SESM Tag Libraries

This appendix provides information on configuring an SESM tag library and describes the SESM tag libraries other than the Localization tag library:

- Iterator Tag Library, page A-2
- Navigator Tag Library, page A-3
- Shape Tag Library, page A-4

For information on the Localization tag library, see the "Using the Localization Tag Library" section on page 5-6.

Configuring a Tag Library

If you are using an SESM tag library in a web application, you must perform the following steps to configure the tag library.

For a sample SESM web application, you *do not* need to perform this procedure. The SESM tag libraries are already defined in the web.xml file of each SESM web application. The tag library descriptor files and the `com.cisco.sesm.contextlib.jar` file already exist in, respectively, the \WEB-INF and \WEB-INF\lib directories.

Step 1 Copy the tag library's descriptor file from *install_dir*\nwsp\webapp\WEB-INF to the \WEB-INF directory of your web application. Table A-1 lists the tag library descriptor files.

Table A-1 SESM Tag Library Descriptor Files

SESM Tag Library	Tag Library Descriptor File
Iterator	iterator.tld
Localization	localization.tld
Navigator	navigator.tld
Shape	shape.tld

Step 2 Copy the `com.cisco.sesm.contextlib.jar` from *install_dir*\nwsp\webapp\WEB-INF\lib directory to the \WEB-INF\lib directory of your web application.

Step 3 Add the appropriate `<taglib>` element to your web application deployment descriptor in \WEB-INF\web.xml. You can cut and paste the `<taglib>` elements for the libraries from the web.xml file for NWSP. For example, the `<taglib>` element for the Localization tag library is:

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

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

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

Iterator Tag Library

The Iterator tag library provides tags that allow a JSP page to iterate over a Java collection. The Iterator tag library implements a `java.util.Iterator` and includes these tags:

- `start`—Marks the start and end of an iteration.
- `val`—Obtains the value of the current element of an iteration loop.

start Tag

Table A-2 lists the attributes of the `start` tag. The `start` tag uses either the iteration instance given in `over`, or the `Iterator` instance given in `value`. Either the `over` or `value` attribute must be supplied.

Table A-2 Start Tag Attributes

Attribute	Description	Required	Runtime Expression
`type`	Specifies the class of the variable given in `name`.	Yes	No
`name`	Specifies a variable name for accessing the value of the current element of the iteration.	Yes	No
`over`	Provides an instance of an iteration. The `start` tag invocation must include either `over` or `value`.	No	Yes
`value`	Provides an instance of an `Iterator`. The `start` tag invocation must include either `over` or `value`.	No	Yes

The following example uses the `start` tag to iterate through a collection of messages.

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

<it:start over="<%=messagesBean.getMessages()%>" name="message"
type="com.cisco.sesm.i18n.I18nObject">
<tr>
<td><l10n:format shortDateTime="<%= new Date()%>">timestamp</l10n:format></td>
<td><l10n:format object="<%= message %>">message</l10n:format></td>
</tr>
</it:start>
```

In the preceding example, the `messages` variable is used to access the current value of the iteration. Each message is an internationalized object of the type `com.cisco.sesm.i18n.I18nObject`.

val Tag

Table A-3 lists the attributes of the `val` tag. When the `val` tag is specified without the `value` attribute, the tag obtains the value of the current element of an iteration loop.

Table A-3 Val Tag Attributes

Attribute	Description	Required	Runtime Expression
`value`	Specifies Java code to evaluate for each iteration through the collection. The `value` attribute is optional. When a `val` tag is specified but no `value` attribute is given, the value of this iteration is the value of the current element.	No	Yes

The following example uses the `start` and `val` tags to iterate through a collection of titles.

```
<html>
 <%@ page import="java.util.*" %>
 <%@ taglib uri="http://www.cisco.com/taglibs/iterator" prefix="it" %>

 <head>
 <title>Iterator Example</title>
 </head>
 <body>
 <%
  Vector titles = ... A vector of titles of type java.lang.String
 %>
 <table>
 <!-- Define a header row for the table -->
   <tr>
     <th>Title</th>
     <th>Description</th>
   </tr>

 <!-- Iterate over titles. The current value will be in a variable called title -->
 <it:start over="<%=titles%>" name="title" type="java.lang.String">
   <tr>
     <td><it:val>a title</it:val></td>
     <td><it:val value="<%=getDescription(title)%>">a
 description</it:val></td>
   </tr>
 </it:start>

 </table>
 </body>
 </html>
```

Navigator Tag Library

The Navigator tag library and its `decorate` tag allow a JSP page to invoke a decorator if decoration is needed.

decorate Tag

The `decorate` tag invokes a decorator if decoration is needed. The `decorate` tag is an efficient way to invoke a decorator because the `decorate` tag determines whether decoration is needed by calling the decorator's `decorateIfNecessary` method. For information on decorators and `decorateIfNecessary`, see the description of the `Decorator` class in the Javadoc documentation for SESM.

The decorator specified in the tag's `name` attribute must be declared as a servlet in the web application's web.xml file. For example, the servlet name `Cookie` is declared in the following entry to be associated with the class `com.cisco.sesm.navigator.CookieDecorator`.

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

The decorator specified in the `name` attribute must be registered with the `DecoratorPool` method. If it is not registered, the `decorate` tag throws a JSP-page exception.

Table A-4 lists the attributes of the `decorate` tag.

Table A-4 decorate Tag Attributes

Attribute	Description	Required	Runtime Expression
`name`	Invokes a decorator if decoration is needed.	Yes	Yes

The following example uses the `decorate` tag to invoke the decorator having the servlet name `Cookie`.

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

The effect of the preceding `decorate` tag is to call the `CookieDecorator.decorateIfNecessary` method. Any pre-decorators or post-decorators specified for `CookieDecorator` in the web.xml file are also invoked. For information on specifying pre-decorators and post-decorators, see the "Using the preDecorate and postDecorate Parameters" section on page B-2.

Shape Tag Library

The Shape tag library provides a set of tags that translate a virtual file name into an actual file name according to the current values for the dimensions of the user shape. The JSP pages of an SESM web application use the tags of the Shape tag library when retrieving a shape-specific web resource. For information on the user shapes and finding an actual file, see the "User Shapes and User-Shape Decoration" section on page 3-8 and the "Mapping a Virtual File Name to an Actual File Name" section on page 3-34.

The Shape tag library includes these tags:

- `file`—Translates a virtual file name into an actual file name according to the current values for the dimensions of the user shape.
- `path`—Translates a virtual file name into an actual file name according to the current values for the dimensions of the user shape.

The path tag provides some functionality that is not provided by the file tag and that is useful in Dreamweaver Design View (as opposed to Dreamweaver Live Data View).

Tip Use the file tag, in most situations, rather than the path tag. The file tag does not evaluate the tag body and, therefore, is more efficient. The exception is that you should use the path tag when linking a JSP page to a style sheet. The file tag does not work for linking to a style sheet.

file Tag

The file tag translates a virtual file name into an actual file name according to the current values for the dimensions of the user shape. The actual file name is a URI identifying a web resource. A Cisco SESM web application employs the user-shape mechanisms for customizing the web resources served to a subscriber. In its JSP pages, the web application uses the file tag to specify each web resource that can differ according to the user's shape.

For example, if the banner.jpg file can be different depending on the user's shape, an SESM web application uses the file tag in a JSP page when specifying that resource:

```
<img src="<shape:file name='/images/banner.jpg'/>"
```

The file tag translates the virtual file name specified by the tag's name attribute into an actual file name (a URI) based on the following:

- Values for the dimensions of the user shape. The user shape is stored in an HTTP session attribute named "shape".
- Files that reside within the SESM web application (starting at document root).

If the file tag finds the web resource within the web application, it returns the actual URI. If the file tag does not find the resource, it throws an exception. The file tag is similar to calling Shape.getPath(name, application), where application is the web application.

Note The file tag does not work when linking a JSP page to a style sheet. Use the path tag when linking a JSP page to a style sheet.

Table A-5 lists the attributes of the file tag.

Table A-5 File Tag Attributes

Attribute	Description	Required	Runtime Expression
name	Specifies the path name of the virtual file that will be translated into an actual file name (URI).	Yes	Yes

The following example uses the file tag to get the actual file name for /images/nwsp_mid_banner.jpg.

```
<%@ taglib uri="http://www.cisco.com/taglibs/shape" prefix="shape" %>
...
<td><img src="<shape:file name='/images/nwsp_mid_banner.jpg'/>" width="465"
height="53"></td>
```

In the example, the `file` tag translates the path for the virtual file `/images/nwsp_mid_banner.jpg` into an actual file name (for example, `/gold/de/images/nwsp_mid_banner.jpg`) based on the values of the dimensions of the user's shape.

path Tag

Note Use the `file` tag, in most situations, rather than the `path` tag. The `file` tag does not evaluate the tag body and, therefore, is more efficient. The exception is that, in a production SESM web application, you should use the `path` tag when linking a JSP page to a style sheet. The `file` tag does not work for linking to a style sheet.

The `path` tag translates a virtual file name into an actual file name according to the current values for the dimensions of the user shape. The actual file name is a URI identifying an existing web resource. The path for the virtual file is given in an HTML tag attribute, such as `src` or `href`.

In the following example, the `path` tag evaluates its tag body and replaces the virtual file name for the style sheet specified for the `href` attribute with an actual file name:

```
<shape:path attrib="href">
   <link rel="stylesheet" href="/styles/sesm.css" type="text/css">
</shape:path>
```

For example, if the user shape is determined by the `brand` dimension, the `path` tag might locate the style sheet for a gold-brand user at `/gold/styles/sesm.css`. In this case, the preceding example generates the following:

```
<link rel="stylesheet" href="/gold/styles/sesm.css" type="text/css">
```

The `path` tag can be useful in the following situation:

- If you use Dreamweaver's Design View mode (as opposed to Live Data View) to edit a JSP page
- If the runtime value for a web resource will be determined dynamically and differ from the value specified for an HTML tag attribute such as `src`, `href`, and `background`

In Design View, the value specified for an attribute such as `src` can cause Dreamweaver to display a broken link if the file does not exist at the specified location because the actual file name (URI) is determined at runtime.

If a `file` attribute is specified, the `path` tag evaluates its tag body at runtime and replaces the specified value of the `src` attribute with the value specified in the `file` attribute. Consider the following example:

```
<shape:path file="<%=getCurrentImageName(request, name)%>"
   <img src="/images/up1.gif" width=32 height=18>
</shape:path>
```

At runtime, the preceding example produces the following virtual file name, which the `path` tag then translates into an actual file name (URI) based on the user shape.

```
<img src="value_of_getCurrentImageName_call" width=32 height=18>
```

In Design View mode, Dreamweaver uses the tag body, and the preceding example produces this result:

```
<img src="/images/up1.gif" width=32 height=18>
```

Table A-6 lists the attributes of the path tag.

Note If you use the id attribute when the body of the path tag has dynamic content, different id values are needed to reflect the possible states of the dynamic content. With dynamic content, if the id attribute is used with the same value for all states, the cache must be cleared in order to see a change to the dynamic content.

Table A-6 Path Tag Attributes

Attribute	Description	Required	Runtime Expression
attrib	Defines the HTML tag attribute (for example, src, href, or background) that appears in the path tag body and whose value will be translated into an actual file name (a URI) at runtime. If no attrib attribute is given, the HTML tag attribute whose value is translated is src.	No	Yes
file	Specifies a replacement path for the file that will provide the web resource. At runtime, the replacement path given for file is the virtual file name that the SESM software translates into an actual file name (a URI) based on the user shape.	No	Yes

id	Provides caching of the web resource. If the id attribute is not used in an invocation of the path tag, a new instance of the path tag object is created every time the JSP page with the invocation is requested. The id value specifies a unique identifier for the path tag. For example:
	`<shape:path id="UP1"`
	`<img src="/images/up1.gif" width=32 height=18>`
	`</shape:path>`
	When an id attribute is specified, the path tag reuses the results of processing the initial instance of the tag if an identical use of the path tag is encountered. With the id attribute, tag instances are considered identical if the user shape in the first tag invocation matches the user shape in the second tag invocation.
	The value given for id must be globally unique within the scope of the web application.
		No	Yes

idFile	Provides caching of the web resource. The idFile value specifies a static replacement path for the file that will provide the web resource. At runtime, the replacement path given for idFile is the virtual file name that the SESM software translates into an actual file name (a URI) based on the user shape. The idFile attribute can be used instead of the id attribute when a runtime computation is not used for the replacement path. If an idFile attribute is specified, the path tag reuses the results of processing the initial instance of the tag if an identical use of the path tag is encountered. With the idFile attribute, tag instances are considered identical if the user shape in the first tag invocation matches the user shape in the second tag invocation. The use of an id or file attribute takes precedence over an idFile attribute.	No	Yes
shape	Specifies a user Shape object explicitly rather than using the Shape stored in the "shape" attribute of the current HTML session. The "shape" attribute of the HTML session is overridden if the shape attribute is specified in this tag or in a previous shape attribute that appears on the same JSP page. The most recent definition for the tag's shape attribute has precedence.	No	Yes