Content Portlet Suite (CPS) Portlet Developer Guide
10g Release 3 (10.1.3.3.1)

May 2007

ORACLE

Content Portlet Suite (CPS) Portlet Developer Guide, 10g Release 3 (10.1.3.3.1)

Contributing Author: Will Harris
Contributors: Adam Stuenkel, David Truckenmiller

Table of Contents

Chapter

1

INTRODUCTION

OVERVIEW

The information contained in this guide is subject to change as product technology evolves and as hardware and operating systems are created and modified.

Due to the technical nature of browsers, databases, web servers, and operating systems, Oracle, Inc. cannot warrant compatibility with all versions and features of third-party products.

This section contains the following topics:

ABOUT THIS GUIDE

This guide is intended for application developers and programmers. It offers an overview of the portlets, a presentation of their framework and architecture, and information on using the portlet Software Development Kit (SDK).

PRODUCT OVERVIEW

One key to implementing a corporate portal is ensuring a secure, personalized way to aggregate data for consumption and processing. To provide the best possible combination of a portal solution and a content management solution, Oracle has created Content Portlet Suite (CPS).

With Content Portlet Suite (CPS), you can manage content creation and the distribution process through a set of easy-to-use portlets. By providing access to Content Server and Image Server, Content Portlet Suite enables users to update, search, and view content in an easy, efficient way.

The portlets in Content Portlets Suite can be enabled for different users, based on their roles and permissions in the organization, and can be easily customized. Users can browse or search content based on their permissions, contribute new content (with the appropriate level of access), and view the progress of their content through workflow.

Keeping portals up-to-date

The ease of use of Content Portlet Suite addresses a key issue: how to keep content up-to-date. By driving content updates and additions through the portal interface, the process of updating the portal becomes part of using the portal, as opposed to a separate task performed outside of the portal.

Ease of use for contribution

With CPS, content contribution is simple; users can contribute content by checking in a document. CPS takes care of normalizing the data with its ability to convert files through templates to your specified markup for viewing in the portal.

Ease of use for workflow

After checking in or updating content, users can track content status through the portal. Users are notified right in the portal of their workflow status—whether to review, edit, or approve content—with links to the content itself. They can click a link to view a content item, approve or edit it, and then send the item on its way for further workflow or publishing. When content is approved, it is published and made available for viewing. Approved content can be published and expired at predetermined times (as in the case of a promotional offer or classified ad).

Ease of use for browsing

Content can be presented to users based on their role in the organization. The support team may log in to see a portlet that shows new updates and fixes, with another portlet that shows current cases. These are automatically updated through the content management system and require no manual intervention. They appear to the users as titles with a link to the content; they may also have a short summary of the content.

Ease of use for searching

Content can be easily searched based on metadata categories and full-text content through one of the portlets. For instance, users may search for a “support note” content type containing the word “policy,” which removes the need to search through all other occurrences of the word “policy” in other content types. CPS provides separate portlets for basic and authenticated search. CPS even provides a portlet that allows users to save searches they use regularly.

Easy to administer

Content Server and Content Portlet Suite are easy to administer. They can be set up and left to run indefinitely. Once you have decided your metadata model and workflow paths, Content Server and Content Portlet Suite can be installed and linked to the portal, and users can begin contributing and viewing content.

Metadata Admin portlet

Content Portlet Suite includes the Metadata Admin portlet that allows you to modify properties of the content server metadata fields, which affect the behavior of the Contribution portlet. With the Metadata Admin portlet, an administrator can specify which metadata fields users see, as well as the default value for each field. Administrators can even hide metadata fields and specify what values should be set for those items.

High performance

Content Portlet Suite is built on top of Content Integration Suite. This powerful integration layer offers speedy asynchronous access to content.

Easy to customize

The portlets in the Content Portlet Suite are designed to be customizable. Customizing the portlets will help you make the most of your portal investment. For example, to create a

new portlet to show content of a certain metadata value, you can copy the existing Browse portlet, tweak one simple parameter, and publish the portlet.

By integrating Content Portal Suite and your portal server with Content Server, you provide an easy way to keep your portal up-to-date. You also improve efficiency, lower costs, and increase your return on your portal investment.

CONVENTIONS USED IN THIS GUIDE

This guide uses the forward-slash (/) to separate directories. Depending on your operating system, you may need to change the separation markers when defining directories. Other conventions are:

Convention	Definition
Bold	Indicates an item that you select in the interface, such as a button or menu, in order to perform a specific task, for example, "Click **OK**."
>	Indicates a menu choice. For example, "**Choose File > Open**" means "Click the **File** menu, and then click **Open**."
`Code`	Indicates the actual code used.

Notes, technical tips, important notices, and cautions use these conventions:

Symbols	Description
	This is a note. It is used to bring special attention to information.
	This is a technical tip. It is used to identify information that can be used to make your tasks easier.
	This is an important notice. It is used to identify a required step or required information.
	This is a caution. It is used to identify information that might cause loss of data or serious system problems.

Chapter

2

CPS PORTLETS

OVERVIEW

A portlet is a Java technology–based web component, managed by a portlet container, that processes requests and generates dynamic content. Portlets are used by portals as pluggable, user-interface components that provide a presentation layer to information systems.

This section contains the following topics:

See also:

- CPS JavaDoc for information on the Class/Interface, Field, and Method description (located in the /docs/cps-javadoc.zip file).
- CIS Developer Guide for information on command invocation, execution, and extending commands (cis-developer-guide.pdf in the /docs directory of the unbundled CIS distribution file).

PORTLET DESCRIPTIONS

The CPS Portlets use the Universal Content and Process Management API (UCPM API) within the CIS distribution to communicate back to the Content Server and/or Image Server. The Portlet API facade abstracts the common operations within portlet containers so our framework will work on a variety of platforms (WebSphere, WebLogic, Plumtree, and SunONE) using the same handler code. Portlet Actions are mapped to a custom MVC framework that uses the UCPM API to perform the desired task.

The Content Portlet Suite implements the following set of portlets that interact with Content Server and/or Image Server:

Content Server

- **Library:** Presents content to users based on their role in the organization.
- **Search:** Allows the user to perform a keyword or full-text search on the content server and permits read-only access to the returned content.
- **Saved Search:** Allows the user to save frequently used queries.
- **Contribution:** Allows the user to contribute content to the content server.
- **Workflow Queue:** Notifies users of their workflow tasks.
- **Authenticated Library:** Presents content to users based on their role in the organization, and provides read/write access to the returned content.
- **Authenticated Search:** Allows the user to perform a selected metadata and keyword search on the content server and provides read/write access to the returned content.
- **Metadata Admin:** Allows the user to modify the properties of custom metadata.

Image Server

- **Image Server Search:** Allows the user to perform a search on the image server and permits read-only access to the returned content.

Content Server and Image Server

- **Federated Search:** Facilitates the integration of an image server with one or more content servers and allows queries of all the repositories.

Each of these portlets requires that Content Integration Suite (CIS) be installed and available. CIS provides a Java API into the content and image servers, and is capable of

running in either a J2EE application server environment (e.g., WebSphere or WebLogic) or a servlet container environment (e.g., Tomcat).

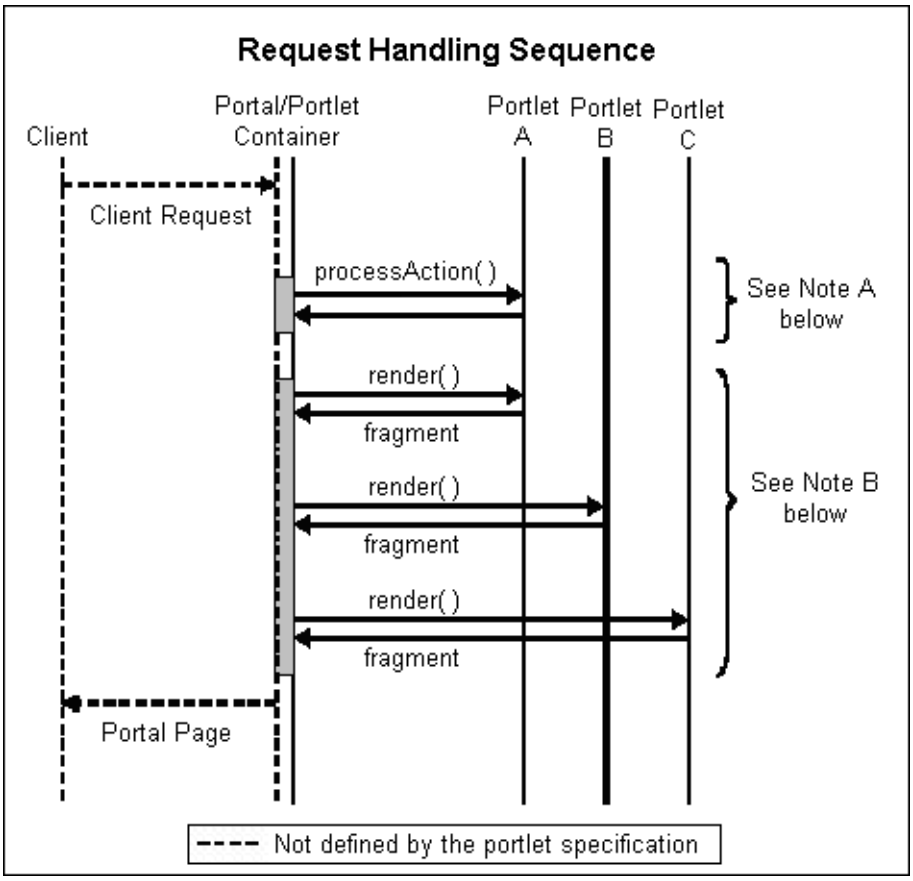
The portlets are consumers of standard content server services (IdcCommand services), such as CHECKIN_UNIVERSAL and GET_SEARCH_RESULTS. However, these services are not called directly by the dispatch handlers from the portlet controller. Rather, the UCPM API abstracts the portlets from the details of talking to the server. The UCPM API allows for rigid parameter validation, dynamic command selection, and standardized integration with a J2EE environment.

Note: See the CIS Developer Guide for more information (cis-developer-guide.pdf in the /docs directory of the unbundled CIS distribution file).

REQUEST HANDLING

As an example of request handling, here is the high-level sequence of events using the Search portlet:

1. A user enters a query and clicks the Search button.
2. An ‘action’ URL is built and routed to the portlet container, which, in turn, routes the command to the appropriate portlet (in this case, the Search portlet).
3. A ‘processAction’ is called on the Search portlet.
4. The Search portlet retrieves the search parameters (they are part of the URL that was built), and calls the ‘search’ method on the CIS layer.
5. The CIS layer queries the content server, retrieves the data, and passes the data object to the Search portlet.
6. The portlet container calls *render* on each of the portlets on the page (including the Search portlet), and each portlet uses the received data, or refreshes the data, and displays HTML fragments to the user.



Note A: The *action* requests must end before the *render* requests begin.

Note B: The *render* requests are not triggered in a specific order and may be executed sequentially or simultaneously.

MIGRATING TO VERSION 10GR3

CPS version 10gR3 depends on the Content Integration Suite (CIS) UCPM 8.0.0 API. If you are upgrading from CPS version 7.6, you should be aware of a change in the product.

Note: See the CIS Developer Guide for more information (cis-developer-guide.pdf in the /docs directory of the unbundled CIS distribution file).

Changes for this version of CPS:

- Calls to createSCSContext() and createSISContext() where changed.

  From:

```
cisApplication.getCommandFacade.createSCSContext();
cisApplication.getCommandFacade.createSISContext();
```

To:

```
cisApplication.getUCPMAPI().getActiveAPI()._createSCSContext();
cisApplication.getUCPMAPI().getFixedAPI()._createSISContext();
```

- The location of com.stellent.web.servlets have moved to com.stellent.cis.web.servlets. Thus, these .java files have changed location:
  - SCSPortletDynamicConverterServlet.java
  - SCSPortletDynamicURLServlet.java
  - SCSPortletFileDownloadServlet.java

  From:

  `core/framework/src/java/com/stellent/web/servlets/`

  To:

  `core/framework/src/java/com/stellent/cis/web/servlets/`

Chapter

3

CPS Portlet Software Development Kit

Overview

The CPS Portlet Software Development Kit (SDK) provides everything you need to build, customize, and distribute portlets.

This section contains the following topics:

See also:

- Chapter 4 (*Using the CPS Portlet SDK)* for more information on how to create portlets.
- CPS JavaDoc for information on the Class/Interface, Field, and Method description (located in the /docs/cps-javadoc.zip file).

SDK DIRECTORY STRUCTURE

The Portlet SDK can be found in the sdk/ directory of the unbundled CPS distribution file. It consists of these sub-directories:

- **ReferencePortlets**: Contains the source code for the CPS Portlets, including the Java code, JSP pages, and Ant build.xml file that is used to create customized portlets. Thus, those who wish to customize the portlets have access to the source code for the portlet JSP pages and the Model-View-Controller framework.
- **PortletBuilder**: Provides the structure for creating new portlets that use the Model-View-Controller framework and the CIS layer. It includes an Ant build.xml file that can be used to create custom portlets for a target platform (WebSphere, WebLogic, Plumtree, or Sun ONE).
- **lib:** Contains the CPSSDK tag libraries bundled in JAR files.
- **sample:** Contains a sample development portlet that shows how to use the PortletBuilder directory to create a custom CPS Portlet.

Note: Apache Ant is a Java-based build tool that must be installed in order to build customize portlets. This tool is available at: http://ant.apache.org

PORTLET DEVELOPMENT TIPS

Whenever possible, CPS follows web standards and uses the Model-View-Controller design pattern. By following these best practices your portlets will be portable and maintainable. Portlet developers should use these coding guidelines when designing and developing portlets.

This is not intended as a primer for portlet development, as it does not address the fundamentals of portlet programming. Instead, use this as a checklist during design and code reviews to help promote consistent and quality portlet implementations.

Use these practical recomendations when developing portlets:

- Use taglibs whenever possible

  Encapsulating Java code within JSP Tag Libraries allows you to more easily reuse common functions and makes the JSP pages easier to update.

- Do not give portlets and servlets the same name

  Some portal servers use the portlet name to identify the portlet within a web application and may cause errors if encounters a servlet with the same name.

- Do not use head or body tags

  The portlet JSP page contributes to the content of a larger page. Because the HTML fragment is being added to a table cell <td></td> in the portal, it should not include <html>, <head>, or <body> tags.

- Avoid client-side JavaScript

  Using JavaScript executed on the browser makes your portlets browser-dependant and requires additional cross-browser testing.

- Follow the Model-View-Controller design pattern

  CPS uses a Model-View-Controller design pattern based on the open source Struts and Tiles framework. Thus, the presentation of data should be separated from the logic that obtains and organizes the data.

- Use the JavaServer Pages Standard Tag Library (JSTL)

  The JSTL defines many commonly needed tags for conditions, iterations, formatting, etc. When you see the c: prefix in the code of JSP pages, these tag libraries are being used. You can find more information about these tag libraries at: http://jakarta.apache.org/taglibs/

USING REFERENCEPORTLETS AND PORTLETBUILDER

The CPS Portlet SDK includes the **ReferencePortlets** and **PortletBuilder** directories. The ReferencePortlets directory contains source code and the PortletBuilder directory contains the portlet build files. These directories share a similar build environment and Ant scripts.

This directory structure is used by the supplied Ant file to build a portlet distribution. The PortletBuilder Ant script builds a single portlet as an example of how to package the needed portlet files for a few portal vendors (WebSphere, WebLogic, Plumtree, and Sun ONE). Users wishing to build many portlets should adapt the scripts accordingly.

Directory Structure	Definition
lib/compile/$portalvendor	Contains the libraries needed for building the portlets.
lib/deploy/$portalvendor	Contains the libraries needed for deploying the portlets.

Directory Structure	Definition
resources/$portalvendor	Contains global and portal vendor-pecific files needed for portlet packaging.
src	The source files for the new portlet.
build/$appserver	The directory generated during the build to hold the classes and other build-related files.
build/$appserver	The directory generated after the build is run to hold the built portlet.
dist/$appserver	Contains the libraries needed for building the portlets.

USING ANT TO BUILD PORTLET DISTRIBUTIONS

Both the PortletBuilder and ReferencePortlets root directories contain an Ant file that performs the compilation and packaging of the portlet. This root directory will be referred to as $workingdirectory. The distribution process is invoked by the following commands:

```
cd $workingdirectory
ant dist
```

For this distribution to work correctly, the following two environment variables should be set in the build.properties file in the $workingdirectory directory.

Property Name	Definition
portal.vendor	The name of the portal vendor that is the target for the current distribution.
portlet.name	The name the user wishes to use for the current build. This name will be used in the generation of the descriptor files for the portlet.

The newly built portlet can be found in the $workingdirectory/dist/$portal.vendor directory.

Note: Apache Ant must be installed for this process to work properly. This tool is available at: http://ant.apache.org

USING THE CPS PORTLET TAG LIBRARIES

The CPS Portlet Tag Libraries includes several tags that may be useful when building customized portlets. The CPS Portlet Tag Libraries are located in the lib/ directory and are bundled in JAR files.

URI Creation

`<SCS:CreateURI mode=("edit" | "help" | "view")>`

Creates a URL through the PortletAPIFacade. The mode parameter is optional and, if used, the created URL will cause the portlet mode to be switched to the user-specified value. This tag is often used in conjunction with the following two nested tags:

`<SCS:URIAction name="$actionName">`

Modifies the created URL by specifying an action to perform. This action name is defined in the PortletDispatch.xml file (see Portlet Dispatch Framework).

`<SCS:URIParameter name="$paramName" value="$paramValue">`

Add the name/value pair to the generated URL.

Code sample:

```
<a href="
<scsportlet:createURI>
<scsportlet:URIAction value="checkOut"/>
<scsportlet:URIParameter name="documentID" value='<%=id%>' />
</scsportlet:createURI>">
Check Out File
</a>
```

Error Handling

`<SCS:Error id="$errorObject">`

Determines if an error is present. If an error is found, the body of the tag is evaluated and the error object variable is set.

Code sample:

```
<scsportlet:error id="error">
<div class="portlet-msg-error">
<%=error.getMessage ()%></div>
</scsportlet:error>
```

Portlet Preferences

```
<SCS:GetPreference preference="$prefName" result="$esultVar">
```

Retrieves the specified portlet preferences and stores the result in the specified variable name.

Code sample:

```
<scsportlet:getPreference preference="maxResults" result="maxResultsVar" />
```

Chapter

4

USING THE CPS PORTLET SDK

OVERVIEW

The Content Portlet Suite (CPS) includes the CPS Portlet Software Development Kit (SDK), which is used to develop new portlets.

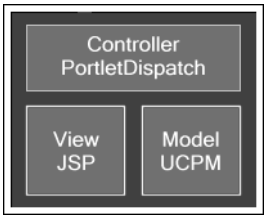
This section contains the following topics:

See also:

- Chapter 3 (*CPS Portlet Software Development Kit)* for an overview of the CPS Portlet SDK, an explanation of the various subdirectories, and a description of how Ant and the tag libraries are used to build portlet distributions.
- CPS JavaDoc for information on the Class/Interface, Field, and Method description (located in the /docs/cps-javadoc.zip file).

MODEL-VIEW-CONTROLLER FRAMEWORK

The Content Portal Suite uses a Model-View-Controller design pattern based on the open source Struts and Tiles framework. The View is usually handled by JSP pages. The Model is generally a UCPM API ICISObject that represents data retrieved from the content server or image server. The Controller are the Java classes in the com.stellent.portlet.framework package that controls the execution of actions.



Note: See the CIS Developer Guide for more information (cis-developer-guide.pdf in the /docs directory of the unbundled CIS distribution file).

PORTLET CONSTRUCTION

Any portlet you build with the CPS Portlet SDK contains a dispatch configuration file, a set of JavaServer Pages, and a set of action handlers. When the user clicks a link in a specific portlet, the associated action handler is executed and the result of the action is placed on the request. The Tile configured to be the destination after the action is executed is then found, and its associated JSP pages are inserted. The JSP page then models the data that was the result of the action.

CREATING A DISPATCH CONFIGURATION FILE

A dispatch configuration file defines each action handler, each Tile, and information about the portlet itself. By default, the naming convention is “stellent <portletname> dispatch.xml”.

Example: stellentactivesearchdispatch.xml

The entry point to CPS Portlets is the SCSPortlet class (there may be different implementations of this per container). This class extends the GenericPortlet class. At initialization, it looks for its configuration file in the "WEB-INF/config/" directory.

Keywords

These special keywords can be used as view targets:

- **default:** Renders the default page for the portlet as defined by the default-action node; if the user is in edit mode, the default edit mode page is displayed.
- **previous:** Renders the previous page in the stack.
- **login:** Rpecifying an action node with this name will cause the framework to execute the action handler upon detection of a new login
- **error:** CPS displays a default error page that assumes a throwable has been placed on the request, you may override this error page by creating a new action definition with the keyword name ‘error’.

Active Search Dispatch Configuration

Here is an example of the active search dispatch configuration coding:

```
<portletdispatch-config>

<!--
  Default action parameters, name for the default action, cacheResult is a
  boolean that specifies whether the default behavior is to cache the action
  result on the session. If the value is set to false the action will be
  performed each time the portlet is rendered, the result data is discarded
  each time.

  The cacheResult value here can be overriden by the action definition itself,
  it the action does not specify, the default value is used.
-->

<default-action view="showHome" edit="showEdit" cacheResult="true"/>

  <!--
  Portlet-id is used to ensure that unique HTML form, javascript names are
  used, this value will be available on the request object as
  ISCSAction.PORTLET_ID
  -->

<portlet-id value="active_search_portlet"/>

  <!--
  Definitions for all the action types available to this portlet
  -->

<action-mappings>
```

```
  <forward name="showHome" authRequired="true" path="active.search.main.page"/>
  <forward name="showEdit" authRequired="true" path="active.search.edit.page"/>
    <location path="/WEB-INF/actions/active_search_actions.xml" />
    <location path="/WEB-INF/actions/active_document_actions.xml" />
</action-mappings>

  <!--
  Definitions for UI components available to this portlet
  -->

<tiles-definitions>
  <definition name=".mainLayout" path="/stellent/ui/layouts/mainlayout.jsp">
    <put name="header" value="/stellent/ui/fragment/header.jsp"/>
    <put name="footer" value="/stellent/ui/fragment/footer.jsp"/>
    <put name="content" value="/stellent/ui/layouts/defaultContent.jsp"/>
  </definition>
  <location path="/WEB-INF/tiles/active_search_tiles.xml" />
  <location path="/WEB-INF/tiles/active_document_tiles.xml" />
</tiles-definitions>
</portletdispatch-config>
```

Types of Child Nodes

The top-level node, <portletdispatch-config>, can have these types of child nodes:

- Default Action Node (page 4-4)
- Portlet ID Node (page 4-5)
- Location Node (page 4-5)
- Action Mappings Node (page 4-5)
- Tiles-Definitions Node (page 4-7)

Default Action Node

The <default-action> node is used to specify the action to execute or Tile to display when the portlet or the edit mode is first visited. It will also be the action executed when the keyword 'default' is the target of another action.

view/edit attribute: Specifies the view/edit default; the values are the name of a defined action and the default edit action. For example, the defined action of ‘showHome’ and the default edit action of ‘showEdit’.

cacheResult attribute: Indicates whether or not the result of the action should be cached on the session or re-executed each time the render () method is called. When users perform actions on other portlets it generates a render call which asks the portlet to redraw itself. If

the cachResult is set to ‘true’ this redraw will not re-execute the action but instead uses the cached result. In the case of the Active Search portlet it is set to true by default. individual actions can override the default for the portlet, this value is only used when not specified by the action definition.

Portlet ID Node

The <portlet-id> node is used to specify a unique name for the portlet. The string value specified here is made available on the request with the parameter name ISCSAction.PORTLET_ID. This ID is mainly used to uniquely identify HTML elements such as forms and JavaScript functions so they do not collide with other portlets on the same page.

Location Node

The <location> node is used to specify another dispatch configuration file in which to load definitions from. It simply takes a path attribute and it indicates where to look for the configuration file to be loaded.

This node can be a child of the Action Mappings node or the Tile Definitions node. If there is a name conflict the Action Mappings in the current configuration XML take precedence over the loaded action definitions.

Action Mappings Node

The <action-mappings> node is the container for action definitions, which are usually defined by an Action node; two exceptions are the Forward node and the Location node.

Action Node

The <action> node specifies several attributes, which are used to perform the desired action. Here is an example of an action definition:

```
<!--
Shows the form to add new saved search.
-->

<action
  name="active.search.showAddSavedSearch"
  class="com.stellent.portlet.components.search.active.handlers.
         ShowAddSavedSearchHandler"
  bean="com.stellent.portlet.components.search.active.forms.
```

```
        AddSavedSearchForm"
  authRequired="true"
  addToStack="false">
  <forward name="success" path="active.search.savedsearch.add.page"/>
</action>
```

The attributes are:

- **name**: The name of the action. This is used when executing this action within a JSP page.
- **class**: The fully qualified class name of the class that implements the ISCSActionHandler interface. This class is where you will add your action handler code. Both the name and class attributes are required to define an action.
- **bean**: The fully qualified class name of a class that implements the ISCSActionForm interface. This is passed into your action handler when the handleAction method is called. This bean can be populated through an HTML form post or by explicit definition through a special CPS portlet tag. This is an optional attribute, if the action does not need any input parameters this attribute can be omitted, the ISCSActionForm passed into the handleAction will be null.
- **authRequired**: Controls whether the framework will actually execute the action if an unauthenticated portal user tries to execute the action. It defaults to false and is an optional parameter. If it is set to true and an unauthenticated user attempts to execute the method a special system JSP page will displayed asking the user to first login before attempting to use the portlet.
- **addToStack**: Defines whether the portlet framework will execute this action again or cache the result when *render* is called for a redraw. It defaults to true so that portlets will show the last state when redrawing, however, some actions should not be performed more than once, so you may tell the framework not to save the result or remember it as the last action.

The <action> node is also a container for any number of forward actions, which specify which Tile or JSP page the portlet should display upon completion of the action. You may specify as many different ‘forwards’ as you want in this list, as long as they have unique names. The action handler code itself specifies which ‘forward’ to use upon completion of the action. If the handler does not explicitly state the view name to forward to upon completion of the action, it defaults to the *success* forward.

In addition to these framework properties, you may specify an arbitrary number of custom attributes for an action definition. These attributes will be available to the action handler via the ISCSActionHandler getAttributes() method. For example, the Contribution portlet

adds a custom property called 'async' that indicates whether contributions should leverage the Java Messaging Service (JMS) to do document contributions asynchronously.

Forward Node

The <forward> node is a special type of action that does not actually execute any code but automatically forwards the display to the specified Tile definition or explicit JSP page location. The path attribute accepts either of these values.

Tiles-Definitions Node

The Tiles and Struts design pattern tells the framework how to render a particular view by specifying a main JSP page, various regions of content, and an optional controller class that are all used to create the final view.

Example:

```
<definition name=".mainLayout" path="/stellent/ui/layouts/mainlayout.jsp">
  <put name="header" value="/stellent/ui/fragment/header.jsp"/>
  <put name="footer" value="/stellent/ui/fragment/footer.jsp"/>
  <put name="content" value="/stellent/ui/layouts/defaultContent.jsp"/>
</definition>
```

This defines a Tile of the name ".mainLayout" and specifies the JSP page "/stellent/ui/layouts/mainlayout.jsp" to be the main JSP page to use when rendering this view. Notice the Put nodes, which specify the regions of content available to the main JSP page. In this example, three regions are available: a header, footer, and content region. Each of these Put nodes specify a name for the region and the corresponding JSP page to use to render the region.

You can also specify a controller for Tile definitions and specify inheritance, as in the following definition:

```
<definition name="active.search.edit.page" extends=".mainLayout"
                  controllerClass="com.stellent.portlet.components.search.active.
                  controllers.EditController">

<put name="content" value="/stellent/ui/layouts/search/active/
                  active_search_edit.jsp" />

</definition>
```

This Tile extends the ".mainLayout" Tile that we defined earlier and inherits its configuration. We add a controllerClass to this Tile which is an object that implements the ISCSController interface and provides a hook to execute Java code before the Tile is

rendered in case processing is needed. Notice that this Tile definition overrides the "content" region and changes the JSP page that is used to render this region.

GETTING A REFERENCE TO THE PORTLET API FACADE

You may at any point in the code execute the following line to get the stateless API facade object:

```
IPortletAPIFacade facade = PortletAPI.getInstance ().getPortletAPIFacade ();
```

This will return a facade object that is relevant to your current container. If the facade is unable to determine which vendor the container was implemented for, it creates a generic facade object. To write a new detector, see the com.stellent.portlet.api.PortalVendor class and add your new detector based on the other implementations.

CREATING AN ACTION HANDLER

An action definition is invoked through a JSP page, like the following:

```
<form name="subAuthSearch" method="POST" onSubmit="prepareAuthScsSearch()"
            action='<scsportlet:createURI><scsportlet:URIAction
            value="active.search.doSearch" />
            </scsportlet:createURI>'>
```

In this example, the CPS tag “createURI” invokes the “active.search.doSearch” action when the form is submitted. The name “active.search.doSearch” maps to an action definition created in the configuration file. See Action Node (page 4-5) for additional information.

The action definition specifies a class name. The class name should be an object that implements the ISCSActionHandler interface, which has a variety of methods to implement that the framework uses to exercise the object. However, by extending the abstract base class SCSActionHandler, the developer need only implement one method:

```
/**
   * Handle an action from the portlet
   * @param portletRequest
   * @throws com.stellent.portlet.dispatcher.PortletDispatcherException
   */
public ISCSActionResult handleAction (ISCSActionForm form, Object portletRequest)
    throws PortletDispatcherException, CommandException, RemoteException;
```

This method will be called each time the action is invoked through the portlet framework. The method is passed in an ISCSActionForm, which is a bean that represents the parameters that are made available to this action.

Note: This class will be of the type specified in the Action node.

The already initialized CISApplication object will be available to the action handler through the getCISApplication() method when inside the handleAction method, as will any other attributes specified on the Action node via the getAttributes() method. You may also access a unique ID for this handler via the getID() method. This can be used to store information on the session without fear of collision.

The return type is that of an action result object. Usually, this is simply a container for the result parameters that are to be stored on the request for access within the JSP page. However, you may specify other parameters to this result, such as the view that should be used upon return. (It defaults to “success” if you use the base class SCSActionResult.)

Sample Action Handler

The action definition specifies the action name “active.document.checkOut”; the ISCSActionHandler class that performs the action; the ISCSActionForm, which is a bean that represents the parameters passed into the handler; and the resulting Tile that is to be displayed upon completion of the action.

```
<!--
Attempts to check out the specified document.
-->

<action
  name="active.document.checkOut"
  class="com.stellent.portlet.components.document.active.handlers.CheckOutHandler"
  bean="com.stellent.portlet.components.document.active.forms.CheckOutForm"
  authRequired="true" >

  <forward name="success" path="active.document.checkout.page" />
</action>
```

The SCSActionForm code represents the parameters the checkout handler needs to complete its action. In the following example, checkout needs only the document ID of the document that we are planning to check out:

```
public class CheckOutForm extends SCSActionForm {
  private String m_documentID;
```

```
  public String getDocumentID () {
    return m_documentID;
  }

  public void setDocumentID (String documentID) {
    m_documentID = documentID;
  }
}
```

The SCSActionHandler code first checks to see if the passed-in form is really an instance of CheckOutForm. It errors out if this is not the case; otherwise, it checks out the file through the UCPM API. This puts the resulting ICISObject on the request by calling result.setVariable(name, object). These objects will now be available to the JSP page rendering the view.

Note: See the CIS Developer Guide for more information (cis-developer-guide.pdf in the /docs directory of the unbundled CIS distribution file).

```
public class CheckOutHandler extends SCSActionHandler {
/**
   * Checks out the specified content.
   *
   * @param portletRequest
   * @throws com.stellent.portlet.dispatcher.PortletDispatcherException
   */

public ISCSActionResult handleAction (ISCSActionForm form,
            Object portletRequest)
            throws PortletDispatcherException, CommandException, RemoteException {
  ISCSActionResult result = new SCSActionResult ();

  if (form instanceof CheckOutForm) {
    CheckOutForm cof = (CheckOutForm)form;
    ISCSContext ctx = SCSSession.getSCSContext (portletRequest);

    //Get checkout response
    ISCSDocumentCheckoutAPI checkoutAPI =
            getCISApplication ().getUCPMAPI ().getActiveAPI
            ().getDocumentCheckoutAPI ();
    ISCSDocumentID docID =
            getCISApplication ().getUCPMAPI ().getActiveAPI
            ()._createDocumentID (cof.getDocumentID (),
            ctx.getAdapterName ());
    ISCSDocumentActionResponse resp =
            checkoutAPI.checkoutFileByID (ctx, docID);

    //Get document info
    ISCSDocumentInformationAPI docInfoAPI =
```

```
            getCISApplication ().getUCPMAPI ().getActiveAPI
            ().getDocumentInformationAPI ();
    ISCSDocumentInformationResponse docInfoResp =
            docInfoAPI.getDocumentInformationByID
            (SCSSession.getSCSContext (portletRequest), docID);

            result.setVariable ("checkoutResponse", resp);
            result.setVariable ("infoResponse", docInfoResp.getDocNode ());
    } else {
            throw new PortletDispatcherException ("Unexpected form type,
                                expected 'CheckOutForm', got " + form);
    }

  return result;
}
```

CREATING A TILE

A Tile consists of a definition, an optional controller class, and a collection of JSP classes that will make up a portlet view. The definition section contains XML code that identifies the main layout JSP page.

The following example specifies three different regions that reference three JSP pages:

```
<%@ include file="/stellent/ui/fragment/jspimport.inc" %>
<scsportlet:insert name="header" />
<scsportlet:insert name="content" />
<scsportlet:insert name="footer" />
```

The ‘include’ at the top includes commonly defined imports and taglib definitions. For example, this is an example of the file for the portlets included with Content Portlet Suite:

```
<%@ page import="com.stellent.portlet.api.IPortletAPIFacade" %>
<%@ page import="com.stellent.portlet.api.PortletAPI" %>
<%@ include file="/stellent/ui/fragment/page.inc" %>
<%@ taglib uri="/WEB-INF/tlds/i18n.tld" prefix="i18n" %>
<%@ taglib uri="/WEB-INF/tlds/scsportlet.tld" prefix="scsportlet" %>
<%@ taglib uri="/WEB-INF/tlds/c.tld" prefix="c" %>
<%@ taglib uri="/WEB-INF/tlds/scs-databinder.tld" prefix="db" %>

<%
  //the api facade class
  IPortletAPIFacade apiFacade = PortletAPI.getInstance ().getPortletAPIFacade ();
%>
```

In the JSP page, three simple lines use the insert tag included with CPS to tell the view to put the header first, the content next, and the footer last. For each region, the insert tag tells

the framework to look up the definition and include the JSP page it finds defined in the location specified by the insert tag.

CREATING A CONTROLLER

A controller is a hook that allows the Tile author to execute Java code before the Tile itself is rendered. To create a controller, you need to implement only the ISCSController class, which requires a variety of methods that the portlet framework uses to control its lifetime.

Fortunately, there is an abstract base class included (the SCSController class), which in most cases performs all the operations you need, save one:

```
/**
   *  Method is called before a Tile is rendered.
   *
   * @param portletRequest The portlet request that generated the Tile render.
   * @param portletResponse The portlet response associated with Tile render.
   * @throws ServletException If a portlet container error occurs.
   * @throws IOException If a portlet container error occurs.
   * @throws CommandException If a CIS framework error occurs.
   * @throws RemoteException If a CIS communication error occurs.
 */

public void perform (Object portletRequest,
            Object portletResponse)
            throws ServletException, IOException, CommandException,
RemoteException;
```

This method will get called right before the Tile is rendered and any objects you place on the request will be available to the resulting JSP page.

Appendix

A

THIRD PARTY LICENSES

OVERVIEW

This appendix includes a description of the Third Party Licenses for all the third party products included with this product.

APACHE SOFTWARE LICENSE

```
* Copyright 1999-2004 The Apache Software Foundation.

* Licensed under the Apache License, Version 2.0 (the "License");

* you may not use this file except in compliance with the License.

* You may obtain a copy of the License at

*      http://www.apache.org/licenses/LICENSE-2.0

*
```

```
* Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
```

W3C® SOFTWARE NOTICE AND LICENSE

```
* Copyright © 1994-2000 World Wide Web Consortium,
* (Massachusetts Institute of Technology, Institut National de
* Recherche en Informatique et en Automatique, Keio University).
* All Rights Reserved.  http://www.w3.org/Consortium/Legal/
*
* This W3C work (including software, documents, or other related items) is
* being provided by the copyright holders under the following license. By
* obtaining, using and/or copying this work, you (the licensee) agree that
* you have read, understood, and will comply with the following terms and
* conditions:
*
* Permission to use, copy, modify, and distribute this software and its
* documentation, with or without modification, for any purpose and without
* fee or royalty is hereby granted, provided that you include the following
* on ALL copies of the software and documentation or portions thereof,
* including modifications, that you make:
*
*   1. The full text of this NOTICE in a location viewable to users of the
*      redistributed or derivative work.
*
*   2. Any pre-existing intellectual property disclaimers, notices, or terms
```

```
*      and conditions. If none exist, a short notice of the following form
*      (hypertext is preferred, text is permitted) should be used within the
*      body of any redistributed or derivative code: "Copyright ©
*      [$date-of-software] World Wide Web Consortium, (Massachusetts
*      Institute of Technology, Institut National de Recherche en
*      Informatique et en Automatique, Keio University). All Rights
*      Reserved. http://www.w3.org/Consortium/Legal/"
*
*   3. Notice of any changes or modifications to the W3C files, including the
*      date changes were made. (We recommend you provide URIs to the location
*      from which the code is derived.)
*
* THIS SOFTWARE AND DOCUMENTATION IS PROVIDED "AS IS," AND COPYRIGHT HOLDERS
* MAKE NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT
* NOT LIMITED TO, WARRANTIES OF MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR
* PURPOSE OR THAT THE USE OF THE SOFTWARE OR DOCUMENTATION WILL NOT INFRINGE
* ANY THIRD PARTY PATENTS, COPYRIGHTS, TRADEMARKS OR OTHER RIGHTS.
*
* COPYRIGHT HOLDERS WILL NOT BE LIABLE FOR ANY DIRECT, INDIRECT, SPECIAL OR
* CONSEQUENTIAL DAMAGES ARISING OUT OF ANY USE OF THE SOFTWARE OR
* DOCUMENTATION.
*
* The name and trademarks of copyright holders may NOT be used in advertising
* or publicity pertaining to the software without specific, written prior
* permission. Title to copyright in this software and any associated
* documentation will at all times remain with copyright holders.
*
```

ZLIB LICENSE

```
* zlib.h -- interface of the 'zlib' general purpose compression library
  version 1.2.3, July 18th, 2005

Copyright (C) 1995-2005 Jean-loup Gailly and Mark Adler
This software is provided 'as-is', without any express or implied
  warranty.  In no event will the authors be held liable for any damages
  arising from the use of this software.

  Permission is granted to anyone to use this software for any purpose,
  including commercial applications, and to alter it and redistribute it
  freely, subject to the following restrictions:

  1. The origin of this software must not be misrepresented; you must not
     claim that you wrote the original software. If you use this software
     in a product, an acknowledgment in the product documentation would be
     appreciated but is not required.
  2. Altered source versions must be plainly marked as such, and must not be
     misrepresented as being the original software.
  3. This notice may not be removed or altered from any source distribution.

  Jean-loup Gailly jloup@gzip.org
  Mark Adler madler@alumni.caltech.edu
```

GENERAL BSD LICENSE

Copyright (c) 1998, Regents of the University of California

All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

"Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

"Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

"Neither the name of the <ORGANIZATION> nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

GENERAL MIT LICENSE

Copyright (c) 1998, Regents of the Massachusetts Institute of Technology

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

UNICODE LICENSE

UNICODE, INC. LICENSE AGREEMENT - DATA FILES AND SOFTWARE

Unicode Data Files include all data files under the directories http://www.unicode.org/Public/, http://www.unicode.org/reports/, and http://www.unicode.org/cldr/data/ . Unicode Software includes any source code published in the Unicode Standard or under the directories http://www.unicode.org/Public/, http://www.unicode.org/reports/, and http://www.unicode.org/cldr/data/.

NOTICE TO USER: Carefully read the following legal agreement. BY DOWNLOADING, INSTALLING, COPYING OR OTHERWISE USING UNICODE INC.'S DATA FILES ("DATA FILES"), AND/OR SOFTWARE ("SOFTWARE"), YOU UNEQUIVOCALLY ACCEPT, AND AGREE TO BE BOUND BY, ALL OF THE TERMS AND CONDITIONS OF THIS AGREEMENT. IF YOU DO NOT AGREE, DO NOT DOWNLOAD, INSTALL, COPY, DISTRIBUTE OR USE THE DATA FILES OR SOFTWARE.

COPYRIGHT AND PERMISSION NOTICE

Copyright © 1991-2006 Unicode, Inc. All rights reserved. Distributed under the Terms of Use in http://www.unicode.org/copyright.html.

Permission is hereby granted, free of charge, to any person obtaining a copy of the Unicode data files and any associated documentation (the "Data Files") or Unicode software and any associated documentation (the "Software") to deal in the Data Files or Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, and/or sell copies of the Data Files or Software, and to permit persons to whom the Data Files or Software are furnished to do so, provided that (a) the above copyright notice(s) and this permission notice appear with all copies of the Data Files or Software, (b) both the above copyright notice(s) and this permission notice appear in associated documentation, and (c) there is clear notice in each modified Data File or in the Software as well as in the documentation associated with the Data File(s) or Software that the data or software has been modified.

THE DATA FILES AND SOFTWARE ARE PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT OF THIRD PARTY RIGHTS. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR HOLDERS INCLUDED IN THIS NOTICE BE LIABLE FOR ANY CLAIM, OR ANY SPECIAL INDIRECT OR CONSEQUENTIAL DAMAGES, OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THE DATA FILES OR SOFTWARE.

Except as contained in this notice, the name of a copyright holder shall not be used in advertising or otherwise to promote the sale, use or other dealings in these Data Files or Software without prior written authorization of the copyright holder.

_____Unicode and the Unicode logo are trademarks of Unicode, Inc., and may be registered in some jurisdictions. All other trademarks and registered trademarks mentioned herein are the property of their respective owners

MISCELLANEOUS ATTRIBUTIONS

Adobe, Acrobat, and the Acrobat Logo are registered trademarks of Adobe Systems Incorporated.

FAST Instream is a trademark of Fast Search and Transfer ASA.

HP-UX is a registered trademark of Hewlett-Packard Company.

IBM, Informix, and DB2 are registered trademarks of IBM Corporation.

Jaws PDF Library is a registered trademark of Global Graphics Software Ltd.

Kofax is a registered trademark, and Ascent and Ascent Capture are trademarks of Kofax Image Products.

Linux is a registered trademark of Linus Torvalds.

Mac is a registered trademark, and Safari is a trademark of Apple Computer, Inc.

Microsoft, Windows, and Internet Explorer are registered trademarks of Microsoft Corporation.

MrSID is property of LizardTech, Inc. It is protected by U.S. Patent No. 5,710,835. Foreign Patents Pending.

Oracle is a registered trademark of Oracle Corporation.

Portions Copyright © 1994-1997 LEAD Technologies, Inc. All rights reserved.

Portions Copyright © 1990-1998 Handmade Software, Inc. All rights reserved.

Portions Copyright © 1988, 1997 Aladdin Enterprises. All rights reserved.

Index

L

M

P

R

S

T

U

W