BEA WebLogic Portal®

Customizing the Administration Console

Version 10.2
Revised: February 2008

Contents

Introduction to the Portal Administration Console

The WebLogic Portal Administration Console is the tool that portal administrators use to control the behavior, content, and appearance of portals. Moreover, it is used to perform many traditional system administration activities such as user management and security management.

The WebLogic Portal Administration Console is organized according to the following categories of tasks:

- **Portal Management** – Portals, desktops, books, pages, portlets, and other portal resources.
- **User, Groups, & Roles** – User and group management, security provider configuration, Delegated Administration, and Visitor Entitlements.
- **Configuration Settings** – Server settings for Cache Management, Server Maintenance Mode, Personalization, Security, Unified User Profiles, and WSRP.
- **Interaction Management** – Campaigns, placeholders, user segments, and content selectors.
- **Content Management** – Content and repositories.

The Administration Console is a portal application and is built with the portal framework. Using this framework, you can customize the Administration Console by changing the look-and-feel of your Administration Console, adding new portlets (extending) to the Administration Console, or re-using Administration Console portlets within a separate portal application.

This chapter includes the following sections:

- Document Scope and Audience
- Getting Started
- In This Guide

Document Scope and Audience

This document is a resource for software developers who want to customize how the Portal Administration Console is used and looks. It is assumed that the reader is already familiar with using WebLogic Portal to develop portal applications.

Getting Started

Before you begin working with the Portal Administration Console, you should be familiar with working with WebLogic Portal applications. WebLogic Portal provides specific documentation on developing portals, portlets, and deploying portal applications. It is recommended that you become familiar with the following guides and their tasks before customizing your Administration Console.

- *Portal Development Guide*
- *Portlet Development Guide*
- *Production Operations Guide*

In This Guide

This guide includes the following chapters:

- Chapter 2, "Re-using Administration Console Portlets in Your Portal"
- Chapter 3, "Extending the Portal Administration Console"
- Appendix A, "List of Resource IDs"

Re-using Administration Console Portlets in Your Portal

You can re-use portlets from the Administration Console within your portal. Doing this allows you to access portal resources directly from your portal, such as content, user and group information, or campaign settings.

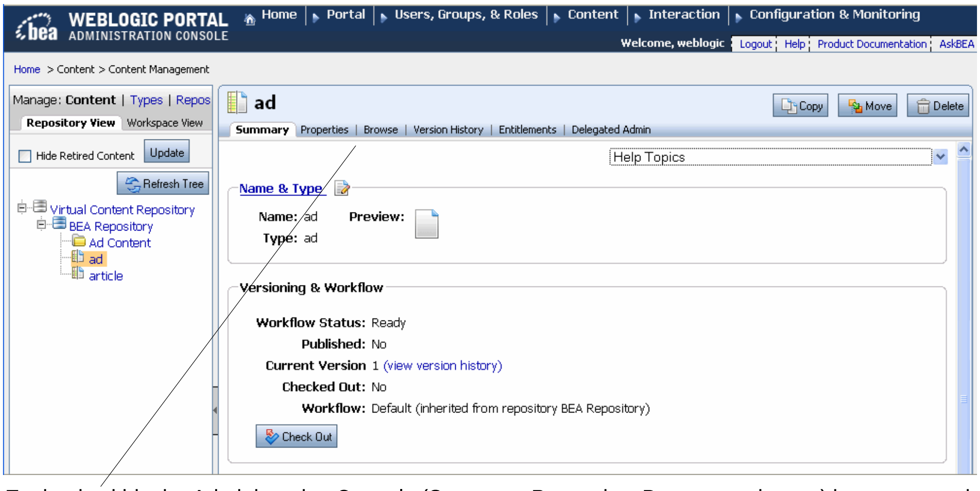
This chapter includes the following topics:

- Overview of the Administration Console Portlets
- Adding Administration Console Portlets to Your Portal Application
- Ensuring Resources Appear in Reused Administration Console Portlets

Overview of the Administration Console Portlets

The Administration Console is comprised of many portlets. Each portlet can access its resources in different ways. Each tab within the Administration Console is a page containing a portlet, see Figure 2-1 for an illustration.

Figure 2-1 Administration Console Tabs


WEBLOGIC PORTAL
ADMINISTRATION CONSOLE
Home | Portal | Users, Groups, & Roles | Content | Interaction | Configuration & Monitoring
Welcome, weblogic | Logout | Help | Product Documentation | AskBEA
Home > Content > Content Management
Manage: Content | Types | Repos
Repository View | Workspace View
Hide Retired Content | Update
Refresh Tree
Virtual Content Repository
BEA Repository
Ad Content
ad
article
ad
Copy | Move | Delete
Summary | Properties | Browse | Version History | Entitlements | Delegated Admin
Help Topics
Name & Type
Versioning & Workflow
Check Out

Name: ad
Type: ad
Preview:

Workflow Status: Ready
Published: No
Current Version 1 (view version history)
Checked Out: No
Workflow: Default (inherited from repository BEA Repository)

Each tab within the Administration Console (Summary, Properties, Browse, and so on) is represented by a page containing a portlet.

To access portal resources without accessing the Administration Console, you can re-use these portlets in your portal application. Portlets associated with the following portal resources are available to add to your portal:

- Interaction Management (campaigns, content selectors, placeholders, and user segments)
- Content Management
- User and Group Management

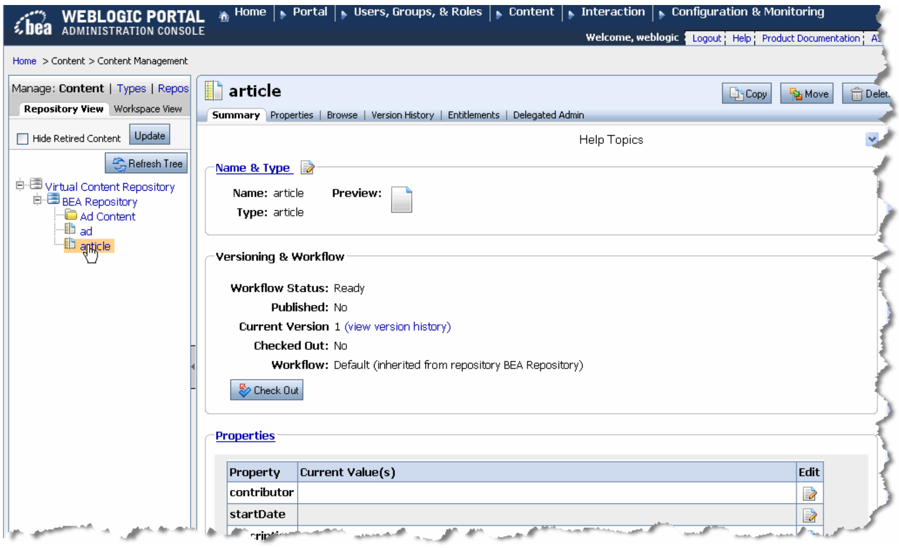
Note: Portal Management, Delegated Administration, Entitlements, and Service Administration portlets cannot be re-used in portal applications.

Portlets within the Administration Console react to tree events. For example, when you use the Content Resource tree to select content, the content Summary tab (the content summary portlet) is activated, see Figure 2-2.

However, within your portal, resource IDs are used in inter-portlet communication events to activate Administration Console portlets. If another portlet refers to the resource ID that is used by a Administration Console portlet, that portlet is activated.

Note: Administration Console tree portlets, such as the Portal Resource tree and the Content Resource tree, cannot be re-used in your portal application. If you want to use the entire Administration Console functionality such as the Content Management Resource tree as well as its portlets, BEA recommends using the Administration Console rather than adding an Administration Console portlet to your portal.

Figure 2-2 Selecting Article in the Repository Tree Calls the Summary Portlet


WEBLOGIC PORTAL
ADMINISTRATION CONSOLE
Home | Portal | Users, Groups, & Roles | Content | Interaction | Configuration & Monitoring
Welcome, weblogic | Logout | Help | Product Documentation
Home > Content > Content Management
Manage: Content | Types | Repos
Repository View | Workspace View
Hide Retired Content | Update
Refresh Tree
Virtual Content Repository
BEA Repository
Ad Content
ad
article
Copy | Move | Delet
Summary | Properties | Browse | Version History | Entitlements | Delegated Admin
Help Topics
Check Out

article

Name & Type

Name: article
Type: article
Preview:

Versioning & Workflow

Workflow Status: Ready
Published: No
Current Version 1 (view version history)
Checked Out: No
Workflow: Default (inherited from repository BEA Repository)

Properties

Property	Current Value(s)	Edit
contributor		
startDate		

WebLogic Portal provides specific APIs for working with Administration Console portlets that provide an interface-based framework that integrates with inter-portlet communication events. Instead of using events and event handlers explicitly, Administration Console portlets generate and receive events using resource IDS and the interface-based framework.

For example, you can use a combination of the Content Presenter portlet and the Edit Content portlet from the Administration Console to allow users to modify content on the fly. To do so, within the Content Presenter portlet, you add an Edit Content button, which activates the edit content portlet.

Adding Administration Console Portlets to Your Portal Application

Before you can add any administration console portlets to your portal, you must first add the respective J2EE libraries to your web project. These library modules provide the common framework and the respective portlets for this purpose.

After ensuring that your web project includes all the library modules you need, you can add administration portlets to your portal.

To be activated, Administration Console portlets require communication from another portlet. Subsequently, you need to build a portlet that activates or calls an Administration Console portlet. Activating an Administration Console portlet is done by using resource IDs. You need to include code in the primary portlet to generate an event using a resource ID that the Administration Console portlet recognizes.

Adding Administration Console Resources to Your Web Project

Administration Console portlets are included in respective library modules that you must add to your web project, as shown in Table 2-1. Copy the J2EE modules that you need. The wlp-tools-common-web-lib and wlp-tools-framework-web-lib libraries are required for all portlets.

To add administration console J2EE libraries:

1. Right-click your web project folder and select **Build Path** > **Add Libraries**.
2. In the Add Library dialog, select **WebLogic J2EE library**. Click **Next**.
3. Click **Add..**. and browse to the respective library module and click **OK**.
4. Click **Finish**.
5. Repeat for each of the modules listed in Table 2-1. The added portlets appear in the list of available portlets in the Design Palette.

Table 2-1 J2EE Library Modules for Administration Console Portlets

Library Module	Portlets	Description
wlp-tools-common-web-lib	All	Contains the common framework needed for all administration console portlets.
wlp-tools-framework-web-lib	All	Contains the common framework needed for all administration console portlets.
wlp-tools-ugm-web-lib	User Management Group Management	Contains the portlets for user and group management
wlp-tools-content-web-lib	Content Management	Contains all content management portlets.
wlp-tools-im-web-lib	Campaigns Content Selectors Placeholders User Segments	Contains all interaction management portlets.

Using Portlet Resource IDs to Generate Events

Administration Console portlets are designed to react to inter-portlet communication events that are generated by resource IDs. When a portal resource, such as content or a user name, is entered or selected in a portlet, the respective Administration Console portlet is activated.

For example, you can allow portal users to edit the content displayed in a portlet by adding code in that portlet that activates the Edit Content Properties portlet, as shown in Listing 2-1. The following WebLogic APIs are used to interact with or activate Administration Console portlets:

- `ResourceID`
- `PortletORB`

Listing 2-1 Code Snippet Showing How to Trigger a Resource ID Event

```
// Create a resource ID (see Javadoc)
final ResourceID resourceID = … ;

// Get a TreeSelectListener
final HttpServletRequest request = …;
final HttpServletResponse response = …;
final PortletORB portletORB = PortletORB.getPortletORB(request, response);
final PortletHandle portletHandle = portletORB.getBroadcastPortletHandle();
final TreeSelectListener listener = portletHandle.
  narrow(TreeSelectListener.class);

// Call selected(…)
final ResourcePath resourcePath = new ResourcePath(new ResourceID[] { resourceID
});
final ResourceType resourceType = resourceID.getResourceType();
listener.selected(resourcePath, resourceType, resourceID);
```

Using Online Help in Reused Portlets

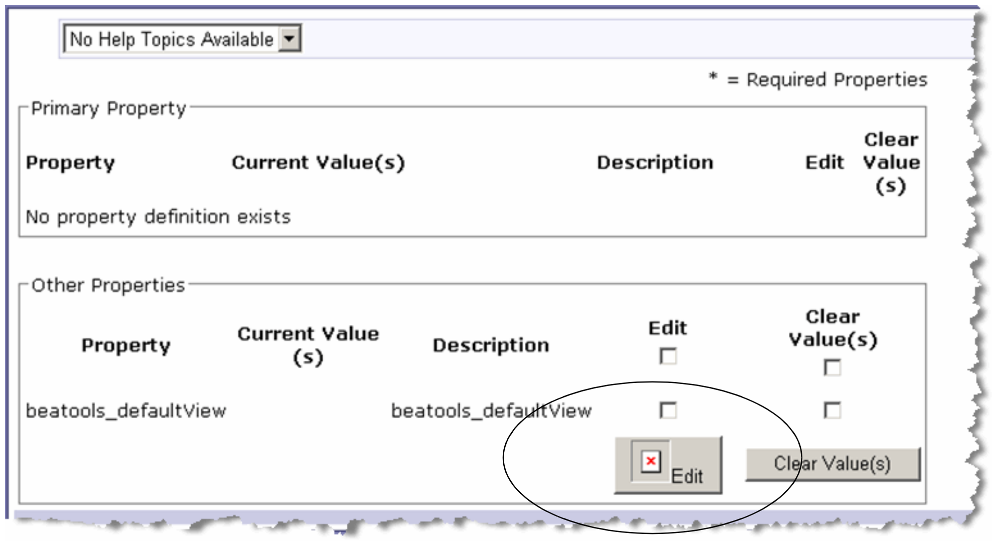
If you are reusing WebLogic Portal Administration Console portlets in a portal, some features and steps that are referred to in the administration console online help system might not apply in the context in which the portlets are reused.

For example, in the context of the administration console, content management portlets react to tree events (events generated by clicking in the content resources tree to select content). In a reuse context, the portlets are configured (by a portlet developer) to react to events fired by other portlets. The tree view will not be present and therefore, some steps described the help system will not apply.

Ensuring Resources Appear in Reused Administration Console Portlets

It is possible that some resources, such as certain images, CSS files, and scripts, will not appear in a reused Administration Console portlet. This is because the portlet references resources that are only specified in the skin file that is used by the Portal Administration Console. If your portal uses a different skin file, these resources will not show up, as shown in Figure 2-3.

Figure 2-3 Missing Graphic in Reused Administration Console Portlet



* = Required Properties

Primary Property

Property	Current Value(s)	Description	Edit	Clear Value (s)
No property definition exists				

Other Properties

Property	Current Value (s)	Description	Edit ☐	Clear Value(s) ☐
beatools_defaultView		beatools_defaultView	☐	☐

The simplest way to fix this problem is to add the appropriate references to the `skin.xml` file used by your portal’s look and feel. To edit the `skin.xml` file, you must first copy it to your project in WorkSpace Studio (right-click the file and select Copy to Project). For more information on skins, see Working with Skins in the *Portal Development Guide*.

Listing 2-2 highlights the kinds of changes you might need to make to your skin file to ensure that Administration Console specific resources show up in your reused portlet. These resources are indicated in Listing 2-2 by the comment:

```
<!--  from the wlp-tools skin -->
```

Note: If your portal is using the Bighorn skin, you also need to make similar changes to the Internet Explorer specific skin file, `.../skins/bighorn/msie/skin.xml`. This ensures that the skin will function properly in Internet Explorer browsers.

Listing 2-2 Adding Administration Console Specific Resources to a Skin File

```
<skin
    xmlns="http://www.bea.com/servers/portal/framework/laf/1.0.0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.bea.com/servers/portal/framework/laf/1.0.0
laf-skin-1_0_0.xsd"
>
    <target-skeleton>
        <skeleton-id>bighorn</skeleton-id>
    </target-skeleton>
    <images>
        <search-path>
            <path-element>images</path-element>

            <!--  from the wlp-tools skin -->
          <path-element>../wlp-tools/images</path-element>

        </search-path>
    </images>
    <render-dependencies>
        <html>
            <links>
                <search-path>
                    <!-- from wlp-tools skin -->
                    <path-element>../wlp-tools/custom/css</path-element>
                    <path-element>../wlp-tools/css</path-element>


                    <path-element>css</path-element>
                </search-path>
                <link href="colors.css" rel="stylesheet" type="text/css" />
                <link href="general.css" rel="stylesheet" type="text/css" />
                <link href="menu.css" rel="stylesheet" type="text/css" />
                <link href="window.css" rel="stylesheet" type="text/css" />
                <link href="wsrp.css" rel="stylesheet" type="text/css" />
                <link href="custom.css" rel="stylesheet" type="text/css" />

                <!--  from wlp-tools skin -->
                <link href="wlp.css" type="text/css" rel="stylesheet"/>
    <link href="wlp-custom.css" type="text/css" rel="stylesheet"/>
</links>

<!--  from wlp-tools skin -->
<scripts>
    <search-path>
        <path-element>../wlp-tools/js</path-element>
        <path-element>../../../js</path-element>
    </search-path>
    <script type="text/javascript"
        src="com/bea/portal/tools/js/Core.js"/>
    <script type="text/javascript"
        src="com/bea/portal/tools/js/Core.jsp"/>
    <script type="text/javascript"
        src="com/bea/portal/tools/js/javascriptConstants.jsp"/>
    <script type="text/javascript"
      src="com/bea/portal/tools/js/Math.js"/>
    <script type="text/javascript"
      src="com/bea/portal/tools/js/skin/FeatureProviderManager.js"/>
    <script type="text/javascript"
      src="com/bea/portal/tools/js/skin/ProgressManager.js"/>
    <script type="text/javascript"
      src="com/bea/portal/tools/js/html/Form.js"/>
    <script type="text/javascript"
      src="com/bea/portal/tools/js/html/Event.js"/>
    <script type="text/javascript" src="progress-manager.js"/>
    <script type="text/javascript" src="progress-manager.jsp"/>
    <script type="text/javascript" src="main-menu.js"/>
    <script type="text/javascript" src="resize-collapse.js"/>
    <script type="text/javascript" src="skin.js"/>
</scripts>

<!--  from wlp-tools skin -->

<styles>
    <search-path>
        <path-element>../wlp-tools/custom/css</path-element>
        <path-element>../wlp-tools/css</path-element>
    </search-path>
    <style content-uri="wlp-ajax-image.css" type="text/css" />
</styles>
```

```
        </html>
    </render-dependencies>

    <!--  from wlp-tools skin -->
    <script-dependencies>
        <script-fragments>
            <fragment
           event-handler="onload">__BEA.framework.skins['wlp-tools'].js.onload()
           </fragment>
        </script-fragments>
    </script-dependencies>
</skin>
```

Extending the Portal Administration Console

Administration Console extensions are web applications that you integrate with the existing Administration Console. You can use an extension to do something as simple as adding a portlet to a page within the console, or as complex as adding an entire editor module that includes multiple books and pages.

To create an extension, you re-create the Administration Console web application within your EAR and import the associated libraries. You then can add or remove functionality to the web application.

After creating your custom Administration Console, including any extensions you added, you package this new custom Administration Console as a J2EE library and add that library to your existing web application. The new library will replace the Administration Console that was previously associated with your portal.

The section includes the following topics:

- What Is an Administration Console Extension?
- Setting up Your Development Environment
- Building Your Extension Module
- Using Online Help in an Extended Console
- Using your Extended Administration Console

What Is an Administration Console Extension?

You can extend the Administration Console by adding books, pages, or portlets to an existing book or page within the Administration Console. For example, you can add a login portlet to the Administration Console that allows users to login to another administrative application, such as a Network Operations Center.

When you extend the Portal Administration Console, you need to first assemble the application files that are needed in the extension (new books, pages, JSPs, portlets, and so on), and then configure your extension so that the Portal Administration Console can locate and communicate with its resources.

The simplest extension adds content to the Administration Console's home page (desktop). The JAR file for such an extension contains:

- A NetUI Extension XML file that describes the location in the UI in which you want your extension to display.
- An editors XML file that provides the metadata used to include your extension in the Administration Console menu.
- A book or page that contains the content you want to display (and all of its contents).

Setting up Your Development Environment

Setting up your development environment involves creating two additional web projects within your EAR. One project will be used to re-create the Administration Console with all of its libraries. This project will become your new extended Administration Console. You use an additional web project to create your extension.

Creating an Administration Console Web Project

To create your custom Administration Console web project:

1. Create a web project:
   a. Right-clicking your EAR project.
   b. Choose **New > Portal Web Project**.
2. In the New Portal Web Project dialog, associate your web project with the same EAR that your portal web application is associated with.

3. In the Project Facets dialog, de-select the Portal Visitor Tools facet.
4. Click **Finish**.

Add the Administration Console J2EE Libraries to your Extension Project

The following J2EE modules are required for the Administration Console. Even though you may be customizing only a page of the console, import *all* of these modules to your web project. Remember you are re-creating the entire Administration Console, which ensures that your deployment remains consistent.

WARNING: J2EE libraries not imported to your web project can be overwritten when you upgrade WebLogic Portal. In most cases, this is desirable. However, upgraded libraries will delete your Administration Console customizations and extensions.

To copy a J2EE module to your web project:

1. In the Project Explorer, right-click the **//*`web_project_name`*/WebLogic Deployment Descriptor > J2EE Libraries** folder and then select **Add Multiple**.
2. In the Select WebLogic J2EE library, select the modules you need. Figure 3-1 lists the required libraries.

Table 3-1 Required Administration Console J2EE Libraries

Required Administration Console J2EE Libraries
wlp-tools-full-console-web-lib
wlp-tools-common-web-lib
wlp-tools-content-web-lib
wlp-tools-framework-web-lib
wlp-tools-im-web-lib
wlp-tools-portal-web-lib
wlp-tools-serviceadmin-web-lib
wlp-tools-ugm-web-lib

3. Click **OK**.

Creating your Extension Web Project

Create a web project where you will build your extension. Your extension can include books, pages, portlets, or an entire portal.

To create your extension web project:

1. Create a web project:
   a. Right-click your EAR project.
   b. Choose **New > Portal Web Project**.
2. In the New Portal Web Project dialog, associate your web project with the same EAR that your portal web application is associated with.
3. In the Project Facets dialog, de-select the Portal Visitor Tools facet.
4. Click **Finish**.
5. Create the books, pages, and other resources that you want to include in your extension. For more information about building portal resources, see the *Portal Development Guide*.

Building Your Extension Module

When you are finished creating the resources for your extension, you are ready to create the configuration files necessary to integrate your extension with the Administration Console project you created. You need to create two configuration files to include with your project, then package the project and add it to your Administration Console web project.

Creating Configuration Files for Your Extension

When you create an extension, you are creating a new web application that can be used in your Administration Console via a `netuix-extension.xml` file that is contained within the extension application itself. If you want your extension to be accessible via the main menu of the Administration Console, you also need to create an `editors.xml` file.

- The `netuix-extension.xml` file defines target locations within the existing Administration Console (via definition labels) where these new portlets, books and pages should be added.
- The `editors.xml` file defines which Administration Console menu to use (Portal Management, Content Management, and so on) to access your extension.

Note: The `beehive-netui-config.xml` file in the WEB-INF/ directory specifies netui-specific tags and handler classes. Add the following elements to the `beehive-netui-config.xml` file between the `<pageflow-config>` and the `<request-interceptors>` elements:

```
<jsp-tag-config>
       <id-javascript>legacy</id-javascript>
       <tree-renderer-class>com.bea.jsptools.patterns.tree.
       ToolsTreeRenderer</tree-renderer-class>
</jsp-tag-config>
```

Determining Where to Place Your Extension

The Administration Console is built from several library modules that represent the features supported within the console. The main portal of the Administration Console resides in the wlp-tools-common-web-lib. This main `portal.portal` file is extended by each subsequent library module. To extend the portal, each library uses an `netuix-extension.xml` that references its resources and indicates where it will extend the `portal.portal`.

You can also extend the Administration Console at the book or page level. To determine which book to extend, you use the `netuix-extension.xml` files for each library module to familiarize yourself with how the Administration Console is structured.

After determining the Administration Console book or page you want to extend, you refer to the definition labels of the respective module’s books or pages in your own extension module’s `netuix-extension.xml` to indicate where your extension should appear within the Administration Console.

To view these definition labels, you open the respective WAR files that contain the associated books for your extension point. Keep in mind that you do *not* modify the existing Administration Console files or modules in any way. Instead, you create an extension that points to the existing files. Table 3-2 lists the books that contain definition labels of commonly used extension points.

Table 3-2 Common Extensions for the Administration Console

Book	Extension Point	netuix-extension.xml Location
/im/interactionToolsMainBook.book	The main book for Interaction Management. Look in this book's XML file for the definition labels of subsequent books that you want to extend.	`//`*`WebLogic_Home`*`/portal/lib/modules/wlp-im-web-lib.war`
/ugm/TopLevelEditor.book	The main book for User and Group Management. Look in this book's XML file for the definition labels of subsequent books that you want to extend.	`//`*`WebLogic_Home`*`/portal/lib/modules/wlp-ugm-web-lib.war`
/portalTools/portalTopLevelEditorBook.book	The main book for Portal Management. Look in this book's XML file for the definition labels of subsequent books that you want to extend.	`//`*`WebLogic_Home`*`/portal/lib/modules/wlp-pm-web-lib.war`
/serverTools/serverToolsTopLevelEditorBook.book	The main book for Service Administration. Look in this book's XML file for the definition labels of subsequent books that you want to extend.	`//`*`WebLogic_Home`*`/portal/lib/modules/wlp-serviceadmin-web-lib.war`

Table 3-2 Common Extensions for the Administration Console (Continued)

Book	Extension Point	netuix-extension.xml Location
/content/ contentTopLevelEditorBook_twoPageTree.book	The main book for Content Management when using library services. Look in this book's XML file for the definition labels of subsequent books that you want to extend.	`//`*`WebLogic_Home`*`/ portal/lib/modules/ wlp-cm-web-lib.war`
/content/ contentTopLevelEditorBook_onePageTree.book	The main book for content management if not using library services. Look in this book's XML file for the definition labels of subsequent books that you want to extend.	`//`*`WebLogic_Home`*`/ portal/lib/modules/ wlp-cm-web-lib.war`

Creating a netuix-extension.xml File

The easiest way to create a new `netuix-extension.xml` file to include in your extension application is to copy an existing file from one of the WAR files included in a library module. The `netuix-extension.xml` file must reside in the `//WEB-INF/` directory of your project.

Before beginning these steps, copy the text from an existing `netuix-extension.xml` to use in your new file. Listing 3-1 provides an example file.

Listing 3-1 Example of a netuix-extension File that Extends the Main Book of the Administration Console

```
<?xml version="1.0" encoding="UTF-8" ?>

<weblogic-portal-extension
xmlns="http://www.bea.com/servers/portal/weblogic-portal/8.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.bea.com/servers/portal/weblogic-portal/8.0
netuix-extension-1_0_0.xsd">

< !-- The provider-info element allows you to enter a description for your
extension -->

<provider-info>

       <title>BEA WebLogic Portal Administration Console - Content Management
Extension</title>

       <description>Content Management books that are inserted into theBEA
WebLogic Portal Administration Console</description>

       <author>BEA Systems, Inc.</author>

</provider-info>

<! -- The portal-file element defines which portal file you are extending. To
extend the Administration Console, the value of this element should be
portal.portal -->

<portal-file>/portal.portal</portal-file>

<! -- Use the book-extension element and its children to define which book in
the Adminstration Console you are extending. The <book-content content-uri>
element is wehre you put that path to your extension. In the following example,
content is the name of the web extension application that contains the
contentTopLevelEditorBook-twoPageTree.book -->

<book-extension>
```

```
      <book-location>

             <parent-label-location label="mainToolsBook" />

             <book-insertion-point page-label="homeBook" action="append" />

      </book-location>

      <book-content
content-uri="/content/contentTopLevelEditorBook_twoPageTree.book" />

</book-extension>

< ! -- This example file includes two extensions. Both of which extend the home
book of the mainToolsBook of the Administration Console -->

<book-extension>

      <book-location>

             <parent-label-location label="mainToolsBook" />

             <book-insertion-point page-label="homeBook" action="append" />

      </book-location>

      <book-content
content-uri="/content/contentTopLevelEditorBook_onePageTree.book" />

</book-extension>

</weblogic-portal-extension>
```

To create a `netuix-extension.xml` file within your web project:

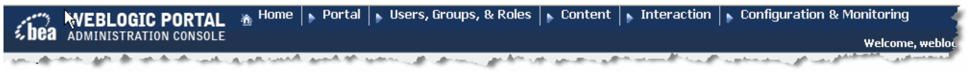
1. Right-click the //WEB-INF folder within your extension web project and select **New** > **Other**.
2. In the Select a Wizard dialog, select **XML** > **XML** and click **Next**.
3. In the Create an XML File dialog, mark **Create XML File from Scratch** and click **Next**.
4. In the XML File Name dialog, type `netuix-extension` as the name of your file and click **Finish**.
5. In the Package Explorer, right-click the `netxuix-extension.xml` file and select **Open With** > **Text Editor**.
6. Paste the text you have copied from a valid `netuix-extension.xml` file.

7. Make modifications to the text that match the extension module you have created and save the file.

Creating an editors.xml File

If you want to include your extension in the header menu of the Administration Console (shown in Listing 3-1), you need to include an `editors.xml` file in your extension web project.

Figure 3-1 The Administration Console Header Menu



The `editors.xml` file allows you to extend an existing menu entry or add a new menu entry.

The easiest way to create a new `editors.xml` file to include in your extension application is to copy an existing file from one of the WAR files included in a library module. The `editors.xml` file must reside in the //WEB-INF/ directory of your project.

Note: For WebLogic Portal 10.x, if you want to add an editor to an existing menu, you need to add a properties file to your system path. For more information, see the *WebLogic Portal Release Notes*.

Before beginning these steps, copy the text from an existing `editors.xml` to use in your new file. Listing 3-1 provides an example file.

To create an `editors.xml` file:

1. Right-click the //WEB-INF folder within your extension web project and select **New** > **Other**.
2. In the Select a Wizard dialog, select **XML** > **XML** and click **Next**.
3. In the Create an XML File dialog, mark **Create XML File from Scratch** and click **Next**.
4. In the XML File Name dialog, type `editors` as the name of your file and click **Finish**.
5. In the Package Explorer, right-click the `editors.xml` file and select **Open With** > **Text Editor**.
6. Paste the text you have copied from a valid `editors.xml` file.
7. Make modifications to the text that match the extension module you have created and save the file. Listing 3-2 shows an example of a valid `editors.xml` file.

Listing 3-2 Example of an editors.xml File

```
<?xml version="1.0" encoding="UTF-8"?>

<!-- @name: the name of the top-level menu item.  This is used as the key for
getting the title from the properties file specified in the bundleName attribute.
-->

<!-- @defLabel: the definition label of the default book to show for this editor
group.  This is not currently used by the menu. -->

<!-- @defaultEditor: the name attribute of the default editor for this group -->

<!-- @bundleName: the name of the applications scoped (EAR) properties file from
which to retrieve the title for @name -->

<edtrs:editor-group

    xmlns:edtrs="http://com.bea.portal/tools/admin/xsd/editors-1_0_0"

    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

    xsi:schemaLocation="http://com.bea.portal/tools/admin/xsd/editors-1_0_0
editors-1_0_0.xsd"

    name="myPAT"

    defLabel="NOT_USED"

    defaultEditor="myPATeditor"

    bundleName="myPATeditor.i18n.myPATeditor" >

<!-- @name: the name of this editor which is used as a second level menu item.
This is used as the key for getting the title from the properties file specified
in the bundleName attribute. -->

<!-- @landingDefLabel: the definition label of the landing page for this editor;
IT MUST END WITH _landingPage -->

<!-- @headerIcon: the icon to show for this editor -->

<!-- @bundleName: the name of the applications scoped (EAR) properties file from
which to retrieve the title for @name -->

       <edtrs:editor name="myPATeditor"

              landingDefLabel="myPATeditor_landingPage"

              headerIcon="wlp-content-mgmt-24.gif"

              bundleName="myPATeditor.i18n.myPATeditor" >
```

```
<!--  tree builders are only used in the Administration Console editors. However,
this entry is required.-->
              <edtrs:tree-builder>
                     <edtrs:mode>NOT_USED</edtrs:mode>
              </edtrs:tree-builder>
       </edtrs:editor>
</edtrs:editor-group>
```

Adding Your Extension to Your Administration Console

When you have finished building your extension resources and added the necessary configuration files (a `netuix-extension.xml` file and an optional `editors.xml` file), you need to export your extension to your custom Administration Console application.

Note: Make sure that you have added your extension to the custom Administration Console project you created, see "Creating an Administration Console Web Project" on page 3-2.

To add your extension to your Administration Console application:

1. In your Administration Console web project, add a folder called **bea-ext** under the WEB-INF directory. You will export your extension project to this directory.
2. In your extension web project, right-click the root directory of your extension web project and select **Export** > **WAR**.
3. In the Export WAR dialog, click **Browse** to navigate to the `\\WEB-INF\bea-ext` directory in your Administration Console web project. For example, `c:\bea\user_projects\workspaces\`*`myWorkspace`*`\`*`adminConsoleWebProject`*`\webContent\web-inf\bea-ext\`
4. Click **Finish**.
5. Using the Project Explorer, navigate to the WAR file you just exported to your custom Administration Console. You may need to press F5 to refresh the directory's contents.
6. Right-click on the *.war file (for example, `myExtension.war`) and select **Copy**.
7. Right-click the //WEB-INF/bea-ext directory and select **Paste**.

8. In the Naming Conflict dialog, rename your WAR file to the same name yet with a *.jar file extension. For example, `myExtension.jar`.
9. Delete the *.war file.

Using Online Help in an Extended Console

In an extended administration console, some features and steps that are referred to in the administration console online help system might not apply. This situation can occur, for instance, if you have removed features or parts of the administration console. Users of the extended administration console need to be aware of this possibility.

Using your Extended Administration Console

As with the WebLogic Portal Administration Console, you access the extended Administration Console with a URL. For example,

```
http://localhost:7001/myCustomConsoleWebProject/portal.portal
```

List of Resource IDs

Administration Console portlets that you add to your portal are designed to react to resource ID calls.

The following sections list the respective resource IDs for the Administration Console portlets:

- Interaction Management Portlet Resource IDs
- Content Management Portlet Resource IDs
- User and Group Management Portlet Resource IDs

Interaction Management Portlet Resource IDs

Interaction management includes user and group management, content selectors and campaigns. The interaction management portlets can be found in the wlp-tools-im-web-lib.

The following tables list each portlet associated with interaction management according to function (user management, campaigns, and so on). Also listed in each table are the resource IDs needed to call each portlet.

- Table A-1, “Campaign Portlets wlp-tools-im-web-lib”
- Table A-2, “Placeholder Portlets in wlp-tools-im-web-lib”
- Table A-3, “Content Selector Portlets in wlp-tools-im-web-lib”
- Table A-4, “User Segments Portlets in wlp-tools-im-web-lib”

Table A-1 Campaign Portlets wlp-tools-im-web-lib

Portlet	Requires Additional portlet(s)?	Library Module Path	Resource ID
Browse Campaigns		./interactionMgmtModule/web/im/campaigns/browse/CampaignBrowse.portlet	::interaction:campaigns
Campaign Details		./interactionMgmtModule/web/im/campaigns/details/CampaignDetails.portlet	::interaction:campaign
Browse Scenarios		./interactionMgmtModule/web/im/campaigns/scenarios/browse/ScenarioBrowse.portlet	::interaction:scenario
Scenario Details		./interactionMgmtModule/web/im/campaigns/scenarios/details/ScenarioDetails.portlet	::interaction:scenario
Browse Rules		./interactionMgmtModule/web/im/campaigns/scenarios/rules/browse/RuleBrowse.portlet	::interaction:campaign:scenario
Rule Details		./interactionMgmtModule/web/im/campaigns/scenarios/rules/details/RuleDetails.portlet	::interaction:campaign:scenario:rule

Table A-2 Placeholder Portlets in wlp-tools-im-web-lib

Portlet	Requires Additional Portlet(s)?	Library Module Path	Resource ID
Browse Placeholder		./interactionMgmtModule/web/im/placeholders/browse/PlaceholderBrowse.portlet	::interaction:placeholders
Placeholder Details		/project/tools/modules/interactionMgmtModule/web/im/placeholders/details/PlaceholderDetails.portlet	::interaction:placeholder
Search Placeholders		./interactionMgmtModule/web/im/placeholders/queries/browse/QueryBrowse.portlet	::interaction:placeholder

Table A-2 Placeholder Portlets in wlp-tools-im-web-lib (Continued)

Portlet	Requires Additional Portlet(s)?	Library Module Path	Resource ID
Search Delegated Administration Roles for Placeholders		./interactionMgmtModule/web/im/placeholders/queries/details/da/QueryDARoles.portlet	::interaction:placeholder:query:definition
Search Placeholder Details		./interactionMgmtModule/web/im/placeholders/queries/details/QueryDetails.portlet	::interaction:placeholder:query:definition

I

Table A-3 Content Selector Portlets in wlp-tools-im-web-lib

Portlet	Requires Additional Portlet(s)?	Library Module Path	Resource ID
Browse Content Selectors		./interactionMgmtModule/web/im/contentselectors/browse/ContentSelectorBrowse.portlet	::interaction:content-selectors
Content Selectors Details		./interactionMgmtModule/web/im/contentselectors/details/ContentSelectorDetails.portlet	::interaction:content-selector

Table A-4 User Segments Portlets in wlp-tools-im-web-lib

Portlet	Requires Additional Portlet(s)?	Library Module Path	Resource ID
Browse Segments		./interactionMgmtModule/web/im/segments/browse/SegmentBrowse.portlet	::interaction:segment
Segment Details		./interactionMgmtModule/web/im/segments/details/SegmentDetails.portlet	::interaction:segment

Content Management Portlet Resource IDs

Content management portlets include the capability to view and modify content, and content types. If you are using library services, you can also view versioning information.

The following tables include information about the content management portlets, including the resource IDs used for each.

- Table A-5, "Content Portets in wlp-tools-content-web-lib"
- Table A-6, "Repository Portlets in wlp-tools-content-web-lib"
- Table A-7, "Content Type Portlets"

Table A-5 Content Portets in wlp-tools-content-web-lib

Portlet	Requires Additional Portlet(s)?	Library Module Path	Resource ID
Content Browse		./contentMgmtModule/web/content/node/nodeSelected/browse/BrowseNodes.portlet	::content:node
Content Properties		./contentMgmtModule/web/content/node/nodeSelected/properties/NodeProperties.portlet	::content:node
Content Summary		./contentMgmtModule/web/content/node/nodeSelected/summary/NodeSummary.portlet	::content:node
Update Link Property Dialog		./contentMgmtModule/web/content/node/nodeSelected/updateLinkDialog/updateLinkDialog.portlet	::content:node
Content Version History		./contentMgmtModule/web/content/node/nodeSelected/versions/VersionHistory.portlet	::content:node
View Version		./contentMgmtModule/web/content/node/nodeSelected/versions/viewVersion/ViewVersion.portlet	::content:node:version
Content Editor		./contentMgmtModule/web/content/node/nodeSelected/wysiwyg/wysiwyg.portlet	::content:node
Browse Repositories		./contentMgmtModule/web/content/node/virtualRepoSelected/browse/BrowseRepos.portlet	::content:repository
Search Repositories		./contentMgmtModule/web/content/node/virtualRepoSelected/search/repoSearch.portlet	::content:repository

Table A-5 Content Portets in wlp-tools-content-web-lib (Continued)

Portlet	Requires Additional Portlet(s)?	Library Module Path	Resource ID
Browse My Workspace		./contentMgmtModule/web/content/node/workspace/browse/BrowseMyWorkspace.portlet	::content:workspaces
Browse Assigned Items		./contentMgmtModule/web/content/node/workspace/browse/myItems/BrowseMyItems.portlet	::content:workspace:checked-out

Table A-6 Repository Portlets in wlp-tools-content-web-lib

Portlet	Requires Additional Portlet(s)?	Library Module Path	Resource ID
Browse Repositories		./contentMgmtModule/web/content/repo/browse/browse/BrowseRepos.portlet	::content:repositories
Repository Details		./contentMgmtModule/web/content/repo/details/DetailsRepo.portlet	::content:repository
Associated Content (with a workflow)		./contentMgmtModule/web/content/repo/workflow/assignedContent/AssignedContent.portlet	::content:workflow
Associated Types (with a workflow)		./contentMgmtModule/web/content/repo/workflow/assignedTypes/AssignedTypes.portlet	::content:workflow
Browse Workflows		./contentMgmtModule/web/content/repo/workflow/browse/BrowseWorkflows.portlet	::content:workflows
Workflow Details		./contentMgmtModule/web/content/repo/workflow/details/DetailsWorkflow.portlet	::content:workflow

Table A-7 Content Type Portlets

Portlet	Requires Additional Portlet(s)?	Library Module Path	Resource ID
Content Type Details		./contentMgmtModule/web/content/type/property/propertyDetails.portlet	::content:type:property
		./contentMgmtModule/web/content/type/repo/listTypes.portlet	::content:repository
Content Type Properties		./contentMgmtModule/web/content/type/type/properties/typeProperties.portlet	::content:type
Content Type Summary		./contentMgmtModule/web/content/type/type/summary/typeSummary.portlet	::content:type
Browse Repositories		./contentMgmtModule/web/content/type/virtualRepo/browseRepos.portlet	

User and Group Management Portlet Resource IDs

The user and group management portlets include portlets that allow you to add, view, or modify users and groups.

The following tables provide information about the user and group management portlets, including the resource ID used for each.

- Table A-8, "Group Management Portlets in wlp-tools-ugm-web-lib"
- Table A-9, "User Management Portlets in wlp-tools-ugm-web-lib"

Table A-8 Group Management Portlets in wlp-tools-ugm-web-lib

Portlet	Requires Additional Portlet(s)?	Library Module Path	Resource ID
Group Details		./ugmModule/web/ugmTools/groupTools/details/details.portlet	::security:group
Groups in Group		./ugmModule/web/ugmTools/groupTools/listGroups/listGroups.portlet	::security:group

Table A-8 Group Management Portlets in wlp-tools-ugm-web-lib (Continued)

Portlet	Requires Additional Portlet(s)?	Library Module Path	Resource ID
Member of Group		./ugmModule/web/ugmTools/groupTools/listMembers/listMembers.portlet	::security:group
		./ugmModule/web/ugmTools/groupTools/listProfile/listProfile.portlet	::security:group
Users in Group		./ugmModule/web/ugmTools/groupTools/listUsers/listUsers.portlet	::security:group
		./ugmModule/web/ugmTools/groupTools/multipleProfiles/multipleProfiles.portlet	::security:group

Table A-9 User Management Portlets in wlp-tools-ugm-web-lib

Portlet	Requires Additional Portlet(s)?	Library Module Path	Resource ID
Browse Users		./ugmModule/web/ugmTools/userTools/browseUsers/browseUsers.portlet	::security:users
User Details		./ugmModule/web/ugmTools/userTools/details/details.portlet	::security:user
		./ugmModule/web/ugmTools/userTools/groupMembership/groupMembership.portlet	::security:user
		./ugmModule/web/ugmTools/userTools/listProfile/listProfile.portlet	::security:user
		./ugmModule/web/ugmTools/userTools/multipleProfiles/multipleProfiles.portlet	::security:user
		./ugmModule/web/ugmTools/userTools/userTools.portlets	::security:user