BEA AquaLogic® .NET Portlet Toolkit

WSRP Development Guide

Version 1.1 MP1
Revised: June 1, 2008

Contents

BEA Documentation and Resources

The following documentation and resources are available from BEA.

Resource	Description
Installation Guide	This guide describes the prerequisites (such as required software) and procedures for installing and upgrading .NET Application Accelerator components. It is available on *edocs.bea.com/alui/dotnetappaccelerator/docs11*.
Release Notes	The release notes provide information about new features, issues addressed, and known issues in the release. They are available on *edocs.bea.com/alui/dotnetappaccelerator/docs11*.
Development Guides	The development guides provide detailed information on authoring portlets using the .NET Application Accelerator. Separate guides are provided for AquaLogic Interaction (ALI) and WebLogic Portal (WLP). They are available on *edocs.bea.com/alui/dotnetappaccelerator/docs11*.
Additional Developer Guides, Articles, API Documentation,	These resources are provided for developers on the BEA dev2dev site (*dev2dev.bea.com*). They describe how to build custom applications using AquaLogic User Interaction and how to customize AquaLogic User Interaction products and features.

Resource	Description
Blogs, Newsgroups, and Sample Code	
AquaLogic User Interaction (ALUI) and AquaLogic Business Process Management (ALBPM) Support Center	The ALUI and ALBPM Support Center is a comprehensive repository for technical information on ALUI and ALBPM products. From the Support Center, you can access products and documentation, search knowledge base articles, read the latest news and information, participate in a support community, get training, and find tools to meet most of your ALUI and ALBPM-related needs. The Support Center encompasses the following communities: **Technical Support** Submit online service requests, check the status of your requests, search the knowledge base, access documentation, and download maintenance packs and hotfixes. **User Group** Participate in user groups; view webinars, presentations, the CustomerConnection newsletter, and the Upcoming Events calendar. **Product Center** Download product updates, maintenance packs, and patches; view the Product Interoperability matrix (supported third-party products and interoperability between products). **Developer Center** Download developer tools, view code samples, access technical articles, and participate in discussions. **Education Services** Review the available education options, then choose courses by role and delivery method (Live Studio, Public Classroom Training, Remote Classroom, Private Training, or Self-Paced eLearning). **Profile Center** Manage your implementation details, local user accounts, subscriptions, and more.

Resource	Description
	If you do not see the Support Center when you log in to *one.bea.com/support*, contact *ALUISupport@bea.com* or *ALBPMSupport@bea.com*for the appropriate access privileges.
Technical Support	If you cannot resolve an issue using the above resources, BEA Technical Support is happy to assist. Our staff is available 24 hours a day, 7 days a week to handle all your technical support needs. E-mail: *ALUISupport@bea.com* or *ALBPMSupport@bea.com* Phone Numbers: USA, Canada +1 866.262.7586 or +1 415.263.1696 EMEA +44 1494 559127 Asia Pacific +61 2.9931.7822 Australia/NZ +61 2.9923.4030 Singapore +1 800.1811.202

Authoring WSRP Portlets

Authoring WSRP Portlets Using the .NET Portlet Toolkit

The AquaLogic .NET Portlet Toolkit provides ease-of-use features for authoring WSRP portlets. The .NET Portlet Toolkit integrates with Microsoft Visual Studio 2005 providing pre-configured project templates, a portlet item template, and a class library that provides easy access to portlet properties, user properties, user profile information, registration information, and more.

WSRP (Web Services for Remote Portlets) is a W3C standard for consuming one or more remote markup "Producers" from a markup "Consumer". The AquaLogic .NET Application Accelerator includes a WSRP Producer for ASP.NET that can be used to produce portlets for any WSRP Consumer, including WebLogic Portal.

Creating a WSRP Portlet Using the .NET Portlet Toolkit

To create a new WSRP portlet, use the Microsoft Visual Studio 2005 templates provided with the .NET Portlet Toolkit. The templates are pre-configured to include references to required BEA assemblies, including the .NET Portlet API.

A new portlet project includes a simple portlet page in the Default.aspx/Default.aspx.cs files. This page includes the namespaces necessary for accessing the WSRP Portlet API classes such as `PortletPropertyAttribute` and the `WSRPPortletContext`.

Follow the steps below to create a new portlet using the `WSRPPortletPage` template.

1. Create a new WSRP portlet project in Microsoft Visual Studio 2005. Note: ALI Portlet and Preference Page templates cannot be used for WSRP portlets.
   a) To create a new Web application project, follow the instructions below:
      1. Click **File ➤ New Project**
      2. Under **Project types**, select Visual C#.
      3. In the **My Templates** list, select WSRP Portlet Web Application.
      4. Enter a name for the project.
   b) To create a new Web Site project, follow the instructions below:
      1. Click **File ➤ New ➤ Web Site** .
      2. Select the language option: C#.
      3. In the **My Templates** list, select WSRP Portlet Project. (ALI Portlet and Preference Page templates cannot be used for WSRP portlets.)
      4. Enter a name for the project.
2. Create a new portlet page:
   a) In the **Solution Explorer**, right-click on the root of the project and select Add New Item….
   b) In the **My Templates** list, select WSRP Portlet Page.
   c) Save the page with an intuitive name (for example, “HelloWorld.aspx”).
3. Add the necessary functionality to the portlet page:
   - *Manipulating WSRP Properties Using the .NET Portlet Toolkit* on page 11
   - *Accessing User Profile Information Using the .NET Portlet Toolkit* on page 13
   - *Implementing WSRP Portlet Modes Using the .NET Portlet Toolkit* on page 15
   - *Implementing WSRP Portlet Window States Using the .NET Portlet Toolkit* on page 18
   - *Managing SSO Authentication with WebLogic Portal* on page 28

   If the portlet should support CSS styling applied by the WSRP Consumer , you must use the CSS class names defined in the WSRP specification.
4. Register the portlet with the WSRP Producer. For details, see *Registering a Portlet with the WSRP Producer* on page 38.

5. Deploy the portlet in WLP. For details, see*Consuming a .NET WSRP Portlet in WebLogic Portal* on page 27.

Manipulating WSRP Properties Using the .NET Portlet Toolkit

To use WSRP properties in a portlet, configure the properties in the WSRP Producer and use the .NET Portlet API. To access and define properties, you can use a property attribute or a property collection.

1. To define properties for use in a portlet, add entries for each property to the `<portlet-property>` element within the `<portlet>` element in the wsrp-producer.xml configuration file. For a full list of configuration elements, see *WSRP Producer Configuration Elements (wsrp-producer.xml)* on page 38.

```
<portlet>
...
 <portlet-properties>
  <property name="stockSymbols" type="xs:string"
defaultValue="BEAS"/>
 </portlet-properties>
...
</portlet>
```

2. Access the property using the .NET Portlet API. You can use a variety of methods to access and manipulate properties.

   - The **property attribute** provides type specific read-write binding of an WSRP property to a page member variable (field). The property attribute can be bound to fields with types of String, int, float, bool, or DateTime. Values for fields marked with a property attribute are set during the Load event of the Page; any changes are persisted during the EndRequest event of the HttpApplication.

     `PortletPropertyAttribute` has two optional properties:

     - Key: A string that provides the key name of the property to which the member variable is bound. If the property is defined for the portlet in the wsrp-producer.xml configuration file, the key value must match the value of the name attribute for the property. If this value is not defined, the key defaults to the name of the member variable on which it is defined. Key values must be unique within a page.
     - DefaultValue: A string that provides the value to use when the property is unavailable or has not yet been set. If this value is not provided, it will default to a value based upon the type of the variable as follows:

Property Type	Default value if not assigned
string	String.Empty
int, long	0
double, float	0.0
bool	false
DateTime	DateTime.MinValue

Note: You must include one of the following access modifiers on all local fields that are bound to a property using a property attribute: [protected | internal | protected internal | public]. The modifier "private" is not supported. Since "private" is the default modifier if none is specified, the modifier must not be omitted.

```
[PortletProperty(Key = "stockSymbols")]
protected string stockSymbols;

[PortletProperty(DefaultValue = "orange")]
protected String colorSelection;
```

- The **property collection** allows you to access all of the property values in the request in dictionary format. The `WSRP.PortletPropertiesCollection` is a true read-write collection that can be shared across pages that use Server.Transfer.

  In this example, a property collection is used to update the property value so it can be passed to another page. This sample code is taken from the Stock Quote Portlet example included with the .NET Portlet Toolkit.

```
public partial class StockQuotePortlet2_Edit :
System.Web.UI.Page
{
 string StockSymbols
 {
  get { return
WSRPPortletContext.Current.Portlet.Properties["stockSymbols"];
 }
  set {
WSRPPortletContext.Current.Portlet.Properties["stockSymbols"]
 = value; }
 }

    protected void Page_Load(object sender, EventArgs e)
```

```
    {
        // set the stockSymbols textbox to the current
property value
        if (!Page.IsPostBack)
   stockSymbolsTextBox.Text = this.StockSymbols;
    }

 protected void SaveButton_Click(object sender, EventArgs e)

    {
        // update the portlet property
  this.StockSymbols = stockSymbolsTextBox.Text;
  // set the mode back to view and transfer back to the view
 page
      WSRPPortletContext.Current.Portlet.Mode = "wsrp:view";

  // Note: since we set the property value using the
Properties collection
  // it will be available on the destination page.
  Server.Transfer("View.aspx");
    }
}
```

For a full description of the API, see the complete class documentation for the WSRP section of the Portlet API, installed into the Microsoft Visual Studio 2005 Combined Help Collection.

Accessing User Profile Information Using the .NET Portlet Toolkit

To access user profile information using the .NET Portlet Toolkit, add the properties to the Web.config file and reference them by name in the portlet page.

ASP.NET 2.0 introduced the Provider model as a design pattern for plugging data providers into the framework. The .NET Portlet Toolkit Portlet API uses the ProfilerProvider API to expose user profile information sent by a WSRP Consumer. For more information about the ProfileProvider model, see the *MSDN documentation*.

The Web.config file for a .NET portlet project contains a `<profile>` element that defines the user profile properties supported by a Provider. In order for user profile properties from the source application to be accessible from the ASP.NET Profile object, they must be registered in the Web.config file.

Note: User Profile information is read-only and can only be modified through the source application.

1. For each custom property set or user profile grouping, create a `<group>` element in the `<properties>` element within the `<profile>` element in the Web.config file for the portlet project. Add each user profile property to its respective group. Each property must be defined with a name and a type. If you provide a default value, it will be used if the property is not available.

   Note: The name of the property **must** match the name sent by the source application. For more information about sending user profile properties from WebLogic Portal, see *Developing User Profiles* in the WLP documentation.

   ```
   <properties>
    <group name="homeInfo">
     <group name="online">
      <add name="email" type="string"/>
     </group>
    </group>
    <group name="CustomProperties ">
     <add name="title" type="string" defaultValue="No Title"/>
    </group>
    <group name="MyProfile">
     <add name="Name" type="String" defaultValue="Guest"/>
     <add name="Age" type="Int32"/>
    </group>
   </properties>
   ```

2. Reference the property in the portlet page by name using the Profile object.

   ```
   string title = Profile.CustomProperties.Title
   string name = Profile.MyProfile.Name;
   int age = Profile.MyProfile.Age;
   ```

3. Configure the .NET WSRP Producer to request user profile properties from the WSRP Consumer. Add a `<user-profile>` element to the end of the corresponding `<portlet>` element and configure the property names that should be passed to the WSRP Producer as shown in the example below. If you are using WLP, the properties from a property set can be filtered individually using a string with the format `<propertyset-name>/<propertyname>`. Using "*" for the `<propertyname>` will

request all properties from a property set from WLP. (If you are using another WSRP Consumer, consult the related documentation for the correct syntax for user profile properties.)

```
<user-profile>
  <item>CustomProperties/Title</item>
  <item>MyProfile/*</item>
</user-profile>
```

Implementing WSRP Portlet Modes Using the .NET Portlet Toolkit

To access or update portlet mode for a WSRP portlet, use the `PortletInfo.Mode` property in the .NET Portlet API.

WSRP portlets can expose functionality in different modes, with each mode catering to a particular function. Many WSRP Consumers provide functionality to let users request portlet markup in various modes. There are 4 "standard" modes defined in the WSRP 1.0 specification:

- wsrp:view
- wsrp:edit
- wsrp:help
- wsrp:preview

WSRP-compliant portlets must support the wsrp:view mode. Producers can also support custom modes, defined as URIs.

Two steps are required to implement portlet modes:

1. Configure the supported portlet modes in the wsrp-producer.xml file.

   The supported portlet modes for each portlet are specified in the wsrp-producer.xml configuration file as children of the `<supports>` element as shown in the example below. For a custom mode, the description may either be included in-line in the `<supports>` element or as part of a top-level `<custom-portlet-mode>` definition.

   Each supported mode must have an associated URL. Either include the URL for the mode within the `<url>` node, or use the `idref` attribute to refer to a URL defined as a child of the `<portlet>` element. If neither an URL nor the idref is specified, the URL is assumed to be the first one found for the portlet.

   If the `<supports>` element contains multiple mime-type elements, the portlet mode support applies to each of the mime-types included. To specify different portlet modes on a per

mime-type basis, add multiple <supports> elements to the portlet definition, each with a different mime-type. If the same mime-type is listed more than once, an exception is thrown.

Note: If the <portlet-mode> element is omitted, the portlet is assumed to support the wsrp:view portlet mode. If wsrp:* is specified, it implies all of the standard wsrp: values. Support for "wsrp:view" is always assumed, even if it is not explicitly indicated.

```
<portlet>
 ...
 <url
id="default">http://localhost:4614/WSRPSamples/StockQuotePortlet.aspx</url>

 ...
 <supports>
  <mime-type>text/html</mime-type>
  <portlet-mode>
   <name>wsrp:view</name>
   <url idref="default"/>
  </portlet-mode>

  <portlet-mode>
   <name>wsrp:edit</name>
   <url idref="default"/>
  </portlet-mode>

  <portlet-mode>
   <name>wsrp:help</name>
   <url idref="default"/>
  </portlet-mode>

  <portlet-mode>
   <name>wsrp:preview</name>

<url>http://localhost:4614/WSRPSamples/StockQuotePortlet_preview.aspx</url>

  </portlet-mode>

  <portlet-mode>
   <name>urn-myproject-portletmodes:search</name>
    <url>http://myhost/myportlet/search.aspx</url>
    <description lang="en">shows the search view for the
portlet</window-state>
  </portlet-mode>
  ...
 </supports>
```

```
...
<custom-items>
 <custom-portlet-mode>
  <name>urn-myproject-portletmodes:search</name>
  <description lang="en">shows the search view for the
portlet</description>
 </custom-portlet-mode>
</custom-items>
...
</portlet>
```

2. Implement the portlet mode in the portlet code.

   A portlet can access the portlet mode via the `PortletInfo.Mode` property during any portlet request, but may only change the current mode during a `performBlockingInteraction` (typically, an ASP.NET form postback). The new portlet mode is returned in the response to the WSRP Consumer. If the WSRP Consumer does not understand the mode returned by a portlet, it can ignore it. If the WSRP Consumer requests a mode that the portlet does not understand, the WSRP Producer will map to wsrp:view.

   The example below toggles the display of panels based on the portlet mode. This code is taken from the StockQuotePortlet.aspx.cs sample included with the .NET Application Accelerator 1.1 MP1.

```
...
protected void Page_PreRender(object sender, EventArgs e)
    {
        if (WSRPPortletContext.Current.Portlet.Mode ==
"wsrp:edit")
        {
            editPanel.Visible = true;
            helpPanel.Visible = false;
            // set the stockSymbols textbox to the current
property value
            stockSymbolsTextBox.Text = stockSymbols;
            viewPanel.Visible = false;
        }
        else if (WSRPPortletContext.Current.Portlet.Mode ==
"wsrp:help")
        {
            editPanel.Visible = false;
            helpPanel.Visible = true;
            viewPanel.Visible = false;
        }
        else // view mode
```

```
            {
                editPanel.Visible = false;
                helpPanel.Visible = false;
                viewPanel.Visible = true;
                // display the quotes
                DisplayQuotes(stockSymbols);
            }
        }
...
```

Implementing WSRP Portlet Window States Using the .NET Portlet Toolkit

To access or update window state for a WSRP portlet, use the `PortletInfo.WindowState` property in the .NET Portlet API.

Window state defines how much markup a portlet should generate. There are 4 "standard" window states defined in the WSRP 1.0 specification:

- wsrp:normal
- wsrp:minimized
- wsrp:maximized
- wsrp:solo

In WLP and ALI, these window states are implemented by changing the screen real estate provided to the portlet. All WSRP-compliant portlets must support the wsrp:normal window state. To support additional window states, configure the portlet as described below. Producers can also support "custom" window states, defined as a URI.

Two steps are required to implement multiple window states in a portlet:

1. Configure the supported window states in the wsrp-producer.xml file.

   The supported window states for each portlet are specified in the wsrp-producer.xml configuration file as children of the `<supports>` element as shown in the example below. For a custom window state, the description may either be included in-line in the `<supports>` element or as part of a top-level `<custom-window-state>` definition.

   If the `<supports>` element contains multiple mime-type elements, the window state support applies to each of the mime-types included. To specify different window states on a per

mime-type basis, add multiple <supports> elements to the portlet definition, each with a different mime-type. If the same mime-type is listed more than once, an exception is thrown.

Note: If the <window-state> element is omitted, the portlet is assumed to support all standard WSRP window states. If wsrp:* is specified, it implies all of the standard wsrp: values. Support for "wsrp:normal" is always assumed, even if it is not explicitly indicated.

```
<supports>
   <mime-type>text/html</mime-type>
    <window-state><name>wsrp:normal</name></window-state>
    <window-state><name>wsrp:solo</name></window-state>
    <window-state><name>wsrp:maximized</name></window-state>
    <window-state><name>wsrp:minimized</name></window-state>

    <window-state>
      <name>urn-myproject-windowstates:header</name>
      <description lang="en">shows just the header for the
portlet</window-state>
    </window-state>

   ...
</supports>

<custom-items>
 <custom-window-state>
  <name>urn-myproject-windowstates:header</name>
  <description lang="en">shows just the header for the
portlet</description>
 </custom-window-state>
</custom-items>
```

2. Implement the window state in the portlet code.

   A portlet can access the window state via the `PortletInfo.WindowState` property during any portlet request, but may only change the current window state during a `performBlockingInteraction` (typically, an ASP.NET form postback). The new window state is returned in the response to the WSRP Consumer. If the WSRP Consumer does not understand the window state returned by a portlet, it can ignore it. If the WSRP Consumer requests a window state that the portlet does not understand, the WSRP Producer will map to wsrp:normal.

```
protected void returnButton_Click(object sender, EventArgs e)
{
 //set the window state back to wsrp:view
 WSRPPortletContext.Current.Portlet.WindowState = "wsrp:view";
```

```
}

protected void Page_PreRender(object sender, EventArgs e)
{
 PortletInfo portlet = WSRPPortletContext.Current.Portlet;
  if (portlet.WindowState == "wsrp:maximize" ||
portlet.WindowState == "wsrp:solo")
  {
   maximizeButton.Visible = false;
   returnButton.Visible = true;
  }
  else if (portlet.WindowState == "wsrp:view")
  {
   maximizeButton.Visible = true;
   returnButton.Visible = false;
  }
}
```

Accessing WSRP Consumer Registration Information Using the .NET Portlet Toolkit

To identify a specific WSRP Consumer and access registration information, use the `RegistrationInfo` class in the .NET Portlet API.

By requiring registration, a WSRP Producer can request additional information about a WSRP Consumer which can be used to tailor the available portlets and resources. WSRP Consumer registration information is available to WSRP portlets through the `RegistrationInfo` class in the .NET Portlet API.

1. Configure the wsrp-producer.xml file to require registration, and define any registration properties. (The ConsumerAgent, ConsumerName and Handle are available by default.)

```
<?xml version="1.0"?>
<wsrp-producer
xmlns="http://www.bea.com/al/dotnet/wsrpproducer/1.1"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns:xs="http://www.w3.org/2001/XMLSchema" >

 <requires-registration>true</requires-registration>

 <registration-properties>
  <property name="ConsumerClass" type="xs:string">
```

```
    <label lang="en">Consumer Class</label>
    <hint lang="en" >the type of consumer</hint>
   </property>
 </registration-properties>
...
</wsrp-producer>
```

2. Access the registration information using the `RegistrationInfo` class in the .NET Portlet API. You can also use `RegistrationPropertyAttribute` to bind a registration property to a page member variable (field).as shown in the example below.

   The `RegistrationPropertyAttribute` preference attribute has two optional properties:

   - Key: A string that provides the key name of the property to which the member variable is bound. This value must match the value of the name attribute for the registration property defined in the wsrp-producer.xml configuration file. If this value is not defined, the key defaults to the name of the member variable on which it is defined. If the key value does not match a registration property defined in the wsrp-producer.xml file, the member variable will use the default value. Key values must be unique within a page.
   - DefaultValue: A string that provides the value to use when the preference is unavailable or has not yet been set. If this value is not provided, it will default to a value based upon the type of the variable as follows:

Property Type	Default value if not assigned
string	String.Empty
int, long	0
double, float	0.0
bool	false
DateTime	DateTime.MinValue

```
public partial class RegistrationPage : System.Web.UI.Page
{
    protected RegistrationInfo registration =
WSRPPortletContext.Current.ConsumerRegistration;

    [RegistrationProperty(Key = "doubleProperty", DefaultValue
 = 32)]
    protected double doubleProp;

    protected void Page_Load(object sender, EventArgs e)
```

```
    {
        registrationTable.Rows.Add(getPrefRow("ConsumerAgent",
 registration.ConsumerAgent));
        registrationTable.Rows.Add(getPrefRow("ConsumerName",
registration.ConsumerName));
        registrationTable.Rows.Add(getPrefRow("Handle",
registration.Handle));
        foreach (KeyValuePair<string, string> kvp in
registration.Properties)
            registrationTable.Rows.Add(getPrefRow("regprop:" +
 kvp.Key, kvp.Value));


registrationTable.Rows.Add(getPrefRow("RegistrationProperty:doubleProperty",
 doubleProp.ToString()));
    }

...
```

Localizing WSRP Portlet Metadata

To provide localized portlet metadata, configure the WSRP Producer to use localized strings and provide the necessary resource files.

The .NET Application Accelerator supports standard ASP.NET mechanisms for localizing portlet markup. In addition, you can localize WSRP portlet metadata through the wsrp-producer.xml file. Localizable elements are those derived from the schema type “localizableStringType”. These are:

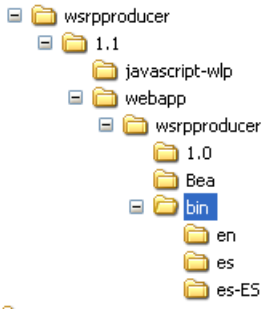
- portlet/description
- portlet/display-name
- portlet/portlet-info/title
- portlet/portlet-info/short-title
- property/label
- property/hint
- portlet/supports/window-state/description
- portlet/supports/portlet-mode/description
- custom-portlet-mode/description
- custom-window-state/description

For a complete list of elements in the wsrp-producer.xml file, see *WSRP Producer Configuration Elements (wsrp-producer.xml)* on page 38.

1. Configure the portlet metadata in wsrp-producer.xml to use “key” attributes to reference strings. You can include a default language and value as shown in the example below..

```
<portlet>
...
 <display-name lang="en" key="my-display-name">My
Portlet</display-name>
 <description lang="en" key="my-description">Displays
data</description>
...
</portlet>
```

2. Create a subdirectory for each supported locale in the directory containing the WSRP Producer assembly. For example, if locales “es”, “es-ES”, and “en” are supported, you would use the following structure:



3. Create resource files that define localized strings for all supported languages and place them in the appropriate locale-specific directory. Resource files must be named according to the filename convention {resource-class}.{locale}.resources. for example myportlet.es.resources and wsrp-producer.es.resources.

   - For all resource strings that are declared as a descendent of a `<portlet>` element, the resource-class name is the value of the portlet handle.
   - For all other resource strings, the resource-class name is “wsrp-producer”.

   Note: .NET resource files must be created using the resgen.exe tool (included in Visual Studio and the .NET SDK).

Logging .NET Application Accelerator Activity

To view logs of .NET Application Accelerator activity, use ALI Logging Spy.

ALI Logging Spy is automatically installed with ALI or installed on a remote server as part of the ALI Logging Utilities, a stand-alone package available for download on bea.dev2dev. ALI Logging Spy listens for log messages transmitted using network broadcast. By configuring a list of message senders, you can listen in on log traffic from multiple BEA applications in a single console.

The .NET Application Accelerator includes four logging message senders:

Logging Message Sender	Description
.NET WSRP Producer	The stand-alone web application designed to serve as a WSRP producer for WLP. This name is not configurable and transmits all messages related to the WSRP Producer. The WSRP producer "loggingName" entry in Web.config is not currently used.
ALIPortletProject	Transmits log messages from the portlet ASP.NET web application. The name of this message sender is defined in the Web.config file of any ASP.NET portlet project built using the .NET Portlet Toolkit. The name of the sender defaults to the project name entered when the project was created, but can be changed by modifying the value attribute of the following entry: `<add key="ptedk.LoggingApplicationName" value="ALIPortletProject"/>`
BEA Portlet API	Transmits log messages from the .NET Portlet API.
ALI .NET Application Accelerator	Reserved for future use. No logging traffic is sent by this sender.

To add a .NET Application Accelerator message sender to your Spy configuration, follow the instructions below.

1. Open ALI Logging Spy.
2. Select **View ➤ Set Filters** .

3. In the **Filter Settings** dialog, click **Edit ➤ Add Message Sender** and choose a message sender (see the table above).

For more information on configuring ALI Logging Spy, see *ALI Logging Utilities* in the ALI 6.0 development documentation.

Configuring WSRP Portlet in WebLogic Portal

Consuming a .NET WSRP Portlet in WebLogic Portal

To consume remote portlet resources in WebLogic Portal (WLP), you must create a Remote Portlet using either the IDE tools (described here) or WLP online administration tools. Once created, Remote Portlets (.portlet files) can be added to a WebLogic Portal portal description (.portal file).

To create the Remote Portlet using the WebLogic Portal IDE tools, follow the steps below. For detailed instructions on using these tools, see the WLP documentation.

Note: In order to produce portlets, the WSRP Producer must be installed and correctly configured within the local IIS instance. You must know the address of the WSRP Producer's WSRPService.wsdl file. The URL will vary depending on how IIS is configured, but it should be similar to the following: http://{wsrpproducer hostname:port}/wsrpproducer1.1/1.0/WSRPService.wsdl. To ensure that the WSRP Producer is running, paste this URL into the address bar of a web browser.

For detailed instructions on creating a WSRP portlet in WebLogic Portal, see *Creating Remote Portlets, Pages and Books* in the WLP documentation.

WebLogic Portal supports WSRP Preferences, but in order to use preferences, the .portal file describing a portal desktop in WebLogic Portal must be converted from "file" mode to "streaming" mode. For details, see *Creating a Desktop* in the WebLogic Portal documentation. Also, in order to modify preferences, a user must be logged into WebLogic Portal. See the WebLogic Portal documentation for additional information on configuring login.

Managing SSO Authentication with WebLogic Portal

To manage single-sign on (SSO) authentication with WebLogic Portal (WLP), configure both WLP and the .NET Application Accelerator to use either SAML or UNT authentication.

WebLogic Portal (WLP) and the WSRP Producer can be configured to work together to automatically authenticate to the remote ASP.NET portlet using credentials determined from the currently logged in user. Because the user only needs to login once, this is often referred to as “Single Sign On” (SSO). Without this feature, each portlet that wishes to authenticate users must manage its own authentication using forms-based mechanisms. This can lead to multiple logins during an end-user interaction with WLP. Using this functionality, portlets can use Windows (IIS), ASP.NET Forms, or SAML authentication, all without requiring the user to complete a separate authentication for the portlet. Once the user logs in to WLP they will be able to access secure remote ASP.NET portlets for which they are authorized.

Using SAML Token Authentication with WebLogic Portal

To implement single-sign on (SSO) authentication with WebLogic Portal using SAML tokens, you must configure the WLP WSRP Consumer, the WSRP Producer, and the remote portlet application.

WebLogic Portal can be configured to send SAML assertions over WSRP. The SAML token is passed directly to the Remote Portlet Application as part of the HTTP headers. The SAML token can be accessed through HTTP Request Headers (`Request.Headers["SAMLToken"]`). The remote portlet host handles authentication using the Custom HttpModule "SAMLAuthenticationModule" and sets the user principal name for the request so that user gain access to remote portlets.

1. Add the SAML security policy declaration to the WSRPService.wsdl and wsrp_v1_bindings.wsdl files.
   a) Open the wsdl file for the WSRP Producer: \wsrpproducer\1.1\webapp\wsrpproducer\1.0\WSRPService.wsdl.
   b) If it is not already present, add the following policy declaration to the WSRPService.wsdl file as a child of the root element <wsdl:definitions> and before the <wsdl:service> element. (The WSRPService.wsdl file installed with the WSRP Producer includes the SAML policy by default.)

```
<wsp:Policy s1:Id="SAMLAuth.xml"
xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy"
xmlns:s1="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">

 <wssp:Identity>
  <wssp:SupportedTokens>
   <wssp:SecurityToken
TokenType="http://docs.oasis-open.org/wss/2004/01/oasis-2004-01-saml-token-profile-1.0#SAMLAssertionID">

    <wssp:Claims>

<wssp:ConfirmationMethod>sender-vouches</wssp:ConfirmationMethod>

    </wssp:Claims>
   </wssp:SecurityToken>
  </wssp:SupportedTokens>
 </wssp:Identity>
</wsp:Policy>
```

   c) Open the wsdl bindings file for the WSRP Producer: \wsrpproducer\1.1\webapp\wsrpproducer\1.0\wsrp_v1_bindings.wsdl.
   d) Find the <wsdl:input> elements with the names “getMarkup” and “performBlockingInteraction”. If not already present, add the <Policy> element shown below. (The bindings file installed with the WSRP Producer includes this code within comments; to enable the code, remove the comment tags and make sure the URI attribute matches the Id value of the SAML policy in the WSRPService.wsdl file..) The complete xml should look as follows:

```
<wsdl:input name="getMarkup">
 <soap:body use="literal"/>
  <wsp:Policy
xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy">
   <wsp:PolicyReference URI="#SAMLAuth.xml"/>
  </wsp:Policy>
</wsdl:input>
```

```
<wsdl:input name=" performBlockingInteraction ">
 <soap:body use="literal"/>
  <wsp:Policy
xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy">
   <wsp:PolicyReference URI="#SAMLAuth.xml"/>
  </wsp:Policy>
</wsdl:input>
```

2. Add a `<securityTokenManager>` element to the WSRP Producer's Web.config file as shown below.

```
<microsoft.web.services2>
 <security>
  <securityTokenManager
   type="Bea.BasicNoAuthSAMLTokenManager, WSRPService"
   xmlns="urn:oasis:names:tc:SAML:1.0:assertion"
   qname="Assertion"
   />
 </security>
</microsoft.web.services2>
```

3. Generate a SAML credential certificate and configure the WLP WSRP Consumer to use the generated key. For instructions, see *Establishing WSRP Security with SAML* and *Configuring Single Sign-On with Web Browsers and HTTP Clients* in the WLP documentation. You must also configure the Relying Party properties as described below.
   a) On the Management tab, click **Relying Parties**.
   b) In the Relying Parties table, double click on **rp_00001**.
   c) Ensure that Sign Assertions and Include Keyinfo are checked .
   d) Click **Save**.
4. Configure the remote portlet application to verify the SAML token.
   a) Import the certificate used to generate the SAML token to **Local machine store – Enterprise Trusted certificates** on the machine that hosts the remote portlet application.
   b) Configure the Web.config file of the ASP.NET application to use the SAML authenticator.
      1. Under the `system.web` element, update the authentication node to use “None”.

```
<authentication mode="None" />
```

2. Configure the SAMLAuthenitcationModule in the `httpModule` section of `system.web` as follows:

```
<httpModules>
 <add name="SAMLAuthentication"
type="BEA.Portlet.Authentication.SAMLAuthenticationModule,
 SAMLAuth"/>
</httpModules>
```

Using UNT Authentication with WebLogic Portal

To implement single-sign on (SSO) with WebLogic Portal using UNT (User Name Token) authentication, you must configure the WSRP Producer.

UNT authentication can be used with IIS (Windows) authentication or ASP.NET Forms Authentication. The steps below cover both options.

Caution: Using UNT in the manner described below results in passwords being sent between the WSRP Consumer and the WSRP Producer in plain text. Ensure that the Consumer-Producer channel is secured by https before using this approach for transmitting a security token.

1. Enable authentication for your remote ASP.NET portlet. For more information on configuring ASP.NET authentication, consult MSDN and the following resources: *http://msdn2.microsoft.com/en-us/library/eeyk640h%28VS.80%29.aspx*, *http://msdn2.microsoft.com/en-us/library/ms978378.aspx*, *http://support.microsoft.com/kb/324274*.
2. Add the UNT security policy declaration to the WSRPService.wsdl and wsrp_v1_bindings.wsdl files.
   a) Open the wsdl file for the WSRP Producer: \wsrpproducer\1.1\webapp\wsrpproducer\1.0\WSRPService.wsdl.
   b) If it is not already present, add the following policy declaration to the WSRPService.wsdl file as a child of the root element (`<wsdl:definitions>`) and before the `<wsdl:service>` element. (The WSRPService.wsdl file installed with the WSRP Producer includes the UNT policy by default.)

```
<wsp:Policy s1:Id="UNTAuth.xml"
xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy"
xmlns:s1="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">

 <wssp:Identity
xmlns:wssp="http://www.bea.com/wls90/security/policy">
```

```
  <wssp:SupportedTokens>
   <wssp:SecurityToken
TokenType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0#UsernameToken">

    <wssp:UsePassword
Type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0#PasswordText"/>

   </wssp:SecurityToken>
  </wssp:SupportedTokens>
 </wssp:Identity>
</wsp:Policy>
```

c) Open the wsdl bindings file for the WSRP Producer: \wsrpproducer\1.1\webapp\wsrpproducer\1.0\wsrp_v1_bindings.wsdl.

d) Find the `<wsdl:input>` elements with the names "getMarkup" and "performBlockingInteraction". If not already present, add the `<Policy>` element shown below. (The bindings file installed with the WSRP Producer includes this code within comments; to enable the code, remove the comment tags.) The complete xml should look as follows:

```
<wsdl:input name="getMarkup">
 <soap:body use="literal"/>
  <wsp:Policy
xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy">
   <wsp:PolicyReference URI="#UNTAuth.xml"/>
  </wsp:Policy>
</wsdl:input>

<wsdl:input name=" performBlockingInteraction ">
 <soap:body use="literal"/>
  <wsp:Policy
xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy">
   <wsp:PolicyReference URI="#UNTAuth.xml"/>
  </wsp:Policy>
</wsdl:input>
```

3. Map each WLP user to a user that can access the ASP.NET portlet application. For instructions, see *Configuring User Name Token Security: Configuring the Consumer* in the WLP Federated Portals guide.
4. If you are using ASP.NET Forms Authentication, you must provide information about the login form to the WSRP Producer. In wsrp-producer.xml, add a `<forms-authentication>` entry as the last child element of the `<portlet>` element for each portlet that uses ASP.NET Forms Authentication. Each entry should include the following elements, as shown in the

example that follows. For a full list of portlet configuration elements, see *WSRP Producer Configuration Elements (wsrp-producer.xml)* on page 38.

Note: If you omit the `<forms-authentication>` element, the WSRP Producer will still try to authenticate to one of the IIS forms of authentication (Basic, Windows Integrated, or even Digest) if the ASP.NET is using one of those and the UNT is provided, but it will be unable to authenticate to ASP.NET Forms.

Element	Description
`<IsSSOConfigured>`	Whether or not SSO through ASP.NET Forms Authentication is enabled or not (true or false).
`<loginpage>`	The name of the login page configured for ASP.NET Forms Authentication in the remote portlet application.
`<username-field-name>`	The name of the login form field that references the user name.
`<password-field-name>`	The name of the login form field that references the password.
`<login-button-name>`	The name of the submit button in the login form.
`<login-button-value>`	The value of the submit button in the login form.
`<always-use-anonymous-login>`	Whether or not to use anonymous login (true or false). If true, you must provide `<anonymous-username>` and `<anonymous-password>` elements.

```
<portlet>
...
<forms-authentication>
 <IsSSOConfigured>true</IsSSOConfigured>
 <loginpage>login.aspx</loginpage>
 <username-field-name>txtUserName</username-field-name>
 <password-field-name>txtPassword</password-field-name>
 <login-button-name>cmdSubmit</login-button-name>
 <login-button-value>Submit</login-button-value>
 <always-use-anonymous-login>true</always-use-anonymous-login>

  <anonymous-username>userid</anonymous-username>
  <anonymous-password>pwd</anonymous-password>
</forms-authentication>
</portlet>
```

Note: The WSRP specification treats external resources used by a portlet application different from the portlet markup. An external resource is anything that is referenced by the portlet markup but not contained in the markup, such as externally referenced javascript files, images, or CSS style sheets. Portlet markup is retrieved from and proxied by the WSRP Producer; in the process the Producer can negotiate the authentication requirements of portlet applications. However, WSRP resources are retrieved directly by the WSRP Consumer without assistance from the WSRP Producer. As a result, external resources generally should not require authentication when using the WSRP Producer unless you can configure your WSRP Consumer to directly authenticate. If you are using Windows (IIS) authentication, you must move all external resources to a virtual directory or separate server that is configured to not require authentication in IIS. Take note that many ASP.NET controls and components use the WebResources feature to dynamically emit external javascript and image references in your markup. To make sure that WebResources references do not require authentication, ensure that any virtual paths from which these references originate do not require authentication.

Debugging WSRP Portlets

To debug WSRP portlets, use WebLogic Portal logging tools and check the list of common configuration problems.

If you encounter trouble accessing the WSRP Producer from the WebLogic Portal IDE or online administrative tools, attempt to access the WSRPService.wsdl using a web browser at this address: http://<IP-address>:<port-number>/wsrpproducer/1.0/WSRPService.wsdl If errors are returned, correct them; if no errors are returned, switch to the WLP Eclipse tools and examine the error log.

WebLogic Portal also provides a SOAP monitor to track SOAP traffic between a WSRP Consumer and Producer. For detailed instructions on accessing the SOAP monitor, see *Using the Monitor Servlet* When using the SOAP monitor, it is helpful to create a new .portal file with a single portlet so you can view the traffic between the Consumer and a single Producer.

The following list provides solutions to common configuration problems.

- **Preferences don't work over WSRP in WebLogic Portal**: Ensure that the WebLogic .portal file has been converted to streaming mode. For details, see *Managing Portal Desktops* in the WebLogic Portal documentation.
- **Images do not appear in WSRP portlets**: The WebLogic Portal WSRP Consumer allows access to resources hosted in remote web applications acting as WSRP Producers. The AquaLogic Interaction .NET Application Accelerator's WSRP Producer is a web application

that is separate from the ASP.NET web sites being consumed. WebLogic Portal must be configured to allow URL-addressable content such as images, style sheets, and JavaScript includes to be returned to WLP and users' web browsers. To do this, a `ResourceConnectionFilter` must be added to the WebLogic Portal applications as shown in the example below. This code sample is for WebLogic Portal 9.2.

```
<code>
package local;
import com.bea.wsrp.consumer.resource.ResourceConnectionFilter;
public final class AllowAllResourceConnectionFilter
 implements ResourceConnectionFilter
 {
  /**
      * Accept all resource URIs.
      * @return always returns <code>true</code>
      */
         public boolean allowedURL(String resourceURI) {
         return true;
         }
 }
</code>
```

This class is registered in WLP by adding an entry in the appropriate location in the WEB-INF/web.xml file. For example, the following code registers the permissive `ResourceConnectionFilter` implemented above.

```
<!--  WLP 9.2 ResourceProxyServlet -->
<servlet>

<servlet-name>com.bea.wsrp.consumer.resource.ResourceProxyServlet</servlet-name>


<servlet-class>com.bea.wsrp.consumer.resource.ResourceProxyServlet</servlet-class>

  <init-param>
   <param-name>resourceConnectionFilter</param-name>

<param-value>local.AllowAllResourceConnectionFilter</param-value>

 </init-param>
</servlet>
 <servlet-mapping>

<servlet-name>com.bea.wsrp.consumer.resource.ResourceProxyServlet</servlet-name>
```

```
    <url-pattern>/resource/*</url-pattern >
  </servlet-mapping>
```

Note: This code is intended as an example and is not suitable for production. In production environments, we recommend that applications constrain allowable URLs to those that are known to produce remote ASP.NET web sites. For more information, see the *WLP 9.2 API documentation*. For details on configuring a `ResourceConnectionFilter` in WLP 8.1, see *Establishing WSRP Security*

- **Portlet styles and themes are not displayed**: Some portals strictly enforce the *HTML 4.01 DTD*. Portlets that use deprecated HTML might not be displayed correctly. Confirm the DTD used by the target portal and design your portlets accordingly.

Configuring the WSRP Producer

Configuring WSRP Producer Service URLs (WSRPService.wsdl)

To update global WSRP Producer configuration settings after installation, update the WSRPService.wsdl file. Additional configuration of the WSRP Producer web site can be made via IIS administrative tools.

During installation, the WSRP Producer web site is deployed to IIS based on the configuration settings provided to the installer. This configuration information is used to parameterize the WSRP Producer's WSDL file, which describes the locations of various WSRP web services. After installation has completed, any changes to the IP address, port number, or web site deployment path must be made to the WSRPService.wsdl file (located under \wsrpproducer\1.1\webapp\wsrpproducer\1.0\) If these settings are incorrect, the WSRP Producer will not be able to locate WSRP web services.

To update configuration settings, edit WSRPService.wsdl in a text editor and change the necessary entries. For example, to update the location of the WSRP Producer, change the following entry:

```
<soap:address
location="http://192.168.123.456:8888/wsrpproducer1.1/1.0/WSRPBaseService.asmx"/>
```

To configure portlets in the WSRP Producer, use the wsrp-producer.xml configuration file. For details, see *WSRP Producer Configuration Elements (wsrp-producer.xml)* on page 38.

Registering a Portlet with the WSRP Producer

To deploy a portlet in the WSRP Producer, you must enter the URLs of the portlet content and additional metadata.

All the portlets available from the WSRP Producer must be registered in the wsrp-producer.xml file at the root of the WSRP Producer's web site. To add a portlet, edit the wsrp-producer.xml file in a text editor and add a `<portlet>` entry for each portlet within the `<portlets>` element. Once this step is complete, the portlet can be consumed by a WSRP Consumer. For a full list of elements in the wsrp-producer.xml file, see *WSRP Producer Configuration Elements (wsrp-producer.xml)* on page 38.

To upgrade an existing WSRP Producer configuration file (previously called portlets.xml), see Upgrading Existing WSRP Producer Implementations in the *Installation Guide for the .NET Application Accelerator*.

WSRP Producer Configuration Elements (wsrp-producer.xml)

The wsrp-producer.xml file provides configuration information about all portlets deployed in the WSRP Producer. This file is also used to map portlet modes to specific URLs and define portlet info and portlet properties for use within portlets.

The table below summarizes the configuration elements required in wsrp-producer.xml. For a complete listing of elements and their restrictions, see the wsrp-producer.xsd schema file in the WSRP Producer installation directory (\wsrpproducer\1.1\webapp\wsrpproducer\).

Note: Some metadata elements can be localized using a “key” attribute. For details, see *Localizing WSRP Portlet Metadata* on page 22.

Element	Description	Accepted Values
<requires-registration>	A flag indicating wether or not the WSRP Producer requires Consumers to register before providing a description of the portlets offered by the Producer.	true or false
<registration-property>	The list of registration properties required by the Producer.	For details on using registration properties, see *Accessing WSRP Consumer Registration Information Using the .NET Portlet Toolkit* on page 20.
<custom-items>	The list of custom window states and custom portlet modes that are used by the Producer. Custom items defined at this level may be referenced by any portlet offered by the Producer. Custom items that are shared by multiple portlets should always be defined here.	<custom-window-state> and <custom-portlet-mode> For details on window states and portlet modes, see *Implementing WSRP Portlet Window States Using the .NET Portlet Toolkit* on page 18 and *Implementing WSRP Portlet Modes Using the .NET Portlet Toolkit* on page 15.
<portlets>	A list of the portlets offered by the Producer.	<portlet> elements
<portlet>	Element within <portlets> that contains metadata for an individual portlet.	portlet configuration elements (see below)
Portlet Configuration Elements (used within each <portlet> element)		
<handle>	WSRP portlet handle, a unique token used to refer to the portlet shared between the Consumer and Producer.	string
<url id="">	Defines a URL for later use within portlet mode mappings; requires “id” parameter	URL
<expiration-cache>	Cache expiration setting in minutes	int

Element	Description	Accepted Values
<supported-locale>	A locale in which the portlet is able to generate markup. If not provided, indicates that the portlet will attempt to generate markup in any requested locale.	standard locale (as in Accept-Language header)
<description lang=" ">	A localizable description of the portlet intended for display in Consumer-generated dialogs.	string
<display-name lang=" ">	A localizable value intended for display in a Consumer's tooling for building aggregated pages. This value is usually shorter than either title element.	string
<supports>	Defines mime-type, window state, and portlet mode combinations supported by the portlet. If omitted, a default <supports> element for the mime-type"text/html" and all standard WSRP window-states and portlet modes is applied.	<mime-type>, <portlet-mode> and <window-state>
<mime-type>	Element within the <supports> element that defines the MIME type(s) supported by the portlet. **Note:** If the <supports> element contains multiple <mime-type> elements, the specified portlet mode or window state settings apply to all. To specify different portlet modes or window states on a per MIME type basis, add multiple <supports> elements to the portlet definition, each with a different <mime-type> element. If the same <mime-type> is listed more than once, an exception is thrown.	In addition to fully-specified mime-types, '*' and type specific wildcards (e.g., 'text/*') may be used.

Element	Description	Accepted Values
<portlet-mode> <name></name> <url></url>	Defines support for a portlet mode for the mime-type(s) of the parent <supports> element. If omitted, defaults to supporting all standard wsrp portlet modes.	The <name> element must contain a standard WSRP portlet mode (wsrp:view, wsrp:edit, wsrp:help or wsrp:preview) or a URI to a custom portlet mode. Custom portlet modes should include a <description>. Each supported mode must have an associated URL. Either include the URL for the mode within the <url> node, or use the idref attribute to refer to a URL defined as a child of the <portlet> element (see above). If neither an URL nor the idref is specified, the URL is assumed to be the first one found for the portlet. For details on using portlet modes, see *Implementing WSRP Portlet Modes Using the .NET Portlet Toolkit* on page 15.
<window-state> <name></name>	Defines support for a window state for the mime-type(s) of the parent <supports> element. If omitted, defaults to supporting all standard wsrp window states.	The <name> element must contain a standard WSRP window state (wsrp:normal, wsrp:solo, wsrp:minimized or wsrp:maximized) or a URI to a custom window state. Custom window states should include a description. For details on using window states, see *Implementing WSRP Portlet Window States Using the .NET Portlet Toolkit* on page 18.
<portlet-info>	Localized portlet metadata intended for use in Consumer-generated dialogs.	<title>: Localized title for the portlet. This value is intended for display in a titlebar decoration for the portlet markup.

Element	Description	Accepted Values
		<short-title>: Localized short title for the portlet. For details on localization, see *Localizing WSRP Portlet Metadata* on page 22.
<portlet-properties><property>	A list of portlet properties supported by instances of the portlet.	Each property element contains the following attributes: name: A unique name for the property used for indexing the property in a collection. (Required.)label: A short, human-readable name for the property intended for display in any Consumer-generated user interface for administering the portlet.hint: A short description of the property intended for display.type: The datatype of the property. Defaults to xs:string if omitted.defaultValue: The value sent to the portlet for this property if the property has not been explicitly set for the portlet instance. For details on using portlet properties, see *Manipulating WSRP Properties Using the .NET Portlet Toolkit* on page 11.
<user-profile>	A list of user profile properties requested by the portlet.	<item>: The full name of the user profile property.
<forms-authentication>	Defines support and configuration for SSO forms authentication.	For details on using this element, see *Using UNT Authentication with WebLogic Portal* on page 31.

The example below defines two portlets. The first uses one URL for all portlet modes, while the second uses separate URLs for the view mode and edit

mode. This code is from the WSRP sample code provided with the .NET Application Accelerator.

```
<?xml version="1.0"?>
<wsrp-producer
xmlns="http://www.bea.com/al/dotnet/wsrpproducer/1.1"


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

    xmlns:xs="http://www.w3.org/2001/XMLSchema">


<requires-registration>false</requires-registration>


 <portlets>

  <portlet>
   <handle>quote</handle>
   <url
id="default">http://localhost:4614/WSRPSamples/StockQuotePortlet.aspx</url>

   <expiration-cache>10</expiration-cache>
   <supported-locale>en</supported-locale>
   <description lang="en">Portlet that displays
stock quotes.</description>
   <display-name lang="en">Stock Quote
Portlet</display-name>
   <supports>
    <mime-type>text/html</mime-type>
    <portlet-mode>
     <name>wsrp:view</name>
     <url idref="default"/>
    </portlet-mode>
    <portlet-mode>
     <name>wsrp:edit</name>
     <url idref="default"/>
    </portlet-mode>
    <portlet-mode>
     <name>wsrp:help</name>
     <url idref="default"/>
    </portlet-mode>
   </supports>
```

```
    <portlet-info>
     <title lang="en">Stock Quote Portlet</title>
     <short-title lang="en">quote</short-title>
    </portlet-info>
    <portlet-properties>
     <property name="stockSymbols" type="xs:string"
 defaultValue="BEAS"/>
    </portlet-properties>
   </portlet>

   <portlet>
    <handle>quote2</handle>
    <url
id="view">http://localhost:4614/WSRPSamples/StockQuotePortlet2/View.aspx</url>

    <url
id="edit">http://localhost:4614/WSRPSamples/StockQuotePortlet2/Edit.aspx</url>

    <expiration-cache>10</expiration-cache>
    <supported-locale>en</supported-locale>
    <description lang="en">Portlet that displays
stock quotes using a seperate edit
page.</description>
    <display-name lang="en">Stock Quote Portlet
2</display-name>
    <supports>
     <mime-type>text/html</mime-type>
     <portlet-mode>
      <name>wsrp:view</name>
      <url idref="view"/>
     </portlet-mode>
     <portlet-mode>
      <name>wsrp:edit</name>
      <url idref="edit"/>
     </portlet-mode>
    </supports>
    <portlet-info>
     <title lang="en">Stock Quote Portlet 2</title>
     <short-title lang="en">quote#2</short-title>
    </portlet-info>
    <portlet-properties>
     <property name="stockSymbols" type="xs:string"
 defaultValue="BEAS"/>
    </portlet-properties>
```

```
  </portlet>
 </portlets>
</wsrp-producer>
```