SESM Utility Servlets Quick Reference

This appendix provides brief descriptions of the utility Java servlets that are used in an SESM web application. The SESM components include a set of utility Java servlets that the web application uses to perform a variety of common tasks, such as:

- Ensuring that a session ID is present when an HTTP client does not support cookies
- Finding a resource customized for the user shape
- Redirecting a request to an insecure (HTTP) or secure (HTTPS) version of the same request
- Ensuring that the HTTP client does or does not cache an HTTP response

Most SESM utility servlets are decorators that extend the `Decorator` abstract class and are subclasses of the `Navigator` class. The SESM utility servlets that are decorators have `Decorator` in their names.

The SESM utility servlets are preprogrammed with specific functionality. They do not require customization, but because they are decorators, you can extend a utility servlet with a post-decorator.

This appendix does not provide descriptions of two specialized sets of servlets: SESM controls and dimension decorators. You can find information on these servlets in the following locations:

- For overview information on the control servlets, see the "Controls" section on page 3-2.
- For information on the dimension decorators, see the "Decorating a User Shape" section on page 3-18.

All SESM servlet classes including those for dimension decorators and controls are documented in the Javadoc documentation that is installed with the SESM software.

Tip A few utility servlets are very helpful for testing and debugging an SESM web application: `Snoop`, `TestDimensionDecorator`, `LocaleDecorator`, and `TestUserDecorator`. For information on debugging, see the "Debugging an SESM Web Application" section on page 2-16.

Using the preDecorate and postDecorate Parameters

An SESM utility servlet that is a subclass of either `Navigator` or `Decorator` can have `preDecorate` and `postDecorate` initialization parameters:

Initialization Parameter	Description
`preDecorate`	Specifies a list of names for other decorator servlets that are invoked, in the order listed, *before* the decorator declared in the `<servlet-class>` attribute is invoked.
`postDecorate`	Specifies a list of names for other decorator servlets that are invoked, in the order listed, *after* the decorator declared in the `<servlet-class>` attribute is invoked.

Depending on whether a servlet being declared in `<servlet-class>` is a subclass of `Navigator` or `Decorator`, the behavior of the `preDecorate` and `postDecorate` initialization parameters is different:

- If the servlet named in `<servlet-class>` is a subclass of `Navigator`, each decorator in the `preDecorate` list and the `postDecorate` list is always invoked. Each decorator is called by invoking its `decorateIfNecessary` method.
- If the servlet named in `<servlet-class>` is a subclass of `Decorator` and if the decoration by this servlet is needed, each decorator in the `preDecorate` list and the `postDecorate` list is invoked by calling its `decorateIfNecessary` method.

If a pre-decorator or post-decorator throws a `ServletException`, all decoration stops. No further pre-decorators or post-decorators are invoked. The servlet declared in the `<servlet-class>` attribute is not invoked if it has not already been called. If the servlet declared in the `<servlet-class>` attribute throws a `ServletException`, no post-decorators are invoked.

In the web.xml file, you configure the `preDecorate` and `postDecorate` initialization parameters in the servlet declaration. Consider the following declaration for `MyAccountView` that specifies the `VirtualFile` servlet:

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

In the preceding example, the pre-decorators `User` and `NoCache` are invoked by calling their `decorateIfNecessary` methods. With subclasses of `Navigator` (such as `VirtualFile`), the pre-decorators and post-decorators *are always invoked*. They are called by invoking their `decorateIfNecessary` methods.

In the next example, the `UserDecorator` servlet is a subclass of `Decorator`.

```
<servlet>
    <servlet-name>User</servlet-name>
    <servlet-class>
    com.cisco.sesm.webapp.decorator.UserDecorator
    </servlet-class>
    <load-on-startup>1</load-on-startup>
    <init-param>
        <param-name>postDecorate</param-name>
        <param-value>InitUser, RemovePermission, RefreshShape</param-value>
    </init-param>
  </servlet>
```

In the preceding example, if decoration by `UserDecorator` is needed, the post-decorators `InitUser`, `RemovePermission`, and `NoCache` are invoked by calling their `decorateIfNecessary` methods. With subclasses of `Decorator` (such as `UserDecorator`), the pre-decorators and post-decorators *are invoked only if decoration is needed* by the decorator being declared in `<servlet-class>` (in this example, `UserDecorator`).

For more information on the `Navigator` and `Decorator` classes, see the Javadoc documentation for these classes.

SESM Utility Servlet Quick Reference

The following descriptions briefly explain the SESM utility servlets. For more information on a servlet, see the Javadoc for the servlet class. In each description, the Servlet Mapping entry is the URL-to-servlet mapping that is defined in the NWSP web.xml file. You can add a servlet mapping for any servlet or modify an existing mapping.

Alias Servlet

Servlet Mapping: None
Class: `com.cisco.sesm.navigator.Alias`

Maps a servlet name to a servlet chain, allowing servlets to be invoked in sequence. The `Alias` servlet's initialization parameters include the following:

Initialization Parameter	Description
`to`	Specifies the URI to which `Alias` forwards the request.

In the following example, `Alias` is used with the servlet name `StatusView`:

```
<servlet>
    <servlet-name>StatusView</servlet-name>
     <servlet-class>com.cisco.sesm.navigator.Alias</servlet-class>
    <load-on-startup>1</load-on-startup>
    <init-param>
        <param-name>to</param-name>
        <param-value>/user/nocache/vfile/pages/status.jsp</param-value>
    </init-param>
  </servlet>
```

When the servlet named `StatusView` is requested, the `Alias` servlet forwards the HTTP request to the servlet chain `/user/nocache/vfile/pages/status.jsp`. The `to` initialization parameter for the `Alias` servlet specifies the URI to which the request is forwarded. For information on servlet chains, see the "Servlet Chaining" section on page 3-33.

BuildVersion Servlet

Servlet Mapping: `None`
Class: `com.cisco.sesm.webapp.decorator.BuildVersionDecorator`

Discovers the build version and adds a session scope attribute named `"buildVersion"`. The build version is obtained from the configuration files and identifies which version of this software is running.

The attribute is an internationalized object (`I18nObject`). The possible values of the internationalized object include resources with keys `versionNotAvailable` and `versionError`.

CacheDecorator Servlet

Servlet Mapping: `/cache/*`
Class: `com.cisco.sesm.navigator.CacheDecorator`

Tells the HTTP client to cache the HTTP response. The response is cached by the HTTP client until the shape of the subscriber changes. The `CacheDecorator` servlet writes HTTP headers into the response indicating that the response is to be cached.

If `CacheDecorator` can determine that the HTTP response content is not required because it is already cached, the request is quickly terminated with an empty response. In this case, `CacheDecorator` does not forward to any other decorator or servlet. The HTTP client will use the response that it has already cached.

For information on preventing caching of the HTTP response, see the `NoCacheDecorator` servlet.

CookieDecorator Servlet

Servlet Mapping: None
Class: `com.cisco.sesm.navigator.CookieDecorator`

Adds an attribute named `"cookie"` to the current HTTP session. The value of the attribute represents the HTTP session ID.

Each SESM JSP page is responsible for inserting the cookie into each URL just before the place where a question mark is located at the beginning of the query-string. If there is no question mark, it is inserted just before the place where a question mark would be located.

Tip The SESM web developer does not have to modify the NWSP web application for browsers that do not support cookies. The needed code is already present in NWSP. If the client browser *does support* cookies, the web server automatically adds a cookie with the HTTP session ID to the HTTP headers. If the client browser *does not support* cookies, the NWSP web application has the code, where it is needed, to add a session ID to the URLs that require them. NWSP does not require any additional URL rewriting.

If the HTTP client does not support cookies (for example, because the subscriber has disabled cookies), an SESM web application uses the `"cookie"` attribute to write the session ID into every URL that is returned to the client.

If `CookieDecorator` can determine that the HTTP client supports cookies, the `"cookie"` attribute is set to the empty string. The `"cookie"` attribute is never null once the web application invokes `CookieDecorator`.

If this is the first HTTP response for this HTTP session, `CookieDecorator` does not know whether the HTTP client supports cookies in the HTTP headers. In this case, the `"cookie"` attribute is set to the session ID.

The following example shows a typical use of `CookieDecorator`. (In the NWSP web.xml file, `CookieDecorator` is declared with the servlet name `Cookie`.) In the example, the JSP-page code does the following:

- Uses the `decorate` tag of the Navigator tag library to call the `CookieDecorator.decorateIfNecessary` method to decorate (if needed) the HTTP session with a `"cookie"` attribute.
- Declares the `"cookie"` attribute as a JavaBean instance so that it accessed as a scripting variable.
- Embeds the `cookie` variable into the URL as `"/myAccount<%=cookie%>"`.

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

<nav:decorate name="Cookie" />
<jsp:useBean id="cookie" class="java.lang.String" scope="session" />
...
<th><a href="/myAccount<%=cookie%>">My-Account</a></th>
```

Notice that the JSP-page code inserts `<%=cookie>` into the URL just before the place where a question mark would be located at the beginning of the query-string.

For information on the `decorate` tag, see the "decorate Tag" section on page A-4.

DecoratorPool Servlet

Servlet Mapping: `/pool/*`
Class: `com.cisco.sesm.navigator.DecoratorPool`

Maps from a decorator name to a `Decorator` instance. When a JSP page is used as a decorator, it must register itself in the `jspInit` method by calling `DecoratorPool.register(JspPage)`. For information on JSP-page decorators, see the "Creating or Customizing Dimension Decorators" section on page 3-23.

EndSessionDecorator Servlet

Servlet Mapping: `/end/*`
Class: `com.cisco.sesm.navigator.EndSessionDecorator`

Ends the HTTP session by calling the `HttpSession.invalidate` method.

InitServlet Servlet

Servlet Mapping: `None`
Class: `com.cisco.sesm.core.http.InitServlet`

Initializes the SESM model.

InsecureDecorator Servlet

Servlet Mapping: None
Class: `com.cisco.sesm.navigator.InsecureDecorator`

If the current HTTP request uses Secure Sockets Layer (SSL) encryption, `InsecureDecorator` redirects the request to an insecure (HTTP) version of the same request.

In the redirect, the new URL is based upon the current request not the original request. The current request is different from the original request when the original request invoked a servlet that has forwarded to the current servlet. The query string in the redirect is the same as the query string in the current request.

If the `InsecureDecorator` servlet is successful in redirecting the request, it sends the HTTP client an `SC_MOVED_TEMPORARILY` (302) status code in the response. In addition, `InsecureDecorator` terminates the current request by throwing a `ResponseCompleteNotice`, which is wrapped inside a `ServletException`.

For information redirecting a request with SSL encryption, see the `SecureDecorator` servlet.

HttpSniffDecorator Servlet

Servlet Mapping: None
Class: `com.cisco.sesm.navigator.HttpSniffDecorator`

Adds an `HttpSniffBean` attribute named `"httpSniffBean"` to the HTTP session. `HttpSniffBean` contains information about the HTTP client device. `HttpSniffBean` obtains the information from the HTTP headers. For example, `HttpSniffBean` sets the bean property `clientOSName` to the name of the operating system that is running on the HTTP client device. `HttpSniffBean` properties are as follows:

- `clientBrowserName`
- `clientColor`
- `clientDeviceName`
- `clientMarkupLanguage`
- `clientOSName`
- `clientScriptLanguage`
- `clientSize`
- `connection`

For information on all `HttpSniffBean` properties and the possible values of each, see the Javadoc description of the `get*` methods in the `HttpSniffBean` class. The Javadoc documentation is installed with the SESM software.

Tip Many of the `HttpSniffBean` properties are used to set the values of the dimensions of the user shape. For information on the user-shape dimensions that are set by the SESM software, see the "SESM-supplied Dimension Decorators" section on page 3-21.

In the NWSP web.xml file, the `HttpSniffDecorator` servlet is configured with a deployer-customizable post-decorator (`httpSniff.jsp`) that provides additional "browser sniffing" capabilities. Using this additional information about the client device, `httpSniff.jsp` and some related JSP pages modify `HttpSniffBean` properties based on the characteristics that it detects. The intention is that the service-provider developer, who has knowledge of the client devices to expect, can modify `httpSniff.jsp` to provide better information on these devices. For example, the developer could modify `httpSniff.jsp` to use third-party browser-sniffing software.

In NWSP, `httpSniff.jsp` uses a JavaScript probe to detect client-device characteristics and sets `HttpSniffBean` properties to the appropriate values. If the subscriber's client device supports HTML 4, NWSP uses a service list implemented with JavaScript. The `httpSniff.jsp` also performs some additional checks and sets the relevant bean properties. For example, it sets the `clientDeviceName` property to `wap` if it detects a WAP phone simulator.

L10nContextDecorator Servlet

Servlet Mapping: `/l10n/*`
Class: `com.cisco.sesm.navigator.L10nContextDecorator`

Sets the value of the web application default localization (`L10nContext`) context. In the web.xml file, the initialization parameters for `L10nContextDecorator` define the default values for the current localization context. For information on the `L0nContextDecorator` servlet initialization parameters, see the "Setting a Default Localization Context" section on page 5-14.

LocaleDecorator Servlet

Servlet Mapping: Varies
Class: `com.cisco.sesm.navigator.LocaleDecorator`

Sets the language, country, and (if used) variant of two HTTP session attributes:

- The current localization context (`L10nContext`) attribute—`"com.cisco.aggbu.l10n.context"`
- The `locale` dimension of the user shape attribute—`"shape"`

The `LocaleDecorator` servlet's initialization parameters include the following:

Initialization Parameter	Description
`language`	Specifies the language.
`country`	Specifies the country.
`variant`	Specifies the variant (if any).

In the web.xml file, the initialization parameters in a servlet declaration for a locale decorator define the locale. In the NWSP web.xml file, the servlet declaration for `France` is:

```
<servlet>
    <servlet-name>France</servlet-name>
    <servlet-class>com.cisco.sesm.navigator.LocaleDecorator</servlet-class>
    <load-on-startup>1</load-on-startup>
    <init-param>
        <param-name>language</param-name>
        <param-value>fr</param-value>
    </init-param>
    <init-param>
        <param-name>country</param-name>
        <param-value>FR</param-value>
    </init-param>
    ...
  </servlet>
```

In the web.xml file, `LocaleDecorator` is used with a servlet mapping, which indicates the locale value that will be tested. For example:

```
<servlet-mapping>
    <servlet-name>France</servlet-name>
    <url-pattern>/locale=fr/*</url-pattern>
</servlet-mapping>
```

Given the preceding servlet declaration and servlet mapping for `France`, the following URL would invoke `LocaleDecorator` and set the language and country of the current localization context (`L10nContext`) to `fr` and `FR`, respectively. It also sets the `locale` dimension of the user shape to `fr/FR`.

http://someserver:8080/locale=fr/home

MemoryCheckDecorator Servlet

Servlet Mapping: `None`
Class: `com.cisco.sesm.webapp.decorator.MemoryCheckDecorator`

Determines whether there is sufficient memory on the web server to create an `SESMSession`. If there is sufficient memory, `MemoryCheckDecorator` sets the request attribute `"MemoryOK"` to `Boolean.TRUE`.

If there is not sufficient memory, `MemoryCheckDecorator` first attempts to free memory on the web server. If there is still not sufficient memory to create an `SESMSession`, the servlet does the following:

- Forwards the request to the `busyURL`
- Sets the request attribute `"MemoryOK"` to `Boolean.FALSE`
- Throws a `ResponseCompleteNotice`

In the web.xml file, `busyURL` is an initialization parameter of `MemoryCheckDecorator`. By default, `busyURL` has the value /pages/serverBusy.html.

MessageDecorator Servlet

Servlet Mapping: `None`
Class: `com.cisco.sesm.navigator.MessageDecorator`

Adds a message to the list of messages for the HTTP session by calling `MessagesControl.addMessage(javax.servlet.http.HttpServletRequest, I18nObject)`.

The content of the message is derived from the following request attributes, which are defined by the *Java Servlet Specification Version 2.3*.

- `javax.servlet.error.message`
- `javax.servlet.error.status_code`
- `javax.servlet.error.exception_type`
- `javax.servlet.error.exception`
- `javax.servlet.error.request_uri`

If the response is not committed, `MessageDecorator` forwards to the URL "/messages" and throws a `ResponseCompleteNotice` to prevent further processing.

Note In the web.xml file, the deployer must map the URL "/messages" to a JSP page that displays the messages of the HTTP session.

If the response is committed, `MessageDecorator` allows the request to continue. The error message is not displayed until a new request displays all the messages of the HTTP session.

NoCacheDecorator Servlet

Servlet Mapping: `/cache/*`
Class: `com.cisco.sesm.navigator.NoCacheDecorator`

Prevents caching of the HTTP response by the HTTP client. The `NoCacheDecorator` servlet writes HTTP headers into the response indicating that the response is not to be cached.

For information on caching the HTTP response, see the `CacheDecorator` servlet.

OriginalURLDecorator Servlet

Servlet Mapping: None
Class: `com.cisco.sesm.navigator.OriginalURLDecorator`

Decorates the current HTTP request with an attribute named "originalURL". The attribute value is the original URL that was sent from the HTTP client to the HTTP server. The original URL is the URL of the current HTTP request before any servlet forwarding takes place.

The "originalURL" attribute contains the complete URL of the request, including any parameters. The parameters are included as a query string regardless of whether the method is GET or POST.

To capture the original URL for later use, a JSP page can use OriginalURLDecorator (whose servlet name in the web.xml file is OriginalURL). For example:

```
<%-- Always capture original URL at start of each request. --%>
<nav:decorate name="OriginalURL" />
<jsp:useBean id="originalURL" type="java.lang.String" scope="request"/>
...
<jsp:param name = "okPage" value = "<%=originalURL%>" />
```

PermissionCheckDecorator Servlet

Servlet Mapping: None
Class: com.cisco.sesm.webapp.decorator.PermissionCheckDecorator

Determines whether the subscriber has a specified permission. The PermissionCheckDecorator servlet's initialization parameters include the following:

Initialization Parameter	Description
permission	Specifies the permission for which to check.

The permissions that can be specified in the permission initialization parameter are:

- firewallManage—Permission for managing a personal firewall
- selfManage—Permission needed for modifying account information, such as names, addresses, and so on
- serviceSelection—Permission for service selection
- serviceSubscription—Permission for service subscription
- subAccountManage—Permission for creating, deleting, and managing subaccounts

PermissionCheckDecorator uses the corresponding permission attribute of permissionBean to check whether the subscriber has the permission. This attribute, in turn, has been set depending on whether the user account has the needed privileges.

As an example, if the permission parameter is selfManage, the permissionBean.isSelfManage method is called to check the permission.

- If the user has the needed permission, the corresponding session attribute is set to Boolean.TRUE. The session attribute that corresponds to the permission value given in the initialization parameter permission is named *permission_value* + "Permission". For example, the attribute that corresponds to the permission value firewallManage is "firewallManagePermission".
- If the user does not have the needed permission, a ServletException is thrown, which wraps a PermissionFailedException. The resource key used for this exception is PermissionFailedMessage.

PermissionDecorator Servlet

Servlet Mapping: None
Class: class com.cisco.sesm.webapp.decorator.PermissionDecorator

Adds a `permissionBean` to the HTTP session. The bean is added as an HTTP session attribute named `"permissionBean"`. The `permissionBean` contains Boolean variables indicating whether the current subscriber has permission to perform certain Cisco SESM web portal tasks, such as creating a subaccount.

The `PermissionDecorator` servlet is invoked on any JSP page where the page needs a `permissionBean` in order to determine the permissions that the subscriber has. For example, many JSP pages in NWSP need knowledge of subscriber permissions to determine what buttons (for example, the My Account and Accounts buttons) to display in the navigation bar.

Table B-1 provides information on the `permissionBean` properties.

Table B-1 permissionBean Properties

Bean Property	Description
`isFirewallManage`	The value `true` indicates that the subscriber has the permissions needed to manage a personal firewall. The value `false` indicates that the subscriber does not have the needed permissions.
`isSelfManage`	The value `true` indicates that the subscriber has the permissions needed to modify account information, such names, addresses, and so on. The value `false` indicates that the subscriber does not have the needed permissions.
`isServiceSelection`	The value `true` indicates that the subscriber has the permissions needed for service selection. The value `false` indicates that the subscriber does not have the needed permissions.
`isServiceSubscription`	The value `true` indicates that the subscriber has the permissions needed for service subscription. The value `false` indicates that the subscriber does not have the needed permissions.
`isSubAccountManage`	The value `true` indicates that the subscriber has the permissions needed for creating, deleting, and managing subaccounts. The value `false` indicates that the subscriber does not have the needed permissions.

RedirectRemainder Servlet

Servlet Mapping: `/redirect/*`
Class: `com.cisco.sesm.navigator.RedirectRemainder`

Sends a redirect response to the HTTP client. The redirection is from the current URL to a new URL, where the new URL is the remainder of the current URL after the current servlet name.

For example, because `RedirectRemainder` is mapped to the URL pattern `/redirect/*` in the NWSP web.xml file, the URL `/redirect/next` redirects to `/next`.

RemoveAttributeDecorator Servlet

Servlet Mapping: None
Class: `com.cisco.sesm.navigator.RemoveAttributeDecorator`

Removes an attribute from one of the following scopes: request, session, or application. The `RemoveAttributeDecorator` servlet's initialization parameters include the following:

Initialization Parameter	Description
name	Specifies the name of the attribute.
scope	Specifies the scope of the attribute: request, session, or application.

You can use this decorator to undo the decoration of other decorators. Given the following web.xml file declaration, when RefreshShape is requested, RemoveAttributeDecorator removes the "shape" attribute, which has session scope.

```
<servlet>
    <servlet-name>RefreshShape</servlet-name>
    <servlet-class>
    com.cisco.sesm.navigator.RemoveAttributeDecorator
    </servlet-class>
    <load-on-startup>1</load-on-startup>
    <init-param>
        <param-name>name</param-name>
        <param-value>shape</param-value>
    </init-param>
    <init-param>
        <param-name>scope</param-name>
        <param-value>session</param-value>
    </init-param>
    <...
</servlet>
```

SecureDecorator Servlet

Servlet Mapping: None
Class: com.cisco.sesm.navigator.SecureDecorator

If the current HTTP request does not use Secure Sockets Layer (SSL) encryption, SecureDecorator redirects the request to a secure (HTTPS) version of the same request. As an example, if the request were:

http://someserver:8080/home

The SecureDecorator servlet redirects the request to:

https://someserver:8080/home

In the redirect, the new URL is based upon the current request not the original request. The current request is different from the original request when the original request invoked a servlet that has forwarded to the current servlet. The query string in the redirect is the same as the query string in the current request.

If the SecureDecorator servlet is successful in redirecting the request, the servlet sends the HTTP client an SC_MOVED_TEMPORARILY (302) status code in the response. In addition, SecureDecorator servlet terminates the current request by throwing a ResponseCompleteNotice, which is wrapped inside a ServletException.

For information on redirecting a request without using SSL encryption, see the InsecureDecorator servlet.

ServiceDisplayDecorator Servlet

Servlet Mapping: None
Class: com.cisco.sesm.webapp.decorator.ServiceDisplayDecorator

Sets the session attribute "openWindowName" to the service name, and the session attribute "openWindowURL" to the service URL if it finds a service URL in the profile for the service with the name given by request parameter "service". When the service starts, the NWSP web application uses these session attributes to open a popup window with the specified "openWindowURL" in the window HTTP address field and "openWindowName" as the window name.

If an optional request parameter "serviceURL" is present, it will override any URL retrieved from the service profile.

A URL is considered valid only if its length is greater than one.

ShapeDecorator Servlet

Servlet Mapping: /shape/*
Class: com.cisco.sesm.navigator.ShapeDecorator

Creates a Shape object for the user and ensures that there is a "shape" attribute with session scope that holds the Shape object. The Shape servlet's initialization parameters include the following:

Initialization Parameter	Description
dimensions	Specifies a set of zero or more dimension IDs that the SESM software uses when it creates the dimensions for a user shape.

A Shape object encapsulates the set of characteristics that define the web resources available for a specific subscriber. A Shape object consists of one or more dimensions. Each dimension corresponds to a characteristic of the subscriber (for example, the browser software of the subscriber). The value of each dimension specifies one or more directories that the SESM software searches for requested resources for this subscriber.

For information on the ShapeDecorator servlet, see the "Configuring User-Shape Dimensions" section on page 3-19.

Snoop Servlet

Servlet Mapping: /snoop/*
Class: com.cisco.sesm.navigator.Snoop

For testing purposes, displays information about the HTTP session and request. The content type of the response that Snoop sends is "text/plain" so that you can view the information on different types of devices (for example, HTML browsers and WAP phones).

Snoop is used for debugging an SESM web application and is not used in production applications.

SESMSessionDecorator Servlet

Servlet Mapping: /session/*
Class: com.cisco.sesm.webapp.decorator.SESMSessionDecorator

Adds an attribute named "sesmSession" to the current HTTP *request*. The value of the attribute represents the active SESM session, which has been synchronized with the SSG.

If a new "SESMSession" attribute is successfully created and authenticated, initUser.jsp executes. The initUser.jsp page is a deployer-customizable decorator that you can modify to perform any subscriber-related initialization tasks that need to be accomplished after it is known that the user is authenticated. For example, initUser.jsp could be used to set the current localization (L10nContext) to the locale defined in the subscriber profile.

TestDimensionDecorator Servlet

Servlet Mapping: Varies
Class: `com.cisco.sesm.navigator.TestDimensionDecorator`

For testing purposes, sets a dimension of the user shape to a specified value. The `TestDimensionDecorator` servlet's initialization parameters include the following:

Initialization Parameter	Description
`id`	Specifies the dimension ID.
`value`	Specifies a value for the dimension identified by `id`.

In the web.xml file, the initialization parameters in a servlet declaration for the test dimension decorator define the dimension ID and the value. In the NWSP web.xml file, the servlet declaration for `WAPDevice` is:

```
<servlet>
    <servlet-name>WAPDevice</servlet-name>
    <servlet-class>com.cisco.sesm.navigator.TestDimensionDecorator</servlet-class>
    <load-on-startup>1</load-on-startup>
    <init-param>
        <param-name>id</param-name>
        <param-value>device</param-value>
    </init-param>
    <init-param>
        <param-name>value</param-name>
        <param-value>wap</param-value>
    </init-param>
    ...
</servlet>
```

In the web.xml file, `TestDimensionDecorator` is used with a servlet mapping, which indicates the dimension value that will be tested. For example:

```
<servlet-mapping>
    <servlet-name>WAPDevice</servlet-name>
    <url-pattern>/device=wap/*</url-pattern>
</servlet-mapping>
```

Given the preceding servlet declaration and servlet mapping for `WAPDevice`, the following URL would invoke `TestDimensionDecorator` and set the `device` dimension to `wap`.

n

TestUserDecorator Servlet

Servlet Mapping: Varies
Class: `com.cisco.sesm.navigator.TestUserDecorator`

For testing purposes, authenticates a user name and password. The user names and passwords that can be authenticated with `TestUserDecorator` are those defined in the aaa.properties file, which is used for demonstration mode. The `TestDimensionDecorator` servlet's initialization parameters include the following:

Initialization Parameter	Description
username	Specifies a user name given in the aaa.properties file.
password	Specifies the password for the user identified in username.

In the web.xml file, the initialization parameters in a servlet declaration for a test user decorator define the user name and the password. In the NWSP web.xml file, the servlet declaration for golduser is:

```
<servlet>
    <servlet-name>golduser</servlet-name>
    <servlet-class>com.cisco.sesm.webapp.decorator.TestUserDecorator</servlet-class>
    <load-on-startup>0</load-on-startup>
    <init-param>
        <param-name>username</param-name>
        <param-value>golduser</param-value>
    </init-param>
    <init-param>
        <param-name>password</param-name>
        <param-value>cisco</param-value>
    </init-param>
...
</servlet>
```

In the web.xml file, TestUserDecorator is used with a servlet mapping, which indicates the user name and password value that will be tested. For example:

```
<servlet-mapping>
    <servlet-name>golduser</servlet-name>
    <url-pattern>/user=golduser/*</url-pattern>
  </servlet-mapping>
```

Given the preceding servlet declaration and servlet mapping for golduser, the following URL would invoke TestUserDecorator and try to authenticate the user name golduser with the password cisco.

http://someserver:8080/user=golduser/home

UserDecorator Servlet

Servlet Mapping: /user/*
Class: com.cisco.sesm.webapp.decorator.UserDecorator

Ensures that the user is known. UserDecorator adds the user name as an attribute to the HTTP session. This may require the user to authenticate. If the user is not known, the getNewUser method is called.

Tip Many SESM JSP pages use UserDecorator as a pre-decorator to ensure that the subscriber cannot view a web page until authentication has occurred.

With UserDecorator, decoration is necessary until the user is known and the user is authenticated. SESM/SSG authentication can be lost without notification, leaving the web application with a user name but no SESM authentication. If you want to ensure that there is an authenticated user, invoke UserDecorator. For example, a JSP-page view might not care what the user name is, but by invoking UserDecorator, it ensures that the user is authenticated and that authentication takes place.

In NWSP, if the user is unknown, UserDecorator forwards to AccountLogonControl.

VirtualFile Servlet

Servlet Mapping: `/vfile/*`
Class: `com.cisco.sesm.navigator.VirtualFile`

Translates a virtual file name into an actual file name according to the current values for the dimensions of the user shape. The actual file name is a URI for a web resource. `VirtualFile` also attempts to find the resource located by the actual file name and, if it finds the resource, forwards the HTTP request to the resource. The `VirtualFile` servlet's initialization parameters include the following:

Initialization Parameter	Description
`vfile`	Specifies the path name for a virtual file.

You can specify the virtual file name in either of two ways:

- In the web.xml file, as the value of the `vfile` initialization parameter to `VirtualFile`
- In a JSP page of an SESM web application, as the remainder of the URL in a servlet chain

In the web.xml file, you can specify the virtual file name as the value of the `vfile` initialization parameter to `VirtualFile`. Many of the NWSP views use this method to invoke `VirtualFile`. For example:

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

In a JSP page of an SESM web application, the `VirtualFile` servlet can be invoked when the web application needs to find a resource that may differ according to the user shape. The virtual file name is the remainder of the URL in a servlet chain. In the following example, a URL for `/pages/help.jsp` contains a servlet chain that invokes `VirtualFile` (which is mapped to the URL `/vfile/*`).

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

In a servlet chain, `VirtualFile` (`vfile`) must be located immediately before the virtual file name for the web resource—in the preceding example, immediately before `/pages/help.jsp`. For more information on the `VirtualFile` servlet, see "Mapping a Virtual File Name to an Actual File Name" section on page 3-34.