The \l10n directory contains JSP pages that provide information on the current localization context and the locales and timezones supported by the JRE. For information on the JSP pages used in NWSP, see the "NWSP JavaServer Pages and Servlets" section on page 4-6.

### 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 application can use in Demo mode.

### 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" section on page 4-18.

### 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" section on page A-1.
- The web application's deployment descriptor file, web.xml, is in \WEB-INF. For information on this file, see the "Using the SESM Deployment Descriptor File" section on page 3-31.
- The \lib directory contains the Java archive (JAR) files for the classes used by the SESM web application. For information on the JAR files required for an SESM web application, see the "SESM Class Libraries and Tag Library Descriptor Files" section on page 2-7.
- 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 application includes a set of JSP pages and servlets that generate content for the web pages and perform other tasks, such as authentication, SESM session handling, and service selection and subscription. The JSP pages contain the elements that the developer modifies for the specific requirements of the service provider. No servlet programming is required.

This section provides guidance on the JSP pages that generate the content for NWSP. For information on the architecture of an SESM web application and the servlets and decorator JSP pages 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 application. From the home page, the subscriber can control service subscription and selection and, in LDAP mode, 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 JSP Pages

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

### Wrapper JSP Pages

Each *wrapper JSP page* contains standard functionality found on many JSP pages 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 page containing JavaScript in its `<head>` section, the NWSP banner, the navigation bar, and the service list.

Most content for the wrapper JSP pages comes from the Dreamweaver template mainTemplate.dwt. If you are not using Dreamweaver as your HTML editor or your SESM web 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 JSP Pages

Each *body JSP page* 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 page. Each body JSP page is included into the corresponding wrapper JSP page. The subscriptionManageBody.jsp page, 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 JSP Pages

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

*Table 4-1 NWSP JSP Pages for User-Interface Controls*

| JSP Page | Use of Control |
|---|---|
| cancelButton.jsp | Canceling an action. Links to the URL specified in the request parameter "`url`". |
| 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 is localizable 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.

## JSP Pages for Basic SESM Web Application Functions

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

*Table 4-2 NWSP JSP Pages for Basic SESM Web Application Functions*

| JSP Page | Description |
|---|---|
| accountLogoff.jsp | After the subscriber clicks the Log Out button, asks the subscriber to confirm or cancel session termination. |
| accountLogon.jsp<br>accountLogonBody.jsp | Allows a subscriber to log on to the NWSP web application. The subscriber's user name and password are authenticated by the SESM software. |

*Table 4-2 NWSP JSP Pages for Basic SESM Web Application Functions (continued)*

| JSP Page | Description |
|---|---|
| accountLogon3Key.jsp<br>accountLogon3KeyBody.jsp | Allows a subscriber to log on to the NWSP web application when 3-key authentication is used. When 3-key authentication is used, the subscriber credentials include a user name, 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<br>helpBody.jsp | Displays help information. The help information is for the demonstration mode NWSP. |
| home.jsp | Displays the NWSP home page. See Figure 2-1 on page 2-4. |
| locale.jsp<br>localeBody.jsp | Displays locale settings from which the subscriber can choose. |
| messages.jsp<br>messagesBody.jsp | Displays messages associated with the current session. |
| messagesText.jsp | Displays one or more messages from a `MessagesBean`. |
| navbar.jsp | Contains the NWSP navigation bar. For more information, see the "JSP Page for the Navigation Bar" section on page 4-16. |
| serviceList.jsp<br>serviceListGroup.jsp<br>serviceListService.jsp | Displays the service list, a tree of subscribed services and service groups. For more information, see the "JSP Page for the Navigation Bar" section on page 4-16. |
| serviceLogon.jsp<br>serviceLogonBody.jsp | Allows a subscriber to enter a user name and password when logging on to a service. |
| serviceStart.jsp<br>serviceStop.jsp | Asks whether the subscriber wants to start a service or stop a service. |
| status.jsp<br>statusBody.jsp | Displays the status of each connected service:<br>• User name<br>• Service description from the service profile<br>• Service status<br>• Connected as—the user name specified for a service that requires authentication<br>• Elapsed connect time<br>• Packets sent |

## Standard and Secure Mode

The accountLogonBody.jsp and accountLogon3KeyBody.jsp pages include Standard | Secure hyperlinks below the Log In button. These links let the user choose either standard mode or secure mode. On the logon page that accountLogon.jsp or accountLogon3KeyBody.jsp displays, if the subscriber is currently using secure mode, only the standard mode link is available. Similarly, if the subscriber is currently using standard mode, only the secure mode link is available.

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

If the service provider does not require SSL or does not have a certificate, the developer removes the Standard | Secure elements in the accountLogonBody.jsp and accountLogon3KeyBody.jsp pages. In addition, the deployer removes the Jetty web server's SSL listener, which is configured in the \nwsp\config\nwsp.xml file. For more information on SSL and security, see the *Cisco Subscriber Edge Services Manager and Subscriber Policy Engine Installation and Configuration Guide*.

## JSP Pages for LDAP-mode Functions

Table 4-3 lists JSP pages for the LDAP-mode SESM web 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 application is deployed in LDAP mode. In addition to all basic SESM web application functions (Table 4-2), the LDAP-mode functions include service subscription, account management (subscriber self-care), subaccount creation, and personal firewall protection. These JSP pages are located in the \nwsp\webapp\pages directory.

***Table 4-3 NWSP JSP Pages for LDAP-mode Functions***

| JSP Page | Description |
|---|---|
| accountPassword.jsp<br>accountPasswordBody.jsp | Displays a form for changing a password. |
| advFirewall.jsp<br>advFirewallBody.jsp | Displays a form that allows the subscriber to create or modify Cisco IOS access control list (ACL) entries for a personal firewall. |
| myAccount.jsp<br>myAccountBody.jsp | Displays a form that allows a subscriber to change account information such as an address, telephone number, email address, and so on. |
| firewall.jsp<br>firewallBody.jsp | Displays a form that allows the subscriber to specify user-friendly definitions for a personal firewall. |
| subaccountConfirm.jsp<br>subaccountConfirmBody.jsp | Displays and asks the subscriber to confirm changes to subaccounts specified in subaccountSubscriptions.jsp. |
| subAccountList.jsp<br>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<br>subaccountSubscriptionBody.jsp | Displays a form that allows a subscriber to modify subaccount details, such as subscribed and blocked services and, if applicable, the user name and password for a service. |
| subscriptionConfirm.jsp<br>subscriptionConfirmBody.jsp | Displays and asks the subscriber to confirm changes to the subscriptions specified in subscriptionManage.jsp. |
| subscriptionManage.jsp<br>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 user name and password to access a service. |

In LDAP mode, the permissions defined for each subscriber determine whether the subscriber can subscribe to services, manage account details, and create subaccounts. The NWSP web 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 application has the needed logic to omit the Accounts button from the navigation bar.

## JSP Pages for the Service List

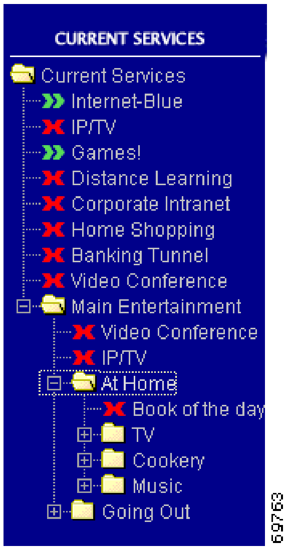
The *service list* (Figure 4-3) is a tree structure with the subscribed services and service groups from which the subscriber can select. The service list is dynamically created by the JSP pages based on the service and subscriber information stored in the data repository. In LDAP mode, the subscriber can use an SESM web application to add or remove services from the set of subscribed services that are displayed in the service list. This feature is called *self-subscription*.

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 JSP pages 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 an SESM web 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()%>">
<img src="<shape:file
name='<%="/images/serviceList/service"+serviceBean.getStatus()+".gif"%>'/>"
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 page 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's status.

- If the current status of the service is On, the action is Stop.
- If the current 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" section on page A-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 I18nObject 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 | serviceStopDesc | Stop |
| Start | serviceStartDesc | 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
```

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

As shown in the preceding code, the JSP page 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 resouce 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" section on page 5-10.

## 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 LDAP mode, the subscriber can use an SESM web 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. | folder.gif |
| 📁 | Indicates a mutually exclusive connection group. | mutexConnectfolder.gif |
| 📁 | Indicates a mutually exclusive subscription group. | mutexSubscribefolder.gif |

## 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 an SESM web application determines the preceding elements, consider this code from /webapp/pages/serviceListGroup.jsp.

```
<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">
   <img src="/cache/vfile/images/serviceList/<%=groupBean.getType()%>folderopen.gif"></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>
```

**Note** The preceding 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 page 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.

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">
   <img src="/cache/vfile/images/serviceList/<%=groupBean.getType()%>folderopen.gif"></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 B, "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_name`*`Description`

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

As shown in the preceding code, the JSP page 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" section on page 5-10.

## JSP Page for the Navigation Bar

In the NWSP web 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 page in the \nwsp\webapp\pages directory contains the navigation bar.

*Figure 4-4 Navigation Bar with Buttons for LDAP Mode*

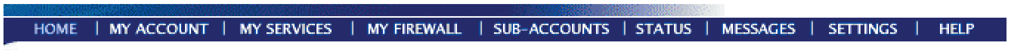


Figure 4-4 shows the NWSP navigation bar with the buttons for the dynamic update features: My Account, My Services, and Sub-Accounts. These buttons are supported only when the dynamic update functionality associated with LDAP mode is available. The Home, Status, Messages, Settings, and Help buttons are displayed in both RADIUS and LDAP modes.

In LDAP mode, the set of buttons that appear in the NWSP navigation bar varies depending on the user's permissions. Various SESM features require that subscriber have appropriate permissions. For example, the ability to create subaccounts or change account information such as a password require that the subscriber have the required permissions. The NWSP web 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 C, "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 application that uses the NWSP components, the developer provides 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 sample 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. A developer can use Fireworks or any graphics tool to modify the button images.

## JSP Pages for User-Shape Decoration

Another smaller set of JSP pages performs some user-shape decoration functions. These JSP pages reside in the \nwsp\webapp\decorator directory. The SESM developer may modify or extend these decorator JSP pages. For information on the user-shape decoration JSP pages, see the “SESM Software Concepts” section on page 3-8 and the “Decorating a User Shape” section on page 3-18.

## JSP Pages and JSPFragment Class for Include File Log Entries

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

When one JSP page includes another JSP page, you will have difficulty determining the JSP include file or *fragment* that wrote a particular message to the application log file. JSP pages are precompiled into one large Java function. When one JSP page includes other JSP pages, 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 page, 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 application writes logging messages from a JSP include file and you want the specific include file to be identified in the message, the web 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 page. 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 page. 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 an SESM web application uses the Model-View-Control (MVC) design pattern. A Cisco SESM web 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. The SESM developer performs no servlet programming. For information on the MVC design pattern and the SESM control servlets, see the "SESM Architecture: An Overview" section on page 3-2.

# NWSP Templates

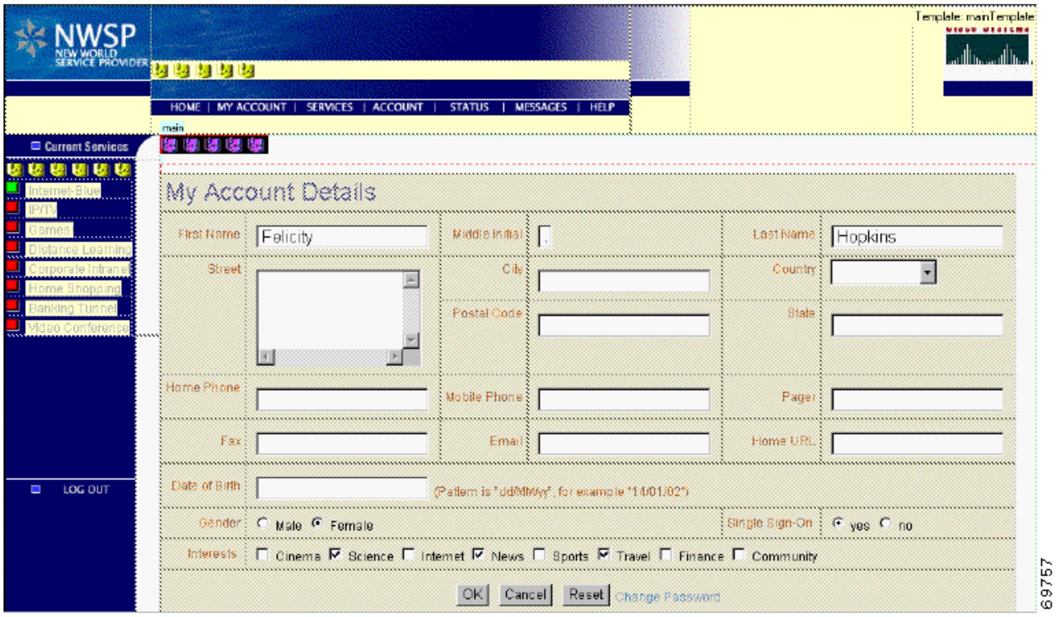
The NWSP components include two Dreamweaver templates for customizing and maintaining the JSP pages:

- 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 JSP pages, including home.jsp, use this template. Figure 4-5 shows the home.jsp page, which is derived from mainTemplate.dwt, as its appears in Dreamweaver with its table borders visible.

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



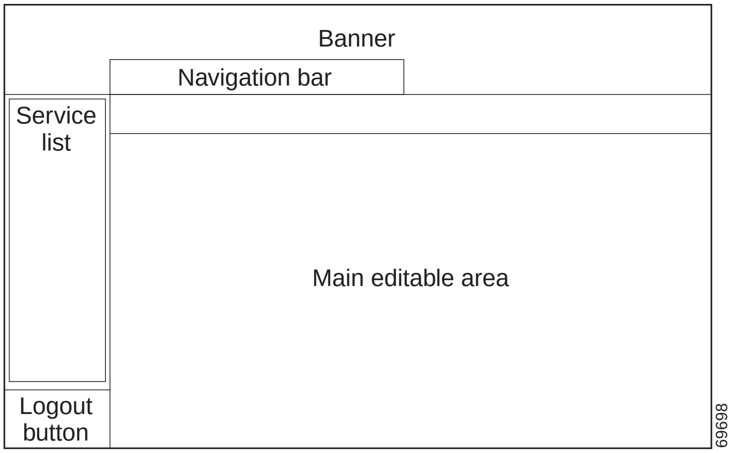
My Account Details

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 "dd/MM/yy", for example "14/01/02")
Gender: ○ Male ● Female
Single Sign-On: ● yes ○ no
Interests: □ Cinema ☑ Science □ Internet ☑ News □ Sports ☑ Travel □ Finance □ Community

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 Page for the Navigation Bar" section on page 4-16.

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 "JSP Pages for the Service List" section on page 4-10.

In the template, there are two editable areas: main and head. The JSP pages 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 JSP pages, 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 page is loaded. The files usually contain images for buttons used on the JSP page. 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'/>',
...
);
}
```

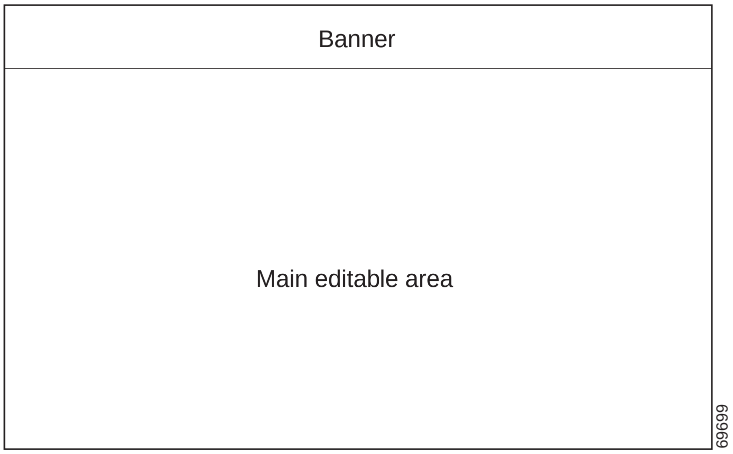
**Tip** Preloading images can cause a page to load more slowly. Because the client browser caches these image files, files that have been loaded once do not need to be reloaded by the other JSP pages. The general rule is: if possible, preload a file once only.

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 that do require a banner with brand icons.

*Figure 4-7 Structure of the Template bannerOnlyTemplate.dwt*



Most JSP pages that use the template `bannerOnlyTemplate.dwt` contain message or help text: help.jsp, and message.jsp. Other JSP pages that use the template have a simple form or button or both: accountLogon.jsp, accountLogon3Key, and accountLogoff.jsp.

# PDA Web Application

The sample PDA web application provides a subset of the SESM functionality for subscribers who are using a personal digital assistant (PDA) or other handheld devices. This application demonstrates how SESM might be used on a handheld for service selection. The PDA web application also shows how to provide a customized look and feel based on a brand.

The files for the PDA web application are located in the *install_dir*\pda directory. For Demo mode, 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 mode, see aaa.properties.

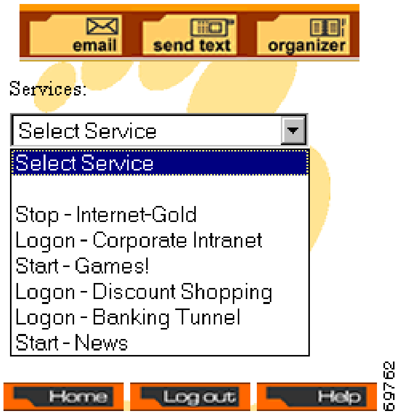
## Branding Based on User Group

The PDA web application is an example of how an SESM web 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 sample PDA web application, there are three user groups: gold, silver, and bronze. The user `golduser` belongs to the `gold` user group, `silveruser` belongs to the `silver` user group, and `bronzeuser` belongs to the `bronze` user group. The appearance of the PDA web 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 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 mode only and are not fully functional.

# PDA Functionality

The SESM-related functionality of the sample PDA web application allows the subscriber to:

- Log on to and log off from an 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 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 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” section on page 3-8.

So that the PDA web 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 Application

The sample WAP web 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 application is an example of how the view JSP pages can serve dynamic WAP content in markup language other than HTML. The JSP pages of the WAP web application are coded in WML.

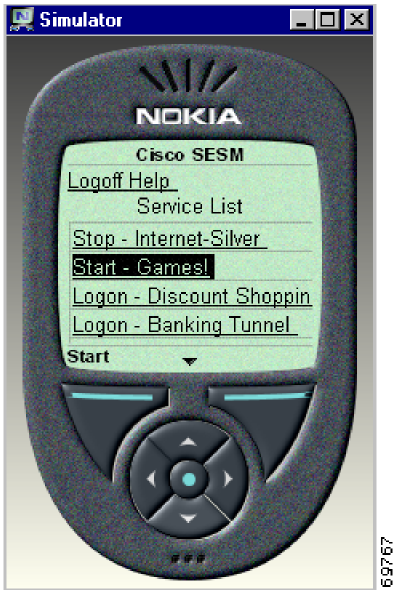
**Note** A WAP phone or WAP phone simulator is needed to view the sample WAP web application. The freely available Nokia Mobile Internet Toolkit contains two simulators and tools for developing a web application that uses WML. For information on using a WAP phone simulator, see the "Using Device Simulators for WAP and WML" section on page 2-20.

The files for the WAP web application are located in the *install_dir*\wap directory. For Demo mode, 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 mode, 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 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 sample WAP web application allows the subscriber to:

- Log on to and log off from an 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 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 sample SESM web 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 Installation 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 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 application functionality. On the SSG, TCP Redirect is configured so that, in certain defined situations, TCP packets from a subscriber HTTP request are redirected by the SSG to an SESM Captive Portal web application.

The SESM Captive Portal application is a servlet that determines the reason for the redirection (for example, an unauthenticated subscriber). The SESM Captive Portal application then redirects the HTTP request to an SESM web application—the “content application”—that provides service content or message content through a JSP page.

- 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 application:

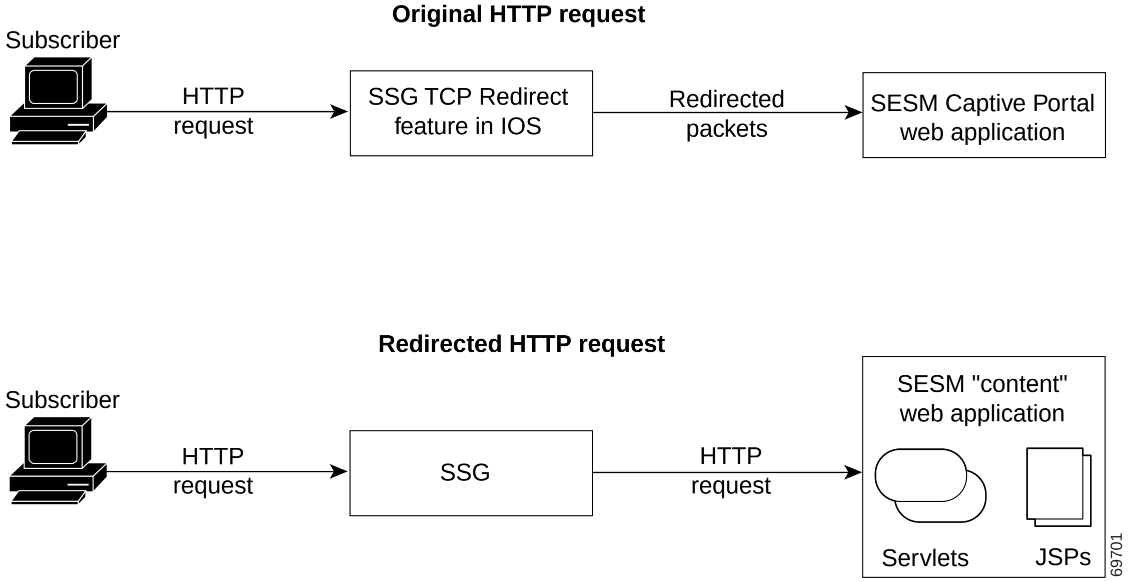
- A SESM Captive Portal web application
- One or more SESM web applications ("content applications") that provide service content or message content or both

TCP Redirect and the SESM Captive Portal application 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.

As an 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, an SESM Captive Portal web application, and an SESM web application, which is responsible for servicing unauthenticated subscribers.

1. **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.)
2. **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.
3. **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.
4. **SESM content web application:** After the subscriber sends a user name 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.<br>• 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.<br>• 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 LDAP mode, a service self-subscription page could be displayed. |
| Initial logon redirection | Displays advertising or other messages to the subscriber upon login, for a specified duration. |
| Advertising redirection | Displays advertising messages to the subscriber at timed intervals during an active SESM session. |

## Captive Portal Solution Configuration

The SSG and SESM configurations define the needed network and device information:

- The SSG configuration defines the TCP ports on which the SSG will redirect incoming TCP packets.
- The SSG configuration also specifies the SESM web server and ports where the SESM Captive Portal application 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 applications, or different pages within the same web application.

For information on configuring the SSG and SESM Captive Portal solution, see the *Cisco Subscriber Edge Services Manager and Subscriber Policy Engine Installation and Configuration Guide* and *SSG Features in Release 12.2(4)B.*

# Sample Captive Portal Solution Web Applications

The sample SESM Captive Portal solution that is included with the SESM software consists 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 sample SESM Captive Portal web 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 the SESM software 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 an 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 host names 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 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 user name from the subscriber profile. |
| Initial logon redirection and<br>Advertising redirection | `CPURL` | The URL in the subscriber's original HTTP request. |
| | `CPDURATION` | The message duration defined in the relevant `captiveportal` MBean attribute:<br>• `initialCaptivateDuration` for initial logon redirection<br>• `advertisingCaptivateDuration` for advertising redirection |
| | `CPSUBSCRIBER` | The user name from the subscriber profile. |

The parameters in Table 4-8 are configurable 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. As an 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 the *Cisco Subscriber Edge Services Manager and Subscriber Policy Engine Installation and Configuration Guide.*

## Content Web Applications

The content web applications provide content to the subscriber. The following basic functionality should be included in the content applications:

- For unauthenticated user redirections—The content application should present an 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 the SSG to authenticate to the service and then connect to the service.

  - Presenting subscription information if the subscriber is not subscribed. For example, in an LDAP deployment, 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 display on either a global or individual subscriber basis. The web application should accept a parameter `CPURL` from the Captive Portal application identifying the subscriber's originally requested URL. Using the original URL, the application can perform another redirection when it is finished 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 sample service portal web application and message portal web application provide this basic functionality.

### Service Portal Web Application

The sample service portal web application is the NWSP web application. NWSP provides the content application for unauthenticated user redirection and unauthorized service redirection. The NWSP web application, which includes a set of servlets and JSP pages, 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 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 application. For information on this servlet, see the Javadoc documentation for the `ServiceRedirectControl` class.

### Message Portal Web Application

The sample message portal web 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 JSP pages. `MessagePortalServlet` sets the value of two HTTP request attributes so that the message JSP pages 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 duration that 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 page based on the subscriber's interests as defined in the subscriber profile.

The message JSP page displays its content for `duration`, which is defined in the captiveportal.xml file. The JSP page then forwards the subscriber to the originally requested URL.

Before redirecting to the message JSP page, the sample message portal web application determines the device that the subscriber is using so that the appropriate JSP page serves device-specific content is used.

- If the subscriber's device is a WAP phone, the message web application looks for message JSP pages in the \wap\pages directory.

- If the subscriber’s device is a PDA, the message web application looks for message JSP pages in the \pda\pages directory.
- If the subscriber’s device is not a WAP phone or PDA, the message web application looks for message JSP pages in the \pages directory.

Beneath these directories, the JSP pages for initial logon redirection are in an \initial directory, and the JSP pages for advertising redirection are in an \advertising directory.

To determine what JSP page 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 page 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 the SESM software.