Sun Java System Portal Server 7 Configuration Guide

Sun Microsystems, Inc.
4150 Network Circle
Santa Clara, CA 95054
U.S.A.

Part No: 819–4605

060131@13215

Contents

Preface

The Sun Java™ System Portal Server Configuration Guide explains in detail how to install or upgrade to this version of the software and post installation configuration, discusses the new `psadmin` command line utilities that can be used to perform the basic duties of administrating the Portal Server software, describes the new inter-portlet feature, and includes reference material for the administration tag library.

Who Should Use This Book

This book includes information including new features and enhancements in the Portal Server software. This guide is meant for administrators and other individuals installing and using this version of the product.

Before You Read This Book

Before you read this book, see the *Sun Java System Portal Server 7 Release Notes*.

How This Book Is Organized

Chapter 1, Enabling Access to the Portal Server Through the Gateway, provides instructions on how you can enable access to thePortal Server through the Gateway.

Chapter 2, Enabling User Behavior Tracking, provides instructions for enabling User Behavior Tracking.

Chapter 3, Setting up Federated Search, provides instructions on how to set up Federated Search which enables users to submit a search query to multiple search engines concurrently and have the search results displayed in a unified format.

Chapter 4, Establishing Trust Between Two Cacao Agents, provides instructions for establishing trust between two cacao agents.

Chapter 5, Setting Up Registry Support for WSRP, provides instructions for setting up registry support for WSRP.

Chapter 6, Modifying Proxylet Rules, provides instructions on how to modify Proxylet Rules. These rules help the browser to identify the domains that needs to be routed through Proxylet.

Chapter 7, Creating a New Portal, provides instructions for creating a new empty portal and deploying sample content into an empty portal.

Chapter 8, Deploying Struts Application as a Portlet in Portal Server, provides instructions on how to deploy any existing struts application as a JSR168 portlet in Portal Server.

Chapter 9, Deploying JSF Application as a Portlet in Portal Server, provides instructions on how to deploy any existing JSF application as a JSR168 portlet in Portal Server.

Default Paths and File Names

The following table describes the default paths and file names used in this Configuration Guide.

TABLE P–1 Default Paths and File Names

Term	Description
PortalServer-base	Represents the base installation directory for a previous version of Portal Server. The software default base installation and product directory depends on your specific platform: Solaris™ systems `/opt`

TABLE P–1 Default Paths and File Names *(Continued)*

Term	Description
PortalServer7–base	Represents the base installation directory for this version of Portal Server. The software default base installation and product directory depends on your specific platform: Solaris™ systems `/opt`
AccessManager-base	Represents the base installation directory for Sun Java System Access Manager. The Access Manager default base installation and product directory depends on your specific platform: Solaris systems: `/opt/SUNWam`
DirectoryServer-base	Represents the base installation directory for Sun Java System Directory Server. Refer to the product documentation for the specific path name.
ApplicationServer-base	Represents the base installation directory for Sun Java System Application Server. Refer to the product documentation for the specific path name.
WebServer-base	Represents the base installation directory for Sun Java System Web Server. Refer to the product documentation for the specific path name.

Related Third-Party Web Site References

Third-party URLs are referenced in this document and provide additional, related information.

Note – Sun is not responsible for the availability of third-party web sites mentioned in this document. Sun does not endorse and is not responsible or liable for any content, advertising, products, or other materials that are available on or through such sites or resources. Sun will not be responsible or liable for any actual or alleged damage or loss caused or alleged to be caused by or in connection with use of or reliance on any such content, goods, or services that are available on or through such sites or resources.

Documentation, Support, and Training

The Sun web site provides information about the following additional resources:

- Documentation (`http://www.sun.com/documentation/`)
- Support (`http://www.sun.com/support/`)
- Training (`http://www.sun.com/training/`)

Typographic Conventions

The following table describes the typographic conventions that are used in this book.

TABLE P–2 Typographic Conventions

Typeface	Meaning	Example
`AaBbCc123`	The names of commands, files, and directories, and onscreen computer output	Edit your `.login` file. Use `ls -a` to list all files. `machine_name% you have mail.`
`AaBbCc123`	What you type, contrasted with onscreen computer output	`machine_name%` **`su`** `Password:`
aabbcc123	Placeholder: replace with a real name or value	The command to remove a file is `rm` *filename*.
AaBbCc123	Book titles, new terms, and terms to be emphasized	Read Chapter 6 in the *User's Guide*. A *cache* is a copy that is stored locally. Do *not* save the file. **Note:** Some emphasized items appear bold online.

Shell Prompts in Command Examples

The following table shows the default UNIX® system prompt and superuser prompt for the C shell, Bourne shell, and Korn shell.

TABLE P–3 Shell Prompts

Shell	Prompt
C shell	`machine_name%`
C shell for superuser	`machine_name#`
Bourne shell and Korn shell	`$`
Bourne shell and Korn shell for superuser	`#`

CHAPTER 1

Enabling Access to the Portal Server Through the Gateway

This chapter provides instructions on how you can enable access to thePortal Server through the Gateway.

Enabling Access to the Portal Server

▼ To Enable Access to the Portal Server

Steps

1. **Modify the following tokens in the *PortalServer7-base*/SUNWportal/export/request/enableSRAForPortal.xml file to suit your deployment.**

   `%INST_GWNAME%`
   Gateway Profile you are modifying

   `%FULLY_QUALIFIED_PORTAL_SERVER_URI%`
   Fully qualified portal URL

   `%PORTAL_SERVER_DOMAIN%`
   Domain in which the portal server resides

   `%DEPLOY_URI%`
   Deploy URL for the portal web application

2. **Save the file after making the changes.**

3. **Load the file into the directory server using the Sun Java System Access Manager's `amadmin` command as follows:**

   AccessManager-base`/bin/amadmin -u` *amadmin* `-w` *amadmin-pwd* `-t enableSRAForPortal.xml`

4. **Log in to the Portal Server administration console and navigate to Secure Remote Access —> Profiles —> default —> Core —> Basic Options — Portal Servers and remove `INST_PS_SERVER_LIST`.**

5. **Add `http://`*PS-HOST*`:`*PS-PORT* and restart the Gateway.**

CHAPTER 2

Enabling User Behavior Tracking

This chapters includes instructions for enabling User Behavior Tracking. For more information on this feature, see "User Behavior Tracking" in *Sun Java System Portal Server 7 Release Notes* in *Sun Java System Portal Server 7 Release Notes*.

Enabling User Behavior Tracking

▼ To Enable User Behavior Tracking

Steps

1. **Set `com.sun.portal.ubt.enable=true` in `/`*PortalData-Dir*`/portals/`*PortalID*`/config/UBTConfig.properties` file**
2. **Start accessing portal desktop and continue with normal operations on the channels.**
3. **Observe the running UBT logs in `/`*PortalData-Dir*`/portals/`*PortalID*`/logs/`*InstanceID*`/ubt.0.0.log` file.**

   Here:

PortalData-Dir	Refers to the portal data directory; for example, `/var/opt/SUNWps`
PortalID	Indicates the Portal ID; for example `myPortal`
InstanceID	Indicates the portal instance ID; for example `sprint_80`

   Use the UBT log file to run the `psadmin generate-ubt-report` command to get sample UBT reports.

CHAPTER 3

Setting Up Federated Search

The Federated Search feature enables users to submit a search query to multiple search engines concurrently and have the search results displayed in a unified format. The Federated Search feature provides a single interface for the user to post a search query to both a web meta-repository, such as google.com and an internal directory system such as a local personnel directory. The search results from these two different sites are presented to the user in a single web page.

Federated Search

▼ To Set Up Federated Search

Steps

1. **Create a Google account and download `googleapi.jar` from `http://www.google.com/apis/`**
2. **Obtain the license key for using `googleapi.jar`.**
3. **Set up sample federated databases:**
   a. **From a terminal window, log in to the host where search server is installed.**
   b. **Type the following:**
      ```
      cd /opt/SUNWps/sdk/search
      ```

c. **Modify the `sampledbs.soif` file to change google clientKey value to be your downloaded license key, and modify databaseurl, providerurl, rdmserverurl, and other url values, accordingly.**

Use the SOIF file syntax. The number in curly brackets ({ }) following the attribute is the number of characters you enter for that attribute's value.

d. **Type the following:**

```
cd /var/opt/SUNWps/searchservers/search-server

./run-cs-cli rdmgr -y root
/opt/SUNWps/sdk/search/sampledbs.soif
```

`./run-cs-cli rdmgr -y root -U` to verify that the soif entries containing the configurations for sample federated databases in the `sampledbs.soif` are in the root db.

4. **Add `googleapi.jar` to the web container's class path:**

On the Application Server:

```
cp google-api-install-directory/googleapi/googleapi.jar
/var/opt/SUNWappserver7/domains/domain1/server1/applications/j2ee-modules/search-se
```

On the Web Server:

```
cp /google-api-install-directory/googleapi/googleapi.jar
/opt/SUNWwbsvr/https-host.domain/webapps/https-host.domain/search-server/WEB-INF/lib
```

5. **Restart the web container.**

▼ To Test Federated Search

Steps

1. **Use the `rdmserver` front-end by:**

a. **Go to http://*host-name*.red.iplanet.com/*search-server*/testrdm.html**

b. **Select "rd-request" for Type, select "search" for "Query Language."**

c. **Enter a federated db such as "google" for "Database."**

d. **Enter a query such as "java" for "Scope".**

e. **Click "Submit GET."**

f. **Verify that search results are returned.**

2. **Use the Search channel by modifying Search channel JSPs to add federated databases to the database list and view attributes for federated search results.**

CHAPTER 4

Establishing Trust Between Two Cacao Agents

This chapter includes instructions for establishing trust between two cacao agents.

Establishing Trust Between Two Cacao Agents

With this release, any `psadmin` subcommand can be remotely executed. This means that `psadmin` command can be executed from portal on one machine to a portal on another machine. To do this:

1. Stop the cacao server on the second machine. To stop, type *PortalServer7-base*`/SUNWcacao/bin/cacaoadm stop`.
2. Copy the `/etc/opt/SUNWcacao/security` directory from the first machine to the other.
3. Start the cacao server on the second machine. To start, type *PortalServer7-base*`/SUNWcacao/bin/cacaoadm start`.

Verify this by running the `psadmin list-portals` command from the first machine.

CHAPTER 5

Setting Up Registry Support for WSRP

This chapter includes instructions for setting up registry support for WSRP.

Setting Up Registry Support

▼ To Set Up Registry Support

Steps

1. **Install and configure Sun Service Registry Server from Sun Java Enterprise System 4 before installing Portal Server and the stack components.**

   If portal server is on a different node, install just `SUNWsoar-sdk` from Sun Java Enterprise System 4 on the Portal Server host.

2. **Contact the Registry Server admin and obtain the credentials information registry to publish and access artifacts into the registry server. Also, obtain the publish and query URLs of this registry server.**

   If you happen to be the registry server admin, see the Registry Server guides on creating users and providing access to the registry server.

3. **Sun JES Registry Server uses client certificates to authenticate the registery server. Obtain the client certificate and create a keystore (JKS) and import the client certificate into the keystore.**

   You must create the keystore under the following directory`/soar/3.0/jaxr-ebxml/`on the Portal Server node.

4. **Log in to the Portal Server administration console and update the value of the`JES-Registry-Server` service in SSO Adapter, based on the above information.**

5. **Specify the keystore location as relative to `/soar/3.0/jaxr-ebxml/`.**

   For example, if you have created the keystore in `/soar/3.0/jaxr-ebxml/security/keystore.jks`, then specify the value of the keystore location as `/security/keystore.jks`

6. **Log in to the Access Manager administration console and grant SSO Adapter service to `amadmin`.**

Note – Make sure you have installed the following patches for registry functionality to work on the portal server node: 119189–04 (SPARC and x86) and 119190–04 (Linux).

CHAPTER 6

Modifying Proxylet Rules

Proxylet rules specify the domain and proxy settings in the Proxy Auto Configuration (PAC) file on the client machine. These rules help the browser to identify the domains that needs to be routed through Proxylet.

The default behavior of Proxylet is changed as follows:

When a user logs into the Portal desktop, the Proxylet channel contains a list of application URLs (much like the Netlet channel containing Netlet rules). When a user clicks on a link, Proxylet is launched if it is not already running. Once Proxylet is launched, the user is redirected to the application URL page. The Proxylet channel user interface contains controls to stop and start Proxylet. Clicking on the stop button on the user interface restores the proxy settings and stops the server. If a rule contains the string `proxylet-host:proxylet-port` as the proxy server, then the generated PAC file replaces the string with the host and port of Proxylet. You can make a rule so that all FTP traffic is routed through Netlet and all HTTP traffic is routed through Proxylet.

Modifying Proxylet Rules

▼ To Modify the Proxylet Rules

Step

- **Enter the proxy-host and proxy-port, using the following syntax:**
  Domain1`[`,*Domain2*,...,*n*`]`:*Host*:*Port*

  The following list describes the variables you use:

  domain Is any domain such as `sun.com` or your portal domain. Multiple domains are separated by a comma.

Host Specifies the proxy server used for this domain(s). To specify Proxylet as your proxy server, specify the string `proxylethost`.

Port Specifies the proxy server port. To specify Proxylet as your proxy server, specify the string `proxyletport`.

If a rule contains the string `proxylethost:proxyletport` as the proxy server, then the generated PAC file replaces the string with the host and port of Proxylet. You can make a rule so that all FTP traffic is routed through Netlet and all HTTP traffic is routed through Proxylet.

▼ To configure Proxylet for the enterprise domain

Step ● **Provide the portal domain as a part of Proxylet rules. For example, specify `your portal domain:proxylethost:proxyletport`.**

The generated PAC will provide instructions to route the portal domain through Proxylet.

▼ To configure Proxylet for specific applications

Step ● **Provide the specific application domain(s) as a part of Proxylet rule. For example, `enterprise application domain:proxylethost:proxyletport`.**

The generated PAC file uses `dnsDomainIs` Javascript function to compare the configured domain against the incoming domain. Administrators can also choose to provide their own PAC file through the Custom PAC file option instead of using Proxylet rules.

CHAPTER 7

Creating a New Portal

This chapter includes instructions for creating a new empty portal and deploying sample content into an empty portal.

Creating a New Portal

This sections contains the following:

▼ To Create a New Empty Portal

Steps

1. **Create a new web container instance.**
   For example, `second`.

2. **Copy**
   PortalServer7-base`/SUNWportal/template/Webcontainer.properties.JESWS6`
   to *PortalServer7-base*`/SUNWportal/bin/second.properites` **file.**

3. **Edit the following properties in the**
   PortalServer7-base`/SUNWportal/bin/second.properties` **file.**

   - `Host=`*host.domain*
   - `Port=`*port*
   - `WebContainerInstanceName=second`
   - `WebContainerInstanceDir=/opt/SUNWwbsvr/https-second`

4. **Run the following commands:**

   `psadmin create-portal -u` *amadmin* `-f` *ps_password* `-p Second --uri /portal -w second.properties`

5. **Restart the web container.**

6. **Verify that the new portal is created properly. To verify:**

   - Type `psadmin list-portals -u` *amadmin* `-f` *ps_password*
   - Login to Access Manager administration console to see new portal-centric services.
   - Access the new portal via the browser.

▼ To Deploy Sample Content (3 samples) on a New Portal

Steps

1. **Make a copy of the** *PortalServer7-base*`/SUNWportal/samples/portals/shared/input.properties` **file and edit the following properties in the file.**

   - `ps.config.location=/etc/opt/SUNWps`
   - `ps.portal.id=Second`
   - `ps.instance.id=host_port`

     Tip – You can find out the exact instance-ID from the output of `psadmin list-portals` command.

   - `ps.access.url=http://host.domain:port/portal` For example, `http://siroe.com:80/portal`
   - `ps.webapp.uri=/portal`
   - `ps.profiler.email=admin@siroe.com`
   - `ps.profiler.smtp.host=host.domain`
   - `search.access.url=http://host.domain:port/mySearch/search`
   - `search.id=search1`
   - `am.admin.dn=uid=amAdmin,ou=People,dc=siroe,dc=com`
   - `default.org.dn=dc=siroe,dc=com`

2. **Make a copy of** *PortalServer7-base*`/SUNWportal/samples/portals/shared/password.properties` **file and edit the following properties in the file to set proper passwords.**

- `amadminPassword=%AMADMIN_PASSWORD%`
- `amldapuserPassword=%AMLDAPUSER_PASSWORD%`
- `userManagementPassword=%USER_MANAGEMENT_PASSWORD%` (optional; can be ignored if you are not setting up the comm channels)

3. **Remove the following files before running the sample content configuration ant script:**
   - Directory:
     - `/var/opt/sun/portal/tmp/par` on Linux
     - `/var/opt/SUNWportal/tmp/par` on Solaris
   - Files:
     - `community_sample.par`
     - `developer_sample.par`
     - `enterprise_sample.par`
     - `welcome_sample.par`
4. **Change the order of targets in** *PortalServer7-base*`/SUNWportal/samples/portals/developer/build.xml` **file.**

   For example, change:

```
<target name="run"
            depends="config_am, config_portal, config_portlets,
            config_wsrp, par_create, par_import, config_authless,
            config_orgadmin, config_subscriptions, deploy"/>
```

   to

```
<target name="run"
            depends="config_am, config_portal, par_create,
            par_import, config_authless, config_orgadmin,
            config_subscriptions, config_portlets, deploy,
            config_wsrp_dp"/>
```

5. **Type** `/usr/sfw/bin/ant -buildfile` *PortalServer7-base*`/SUNWportal/samples/portals/build.xml`**.**

   You will be required to specify the location of the customized `input.properties` and `password.properties` file.

Note – To capture the output of the sample portal content configuration, specify a log file when invoking ant. For example, type `ant -buildfile` *PortalServer-base*`/SUNWps/samples/portals/build.xml -logfile /tmp/samplesinstall.txt`.

▼ To Deploy Sample Content (Any One Sample) on a New Portal

Steps

1. **Make a copy of the *PortalServer7-base*`/SUNWportal/samples/portals/shared/input.properties` file and edit the following properties in the file.**

   - `ps.config.location=/etc/opt/SUNWps`
   - `ps.portal.id=Second`
   - `ps.instance.id=host_port`

     Tip – You can find out the exact instance-id from the output of `psadmin list-portals` command.

   - `ps.access.url=http://host.domain:port/portal` For example, `http://siroe.com:80/portal`
   - `ps.webapp.uri=/portal`
   - `ps.profiler.email=admin@siroe.com`
   - `ps.profiler.smtp.host=host.domain`
   - `search.access.url=http://host.domain:port/mySearch/search`
   - `search.id=search1`
   - `am.admin.dn=uid=amAdmin,ou=People,dc=siroe,dc=com`
   - `default.org.dn=dc=siroe,dc=com`

2. **Make a copy of *PortalServer7-base*`/SUNWportal/samples/portals/shared/password.properties` file and edit the following properties in the file to set proper passwords.**

   - `amadminPassword=%AMADMIN_PASSWORD%`
   - `amldapuserPassword=%AMLDAPUSER_PASSWORD%`
   - `userManagementPassword=%USER_MANAGEMENT_PASSWORD%` (optional; can be ignored if you are not setting up the comm channels)

3. **Remove the following files before running the sample content configuration ant script:**

   - Directory:
     - `/var/opt/sun/portal/tmp/par` on Linux
     - `/var/opt/SUNWportal/tmp/par` on Solaris
   - Files:
     - `community_sample.par`

- `developer_sample.par`
- `enterprise_sample.par`
- `welcome_sample.par`

4. **Change the order of targets in** *PortalServer7-base*`/SUNWportal/samples/portals/developer/build.xml` **file.**

   For example, change:

```
<target name="run"
            depends="config_am, config_portal, config_portlets,
            config_wsrp, par_create, par_import,
            config_authless, config_orgadmin,
            config_subscriptions, deploy"/>
```

   to

```
<target name="run"
            depends="config_am, config_portal, par_create,
            par_import, config_authless, config_orgadmin,
            config_subscriptions, config_portlets,
            deploy, config_wsrp_dp"/>
```

5. **Type:**

   - **`/usr/sfw/bin/ant -buildfile`** *PortalServer7-base*`/SUNWportal/samples/portals/welcome/build.xml` to deploy the Welcome page content.
   - **`/usr/sfw/bin/ant -buildfile`** *PortalServer7-base*`/SUNWportal/samples/portals/developer/build.xml` to deploy the Developer Sample Portal content.
   - **`/usr/sfw/bin/ant -buildfile`** *PortalServer7-base*`/SUNWportal/samples/portals/enterprise/build.xml` to deploy the Enterprise Sample Portal content.

CHAPTER 8

Deploying Struts Application as a Portlet in Portal Server

This chapter describes how to deploy any existing struts application as a JSR168 portlet in Portal Server. Using the steps mentioned in this document, the entire Struts application can be displayed within a channel on the portal server desktop. It contains the following sections:

Preparing the Struts Application

This section contains the following:

Introduction

The extended struts framework shipped with Portal Server is an extension of Struts version 1.2.4. This requires that you must download Standard Struts binary, version 1.2.4, from the struts archive page, and the application must be tested as a standalone application, using standard `struts.jar` file. This is to ensure that you have the proper version of all the supporting JARs required by the struts framework.

Install Portal Server 7 for extended struts framework (`struts.jar` file) and supporting components (`strutssupport.jar`, `portlet.jar`) required to deploy Struts application as JSR168 portlet.

Modify Struts Application

To deploy any struts application as a portlet, the struts application is required to follow some of the following rules:

1. The Struts application must abide by the restrictions applicable to any application running inside the Portal Server. For example, the request parameters in the struts application can not use keywords reserved by the portal server. The list of reserved words include `"action"`, `"provider"`, `"targetprovider"`, `"containerName"`, `"last"`, `"page"`, `"error"`, `"container"`, `"selected"`, `"editChannelName"`, `"targetPortletChannel"`, and `"currentChannelMode"`.
2. All the forms and links must be created using struts tag library. Struts' tag library provide `<html:form>` and `<html:link>` for this purpose.
3. JSP and HTML must not have HTML title, body, frame and base tags. This is as per the PLT.B section of portlet specification. JSP must not use forward and/or redirect.

Obtain Portlet Objects in Struts Application

It is possible for struts application to get hold of portlet objects like `ActionRequest` and `ActionResponse`. This may be required, for example, to implement EDIT functionality. However, if portlet objects are not used properly, the use of portlet objects in struts application may make it portal dependent and result in the struts application unusable as a standalone application.

The struts `Action` class can obtain `javax.portlet.ActionRequest` and `javax.portlet.ActionResponse` objects using the following calls:

```
ActionRequest aReq = (ActionRequest) request.getAttribute("javax.portlet.request");
ActionResponse aRes = (ActionResponse) request.getAttribute("javax.portlet.response");
```

The above two statements return `javax.portlet.RenderRequest` and `javax.portlet.RenderResponse` respectively, when called from a JSP page.

Session Information

If any struts application, deployed as a portlet, is invalidating the session using `session.invalidate()`, the session obtained by the struts-portlet bridge becomes the invalid one. Because of this, the bridge is unable to store rendering related information. In application server, struts application, deployed as a portlet, must not use `session.invalidate()` as the same session is used by struts portlet bridge.

Creating and Modifying XML Files

This section contains the following:

Modify `struts-config.xml` File

Change the `RequestProcessor` to `org.apache.struts.action.PortletRequestProcessor` or `org.apache.struts.tiles.PortletTilesRequestProcessor`, if the application is using Tiles.

For example:

```
<controller
contentType="text/html;charset=UTF-8"
debug="3"
locale="true"
processorClass="org.apahce.struts.action.PortletRequestProcessor">
<!-- The "input" parameter on "action" elements is the name of a local or global "forward"
rather than a module-relative path -->
<set-property property="inputForward" value="true"/>
</controller>
```

Create `portlet.xml` File

Every portlet WAR must have one `portlet.xml` file in the `WEB-INF` directory of the web application. When creating the `portlet.xml` file, note that:

- The Portlet class must be `org.apache.struts.action.StrutsPortlet`.
- The `initPage` *init* parameter is mandatory and its value must be the welcome page of the struts application. This can be a direct reference to a JSP file (such as `/index.jsp`) or it can be a reference of Action Mapping Definition (such as `/welcome.do`).
- The *editPage init* parameter is not mandatory. If specified, portlet mode `EDIT` must also be specified in `<supports>` tag and vice-versa.
- The *helpPage init* parameter is not mandatory. If specified, portlet mode `HELP` must also be specified in `<supports>` tag and vice-versa. Note that the help page support is limited to a single page and it can not provide navigation to any other page within struts application.
- The *factoryName init* parameter is mandatory and must be set to `com.sun.portal.struts.wrapper.PSServletObjectsFactory`.

- All the init parameters associated with the `ActionServlet` as defined in `web.xml` file must also be configured as init parameter in `portlet.xml` file.
- The URL mapping used for `ActionServlet` as defined in `web.xml` file must be configured as an init parameter of the portlet.

Here is a sample `portlet.xml` file for struts-portlet application:

```
<?xml version="1.0" encoding="UTF-8"?>
<portlet-app xmlns="http://java.sun.com/xml/ns/portlet/portlet-app_1_0.xsd"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="http://java.sun.com/xml/ns/portlet/porlet-app_1_0.xsd" version="1.0

<portlet>
        <portlet-name>StrutsPortlet</portlet-name>
        <portlet-class>org.apache.struts.action.StrutsPortlet</portlet-class>
        <init-param>
            <name>initPage</name>
            <value>/index.jsp</value>
        </init-param>
        <init-param>
            <name>helpPage</name>
            <value>/tour.htm</value>
        </init-param>
        <init-param>
            <name>editPage</name>
            <value>/edit.jsp</value>
        </init-param>
        <init-param>
            <name>factoryPage</name>
            <value>com.sun.portal.struts.wrapper.PSServletObjectsFactory</value>
        </init-param>
        <init-param>
            <name>config</name>
            <value>/WEB-INF/struts-config.xml,/WEB-INF/struts-config-registration.xml</value>
        </init-param>
        <init-param>
            <name>servletPage</name>
            <value>*.do</value>
        </init-param>
            <expiration-cache>0</expiration-cache>
            <supports>
                <mime-type>text/html</mime-type>
                <portlet-mode>HELP</portlet-mode>
                <portlet-mode>EDIT</portlet-mode>
            </supports>
        <portlet-info>
            <title>StrutsPortlet</title>
        </portlet-info>
    </portlet>
</portlet-app>
```

Building and Deploying the Web Application as a Portlet Application

▼ To Deploy the Struts Application as a Portlet

Steps

1. **Replace standard `struts.jar` file, in the `WEB-INF/lib` directory, with the extended `struts.jar` file shipped Portal Server.**
2. **Add `strutssupport.jar` file and `portlet.jar` file shipped with Portal Server in the `WEB-INF/lib` directory.**
3. **Copy the newly created `portlet.xml` file and modified `struts-config.xml` file to the `WEB-INF` directory.**
4. **Create the `.war` file for the application.**
5. **Deploy the newly created WAR file using the `psadmin deploy-portlet` command. For example, type `./psadmin deploy-portlet -u` *amadmin* `-f` *passwordfile* `-p` *portalId* `-i` *portalinstance* `-g` *warfile*.**

CHAPTER 9

Deploying JSF Application as a Portlet in Portal Server

This chapter describes how to deploy any existing JSF application as a JSR168 portlet in Portal Server. Using the steps mentioned in this document, the entire JSF application can be displayed within a channel on the portal server desktop. It contains the following sections:

Overview

This section contains the following:

Introduction

A `jsf-portlet.jar` file is included with the Portal Server to enable communication between the Portal Server and JSF. This component allows the execution of commands (to render the information, to perform actions like EDIT, HELP) received from Portal Server and pass the data (such as user defined parameters, or user input in general) to the JSF-based web application and/or to Portal Server.

State Information and High Availability

The JSF application embedded in a JSF portlet can view all user interactions with the portal page that are outside the user interface for the JSF portlet itself as if they were page reloads. The JSF Portlet maintains whatever state information is needed by the JSF application as other pages are selected or while the user interacts with the portal in other ways. The JSF portlet leverages the portlet container HTTP session failover capabilities to enable highly available JSF applications within portlets.

Accessing Portlet APIs

Developers can access the portlet APIs from the `FacesContext` object as shown here:

```
FacesContext facesContext = FacesContext.getCurrentInstance();
PortletRequest pRequest = (PortletRequest)facesContext.getExternalContext().getRequest();
```

To find the portlet window state (like Maximize and Normalize), do the following:

```
WindowState windowState = pRequest.getWindowState();
```

Mapping Actions of JSF Application to Portal Application and Vice-Versa

TABLE 9–1 JSF to Portal Mapping

When the JSF Application	On the Portal
Execute any action in the application	The `processAction()` method of JSF Portlet is called which calls the `execute()` method of JSF Lifecycle. Then the `render()` method of JSF Portlet is called which calls the render method of JSF Lifecycle and returns content using `PortletRequestDispatcher.include()`.
Executing the action results in a error	The `processAction()` method of JSF Portlet is called which calls the `execute()` method of JSF Lifecycle. Then `render()` method of JSF Portlet is called which calls the `render()` method of JSF Lifecycle and returns the error message using `PortletRequestDispatcher.include()`.

TABLE 9–1 JSF to Portal Mapping *(Continued)*

When the JSF Application	On the Portal
The scope of the JSF Component or backing beans is request	When a JSF application is running in a servlet environment, the JSF request begins and ends within the scope of a servlet request (a user request). However, when a JSF application is running in a portlet environment, the JSF request lifecycle is split in two portlet requests. All JSF lifecycle phases but render happen during the portlet processAction request, with the JSF lifecycle render phase happening during the portlet render request.

TABLE 9–2 Portal to JSF Mapping

On a Portal	The JSF Application
Click on a JSF portlet Edit button	Edit page is displayed if portlet init parameter *com.sun.faces.portlet.INIT_EDIT* is set to the edit page; otherwise, a message indicating what needs to be done is displayed.
Click on a JSF portlet Help button	Help page is displayed if portlet init parameter *com.sun.faces.portlet.INIT_HELP* is set to the help page; otherwise, a message indicating what needs to be done is displayed.
Click on a JSF portlet Maximize button	The *render()* method of JSF Portlet is called which calls the render method of JSF Lifecycle and it returns content using `PortletRequestDispatcher.include()`. Therefore, the maximize window displays the same context which were shown on portlet window before clicking maximize button.
Click on a JSF portlet Minimize button	Request is handled at the portlet level and the JSF application remains unaware of it. The desktop displays the minimized window as it displays for any other JSR 168 portlet.
Click on a Normalize button of the minimized JSF portlet	The `render()` method of JSF Portlet is called which calls the render method of JSF Lifecycle and it returns content using `PortletRequestDispatcher.include()`. Therefore, the normalized window displays the same content which was shown on portlet window before clicking the minimize button.

TABLE 9–2 Portal to JSF Mapping *(Continued)*

On a Portal	The JSF Application
Click on a JSF portlet Detach button	The `render()` method of JSF Portlet is called which calls the render method of JSF Lifecycle and it returns content using `PortletRequestDispatcher.include()`. Portal server uses this content to display in a new window. Therefore, the detached window displays the same content which was shown on portlet window before clicking the detach button.
After detaching, click on a JSF portlet Attach button	The `render()` method of JSF Portlet is called which calls the render method of JSF Lifecycle and it returns content using `PortletRequestDispatcher.include()`. Portal server uses this content to display in the portlet window. The content shown is same as was shown in the detached window before clicking the attach button.
Remove the JSF portlet from a page and then add it again	On removal, the request is handled at the portal/portlet level and on adding it again, the `render()` method is called. The jsf-portlet window displays the same content which was shown on portlet window before removing this portlet (that is, the state is maintained). The remove and add have to occur in the same login session while the same `DesktopContext` object exists. For example, if the desktop session reap interval setting is set low enough (say 30 second), and you remove a JSF portlet, then wait 2 minutes and then add it again, the state will be lost.
Click reload of the portal page	The `render()` method of JSF Portlet is called which calls the render method of JSF Lifecycle and it returns content using `PortletRequestDispatcher.include()`. The jsf-portlet window displays the same content which was shown on portlet window before clicking the reload button.
Click on another tab and then click back on the tab that contains the JSF portlet	The `render()` method of JSF Portlet is called which calls the render method of JSF Lifecycle and it returns content using `PortletRequestDispatcher.include()`. The jsf-portlet window displays the same content which was shown on portlet window before clicking on other tab.

TABLE 9–2 Portal to JSF Mapping *(Continued)*

On a Portal	The JSF Application
Click on the Finish button of the edit page of some other channel, thereby causing a refresh of the portal page containing the JSF portlet	The `render()` method of JSF Portlet is called which calls the render method of JSF Lifecycle and it returns content using `PortletRequestDispatcher.include()`. Therefore, the jsf-portlet window displays the same content which was shown on portlet window before clicking the edit button.
Execute any action on some other channel, thereby causing the portal page containing JSF portlet to be refreshed	The `render()` method of JSF Portlet is called which calls the render method of JSF Lifecycle and it returns content using `PortletRequestDispatcher.include()`. The jsf-portlet window displays the same content which was shown on portlet window before executing any action on some other channel.

Converting JSF-based Applications to JSF Aware Portlets in Portal Server

▼ To Convert JSF-based Applications to Portlets

Steps

1. **Copy `jsf-portlet.jar` from *PortalServer7–base*`/lib` directory to `WEB-INF/lib` directory of the application.**
2. **Add a new deployment descriptor for the portlet by creating a `portlet.xml` file.**

   The `portlet.xml` file must placed in the `WEB-INF` directory of the application.
3. **In the `portlet.xml` file, set the portlet parameter *com.sun.faces.portlet.INIT_VIEW* to point to the first page of your portlet.**
4. **Modify the JSP pages as follows:**

   a. **Remove the `<html>`, `<head>`, and `<body>` tags.**

   b. **Modify use of forward and redirect as the new page will replace the existing portal pages.**

c. **Remove all the HTML tags and Javascript calls which are not allowed (as per JSR168 specification).**

5. **(Optional) Set the portlet parameter `com.sun.faces.portlet.INIT_EDIT` to point to the edit page of your portlet in the `portlet.xml` file to provide EDIT functionality for the JSF portlet.**

6. **(Optional) Set the portlet parameter `com.sun.faces.portlet.INIT_HELP` to point to the help page of your portlet in the `portlet.xml` file to provide HTLP functionality for the JSF portlet.**

7. **Deploy the WAR file using the `psadmin deploy-portlet` command. For example, type `psadmin deploy-portlet -u` *amadmin* `-f` *passwordfile* `-v -d` *dn* `-p` *portalID* `-i` *instanceID warfile*.**

8. **Create a new portlet channel and add it to the desired container.**