

# Chapter 22: Integrating Flex applications with portal servers

Using Adobe LiveCycle Data Services ES, you can configure Adobe Flex client applications as local portlets hosted on JBoss Portal, BEA WebLogic Portal, or IBM WebSphere Portal. The following versions of these products are tested:

- JBoss Portal 2.4 and 2.6
- Oracle BEA WebLogic Portal 8.1, 9.2, and 10.2
- IBM WebSphere Portal 5.1.0.1 and WAS 6.0

These portal servers implement the portlet specification: Java Specification Request (JSR) 168. You can also enable Flex client applications for consumption by a portlet consumer by using portal servers that support Web Services for Remote Portlets (WSRP). WSRP is an OASIS standard that defines a web service interface for accessing and interacting with interactive presentation-oriented web services.

The files required to use the portal servers are located in the *installation\_dir*\resources\wsrp directory where LiveCycle Data Services ES is installed.

## Contents

<a href="#">Using a Flex application on a portal server</a> .....	279
<a href="#">Deploying on a portal server</a> .....	283

## Using a Flex application on a portal server

You can configure Flex client applications as local portlets on a portal server that implements the Java portlet specification (JSR 168). You can also enable Flex client applications for consumption by a portlet consumer by using portal servers that support WSRP.

A portal server facilitates the development, deployment, operation, and presentation of aggregated content and applications that can be personalized for specific users. The Java portlet specification provides a standard way to develop user-facing web application components (portlets) for portal servers. The specification defines a common API and infrastructure that allows for product-agnostic portlets. The specification defines a portlet container contract, window states and portlet modes, management of portlet preferences, user information storage, packaging and deployment, security, and portlet-specific JSP tags. For more information about the portlet specification, see <http://jcp.org/en/jsr/detail?id=168>.

WSRP uses web services to provide a standard way for compliant portals to consume portlets hosted on other portals without any programming effort. A portal that hosts a portlet is called a producer portal. A portal that consumes a portlet using WSRP is called a consumer portal. For more information about WSRP, see the OASIS WSRP page at [http://www.oasis-open.org/committees/tc\\_home.php?wg\\_abbrev=wsrp](http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsrp).

## Enabling a Flex application as a portlet

You enable a Flex application as a portlet on a local portal server by creating a portlet definition that uses the portlet class, `flex.portal.GenericFlexPortlet`. The `GenericFlexPortlet` class handles all WSRP requests and returns an HTML-snippet wrapped SWF file for the requested portlet. When using a portal that supports WSRP, the WSRP implementation sends the snippet back in a WSRP supported SOAP message. LiveCycle Data Services ES also provides three default JSP pages, one for each portlet view mode: view, edit, and help. The `GenericFlexPortlet` class uses these JSP pages by default to satisfy requests for corresponding view modes of a portlet. You can customize the look of each view mode by specifying a custom JSP page for any of the view modes.

Each `portlet-name` value in a portlet definition must be unique. The code in the following example `portlet.xml` portal deployment descriptor file makes a Flex application available as a local portlet:

```
<portlet>
  <portlet-name>CRMPortlet</portlet-name>
  <portlet-class>flex.portal.GenericFlexPortlet</portlet-class>
  <supports>
    <mime-type>text/html</mime-type>
    <portlet-mode>VIEW</portlet-mode>
    <portlet-mode>EDIT</portlet-mode>
  </supports>
  <portlet-info>
    <title>CRM Portlet</title>
  </portlet-info>
  <portlet-preferences>
    <preference>
      <name>full_app_uri</name>
      <value>/samples/dataservice/crm/mini.swf</value>
      <read-only>true</read-only>
    </preference>
  </portlet-preferences>
</portlet>
```

To expose the portlet for WSRP consumption, add the following preference element to the `portlet-preferences` section of the `portlet.xml` file. This setting enables requests from SWF files to be proxied through the consumer server in accordance with the WSRP specification.

```
<preference>
  <name>channel_uri</name>
  <value>/messagebroker/amfpolling</value>
  <read-only>true</read-only>
</preference>
```

To expose the portlet for WSRP consumption in portal servers other than WebLogic Portal, you usually include a `remotable` element in the portlet definition in the portlet-server-specific configuration file and set its value to `true`, as the following example shows:

```
<portlet-app>
  <portlet>
    <remotable>true</remotable>
    <portlet-name>CRMPortlet</portlet-name>
  </portlet>
</portlet-app>
```

The `remotable` element enables the Flex application for WSRP consumption by any consumer portal server that has specified the WSDL of your server in its WSRP configuration file.

You must set the `full_app_uri` preference in the `portlet.xml` file. Additionally, to proxy requests through the consumer server in accordance with the WSRP specification, you must set the `channel_uri` preference in the `portlet.xml` file. When the `channel_uri` value is present, the `WSRP_ENCODED_CHANNEL` variable containing a correctly formatted channel URL, is passed to the Flex application via the FlashVars parameter in the HTML that the `portlet-view.jsp` file generates. All traffic from the portlet's Flex application is proxied through the consumer server. Note that this increases the HTTP load on this server if you use a polling or streaming channel.

The following preferences are supported by all Flex application portlets:

Preferences	Description
<code>full_app_uri</code>	(Required) Maps this portlet to a Flex application. Can be a relative URL or a fully qualified location.
<code>channel_uri</code>	(Optional) Relative path (starting after the context root) to the channel between the Flex application and the LiveCycle Data Services ES server. This should be the path of the first channel used by the destinations. Only a single channel is supported for producer/consumer portlets, so failover is not supported.  If you do not set the <code>channel_uri</code> preference, a Flex portlet can still be consumed directly from the producer server, but it will not be proxied through the consumer server in accordance with the WSRP specification.
<code>view_jsp</code>	Specifies the JSP file to be used when returning markup for the VIEW view mode. Default value is <code>/wsrp_jsp/portlet-view.jsp</code> .
<code>edit_jsp</code>	Specifies the JSP file to be used when returning markup for the EDIT view mode. Default value is <code>/wsrp_jsp/portlet-edit.jsp</code> .
<code>help_jsp</code>	Specifies the JSP file to be used when returning markup for the HELP view mode. Default value is <code>/wsrp_jsp/portlet-help.jsp</code> .
<code>normal_width</code>	Specifies the width of the embedded SWF file when the portlet is in the NORMAL window state. Default value is 300.
<code>normal_height</code>	Specifies the height of the embedded SWF file when the portlet is in the NORMAL window state. Default value is 300.
<code>max_width</code>	Specifies the width of the embedded SWF file when the portlet is in the MAXIMIZED window state. Default value is 500.
<code>max_height</code>	Specifies the height of the embedded SWF file when the portlet is in the MAXIMIZED window state. Default value is 500.
<code>markup_cache_secs</code>	Specifies the time in seconds during which the consumer caches the markup for this particular portlet. The default behavior is to do no caching of the HTML markup.

In addition to these preferences, you can define custom preferences. Custom preferences are ignored by the default JSP files, but can be used in custom JSP pages. Custom preferences must include a `cust` prefix.

If you copy the `wsrp-jsp` directory, which you copy from the `installation_dir\resources\wsrp` directory of the LiveCycle Data Services ES installation, to a location other than directly below the context root of your web application, you must set the optional `wsrp_folder` initialization parameter to that location. If you set the `wsrp_folder` initialization parameter, you also must set initialization parameters for each of the JSP pages. The following example shows the `wsrp_folder` initialization parameter and an initialization parameter for one of the JSP pages, the `portlet-view.jsp` page:

```
<portlet>
  <portlet-name>CRMPortlet</portlet-name>
  <portlet-class>flex.portal.GenericFlexPortlet</portlet-class>
  <supports>
    <mime-type>text/html</mime-type>
    <portlet-mode>VIEW</portlet-mode>
    <portlet-mode>EDIT</portlet-mode>
  </supports>
```

```
<init-param>
  <name>wsrp_folder</name>
  <value>/flex/support</value>
</init-param>
<init-param>
  <name>view_jsp</name>
  <value>/flex/support/wsrp-jsp/portlet-view.jsp</value>
...
</portlet>
```

### View modes

You can view a portlet in one of three modes, VIEW, EDIT, and HELP. By default, the GenericFlexPortlet uses a specific template JSP page corresponding to each of these modes.

#### Default view modes

The default VIEW mode JSP page simply wraps the reference to the specified SWF file in an `<object>` tag, which sets the width and height to the values specified in the configuration file for this portlet. The default EDIT mode JSP page lets users modify any of the non-read-only preferences and view the values of the read-only preferences. The default HELP mode JSP page displays basic help text. An administrator can specify custom JSP pages for any view mode by setting configuration preferences in the portlet metadata file.

If the producer and consumer servers do not both support the Portlet Management interface, all preferences appear as read-only. For a remote portlet being consumed through WSRP to have modifiable preferences, the consumer must be able to clone the producer portlet. To do so, the implementations of WSRP that the consumer and producer use must implement the optional Portlet Management interface. If one of the implementations does not, all preferences of a remote portlet appear as read-only.

#### Custom view modes

Portal servers can also support custom view modes. You enable specific portlets for these custom view modes by setting configuration parameters in the portlet deployment descriptor (portlet.xml) file. Because the set of supported custom view modes for a particular installation is not known ahead of time, the GenericFlexPortlet class and default JSP pages do not have any logic for custom view modes. To customize wrappers based on custom view modes, you use custom JSP pages. For details on custom view modes, see the JSR 286 portal specification at <http://jcp.org/en/jsr/detail?id=286>.

### Window states

An incoming markup request from a portlet consumer contains information that indicates whether the portlet in context is MAXIMIZED, MINIMIZED, or NORMAL. This lets the portlet server adjust the markup based on the size of the portlet window. When the GenericFlexPortlet processes a markup request for a portlet that is in the MINIMIZED state, it does not return a SWF file; there is no reason to cause extra network traffic if the user cannot interact with the Flex application. In this case, the GenericFlexPortlet returns text notifying the user that the Flex application is not available in MINIMIZED state. When configuring the portlet for a Flex application, the administrator can specify the width and height of the embedded SWF by using the portlet preferences previously described.

You might require an application to appear differently between MAXIMIZED and NORMAL window states. For example, it might make more sense for a particular panel or widget to be hidden in the NORMAL state and displayed in the MAXIMIZED state. To support such customization, a PORTLET\_WS flashvar, which contains the current window state, is returned in SWF file URLs. You can access the value of this variable in your MXML or ActionScript code as `Application.application.parameters.PORTLET_WS`. Its value corresponds to one of the window states defined in the `javax.portlet.WindowState` class: `normal`, `maximized`, or `minimized`.

Adaptability to a window state requires that you design a Flex application with portlets in mind, and you use the PORTLET\_WS variable in Flex application logic.

**EOLAS support**

From a February 2006 upgrade to Microsoft Internet Explorer forward, SWF files are not immediately interactive in Internet Explorer unless you specify them through the object or the `<embed>` tag by using JavaScript in a file external to the main file. If an object tag specifies a SWF file inline in an HTML file, the user sees a Click To Activate message when hovering over the Flex application. To avoid this, the default view JSP page uses a JavaScript library to create the wrappers around Flex applications. The required JavaScript library, `AC_OETags.js`, is located in the same directory as the three default JSP pages. If you use your own JSP pages outside of the default directory, you must copy the JavaScript library to the correct directory.

**Flex Ajax Bridge and Ajax Client Library for LiveCycle Data Services ES**

You can use the Flex Ajax Bridge and the Ajax client library for LiveCycle Data Services ES in the portlet-generated JSP pages as long as the necessary parts of `FDMSLib.js`, `FABridge.js` and `FDMSBridge.as` are included in the JSP page. Other non-Flex-based portlets running in the same portal page can communicate with Flex portlets by using the Flex Ajax Bridge and subscribing to the same destinations as the Flex portlets.

Because JavaScript libraries are loaded from the producer server in separate requests from the portlet markup, the `<script src=>` tags must specify the full URL of the library on the producer server and use the `renderResponse.encodeURL()` function to convert the URL to proxy through a consumer server if necessary. See the `wsrp-jsp/portlet-view.jsp` file for examples of using the portlet `encodeURL()` API.

*Note: There is a general issue with using JavaScript with WSRP. To avoid object name collisions, the consumer server modifies the names of all HTML objects in the markup that the producer server returns. The same modification does not occur for JavaScript libraries loaded from the producer server. Thus any references from JavaScript libraries to HTML objects in the markup most likely fail on the portal page hosted by the consumer server unless you use URL rewriting between the producer and consumer servers. In this case, you can use URL rewriting to force the consumer to go through the loaded JavaScript library and modify HTML object name references just as it did for the markup itself. To use URL rewriting, follow the guidelines in section 10 of the WSRP specification.*

**Logging**

The `GenericFlexPortlet` writes log messages under the category `WSRP.General`.

**Caching of producer portlets**

When the producer server returns markup to the consumer for a particular portlet, it can specify a cache markup time in seconds. If the consumer server has to get the portlet markup for an identical portal page URL during this time, it returns the markup from cache and does not communicate with the producer. The `markup_cache_secs` preference lets you customize the time. For this property to be meaningful, you must configure the consumer and producer servers to support markup caching in general. If you do not specify the `markup_cache_secs` preference, there is no markup caching.

**Deploying on a portal server**

You can deploy your Flex applications on portal servers that have a full LiveCycle Data Services ES web application installed, and on portal servers that do not have a LiveCycle Data Services ES web application but host a set of precompiled SWF files.

## Deploying with a LiveCycle Data Services ES web application

When your portal contains a Flex application that is in a LiveCycle Data Services ES web application, you can use standard LiveCycle Data Services ES debug and error logging based on the debug level that you set in the `services-config.xml` file.

- 1 Copy `flex-portal.jar` from the `installation_dir\resources\wsrp\lib` directory to the `WEB-INF\lib` directory of your web application.
- 2 Copy the `wsrp-jsp` directory from the `installation_dir\resources\wsrp` directory to the root of your web application.
- 3 Follow the portal-server-specific steps that follow to create portlets for your Flex applications.

## Deploying without a LiveCycle Data Services ES web application

You can create a portlet that uses a Flex application that is not part of a LiveCycle Data Services ES web application. However, you cannot use standard LiveCycle Data Services ES logging because that functionality is inside the LiveCycle Data Services ES web application.

- 1 Copy `flex-portal.jar` and `flex-messaging-common.jar` from `installation_dir\resources\wsrp\lib` directory to the `WEB-INF\lib` directory of your web application.
- 2 Copy the `wsrp-jsp` directory from `installation_dir\resources\wsrp` directory to the root of your web application.
- 3 Follow the portal server specific steps that follow to create portlets for your Flex applications.

## Deploying on JBoss Portal

When deploying on JBoss Portal, consult the relevant JBoss documentation, including the following:

- General JBoss WSRP information
- JBoss administration portlet information

### Configure a JBoss Portal producer server

- 1 To create a local portlet, modify the `portlet.xml` file in the `WEB-INF` directory of your web application based on the information in [“Using a Flex application on a portal server” on page 279](#).
- 2 To create a local instance of the new portlet, either modify `portlet-instances.xml` in the `WEB-INF` directory of your web application, or use the administration portlet at `http://server:port/admin`.
- 3 To enable your portlet for WSRP consumption, update `jboss-portlet.xml` in the `WEB-INF` directory of your web application (setting the `remoteable` flag to `true`), or use the administration portlet `http://server:port/admin`, and select the `Remoteable` option for your portlet.

The WSDL for your producer server should be  
`http://server:port/portal-wsrp/MarkupService?wsdl`.

Depending on how your DNS is set up, it has a server name that is accessible only from within your local network (server name is computer name) or a server name that is accessible from outside the local network (server name is fully qualified or is an IP address). If the `SERVER` name in the generated WSDL does not appear correctly, you can adjust your DNS settings or manually modify the `.wsdl` file, for example:

```
/server/default/data/wsdl/portal-wsrp.war/wsrp_services.wsdl
```

### Configure a JBoss Portal consumer server

- 1 Add the WSDL of the producer server (does not have to be a JBoss server) to `jboss-portal.sar\portal-wsrp.sar\default-wsrp.xml`.
- 2 Use the administration portlet at `http://server:port/admin` to get a list of available remote portlets.
- 3 Create instances of any remote portlets that you are interested in.

## Deploying on WebLogic Portal

When deploying on WebLogic Portal, consult the relevant WebLogic documentation at <http://e-docs.bea.com>.

### Configure a producer server

Create a local portlet by using the WebLogic Workspace Studio (or WebLogic Workshop in earlier versions) and completing the following steps:

- 1 Create a Portal Web Project.
- 2 Create a portlet (New > Portlet).
- 3 After naming the portlet, select Java Portlet as the portlet type.
- 4 Ensure that you set Class Name to `flex.portal.GenericFlexPortlet`.
- 5 Modify `WEB-INF/portlet.xml` based on the information in “[Enabling a Flex application as a portlet](#)” on [page 280](#). Make sure you add the `full_app_uri` preference.
- 6 Ensure that the `portlet-name` attribute in the `portlet.xml` file matches the `definitionLabel` property in the `.portlet` file.
- 7 Create a `.portlet` file for your portlet.

### Considerations for producer servers

Consider the following when configuring a WebLogic Portal producer server:

- The URL of the WebLogic portal administration user interface is `http://server:port/context-root/Admin/portal.portal`.
- WebLogic supports quite a few different portlet types, as described in the *WebLogic Portlet Development Guide*. Look specifically at “Java Portlets (JSR 168).”
- To use the EDIT mode of a portlet, your users must have sufficient permission access to the `PreferencePersistenceManager EJB`.
- WebLogic portlets by default are enabled as WSRP producers. To disable a portlet from being WSRP consumable, set its `Offer as Remote` property to `false` in the WebLogic Workspace Studio.
- The producer server documentation describes the steps necessary to enable a project as a WSRP producer in the WebLogic Workspace IDE. The WSDL generated for your producer server should be `http://server:port/context-root/producer?wsdl`.

Depending on your DNS setup, the WSDL either has a server name that is accessible only from within your local network (server name is computer name) or a server name that is accessible from outside the local network (server name is fully qualified or is an IP address). If the `SERVER` name in the generated WSDL does not appear correctly, you can adjust your DNS settings or manually modify the `wsrp-wsdl-full.wsdl` file that is generated for your application.

After setting up your portlet on the producer server, you can use the Portal Management section of the portal administration user interface (`http://server:port/context-root/Admin/portal.portal`) or WebLogic Workspace Studio to include your Flex application portlet in portal pages.

### Configure a consumer server

Complete the following steps to add the WSDL of the remote producer server to a WebLogic consumer server. The producer server does not have to be a WebLogic server.

- 1 Go to portal administration application `http://server:port/context-root/Admin/portal.portal`.
- 2 Go to Portal Management.
- 3 Browse to Library > Remote Producers.
- 4 Create an entry for the remote producer.

Your remote portlet appears under the Portlets tab. You can add remote portlets to pages by navigating to the page under the Portals tab.

### Deploying on WebSphere Portal 5.1 or WAS 6.0

When deploying a portlet on WebSphere Portal 5.1 or WAS 6.0, consult the relevant WebSphere Portal documentation.

Consider the following when deploying on WebSphere Portal:

- The default URL of the WebSphere Application Server is `http://SERVER:9080`.
- The default URL of the WebSphere Portal administration application is `http://SERVER:9081/wps/portal`.
- The default URL of the WebSphere Application Server administration application is `http://SERVER:9060/ibm/console/`.
- Unlike JBoss Portal and WebLogic Portal, WebSphere Portal is a completely separate server with a different port and context root than the application server.

### Configure a producer server

- 1 Deploy the WAR file that contains your application logic and portlet deployment configuration (`portlet.xml`) on the application server using the application server administration application. You should be able to test the non-portlet functionality of your application by starting the application server.
- 2 Add the same WAR file to the portal server's module list:
  - a Log in to the portal server administration application with `admin/admin` as the username/password.
  - b Select the Administration tab in the upper-right corner.
  - c Select the Portlet Management tab from the list on the left.
  - d Select Web Modules, and install your WAR file. This adds every portlet configured in your `portlet.xml` file to the list of portlets available from the portal server.
  - e Verify that your portlets appear under the Portlets tab on the left.

Now you can create a local portal page with your new portlets.

To make a portlet available as a WSRP producer, click the star icon next to its name under the Portlets tab. A check mark appears in the Provided column for that portlet.

The WSDL of your producer should be `http://SERVER:PRODUCER_PORT/wps/wsd/wsrp_service.wsd`

### Configure a consumer server

Complete the following steps to add the WSDL of the remote producer server to the consumer server. The producer server does not have to be a WebSphere server.

- 1 Select Portlet Management tab on the left side under the Administration tab.

- 2** Select Web Services.
- 3** Click the New Producer button to add the producer WSDL.
- 4** Under the Web Modules tab, look for an entry with the name WSRP Producer (name you gave to remote producer) Application.
- 5** Navigate to the portlets listed under this application
- 6** Make copies of any portlets to consume.

Your remote portlets appear under the Portlets tab, and you can add them to your pages.