CHAPTER 5

SESM Internationalization and Localization

The Cisco SESM software includes a set of components that can be used for internationalization and localization. This chapter provides some general information on internationalization and localization. It also explains the Cisco SESM components and techniques that help a deployer internationalize and localize an SESM web application. This chapter discusses these topics:

For information on configuring a tag library, see the "Configuring a Tag Library" section on page A-1.

Internationalization and Localization

Internationalization is the process of designing an application so that it can be adapted for various languages and regions without programming changes. The term *i18n* is sometimes used as an abbreviation for internationalization because that word begins with i, ends with n and contains 18 characters in between. Text in status and error messages that varies depending on the culture needs to be internationalized. Other data that vary by culture include labels on web application buttons and fields, and the formats of dates, times, numbers, and currencies.

Localization is the process of adapting an application for a specific language or region by adding locale-specific components and text. The term *L10n* is sometimes used as an abbreviation for localization because that word begins with L, ends with n, and contains 10 characters in between. Typically, the localization process involves translating text messages to another language and, where required, providing locale-specific images.

SESM Components for Internationalization and Localization

The Cisco SESM web components include a set of Java classes and JSP tag libraries that help the deployer to internationalize and localize an SESM web application. These Java classes and techniques extend the classes and techniques that are part of the J2EE classes. The Localization tag library provides methods for setting and changing the localization context associated with the subscriber's HTTP session. For example, the Localization tag library allows a web application to change the locale and time zone of the subscriber's session.

It is not required that a Cisco SESM web application use the extended classes and techniques that are provided with the SESM software. An SESM web application can use conventional Java techniques for internationalization and localization. However, the classes and tags provided with the Cisco SESM software are specifically designed for use with a web application.

Localizing a Web Application

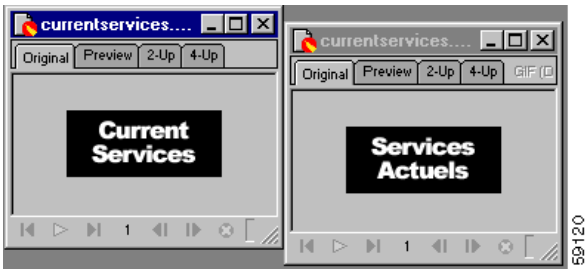
The sample SESM web applications such as NWSP use the SESM classes and techniques for internationalization and localization. The sample NWSP web application is implemented so that it dynamically detects each subscriber's language and country and generates SESM pages with resources appropriate for the language and country. For more information on these mechanisms and techniques, see the "User Shapes and User-Shape Decoration" section on page 3-7 and the "Decorating a User Shape" section on page 3-16.

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

- Changing text, icons, and images
- Creating additional properties files

The first set of modifications involves changing SESM web application components so that they meet the needs of a language other than English. All text in icons and images must be translated into the subscriber's language. For example, the current services image in the NWSP sample contains the English language text "Current Services." If an SESM web application requires localization for the French language, the text in this image would need to be translated into the French language equivalent "Services Actuels" as shown in Figure 5-1.

Figure 5-1 Localizing currentservices.gif



The SESM web components include many of the images and icons in Portable Network Graphics (PNG) format, which is the Fireworks native format. The PNG images and icons are located in the /nwsp/docroot/assets directory. You can change the text in a PNG image using Fireworks or another image editor and then export the GIF image to the *webapp*/docroot/images directory, or a location in a sparse-tree directory structure where locale-specific images reside (for example, *webapp*/docroot/**de**/images for a set of German-language images).

The second set of modifications for localization is that message text and other items in resource bundles must be translated, and an additional properties file must be created for each new language. The NWSP web application includes the logic to determine and set the subscriber's locale so that it uses the appropriate properties file. For information on translating a properties file into another language, see the explanation at the beginning of the messages.properties file in the /nwsp/docroot/Web-inf/classes directory.

For information on resource bundles, see the "Using Resource Bundles" section on page 5-3.

Using Resource Bundles

Resource bundles contain locale-specific data that varies depending on the user's language and region, such as translatable text for status and error messages and for labels on GUI elements. A resource bundle allows a Cisco SESM deployer to separate localizable elements from the rest of the web application.

The localizable elements are stored in a set of properties files, one for each language-region combination. If an SESM web application uses resource bundles, the web application determines the subscriber's locale and then loads the appropriate resource bundle. If the subscriber switches locales, the web application can load a different resource bundle.

Resource bundles allow you to design and write an SESM web application that can be easily localized for the subscriber's language and region. An SESM web application can add additional resource bundles if a new locale is required.

The following sections provide some general information on resource bundles, properties files, and their use with a Cisco SESM web application. For detailed information on resources bundles, see the description of these classes at the Java 2 platform area of the java.sun.com web site:

- java.util.ResourceBundle
- java.util.Locale
- java.util.TimeZone
- java.text.MessageFormat

Using Properties Files

A resource bundle can be implemented with a set of one or more properties files. A *properties file* is a plain-text file that contains key-value pairs for each localizable item. For example, the English version of a sample properties file that contains message text is:

```
# English version of properties file

AMGreeting = Good Morning
PMGreeting = Good Evening
NotValid = Invalid Value
```

The French version of the properties file is:

```
# French version of properties file

AMGreeting = Bonjour
PMGreeting = Bonsoir
NotValid = Valeur Incorrecte
```

The keys and values in a properties file must be string values. A Cisco SESM web application specifies the key when it retrieves the message text from a resource bundle. The key is case-sensitive. The value associated with the key is the localized text. Properties files are not part of the Java source code. Properties files must be located in a directory specified by the `CLASSPATH` variable or in some location where the Java compiler finds web application classes. In the NWSP sample web application, the properties files (for example, messages_en.properties) reside in the \docroot\Web-inf\classes directory.

The value part of the key-value pair can include HTML markup. The value is passed straight through to the HTML page with no changes. Because the `L10nContext` class and the Localization tag library do not process special HTML characters when retrieving a value from a properties file, you can use HTML

markup in the value. Special HTML characters that appear in a value and that are not part of HTML markup must use ISO 8859-1 character encodings. For example, if you want the less-than character (<) to appear in a value as something other than HTML markup, it must be coded as &lt.

You can find information on how to write and retrieve the entries in a properties file from these sources:

- For information on the required syntax for the key-value pairs, see the description of the `java.util.Properties.load` method in the Java 2 platform area of the java.sun.com web site.
- For information on using the Localization tag library to retrieve values from a resource bundle, see the "resource Tag" section on page 5-10 and the "template Tag" section on page 5-11.

The filename of each properties file has a base name and an optional locale identifier. The optional locale identifier can include a language name, a country name, and a variant name. Elements are separated from each other by the underscore (_) character. All properties files must have the .properties extension. As an example, for the resource bundle SESMResources, Table 5-1 shows five examples of properties file names.

Table 5-1 Names of Properties Files in a Resource Bundle

Properties Filename	Description
SESMResources.properties	base name—The file is the default version.
SESMResources_en.properties	base name and language name—The file is the English version.
SESMResources_fr.properties	base name and language name—The file is the French version.
SESMResources_fr_CA.properties	base name and language name and country name—The file is the French version used for Canada.
SESMResources_fr_CA_WIN.properties	base name and language name and country name and variant name—The file is the French version used for Canada on the Windows operating system.

Properties files within the same bundle share the same base name and have the same key-value pairs. The `ResourceBundle` class associates a parent with each bundle. For example, SESMResources_fr is the parent of SESMResources_fr_CA. If the `ResourceBundle` class looks for the file SESMResources_fr_CA.properties and cannot find the file, it uses the parent file SESMResources_fr.properties or the file having the base name if it cannot find a parent file.

Note If an SESM web application uses the `L10nContext` class or the Localization tag library, the algorithm that determines the default resource bundle calls the `L10nContext.getDefault` method (not the `Locale.getDefault` method) to get the default that is associated with the SESM web application. For information on the benefits of using the `L10nContext` class and the Localization tag library, see the "Using the Localization Tag Library" section on page 5-5.

For detailed information on resource bundle properties files and the search algorithm used by the `ResourceBundle` class, see the class description in the Java 2 platform area of the java.sun.com web site.

Using the Localization Tag Library

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

- `context`—Sets the characteristics of the current localization context (`L10nContext`).
- `locale`, `timeZone`, and `language`—Get a string describing the specified characteristic of the current localization context.
- `country`—Gets a string for a country specified by the attribute used with the tag.
- `format`—Converts currency, number, date, and time values according the formatting conventions associated with the Java class `java.text.MessageFormat` and the current `L10nContext` localization context. Also, formats an internationalized object
- `resource`—Obtains a resource from the resource bundle specified for the current localization context and for a specified key.
- `template`—Replaces tokens in a template with parameter values. The `template` tag can be used with the `resource` tag for token replacement in a resource.

The Localization tag library is specifically designed for web application localization. The standard Java `Locale.getDefault` method gets the current value of the default locale for this instance of the Java Virtual Machine (JVM). This default locale is shared across all applications running on that JVM.

In contrast, if the `L10nContext` class and the Localization tag library is used, the class and tag library get the current value of the default locale for the class loader. If each application running on the JVM has its own class loader, then each application has its own localization-context object created using the Cisco SESM `L10nContext` class. Thus, the benefit to using the Localization tag library and the SESM `L10nContext` class is that another application running on the JVM can change the default locale associated with the `Locale` object, but it cannot change a Cisco SESM web application's default `L10nContext` object.

context Tag

The `context` tag can be used to create a scripting variable of the type `L10nContext`, specify the localization context explicitly, and set the characteristics of the current localization context (`L10nContext`). A localization context combines a locale, time zone, resource bundle base name, preferred locales, otherwise locale, and scope.

When a web application uses a `context` tag, the tag implicitly declares a localization context. The `context` tag's current localization context inherits values for locale, time zone, and resource bundle base name from the first of the following localization contexts that exists:

1. A parent `context` tag.
2. If no parent `context` tag is used, the values come from a localization context stored in an `L10nContext` object. The `context` tag software searches for a localization context having (in this order) page, request, session, or application scope. It uses the first context found.
3. If none of the preceding exist, the values come from the default localization context, which it obtains with the `L10nContext.getDefault` method.

A web application can override the current localization context's inherited values with the `context` tag attributes such as `locale`, `preferredLocales`, `timeZone`, and `resourceBundleName`. For example, a web application can set the locale of a user's localization context to Germany for the duration of an HTTP session as follows:

```
<l10n:context locale = "<%= Locale.GERMANY%>" scope = PageContext.SESSION_SCOPE />
```

A localization context object can exist for each of four scopes: page, request, session, and application. As shown in the preceding example, the `scope` attribute defines the scope of a localization context and can be specified with any other `context` tag attribute.

The current localization context, including all its characteristics, can be specified by setting the context using the `context` attribute and an existing `L10nContext` object. For example:

```
<l10n:context context = "<%= someL10nContextObject %>" />
```

If a tag in the Localization tag library is used *inside* the tag body of a `context` tag, the tag's functionality reflects the localization context. In the following example, the currency amount is formatted according to German conventions (`99,99 DM`) because the formatting occurs within the `context` tag body where the localization context's locale is Germany.

```
<% double amount = 99.99; %>

<!-- Set the locale of the current L10nContext localization context -->
<l10n:context locale = "<%= Locale.GERMANY%>" >
Amount formatted as currency:  <l10n:format currency="<%= amount %>" /> <BR>
</l10n:context>
```

If a tag in the Localization tag library is used *outside* the tag body of a `context` tag, the localization context is the same as having a parent context tag with no attributes set.

Table 5-2 lists the attributes of the context tag.

Table 5-2 Context Tag Attributes

Attribute	Description	Required	Runtime Expression
`variable`	Declares a variable that has the type `L10nContext` and the specified name. The named variable can be used as a scripting variable within the tag body. The value assigned to `variable` is the declared `L10nContext` object's name. See the example following this table.	No	No

`context`	Specifies the current localization context explicitly. For example:
	`<l10n:context`
	`context = "<%= someL10nContextObject %>" />`
		No	Yes

`resourceBundleName`	Specifies the resource bundle base name.	No	Yes

`locale`	Specifies the locale of the current localization context. For example:
	`<l10n:context locale = "<%=`
	`Locale.GERMANY%>"/>`
		No	Yes

`timeZone`	Specifies the time zone of the current localization context.	No	Yes

Table 5-2 Context Tag Attributes (continued)

Attribute	Description	Required	Runtime Expression
`preferredLocales`	Specifies the preferred locales of the current localization context. This attribute requires that an otherwise locale be specified with the `otherwise` attribute.	No	Yes
`otherwise`	Specifies the locale to use if no resource bundle can be found for any of the preferred locales. This attribute requires that a preferred locale be specified with the `preferred` attribute.	No	Yes
`scope`	Specifies the scope of the current localization context. Allowed values for scope are: • `PageContext.APPLICATION_SCOPE` • `PageContext.SESSION_SCOPE` • `PageContext.REQUEST_SCOPE` • `PageContext.PAGE_SCOPE` For the meaning of each scope, see the documentation for the Java `PageContext` class in the Java 2 platform area of the java.sun.com web site.	No	Yes

The following example shows how to declare and use the scripting variable that is created with the `context` tag's `variable` attribute. In the example, the scripting variable is used to access the `getLocale` method of the `L10nContext` class.

```
<%@ taglib uri="http://www.cisco.com/taglibs/localization" prefix="l10n" %>
...
<%-- Set the locale and encoding of the response --%>
<%-- to the current locale of the L10nContext.   --%>
<l10n:context variable="l10nContext">
<%
response.setLocale(l10nContext.getLocale());
Log.debug("decorateResponse.jspi, locale=", response.getLocale());
%>
</l10n:context>
```

locale, timeZone, and language Tags

The `locale`, `timeZone`, and `language` tags get a text description of the specified characteristic for the current localization context (`L10nContext`):

- `locale`—Gets a text description of the locale.
- `timeZone`—Gets a text description of the time zone.
- `language`—Gets a text description of the language.

The following example uses the `locale` and `timeZone` tags to obtain a text description for the locale and time zone of the current localization context.

```
<%@ taglib uri="http://www.cisco.com/taglibs/localization" prefix="l10n" %>
...
The current locale is: <l10n:locale /> <BR>
The current timeZone is: <l10n:timeZone /> <BR>
```

The generated HTML page displays the following:

```
The current locale is: English (United States)
The current timeZone is: America/New_York
```

country Tag

The `country` tag gets a text description for a country. The attribute supplied with the `country` tag specifies the country. If no attribute is used, the description is for the country of the current localization context (`L10nContext`). The text description is always in the language of the current localization context. Table 5-3 lists the attributes of the `country` tag.

Table 5-3 Country Tag Attributes

Attribute	Description	Required	Runtime Expression
`code`	Gets the text description for the country specified by a two-character ISO 3166 country code, such as “JP” for Japan. For a list of country codes, see: http://www.chemie.fu-berlin.de/diverse/doc/ISO_3166.html	No	Yes
`locale`	Gets the text description for the country of the specified locale.	No	Yes
`context`	Gets the text description for the country of the specified localization context.	No	Yes

The following example uses the `country` tag and its various attributes to obtain text descriptions for some countries.

```
<%@ taglib uri="http://www.cisco.com/taglibs/localization" prefix="l10n" %>
...
<!-- No attribute specified -->
The country of the current localization context is: <l10n:country /> <BR>

<!-- Country code specified -->
The country associated with the country code "CH" is: <l10n:country code = "CH" /><BR>
<BR>

<!-- For the swissFrench object, set language to French and the country to Switzerland -->
<% Locale swissFrench = new Locale("fr", "CH"); %>

<!-- Set the locale of the current L10nContext localization context -->
<l10n:context locale = "<%= swissFrench %>" >

After changing the locale of the current localization context, <BR>
the country of the localization context <BR>
(in the language of that localization context) is: <l10n:country /> <BR>

</l10n:context>

<BR>

<!-- Set the locale of the current L10nContext localization context -->
<l10n:context locale = "<%= Locale.GERMANY%>" >

After changing the locale of the current localization context, <BR>
the country of the localization context <BR>
(in the language of that localization context) is: <l10n:country /> <BR>
```

```
</l10n:context>
```

The generated HTML page displays the following:

```
The country of the current localization context is: United States
The country associated with the country code "CH" is: Switzerland

After changing the locale of the current localization context,
the country of the localization context
(in the language of that localization context) is: Suisse

After changing the locale of the current localization context,
the country of the localization context
(in the language of that localization context) is: Deutschland
```

format Tag

The `format` tag can be used to convert currency, number, date, time, or internationalized object values into text according to the formatting conventions associated with the Java class `java.text.MessageFormat` and the current `L10nContext` localization context.

Table 5-4 lists the attributes of the `format` tag.

Table 5-4 Format Tag Attributes

Attribute	Description	Required	Runtime Expression
`currency`	Specifies a `Number`, `double`, or `long` value to be converted into a currency.	No	Yes
`number`	Specifies a `Number`, `double`, or `long` value to be converted into a number.	No	Yes
`date`	Specifies a `Date` value to be converted into a date.	No	Yes
`time`	Specifies a `Date` value to be converted into a time.	No	Yes
`dateTime`	Specifies a `Date` value to be converted using date-time format.	No	Yes
`shortDate`	Specifies a `Date` value to be converted using short date format.	No	Yes
`shortTime`	Specifies a `Date` value to be converted using short time format.	No	Yes
`shortDateTime`	Specifies a `Date` value to be converted using short date-time format.	No	Yes
`mediumDate`	Specifies a `Date` value to be converted using medium date format.	No	Yes
`mediumTime`	Specifies a `Date` value to be converted using medium time format.	No	Yes
`mediumDateTime`	Specifies a `Date` value to be converted using medium date-time format.	No	Yes
`longDate`	Specifies a `Date` value to be converted using long date format.	No	Yes

Table 5-4 Format Tag Attributes (continued)

Attribute	Description	Required	Runtime Expression
`longTime`	Specifies a `Date` value to be converted using long time format.	No	Yes
`longDateTime`	Specifies a `Date` value to be converted using long date-time format.	No	Yes
`fullDate`	Specifies a `Date` value to be converted using full date format.	No	Yes
`fullTime`	Specifies a `Date` value to be converted using full time format.	No	Yes
`fullDateTime`	Specifies a `Date` value to be converted using full date-time format.	No	Yes
`object`	Formats an internationalized object of type `I18nObject`.	No	Yes

The following example uses the `format` tag to format:

- A `double` value into a currency amount
- A `Date` value into a time
- A `Date` value into a long time

```
<%@ taglib uri="http://www.cisco.com/taglibs/localization" prefix="l10n" %>
...
<%
double amount = 99.99;
GregorianCalendar calendar = new GregorianCalendar();
Date curDateTime = calendar.getTime();
%>

The current locale is:  <l10n:locale /> <BR>
Amount formatted as currency:  <l10n:format currency="<%= amount %>" /> <BR>
Date value formatted as time:  <l10n:format time="<%= curDateTime %>" /> <BR>
Date value formatted as long time:  <l10n:format longTime="<%= curDateTime %>" /> <BR>
```

The generated HTML page displayed the following:

```
The current locale is: English (United States)
Amount formatted as currency: $99.99
Date value formatted as time: 7:59:18 PM
Date value formatted as long time: 7:59:18 PM EDT
```

resource Tag

The `resource` tag obtains a resource from the resource bundle of the current localization context (`L10nContext`) and for a specified key. A resource obtained is the value part of the key-value pair in a properties file. Table 5-5 lists the attributes of the `resource` tag.

Table 5-5 Resource Tag Attributes

Attribute	Description	Required	Runtime Expression
key	Specifies the key for which to obtain a resource.	Yes	Yes

The following example uses the resource tag to obtain the resource associated with the key PleaseAuthenticate from the messages.properties resource bundle of the NWSP sample web application.

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

The resource for the key "PleaseAuthenticate" is: <l10n:resource key="PleaseAuthenticate">
Text in the tag body appears in Dreamweaver but not in the generated HTML.
</l10n:resource> <BR>
```

The generated HTML page displays the following:

```
The resource for the key "PleaseAuthenticate" is: Please log in
```

template Tag

The template tag can be used with the resource tag for token replacement in a resource. For a template (a message format) that appears in its tag body, the template tag replaces tokens in the template with the values specified with the param or params attributes. The syntax that you use for a token is specified by the class java.text.MessageFormat. The template tag formats locale-sensitive information such as dates, messages, and numbers using the conventions of the current localization context (L10nContext).

Table 5-6 lists the attributes of the template tag.

Table 5-6 Template Tag Attributes

Attribute	Description	Required	Runtime Expression
param	Specifies a single parameter value to substitute for a token. The param attribute is used when an array of parameter values is not required. The type of the parameter can be Object or any of the primitive types.	No	Yes
params	Specifies an array of type Object[] containing parameter values to substitute for one or more tokens.	No	Yes

In the following example, tokens in a message format in the `template` tag body are replaced by the values specified in `nameAndAge`:

```
<%! Object[] nameAndAge = new Object[] {"John", new Long(5)}; %>
<l10n:template params = "<%= nameAndAge %>" >
My name is {0}, I am {1,number,integer} years old.
</l10n:template >
```

The text in braces (`{ }`) is a token. The `template` tag replaces tokens with the values specified with the `params` attribute. In the preceding example, token `{0}` is replaced by element 0 of the `nameAndAge` array, and token `{1, number, integer}` is replaced by element 1 of the `nameAndAge` array. The generated HTML page displays the following:

```
My name is John, I am 5 years old.
```

The `template` tag can be used with the `resource` tag to get a resource from a resource bundle and to replace tokens in the resource with specified parameter values. In a properties file, the value part of a key-value pair is a template, a message format, that can contain one or more tokens. As an example, assume the following key-value pair in a properties file:

```
message=Error is {0} ({1,number,integer}).
```

The following `template` tag uses the `resource` tag in its body to retrieve the value from and replace tokens in the preceding properties file entry:

```
<%! Object[] errorInfo = new Object[] {"Not Found", new Long(403) };  %>

<l10n:template params = "<%= errorInfo %>" >
<l10n:resource key = "message" />
</l10n:template>
```

For the preceding example, the generated HTML page displays the following:

```
Error is Not Found (403).
```

Setting a Default Localization Context

You can use the initialization parameters for the `L10nContextDecorator` servlet to set the values of the web application default localization context. The `L10nContextDecorator` servlet creates an `L10nContext` object and adds it as an attribute having session scope. The initialization parameters for the servlet are defined in the web.xml file of each SESM web application. The value of a localization context (`L10nContext`) is determined as follows:

1. The values for the localization context come from the preferred locale of the subscriber HTTP client machine if the SESM web application supports the preferred locale. For example, a subscriber can set the preferred locale for a Windows client machine with the Regional Settings tool in Control Panel.
2. If the SESM web application does not support the preferred locale, the values for the localization context come from the web application default localization context, which you can specify with the `L10nContextDecorator` initialization parameters.

Table 5-7 lists the `L10nContextDecorator` initialization parameters that you can specify in the web.xml file for an SESM web application.

Table 5-7 L10nContextDecorator Default Initialization Parameters

Parameter	Description
`defaultResourceBundle`	A base name for a resource bundle properties file. For example: SESMResources.
`defaultLanguage`	A language code (for example, `it`). These codes are the lowercase, two-letter codes defined in ISO-639. You can find a full list of these codes at: http://www.ics.uci.edu/pub/ietf/http/related/iso639.txt
`defaultCountry`	A country code (for example, `IT`). These codes are the uppercase, two-letter codes defined in ISO-3166. You can find a full list of these codes at: http://www.chemie.fu-berlin.de/diverse/doc/ISO_3166.html
`defaultVariant`	A variant code (for example, `EURO`). Variant codes are vendor and browser specific.
`defaultTimeZone`	A time zone ID (for example, `Europe/Stockholm`) or the value `default`. • For information on time-zone IDs, see the description of the `TimeZone` class in the Java 2 platform area of the java.com.sun site. • For information on the value `default`, see the explanation that follows this table.

When you use the value `default` for the `defaultTimeZone` parameter, a time-zone map associates a time zone with a specific locale. The `L10nContext` class uses the time-zone mapping for the web application default localization context. If the subscriber subsequently changes the preferred locale, the `L10nContext` software selects a new time zone that matches the new locale.

Two `L10nContextDecorator` servlet initialization parameters are used to define a time-zone map.

- `timeZoneMapCountries` is a set of country codes that defines the countries that are mapped to a time zone.
- `timeZoneMapTimeZones` is a set of time zone IDs that defines the time zones that are associated with the countries given in the `timeZoneMapCountries` parameter.

For both of these parameters, you can separate the values by whitespace or commas.

The following example shows how to use the `timeZoneMapCountries` and `timeZoneMapTimeZones` parameters.

```
<servlet>
    <servlet-name>L10nContext</servlet-name>
    <servlet-class>com.cisco.sesm.navigator.L10nContextDecorator</servlet-class>
    <load-on-startup>1</load-on-startup>

    <!-- Specify associations from Country -> TimeZone. -->
    <init-param>
    <param-name>timeZoneMapCountries</param-name>
    <param-value>AU,NZ,SE,PT</param-value>
    </init-param>
    <init-param>
    <param-name>timeZoneMapTimeZones</param-name>
    <param-value>
    Australia/Sydney
    Pacific/Auckland
    Europe/Stockholm
    Europe/Lisbon
    </param-value>
    </init-param>
```

```
<!-- The value "default" indicates that a time-zone map is used -->
    <init-param>
    <param-name>defaultTimeZone</param-name>
    param-value>default</param-value>
    </init-param>
  </servlet>
```

Given the preceding time-zone map, if the locale of the current localization context (`L10nContext`) were for Australia (country code `AU`), `L10nContextDecorator` uses the corresponding time-zone ID (`Australia/Sydney`) to determine a time zone. For each country specified in `timeZoneMapCountries`, there should be a corresponding time-zone ID given in the `timeZoneMapTimeZones` parameter.

For the default localization context values used for a sample SESM web application like NWSP, see the web.xml for the application.