August 2012

Using the Oracle WebCenter Pagelet Producer to Integrate Markup from Portlets, ADF Taskflows, and Oracle Applications into Oracle WebCenter Sites

An Oracle Whitepaper

Table of Contents

Introduction

Target Audience

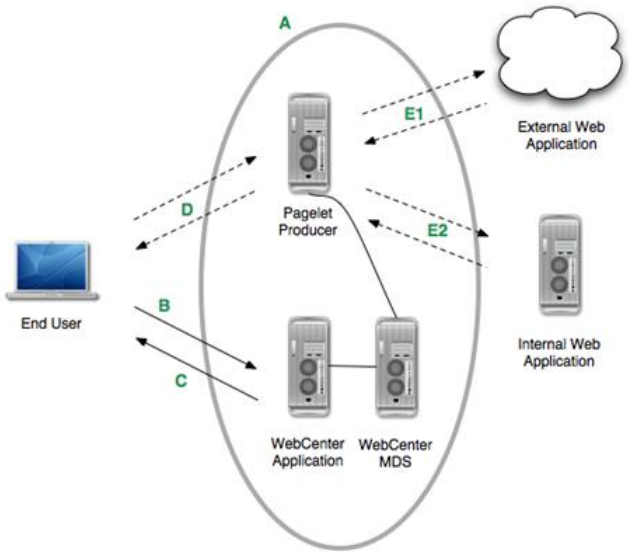
This white paper is aimed at developers who need to integrate content into Oracle WebCenter Sites 11g pages, including existing WSRP portlets or elements of web UI exposed by backend applications such as Oracle EBS. Developers should be familiar with Oracle WebCenter Sites and Oracle WebCenter Pagelet Producer and have a solid understanding of web technologies.

About Oracle WebCenter Portal: Pagelet Producer

The Pagelet Producer proxy provides external access to internal resources including backend applications and secured content, thus helping to leverage investment in existing web infrastructure by making it usable in new ways. Specifically Pagelet Producer provides:

- Tools for consuming existing web UI elements and re-using them as portable components, called pagelets
- Features for integrating non-standard/non-compliant consumers and external markup producers (non-standard refers to consumers or producers that do not support portlet standards, such as JSR 168, JSR 286, or WSRP); these features allow you to:
    - Capture functionality from web applications that do not expose portlets
    - Make existing WSRP, JPDK, or CSP portlets available for use in a non-portal setting
- Injector and Parser components to modify HTML markup, allowing you to adjust functionality or styling to fit the host page
- JavaScript framework for inter-pagelet communication and event handling

A typical use case for the Pagelet Producer consumes existing web UI and delivers it to an HTML page with added value, such as role-based access control to exposed resources and reverse proxying of web resources with HTML mark-up parsing, clipping, injection and inter-pagelet communication. The following diagram shows the high level architecture and interaction flow of a pagelet consumed in WebCenter:



Key Concepts

The following key concepts are useful when working with the Pagelet Producer:

- **Resources** are core objects used to register applications within the Pagelet Producer, including backend applications and WSRP producers. Creating a resource allows the Pagelet Producer proxy to map internal applications to external URLs, manage authentication, and transform applications.
- **Pagelets** are sub-components of a web page accessed through the Pagelet Producer that can be injected into any web application. A pagelet is a reusable component similar to a portlet, but contrary to a portlet, a pagelet does not require a portlet consumer to run. Developers can parameterize pagelets and design them to interact with other pagelets.
- **Injectors** are web injectors that insert content into a specified location in a proxied resource page.
- **Parsers** can be configured to supplement or change built-in logic for parsing content and finding URLs. The parser module in the Pagelet Producer includes default parsers for HTML and JavaScript.

For more information about configuring the Pagelet Producer, creating resources and configuring pagelets, see Managing Oracle WebCenter Portal's Pagelet Producer in the *Oracle Fusion Middleware Administrator's Guide for Oracle WebCenter Portal*. Information about using parameters with pagelets, setting up inter-pagelet communication and other advanced topics are covered in Creating Pagelets with Oracle WebCenter Portal's Pagelet Producer in the *Oracle Fusion Middleware Developer's Guide for Oracle WebCenter Portal*.

Note: In Oracle WebCenter Sites, the term "pagelet" is used to refer to the generated output of one or more elements of a page. The pagelets exposed by the Pagelet Producer are different from these internal pagelets. One way to clarify the difference is to consider Pagelet Producer pagelets as "remote pagelets" vs. the "local pagelets" created within Sites.

Adding Pagelets to Oracle WebCenter Sites

Once a pagelet and its related resources are configured, you can insert it into a web page using JavaScript or REST. For details, see Adding a Pagelet to a Web Page in the *Oracle Fusion Middleware Developer's Guide for Oracle WebCenter Portal*.

This white paper uses an approach in which pagelets are added directly to a page template in Oracle WebCenter Sites using an iFrame with a REST URL for accessing pagelets as a content source. This requires an addition to the page template script tag to load the CSAPI JavaScript library (which adds necessary functions into the parent page), as well as the actual iFrame that loads pagelet content:

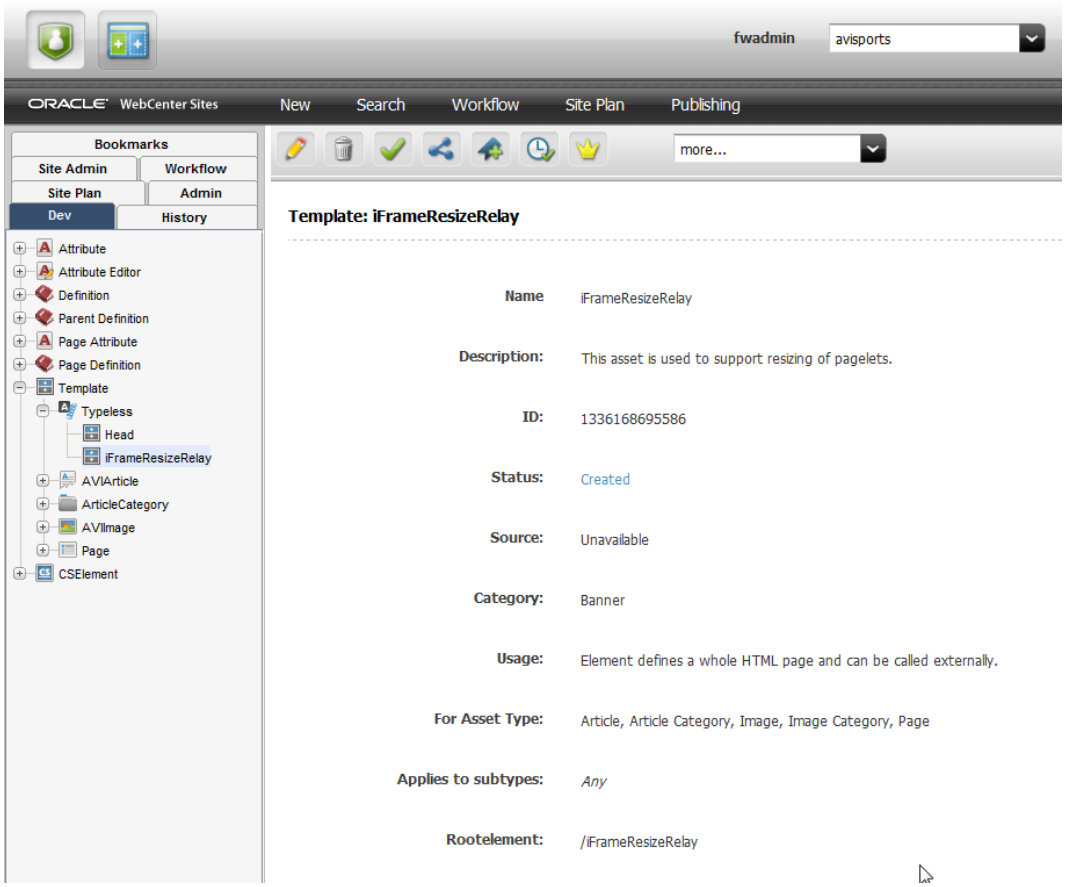
```
<script type="text/javascript" src="http://%PAGELET_PRD_HOST%/pagelets/inject/v2/csapi"/>

<iframe id="pt-pagelet-iframe-1" width="100%" frameborder="0"
src="http://%PAGELET_PRD_HOST%/pagelets/inject/v2/pagelet/lib_name/pagelet_name?content-
type=html&consumepage=true&ifheight=auto"></iframe>
```

Where `lib_name` and `pagelet_name` refer to the library and pagelet configured in the Pagelet Producer. For details on parameters, see Accessing Pagelets Using REST in the *Oracle Fusion Middleware Developer's Guide for Oracle WebCenter Portal*.

Enabling IFrame Auto-Resizing

To provide a more seamless integration of the pagelet into the consuming page, you can make the pagelet iFrame behave more like inline markup by dynamically resizing to accommodate its content. To enable this feature, some extra configuration is required in both Sites and the Pagelet Producer.

This white paper uses the AviSports sample site shipped with WebCenter Sites. To add any new asset to this site, you must log in to the Sites Administration UI and open the Dev section in the left hand navigation as shown below:



First, create a new template with the following settings:

- Name
  - Name: iFrameResizeRelay (or name of your choice)
  - For Asset Type: Applicable to various asset types (typeless)
- Element
  - Usage: Element defines a whole HTML page and can be called externally
  - Element Storage Path/Filename: iframe-resize.html (or name of your choice)
  - Element Logic:

```
<html>
  <head>
    <title>Resizing Page</title>
    <script type="text/javascript">
       function onLoad() {
              var params = window.location.search.substring( 1 ).split( '&' );
              var height;
              var width;
              var iframe;
              for( var i = 0, l = params.length; i < l; ++i ) {
                     var parts = params[i].split( '=' );
                     switch( parts[0] ) {
                     case 'height':
                            height = parseInt( parts[1] );
                            break;
                     case 'width':
                            width = parseInt( parts[1] );
                            break;
                     case 'iframe':
                            iframe = parts[1];
                            break;
                     }
              }
              window.top.updateIFrame( iframe, height, width );
       }
       if (window.addEventListener) {
              window.addEventListener("load", onLoad, false);
       } else if (window.attachEvent) {
              window.detachEvent("onload", onLoad);
              window.attachEvent("onload", onLoad);
       } else {
              window.onload=onLoad;
       }
    </script>
  </head>
  <body>
  </body>
</html>
```

- Site Entry
    - Cache Rules: Cached (default)
- Save
- Record SiteCatalog Pagename as %PAGENAME% (if you used the default name, this would be <site_name>/iFrameResizeRelay)

The new template is addressable as follows:

```
http://%SITES_HOST%/cs/Satellite?pagename=%PAGENAME%
```

Make a note of this URL; it will be used to configure the Injector for the pagelet.

Next, apply the iFrame resizing feature by adding a new Injector to the pagelet via the Pagelet Producer Administration Console. Open the pagelet, select **Injectors** and create a new Injector with the following settings:

- General
    - Name: auto_resizer (or name of your choice)
    - URL Filter: <none>
    - MIME Filter: text/html
    - Inject Location: Before </head>
- Content: Select Text (default) and copy the JavaScript below into the text area
  Note: Replace *SITES_RESIZE_RELAY_PAGE* in the JavaScript below with the URL to the template you created in the previous step.

```
<script type="text/javascript">
var SITES_RESIZE_RELAY_PAGE = "http://%SITES_HOST%/cs/Satellite?pagename=%PAGENAME%"; // CHANGE ME!

if (window.addEventListener) {
    window.addEventListener("load", calculateSizeFixed, false);
} else if (window.attachEvent) {
    window.attachEvent("onload", calculateSizeFixed);
} else {
    window.onload=calculateSizeFixed;
}
function calculateSizeFixed() {
  var PTResizeIFrame = PTResizeIFrame || {};
  if (PTPortalPage && PTPortalPage.portlets) {
    for (var i in PTPortalPage.portlets) {
      if ( PTPortalPage.portlets[i].id != "page") {
        PTResizeIFrame.pageletInstanceID = PTPortalPage.portlets[i].id;
        break;
      }
    }
  }
  else if (!PTResizeIFrame.pageletInstanceID) {
    PTResizeIFrame.pageletInstanceID = 1;
  }

  if (!PTResizeIFrame.pageletResizePage) {
    var match = window.location.search.match(/resizepage=[^&]*/);
    if (match != null && match.length > 0) PTResizeIFrame.pageletResizePage =
unescape(match[0].substr("resizepage=".length));
    else PTResizeIFrame.pageletResizePage = SITES_RESIZE_RELAY_PAGE;
  }
  var agent = navigator.userAgent;
  var ffversion = agent.indexOf("Firefox") >= 0 ?
agent.substring(agent.indexOf("Firefox")).split("/")[1] : -1;
  var FFextraHeight = parseFloat(ffversion)>=0.1? 25 : 0;
  var scrollHeightExtra = 15;
  if (agent.indexOf('MSIE') > 0) {
    scrollHeightExtra = 35;
  }
  if (FFextraHeight > 0) scrollHeightExtra = FFextraHeight;
  var wrapper = document.getElementById( 'pt-inner-iframe-wrapper-' +
PTResizeIFrame.pageletInstanceID);
  if (wrapper == null) wrapper = createRelayIFrame(PTResizeIFrame.pageletInstanceID);
  var iframe = document.getElementById( 'pt-inner-iframe-' + PTResizeIFrame.pageletInstanceID);
  var height = 0;
  var width = 0;
  var iframename = '';

  if( (document.contentDocument) && (document.contentDocument.documentElement.offsetHeight) ) {
    height = document.contentDocument.documentElement.offsetHeight + FFextraHeight;
```

```
  } else if( (document.contentDocument) && (document.contentDocument.body.offsetHeight) ) {
    height = document.contentDocument.body.offsetHeight+FFextraHeight;
  } else if (document && document.documentElement.scrollHeight ) {
    height = document.documentElement.scrollHeight + scrollHeightExtra;
  } else if( document && document.body.scrollHeight ) {
    height = document.body.scrollHeight + scrollHeightExtra;
  } else {
    height = wrapper.offsetHeight;
  }
  width = wrapper.offsetWidth;
  iframename = 'pt-pagelet-iframe-' + PTResizeIFrame.pageletInstanceID;
  var qsSeparator = PTResizeIFrame.pageletResizePage.indexOf("?") >= 0 ? "&" : "?";

iframe.setAttribute("src", PTResizeIFrame.pageletResizePage + qsSeparator + 'height=' + height + '&'
+ 'width=' + width + '&' + 'iframe=' + iframename);
}

function createRelayIFrame(pageletInstanceId) {
  var wrapper = document.createElement("div");
  wrapper.id = "pt-inner-iframe-wrapper-" + pageletInstanceId;
  var iframe = document.createElement("iframe");
  iframe.id =  "pt-inner-iframe-" + pageletInstanceId;
  iframe.setAttribute("height", 0);
  iframe.setAttribute("frameborder", 0);
  iframe.setAttribute("width", 0);
  wrapper.appendChild(iframe);
  document.body.appendChild(wrapper);
  return wrapper;
}
</script>
```

Add the pagelet to the page template in Sites using a REST URL within an IFrame. In the IFrame, the *id* parameter should use the unique number identifying the pagelet IFrame (for example, “pt-pagelet-iframe-1”). You must add the following query string parameters to the pagelet URL to support IFrame auto-resizing:

- resizepage=http://%SITES_HOST%/cs/Satellite?pagename=%PAGENAME%
- ifheight=auto

For example:

```
<script type=”text/javascript” src=”http://%PAGELET_PRD_HOST%/pagelets/inject/v2/csapi”/>

<iframe id=”pt-pagelet-iframe-1” width=”100%” frameborder=”0”
src=”http://%PAGELET_PRD_HOST%/pagelets/inject/v2/pagelet/lib_name/pagelet_name?content-type=html&
consumepage=true&resizepage=http://%SITES_HOST%/cs/Satellite?pagename=%PAGENAME%&ifheight=auto”>
</iframe>
```

Note: In certain situations automatic resizing may not work properly when multiple pagelets are present on a page. This is known to occur with pagelets that require form auto-login to external authentication servers. In this case, only one pagelet can be configured for auto-resizing, while the others should use static IFrame sizing.

Changing Pagelet Styling

To make a pagelet fit better visually in a consuming WebCenter Sites page, an Injector may be used to add styles that override the original CSS. In order for the override to work correctly, it is important that the style definitions supplied by the Injector come after the styles defined by the backend application. The following example does this by injecting replacement CSS into the end of the <HEAD> section.

To restyle a pagelet, add a new Injector to the pagelet via the Pagelet Producer Administration Console. Open the pagelet, select **Injectors** and create a new Injector with the following settings:

- General:
  - Name: new_styles (or name of your choice)
  - URL Filter: <none>
  - MIME Filter: text/html
  - Injector Location: Before </head>
- Content:

```
<style type="text/css">
   /* CSS classes to override original CSS go in here. */
</style>
```

The Consuming E-Business Suite 11i section includes a working example of an Injector used to re-style the EBS UI to match the sample AviSports site in WebCenter Sites.

Related Resources

The following guides are referenced in this document:

- Oracle® Fusion Middleware Installation Guide for Oracle WebCenter Portal
- Oracle® Fusion Middleware Administrator's Guide for Oracle WebCenter Portal
- Oracle® Fusion Middleware Developer's Guide for Oracle WebCenter Portal
- Oracle® Fusion Middleware User's Guide for Oracle WebCenter Portal: Spaces
- Oracle® Fusion Middleware Domain Template Reference
- Oracle® WebCenter Sites Developer's Guide
- Oracle® WebCenter Sites Configuring Supporting Software
- Oracle® Fusion Middleware Portlet Development Guide for Oracle WebLogic Portal
- Oracle® Fusion Middleware Federated Portals Guide for Oracle WebLogic Portal

Using Identity Propagation

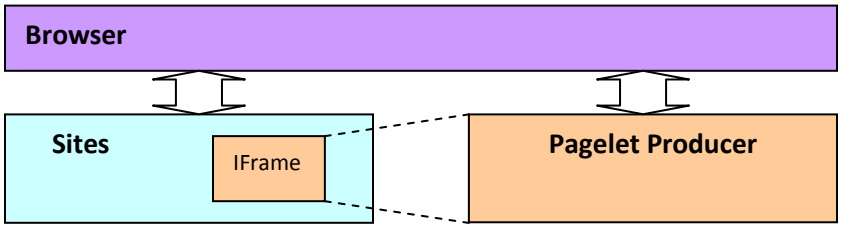
The core purpose of the Pagelet Producer is to proxy backend applications. Since the Pagelet Producer acts as a "middle-man" between the browser and the backend application, identity propagation must be broken into two parts:

- Establishing Identity in the Pagelet Producer
- Propagating Identity from the Pagelet Producer to the Backend Application

Establishing Identity in the Pagelet Producer

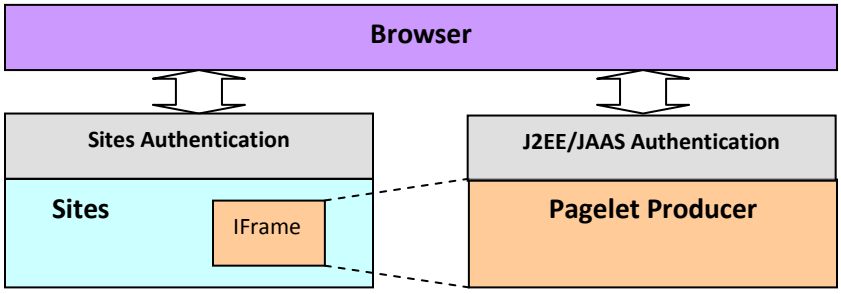
The Pagelet Producer typically injects web content into WebCenter Sites using an IFrame.

Figure 1 – Pagelet Producer content as IFrame in Sites



Since the content is injected as an IFrame, user identity must be established in both the Sites container and the Pagelet Producer container.

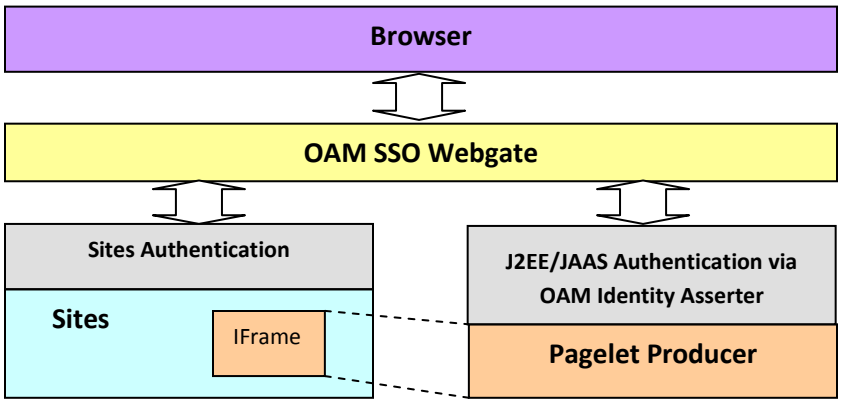
Figure 2 – Different authentication schemes for Sites and Pagelet Producer



**The above figure leaves out the user directory that is assumed to be shared between the two authentication schemes.*

The ideal way to manage identity between the browser and the two containers would be to utilize Oracle Access Manager (OAM).

Figure 3 – Ideal frontend authentication scenario (unsupported pre-11g)



**The above figure leaves out the user directory that is assumed to be shared between the two authentication schemes. It also excludes the OAM access server.*

To set up OAM for WebCenter Sites 11g, refer to the *Oracle Access Manager Integration Setup* chapter in Oracle® WebCenter Sites Configuring Supporting Software.

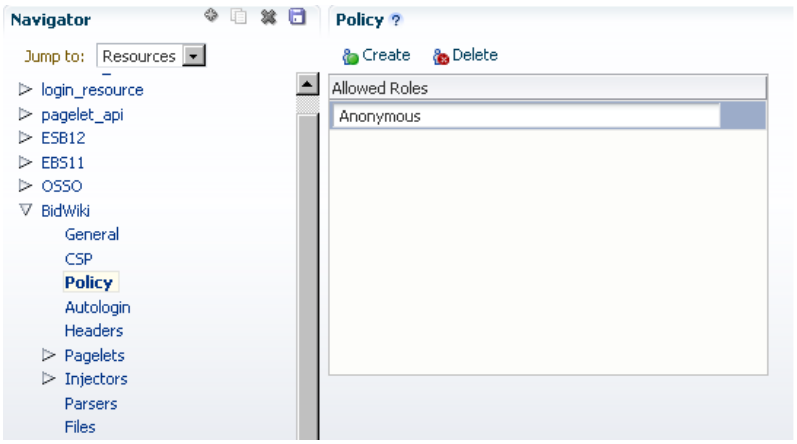
Using a Login Page Supplied by Pagelet Producer

Before setting up OAM, it is sometimes useful to test the integration of the Pagelet Producer in Sites. Without OAM, users must authenticate separately with the Pagelet Producer and with Sites (See Figure 2). This section describes how to configure separate authentication with the Pagelet Producer.

If OAM SSO is not present on either the Pagelet Producer or Sites, the authentication scheme shown in Figure 2 must be supported by ensuring that the Pagelet Producer provides a login form. To force the Pagelet Producer to display a login form:

1. Identify a J2EE role to restrict access to the pagelet(s) or create a new role in the J2EE application server hosting the Pagelet Producer. (This example uses the global WebLogic Server (WLS) "Anonymous" J2EE role that is present in a default WLS installation.)

2. Log in to the Pagelet Producer Administration Console, navigate to the resource(s) containing the pagelet(s) that must be surfaced in Sites and add the role you chose in step 1. Roles are defined on the **Policy** page for the resource.



Propagating Identity from the Pagelet Producer to the Backend Application

The way in which identity is established depends on the type of application:

- WSRP/JPDK Portlet
- Stand-alone Backend Application Protected with Native Authentication (Not SSO)
- Stand-alone Backend Application Protected with SSO

WSRP/JPDK Portlet

WSRP portlets must use WSS tokens for identity propagation. For details on configuring security tokens, see Registering WSRP and Oracle JPDK Portlet Producer in the *Oracle Fusion Middleware Administrator's Guide for Oracle WebCenter Portal* and the Consuming WSRP Portlets as Pagelets section in this document and Registering WSRP and Oracle JPDK Portlet Producers in the *Oracle Fusion Middleware Administrator's Guide for Oracle WebCenter Portal*.

Each Oracle JPDK portlet that requires an identity will be passed credentials based upon information supplied by the external application's login form. For details, see Managing External Applications and Registering Oracle JPDK Portlet Producers in the *Oracle Fusion Middleware Administrator's Guide for Oracle WebCenter Portal*.

Stand-alone Backend Application Protected with Native Authentication (Not SSO)

The Pagelet Producer has a feature called Autologin that allows the Pagelet Producer to interact with the "native" authentication mechanism established by a backend application.

Identity Propagation with Autologin

Since each backend application has its own means of authentication, a separate login prompt must be initiated by the Pagelet Producer to collect any required credentials. Pagelet Producer resources can be configured to send credentials to backend applications as basic authentication headers, NTLM tokens, or Kerberos tokens, as well as manage forms-based authentication (typically via session cookie). For details, see Autologin in the *Oracle Fusion Middleware Administrator's Guide for Oracle WebCenter Portal*.

Note: Set the Username and Password fields to use the User Vault so that each user will be prompted for his or her own unique credentials to access the backend application.

Stand-alone Backend Application Protected with SSO

Determining the proper Autologin settings for a backend application can be very challenging. Doing so requires looking at request headers, identifying redirects, and examining markup (sometimes dynamically generated). This inspection and configuration process can be daunting if there are multiple applications for which Autologin is required. Therefore, if Oracle Access Manager (OAM) SSO or Oracle SSO (OSSO) is available, it is highly recommended that one of these SSO solutions be used to protect backend applications.

Once the backend application is protected with OAM, there are two ways that the Pagelet Producer can supply credentials:

- Using Direct Identity Propagation to OAM Protected Backend Application
- Using Identity Propagation with Autologin and SSO

Using Direct Identity Propagation to OAM Protected Backend Application

An OAM 11g WebGate is an Apache module running on Oracle HTTP Server (OHS) that manages all validation of user credentials with the OAM server. Once it has established the validity of a user it sends the OAM_REMOTE_USER header to the web application on the application server that OHS is proxying. The application server will have an OAM identity asserter running on it that will set the JAAS user principal name to the OAM_REMOTE_USER header value.

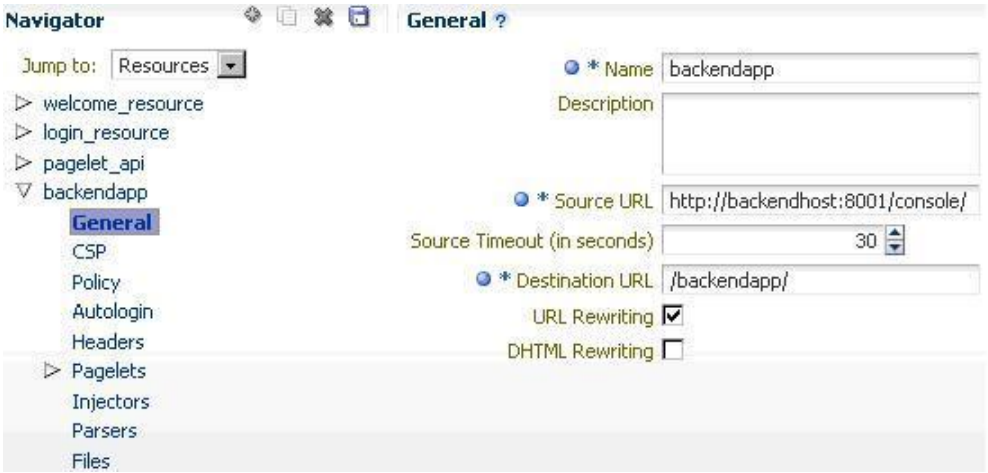
If the Pagelet Producer is protected with an OAM 11g WebGate it will receive an OAM_REMOTE_USER header. All web applications that it proxies will also receive this header. If the web application it proxies is on an application server with the OAM Identity Asserter, then the OAM Identity Asserter will use the OAM_REMOTE_USER header value passed from the Pagelet Producer to set the user principal. This will all work as long as the Pagelet Producer is not trying to proxy the URL to the WebGate for the protected web application. The rest of this section describes in details how this can be achieved in the context of Sites integration.

Example Setup

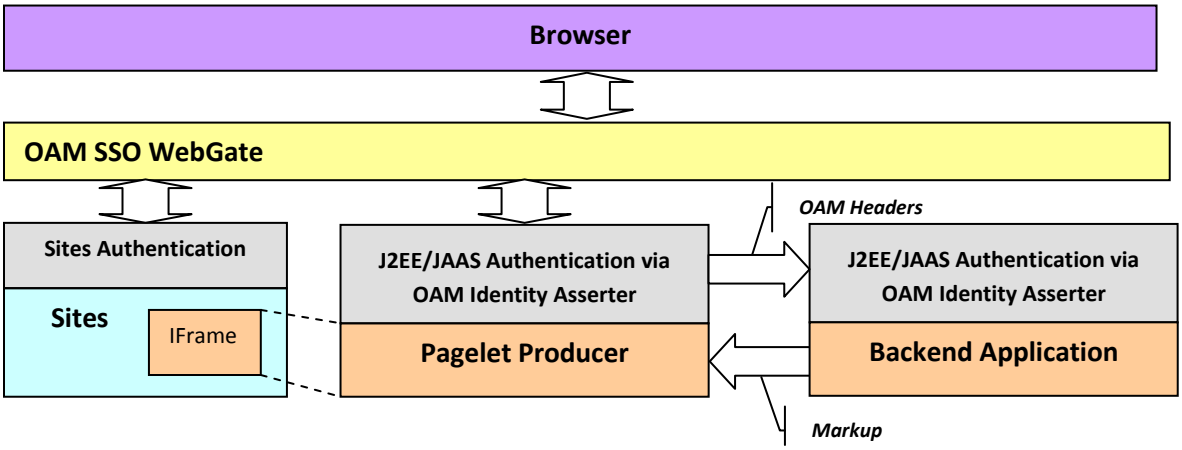
This example assumes that both WebCenter Sites and the Pagelet Producer are protected by OAM WebGate as illustrated in Figure 3 above, and the following hosts and ports are used by the applications:

- OAM SSO WebGate : www.acme.com:80
- WebCenter Sites : sites_host:8080
- WebCenter Pagelet Producer : pp_host:8889
- Backend application : backendhost:8001
- Full URL to backend application as viewed from browser (via WebGate) http://www.acme.com/backend/console
- Full URL to backend application if accessed directly through application server (available only in internal network) http://backendhost:8001/console

Set up the Pagelet Producer resource to use the direct URL to the backend application, thus bypassing the WebGate (i.e., http://backendhost:8001/console).



By setting up the Pagelet Producer to call directly into the backend application (bypassing the WebGate) all the OAM secure headers that the Pagelet Producer receives are passed to the OAM Identity Asserter on the application server hosting the backend application. Once the OAM Identity Asserter on the application server hosting the backend application receives the OAM headers it will set the proper user principal.



**The above figure leaves out the user directory that is assumed to be shared between the two authentication schemes. It also excludes the OAM access server.*

Note: Although setting up the Pagelet Producer to bypass the WebGate will achieve single sign-on, this solution should not always be applied. In cases where the OHS server hosting the WebGate is also managing load balancing for the backend application, bypassing the WebGate will also bypass load-balancing. All traffic to the application will be routed to one server. In this case we recommend configuring Pagelet Producer for Autologin as described in the next section, Using Identity Propagation with Autologin and SSO.

Using Identity Propagation with Autologin and SSO

If bypassing the OAM WebGate (as described in the previous section) is not a feasible option then credentials can still be supplied using the Autologin feature of the Pagelet Producer. The advantage of utilizing Autologin with an Oracle SSO solution (OAM or OSSO) login page as opposed to individual login forms for each backend application (see Identity Propagation with Autologin) is that Autologin only has to be set up once and the configuration steps are the same for every implementation.

The steps for setting up Autologin with OSSO are detailed in the Configure Autologin section of Example 1: Consuming E-Business Suite 11i in this document. In a production environment, these steps can be followed as is with one exception: the *ssousername* and *password* form fields should be linked to the "User Vault".

Backend applications protected with OAM can also be set up following the steps described for OSSO in the Configure Autologin section in this document with the following exceptions:

- The login form URL will be a set location (e.g., http://oam11g.mycompany.com:14100/oam/server/obrareq.cgi).
  Note: Remove the query string after the fixed location argument when setting the Autologin form identification URL.
- The form action URL will be a different value, but it should be a static value. Inspect the OAM form to determine the form action URL.

- The username and password fields may be different values. Inspect the OAM form fields to determine these values.
- Include the following 2 generated values: “request_id” and “OAM_REQ”

Reference: Common Auto-Login Configurations

This section provides a quick reference of autologin settings in the Pagelet Producer for common backend SSO scenarios.

Oracle Access Manager (OAM)

Login Form Identification:

URL: http://%OAM_ROOT%/server/obrareq.cgi

Form Submit Location:

URL: /oam/server/auth_cred_submit

Form Fields:

- actionURL : static : /oam/server/auth_cred_submit
- request_id : Generated
- username : Static or Credential Vault
- OAM_REQ : Generated
- password : Static or Credential Vault

Oracle SSO (OSSO)

Login Form Identification:

URL: %OSSO_HOST%/sso/jsp/login.jsp

Form Submit Location:

URL: %OSSO_HOST/sso/auth - POST

Form Fields:

- appctx: Generated
- locale: Generated
- password : Static or Credential Vault
- site2pstoretoken : Generated
- ssousername: Static or Credential Vault
- v: Generated

Consuming WSRP Portlets as Pagelets

Pagelet Producer incorporates the WebCenter Common Consumer and therefore can expose WSRP and Oracle JPDK portlets as pagelets for use in WebCenter Sites or any other web container that does not include a WSRP consumer.

The WSRP Portlet Producer can be registered with Pagelet Producer using Enterprise Manager, WLST or the Pagelet Producer Administrative Console. For details see Registering WSRP Portlet Producers in the *Oracle Fusion Middleware Administrator's Guide for Oracle WebCenter Portal*. Once registration is complete, the WSRP producer will appear as a resource in the Pagelet Producer Administrative Console, and the portlets associated with the WSRP endpoint will be listed in the pagelets collection for the resource.

Note: Auto-generated WSRP resources and pagelets cannot be modified. To create a version that can be modified, choose the resource in the Pagelet Producer Administrative Console and click Copy. The cloned version of the resource can be edited and various elements such as Injectors can be added to customize pagelet functionality. For example, a custom Injector could be used to inject CSS classes to modify portlet markup to make it look and feel like the Sites that will host the pagelet.

If the WSRP Producer that you are registering with Pagelet Producer requires authentication, the following steps are required:

- Configure WS-Security between Pagelet Producer and the WSRP Producer as described in Configuring WS-Security in the *Oracle Fusion Middleware Administrator's Guide for Oracle WebCenter Portal.* You must ensure that the Java Key Store (JKS) is properly configured between the WSRP producer and the Pagelet Producer; for details, see Setting Up the WebCenter Portal Domain Keystore in the Configuring WS-Security chapter.
- During the WSRP Producer registration, select the appropriate Token Profile in the Security section and enter the necessary configuration information. The path to the keystore must be absolute.

Exposing Custom ADF Taskflows as WSRP Portlets

Although it is possible to consume an ADF taskflow in the Pagelet Producer by 'clipping' it out of the page on which it is hosted, this is not the recommended approach. Considering the complexity of the markup generated by an ADF page and potential dependencies on other taskflows or ADF page parameters, the recommended approach is to create a WSRP portlet based on the taskflow and then consume the portlet in the Pagelet Producer. One advantage of this approach is that ADF taskflow parameters become portlet parameters that can be mapped to pagelet parameters, allowing information to be passed from the Sites page to the taskflow. For step-by-step instructions on how to make a portlet from an ADF taskflow, see How to Create a JSF Portlet Based on a Taskflow in the *Oracle Fusion Middleware Developer's Guide for Oracle WebCenter Portal*.

After the ADF taskflow is exposed as a portlet, deploy a new WSRP Producer to a managed server in your WLS domain (for example, the WC_Portlet server on your WebCenter domain). Obtain the WSDL URL from the info page (http://%PRODUCER_HOST%/%APP_CONTEXT%/info) and make note of it. To register the new WSRP Producer with the Pagelet Producer, see Registering WSRP Portlet Producers in the *Oracle Fusion Middleware Administrator's Guide for Oracle WebCenter Portal*.

Exposing WSRP Portlets Developed for Oracle WebLogic Portal

Oracle WebLogic Portal (WLP) supports a variety of portlet types, including Java Portlets, Java Server Faces Portlets, Java Server Page and HTML Portlets (for a complete list see Portlet Types in the *Oracle Fusion Middleware Portlet Development Guide for Oracle WebLogic Portal*). Remote (Proxy) Portlets are WSRP portlets and can be directly exposed as pagelets by registering the WSRP Producer with the Pagelet Producer. This white paper describes how to expose local WLP portlets such as JSF, JSP or Java Portlets as pagelets for use in Sites.

Exposing WLP Portlets Using WLP as WSRP Producer

WLP can expose Java, Page Flow, JSP, JSF, and Struts Portlets as a "complex producer" that includes the required WSRP interfaces, optional interfaces and some extended interfaces (for details see WebLogic Portal Producers in the *Oracle Fusion Middleware Federated Portals Guide for Oracle WebLogic Portal*). Consequently, native WLP portlets can be consumed by the Pagelet Producer as any other WSRP portlets. Configuration and best practices for WSRP interoperability between WLP and the WebCenter Common Consumer are described in WSRP Interoperability with Oracle WebCenter Portal and Oracle Portal in the *Oracle Fusion Middleware Federated Portals Guide for Oracle WebLogic Portal*.

Note: WLP portlets are WSRP-enabled by default since version 9.2. In earlier versions, you must edit the portlet properties and set the "Offer as Remote" property to *true*.

Configuring WS Security between WLP WSRP Producer and WebCenter Consumer

To consume WSRP portlets exposed by WLP in the WebCenter Common Consumer (Pagelet Producer), configuration is required on both the producer and consumer sides.

Typically it is impossible to consume WSRP portlets exposed by WebLogic Portal as an anonymous user, so it is necessary to configure SAML security on WLP for the producer and the WebCenter Common Consumer. Step-by-step instructions can be found in SAML Security Between a WebCenter Portal: Framework Application Consumer and a WebLogic Portal Producer in the *Oracle Fusion Middleware Federated Portals Guide for Oracle WebLogic Portal*. These instructions are comprehensive, with the following adjustments:

- In section 5 of Configuring the Producer, the steps for configuring keystore information on the WSRP consumer are executed in JDeveloper. You must complete this step in the WebCenter Common Consumer (Pagelet Producer). You can specify the signing alias and keystore password through WLST or Oracle Enterprise Manager. For details, see Configuring the Keystore Using WLST or Configuring the Keystore Using Fusion Middleware Control in the *Oracle Fusion Middleware Administrator's Guide for Oracle WebCenter Portal*.
  Note: Once you have configured the keystore values you must restart both the managed server hosting the Pagelet Producer (by default WC_Portlet) and the WLS Admin Server for the JPS configuration changes to be picked up.
- In section 4 of Configuring the Producer, the instructions do not emphasize the importance of the issuer name. The name placed here must match that sent by the consumer. For Oracle JPS, the default issuer ID sent is *www.oracle.com*.

Registering the WLP WSRP Producer in the Pagelet Producer

To register the WLP WSRP Producer in the Pagelet Producer, follow the standard steps for creating and configuring a new WSRP Producer described in Registering WSRP Portlet Producers in the *Oracle Fusion Middleware Administrator's Guide for Oracle WebCenter Portal*). Make sure to include the following settings:

- WSDL URL:
  http://%WLP_HOST%/%PORTAL_APP_NAME%/producer/wsrp-1.0/markup?WSDL
- Security:
  - Token Profile: Select "*WSS 1.0 Token with Message Integrity*"

After registration, a new Pagelet Producer resource is automatically created and populated with pagelets to represent the WLP portlets associated with this WSRP endpoint.

Note: To allow user identity propagation to the portlets, both WLP and the Pagelet Producer should be configured as described in Propagating Identity from the Pagelet Producer to the Backend Application in this white paper. (For testing purposes or for portlet content that can be accessed without authentication, you can specify a valid username in the Default User field under Security.)

Adding WLP WSRP Portlets to Sites

To add WLP WSRP portlets to a page in Sites, follow the steps in Adding Pagelets to WebCenter Sites in this white paper. The URL to use for the *src* attribute in the *IFrame* tag that loads pagelet content can be found on the Documentation section for the pagelet under "Access Pagelet using REST".

Auto-generated WSRP resources and pagelets cannot be modified. To use Pagelet Producer features to alter the markup of the WLP portlet such as modify look and feel of the portlet UI by injecting custom CSS styles, you must create a new version of the resource. Choose the resource that was created for your WLP WSRP Producer in the Pagelet Producer Administrative Console and click Copy. The cloned version can be modified, including adding elements such as Injectors to customize pagelet functionality.

Consuming WebCenter Portal Services as Pagelets in Sites

Oracle WebCenter Portal: Spaces includes a library of services, such as Announcements, and Document Manager that can be used to facilitate user collaboration in WebCenter Sites. For an overview and usage information about these services, see Services: Announcements through Links and Services: Lists through Worklist in the *Oracle Fusion Middleware User's Guide for Oracle WebCenter Portal: Spaces*.

The recommended approach for exposing these services in Sites is to consume the associated WSRP portlets in the Pagelet Producer by leveraging the Services Producer component included with Oracle WebCenter Portal (11.1.1.6 and above).

The following portal services are exposed as WSRP portlets by default:

Activity Stream	Discussion Forums
Mail	Polls Manager
Document Manager	Lists
Tag Cloud	Blogs
Work list	Take Polls
Announcements	

Note: Additional services can be exposed by extending the Services Producer to a new JSPX page and then adding the services taskflow to the page.

Requirements

Install Oracle WebCenter Portal: Spaces following the instructions in the *Oracle Fusion Middleware Installation Guide for Oracle WebCenter*. Make sure that all dependent components are also installed and configured, including WebCenter Content, Discussions Server and Pagelet Producer.

By default the Services Producer is automatically deployed to the *WC_Portlet* managed server in the WebCenter Portal domain. If the *WC_Portlet* managed server was not created during installation or there is a requirement to run the Services Producer separately, the Oracle WebCenter Portal 11.1.1.6 release added a custom Services Producer template that can be uses to create the *WC_CustomServicesProducer* managed server. This managed server contains the Oracle WebCenter Services Producer component as well as pre-configured JDBC data sources for accessing the Portal, Activities, and MDS schemas. For details, see Oracle WebCenter Custom Services Producer Template in the *Oracle Fusion Middleware Domain Template Reference*.

Configuring Security and Single Sign-On

There are two prerequisites to allow access to WebCenter Portal services from a Sites-driven web site:

- Create a common user base between the web site and WebCenter Portal
- Establish user identity propagation from the web site to WebCenter Portal

Creating a Common User Base: LDAP Integration

Creating a common user base requires configuring a common LDAP repository to be used by WebCenter Sites and all WebCenter Portal components. When selecting an LDAP server, make sure it is compliant with the WebLogic Server that runs WebCenter Portal components.

Things to remember when configuring security on WLS:

- Set the priority of the Authenticators to SUFFICIENT
- Set the Control Flag (in parenthesis) and order of authentication providers as follows:
    - OAM Identity Asserter (REQUIRED)
    - LDAP Authenticator (SUFFICIENT)
    - Default Authenticator (SUFFICIENT)
    - Default Identity Asserter (SUFFICIENT)

Establishing User Identity Propagation: OAM Configuration

To establish user identity propagation, we recommend configuring the SSO solution for WebCenter Sites and WebCenter Portal using Oracle Access Manager (OAM).

If you are using OAM, make sure it is configured to pass the user principal name in the *ORACLE_REMOTE_USER* header. Both *ObSSOCookie* and *OAM_REMOTE_USER* must be set to active on the OAM Identity Asserter on the WebLogic Server.

Configuring the GUID Attribute in the Identity Store

Specify the GUID attribute value to ensure that the value that is used in the identity store matches the value configured in the LDAP authentication provider. This value is configured in the *jps-config.xml* file.

Set the GUID for your LDAP authenticator in the *jps-config.xml* file.

```
<serviceInstance provider="idstore.ldap.provider" name="idstore.ldap">
        <property value="oracle.security.jps.wls.internal.idstore.WlsLdapIdStoreConfigProvider"
name="idstore.config.provider"/>
        <property value="oracle.security.idm.providers.stdldap.JNDIPool"
name="CONNECTION_POOL_CLASS"/>
        <property value="GUID=uuid" name="PROPERTY_ATTRIBUTE_MAPPING"/>
</serviceInstance>
```

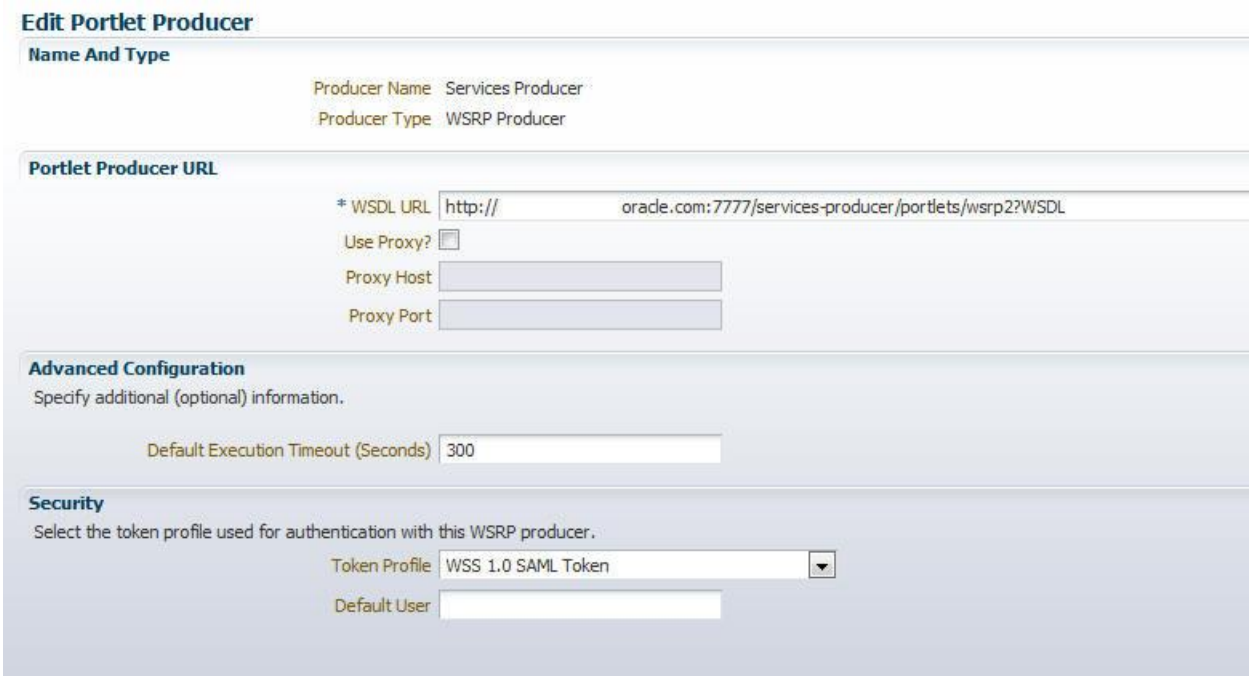
Configuring SSO for Discussion Server

To use Discussion services, configure the Discussion Server for SSO before ordering the Authenticators in WebLogic Server (using the Default Authenticator and Default Identity Asserter). It is also possible to use WLST to configure SSO for the Discussion Server. For details, see Configuring the Discussion Server for SSO in the *Oracle Fusion Middleware Administrator's Guide for Oracle WebCenter Portal*.

Registering WebCenter Services Exposed as WSRP Portlets

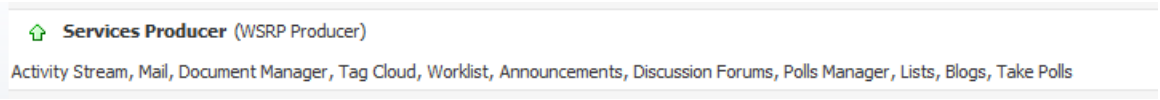
To import the WSRP portlets that expose WebCenter Services into the Pagelet Producer, simply register the Services Producer as a WSRP Producer in the Pagelet Producer using Enterprise Manager, WLST or the Pagelet Producer Administrative Console.

The screenshot below is an example of WSRP endpoint registration in the Pagelet Producer Administrative Console. For details, see Registering WSRP and Oracle JPDK Portlet Producers in the *Oracle Fusion Middleware Administrator's Guide for Oracle WebCenter Portal*.



The Token Profile in the Security section of this page must be set to WSS 1.0 SAML Token to support the Services Producer. You should also enter a suitable execution timeout for requests made by the Pagelet Producer to the Services Producer (the default is 30 seconds).

Once registration is complete, the Services Producer will appear as a resource in the Pagelet Producer Administrative Console and all WSRP portlets exposed by the Producer will be listed in the pagelets collection for the resource:



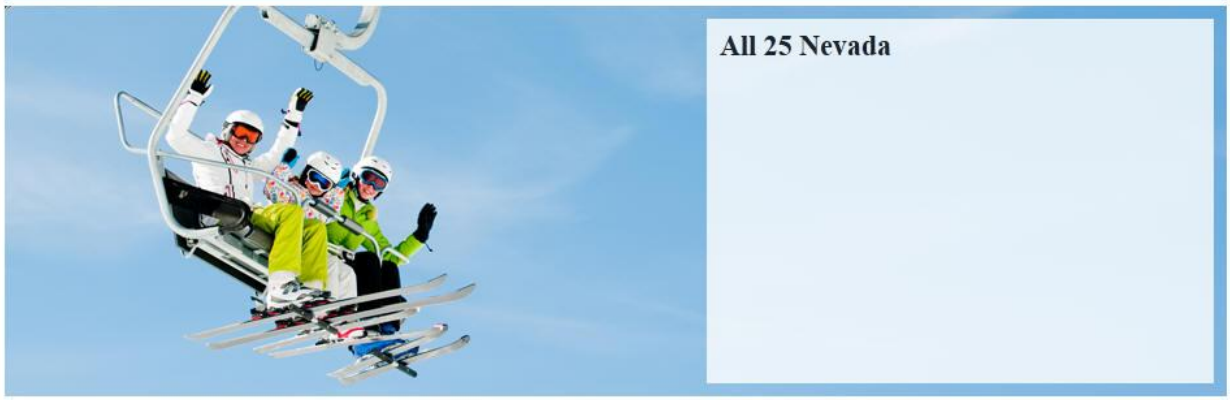
Adding Pagelets to Sites Pages

Any pagelet exposing WebCenter Services can be added to a WebCenter Sites page following the steps described in Adding Pagelets to Oracle WebCenter Sites. The screenshot below shows the Document Manager Service embedded into a page of the AviSports sample site in WebCenter Sites.

Note: Auto-generated WSRP resources and pagelets such as the ones associated with the Services Producer cannot be modified. To add an Injector or make other modifications, you must create a version that can be edited. Choose the resource in the Pagelet Producer Administrative Console and click Copy. The cloned version of the resource can be edited and various elements such as Injectors can be added to customize pagelet functionality.

avisports

HOME SURFING SKIING RUNNING TENNIS BASEBALL



> weblogic

Filter Advanced

File View New Wiki Document Upload Download

Name	Modified By	Last Modified	Like	Description
Blogs	weblogic	5/21/12 3:28 PM		
Public	sysadmin	5/21/12 3:45 AM		
SharepointTests	weblogic	5/21/12 9:02 AM		

Consuming Applications as Pagelets

Example: Consuming E-Business Suite 11i

This section provides detailed instructions for consuming the Oracle E-Business Suite 11i Order Information module UI as a pagelet. It covers autologin, navigation suppression, restyling, and URL rewriting.

This example was created using an EBS 11i instance that is protected with Oracle SSO. The following common terms and variables are used in the example:

- %EBS_HOST%: root URL of the EBS host, e.g. http://my-ebs.company.com:port/
- %OSSO_HOST%: root URL of the OSSO host, e.g. http://sso.company.com:port/

Create Resource for Basic URL Mapping (Proxy)

The following steps set up the basic URL mapping for a new EBS resource in the Pagelet Producer:

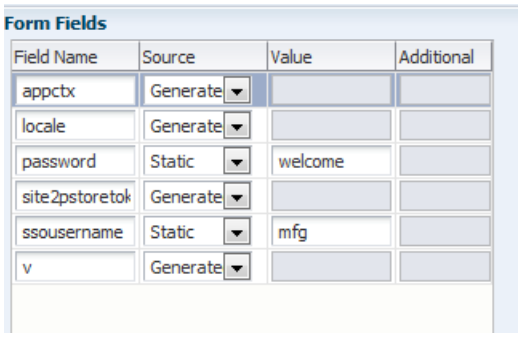
1. Create new "Web" resource with the following settings:
    - Name: EBS 11 (or your choice)
    - Source URL: %EBS_HOST%
      (**Note**: Make sure the URL is not more specific to avoid missing URLs used in the UI)
    - Destination URL: /ebs11/ (or your choice)
    - URL Rewriting: on
    - DHTML Rewriting: on
2. (Click Save.)

Configure Autologin

This step assumes that the EBS System is protected by OSSO and that the Pagelet Producer will be configured to provide form Autologin rather than participate in SSO (a common scenario if the web site is not protected by SSO). EBS credentials may be shared for all users (good for testing) or stored in the Pagelet Producer credential vault for each user.

1. Create a new "Web" resource with the following settings:
    - Name: OSSO (or your choice)
    - Source URL: %OSSO_HOST%
    - Destination URL: /osso/ (or your choice)
    - URL Rewriting: on
    - DHTML Rewriting: off
2. (Click Save.)
3. Under the newly created OSSO resource, configure Autologin using Form Login option as follows:
    - Login Form Identification: URL - %OSSO_HOST%/sso/jsp/login.jsp
    - Form Submit Location: URL - %OSSO_HOST/sso/auth – POST
    - Form Fields:
        - appctx: Generated
        - locale: Generated
        - password: Static or User Vault
        - site2pstoretoken: Generated
        - ssousername: Static or User Vault

- v: Generated



4. (Click Save.)

Note: The names of login form fields are usually obtained by investigating the HTML source for the login page that protects the backend resource.

Create Pagelet

This step associates a pagelet with the Order Status page in the EBS Order Information module. To create a new pagelet, select the Pagelets section under the EBS resource you created and click the Create icon. Create the new pagelet with the following settings:

- Name: order_status (or your choice)
- Library: ebs11 (or your choice)
- URL Suffix: OA_HTML/RF.jsp?function_id=1005664&resp_id=22480&resp_appl_id=660& security_group_id=0&lang_code=US
- Refresh Inline: off

You can test the configured pagelet by accessing it via the REST link on the Documentation page. You should be taken to the Sales Order page without logging in (unless the pagelet uses the User Vault to store credentials and you are accessing it for the first time). From this page, you can use Simple Search to locate orders, using '%' as a wildcard character.

Make Corrective Configurations

After a pagelet has been created, it often requires additional features (Parser, Injector and/or Clipper) to either address issues with URL re-writing in the proxied markup, or simply to modify the markup for style or to hide unnecessary elements. This section describes corrective configurations that were applied to the sample pagelet that exposes the Order Information UI in EBS.

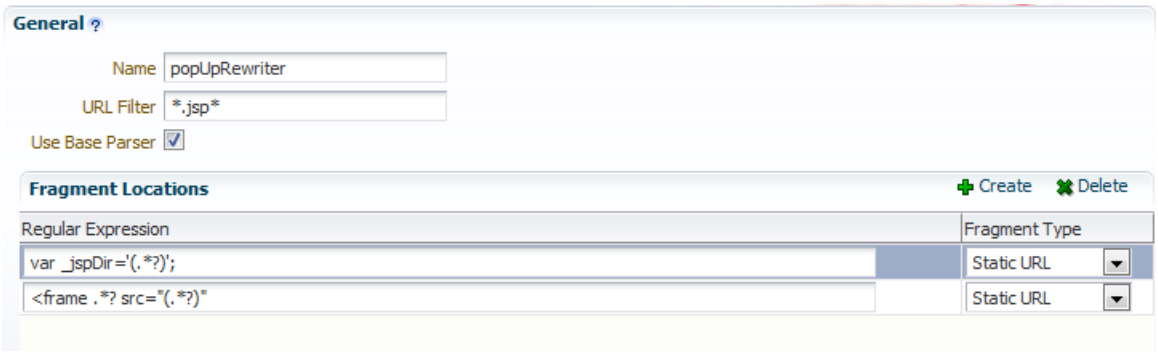
Create a Pluggable Parser for Popup URL Rewriting

Parsers allow you to supplement or change the built-in logic for parsing content and finding URLs that need to be rewritten. When built-in parsers fail to identify URLs, custom parsers can be used to correct this failure. In the case of the EBS 11i UI, this step is required to fix popup handling in Advanced Search.

Under the EBS Resource, create a new Parser with the following settings:

- Name: popupRewriter (or your choice)
- URL Filter: *.jsp*
- Use Base Parser: on
- Fragment Locations

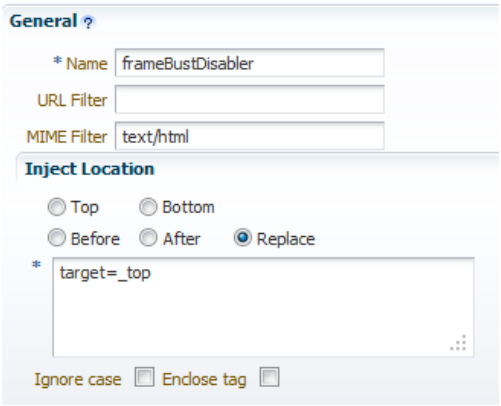
- `var _jspDir='(.*?)';` | type Static URL
- `<frame .? src="(.?)"` | type Static URL



Create an Injector to Disable Frame-busting

Injectors insert content into a specified location in the proxied resource page. The content can be any text, including HTML, CSS, JavaScript, and pagelet declarations. An empty injector may also be used to remove unwanted content from the page. The following injector was created for the EBS pagelet to prevent it from taking over the page, which was happening on the first displayed page where the user is asked to choose responsibility.

Under the EBS resource create a new Injector with the following settings:



- General
  - Name: frameBustDisabler
  - URL Filter: <none>
  - MIME Filter: text/html
  - Inject Location: Replace `target=_top`
- Content: (Text) `alt=''`

Create an Injector to Auto-Resize the iFrame

In order to provide a more seamless integration into the consuming page, the pagelet iFrame can be configured to behave more like inline content by dynamically resizing to accommodate its contents. For details, see Enabling IFrame Auto-Resizing in this document.

Set Up Clipping by JavaScript Injection

One option for suppressing page elements exposed as a pagelet is to create a clipper using graphical or RegEx-based clipping, available in the Pagelet Producer. However, the recommended approach is to use custom JavaScript that is injected into the proxied page. This approach is often more flexible as it can easily accommodate the need to dynamically adjust rules that identify the element(s) to suppress or accommodate rules for suppressing multiple elements on the page.

Custom JavaScript can be added to the proxied resource page using an injector. To suppress EBS 11i standard header and navigation elements, use the following injector definition to insert a snippet of JavaScript that hides the unnecessary elements at page load time. (The standard Clipper feature in the Pagelet Producer is not used due to the complexity of the EBS web UI.)

Under the EBS Resource, create a new Injector with the following settings:

- General
  - Name: nav_suppressor (or your choice)
  - URL Filter: <none>
  - MIME Filter: text/html
  - Inject Location: Before `</body>`
- Content: (Text - paste the content below)

```
<script type="text/javascript">
// this script is injected into the page by ensemble
// it hides header and footer tables at page load

// register handler function to the page load event
if (window.addEventListener) {
  window.addEventListener('load', hideHeaderFooter, false);
} else if (document.attachEvent) {
  window.attachEvent('onload', hideHeaderFooter);
}

// this function does the actual hiding
function hideHeaderFooter() {
  var form = document.forms[0];
  if (typeof(form) == 'undefined') return;
  // main span is form's second child span
  var spansFound = 0;
  var mainSpan = null;
  for (var i = 0; i < form.childNodes.length; i++) {
    var child = form.childNodes[i];
    if (child.tagName && child.tagName.toLowerCase() == 'span') {
      if (++spansFound == 2) {
       mainSpan = child;
       break;
      }
    }
  }
  if (typeof(mainSpan) != "undefined" && mainSpan != null) {
    for (var i = 0; i < mainSpan.childNodes.length; i++) {
      var child = mainSpan.childNodes[i];
       if (child.tagName && child.tagName.toLowerCase() == 'table') {
         child.style.display = 'none';
       }
    }
  }
}
// call this function directly
hideHeaderFooter();
</script>
```

Create Injector for Pagelet Restyling

To make the EBS pagelet fit better visually in the consuming page, an injector can be used to add styles to override the original CSS.

The example injector definition below shows how to do implement restyling for the AviSports sample web site in WebCenter Sites.

Note: In order for the override to work correctly, it is important that the style definitions supplied by the injector come after the styles defined by the backend application. In the example below, the content is injected into the end of the <HEAD> section on the page.

Under the EBS Resource, create a new Injector with the following settings:

- General
  - Name: avisports_styles
  - URL Filter: <none>
  - MIME Filter: text/html
  - Inject Location: Before `</head>`
- Content: (Text - paste the content below)

```
<style>
* {
  font: 11px Arial,Helvetica,sans-serif;
}
body {
  background: url("http://sites-host:port/cs/avisports/images/BlueSliver.png") repeat-x scroll center
bottom transparent;
}
.OraTableCellText, .x1l, .OraTableCellNumber, .x1n, .OraTableCellIconButton, .x1p,
.OraTableCellSelect, .x55,
.OraTableControlBarTop, .x1i, .OraTableControlBarBottom, .x1j {
    background-color: #eceef1;
    border-color: #0b2a55;
}
.OraTableColumnHeader, .x1r, .OraTableSortableColumnHeader, .x20, .OraTableSortableHeaderLink, .x23 {
    background-color: #0b2a55;
    border-color: #F7F7E7;
    color: #336699;
}
.OraTableHeaderLink, .x24, .OraTableColumnHeaderNumber, .x1s, .OraTableColumnHeaderIconButton, .x1t {
    background-color: #0b2a55;
    color: #ffffff;
}
.p_InContextBrandingText, .x2l, .OraHGridNavRowInactiveLink, .x3v, .OraNavBarInactiveLink, .x42,
.OraBINavBarInactiveLink, .x7n, .OraBINavBarInactiveLink_small, .x7o {
    color: #000000;
}
.OraLink:link, .xd:link, .OraBreadCrumbs, .xh, .OraBulletedList a, .xj a, .OraLinkText, .x2v,
.OraHGridNavRowActiveLink, .x3u, .OraNavBarActiveLink, .x41, .OraBINavBarActiveLink, .x7l,
.OraBINavBarActiveLink_small, .x7m {
    color: #0b2a55;
}
.OraBreadCrumbs a, .xh a {
    color: #0b2a55;
}
</style>
```

Test Pagelets

The EBS11 resource now includes the pagelet, the custom injectors, and the custom parser.

Test the pagelet by browsing to the Documentation page and using the URL shown under "Access Pagelet using REST":

Access Pagelet using REST:
http://[redacted]:8889/pagelets/inject/v2/pagelet/ebs11/order_status?content-type=iframe&csapi=true&ifheight=300px

Troubleshooting

If some images are not rendered properly (this often happens in IE), make sure DHTML rewriting is enabled. (The DHTML Rewriting setting is on the General page for the resource.)

Final Result

The screenshots that follow show the EBS 11i Order Information screens embedded into a page of the AviSports sample site in WebCenter Sites using a REST URL for pagelet access in an iFrame.

avisports

HOME SURFING SKIING RUNNING TENNIS BASEBALL

Sales Orders, Quotes

Printable_Page

Simple Search

Advanced Search

Search By Sales Order % Go

Views

View Go Personalize

Previous 1-10 Next 10

Sales Order	Quote Number	Customer PO	Order Type	Ordered Date	Quote Date	Status	Customer Name
777798		AU_IT_PO_GM	AU-STANDARD	21-May-2001 11:29:44		Booked	Pontiac
777788		AU_IT_PO_GM	AU-STANDARD	02-Apr-2001 12:28:41		Booked	Pontiac
13782			Web	14-Oct-2003 23:06:42		Closed	Peter Tong
13811			Web	20-Oct-2003 13:49:10		Closed	Gigi Wong
13810			Web	20-Oct-2003 13:48:51		Closed	Gigi Wong
14057			Web	18-Nov-2003 12:32:12		Closed	w_cw, Inc.
54868			Mixed	17-Feb-2003 16:52:43		Closed	Big 4 Rental
54821			Mixed	13-Feb-2003 11:20:23		Closed	Big 4 Rental
54820			Mixed	13-Feb-2003 11:19:16		Closed	Big 4 Rental
54819			Mixed	13-Feb-2003 11:17:15		Closed	Big 4 Rental

Previous 1-10 Next 10

Printable_Page

avisports

HOME SURFING SKIING RUNNING TENNIS BASEBALL

Order Status >

Sales Order 13782

Printable_Page

General

Customer Name	**Peter Tong**
Customer PO	
Order Date	**14-Oct-2003 23:06:42**
Need By Date	**14-Oct-2003 23:06:42**
Booked Date	**14-Oct-2003 23:06:43**
Status	**Closed**
Quality Plan	View Quality Plan
Invoice Information	View Invoice Information

Shipping

Freight Terms	**Prepay & Add**
Shipment Priority	**Standard Priority**
Delivery Summary	View Delivery Summary
Delivery Line Details	View Line Details

Billing

Bill To	**7695**
	San Jose, CA, 95106, US
Payment Terms	**IMMEDIATE**
Price List	**Corporate**
Total	**2,053.90 US dollar**
Pricing Details	View Pricing Details

Lines

Line Num	Item	Item Description	Ordered Quantity	UOM	Fulfilled Quantity	Unit Price	Extended Price	Line Type	Status	Delivery Name	Expected Ship Date	Details	Return Request	Line Invoice Detail
1.1	AS18947	Sentinel Deluxe Desktop	1	Ea	1	1,870.55	1,870.55	Standard (Line Invoicing)	Closed	59158	14-Oct-2003 23:59:59			

Printable_Page

avisports

HOME SURFING SKIING RUNNING TENNIS BASEBALL

Oracle EBS 11i

Order Status > Sales Order 13782 >

Tax Details | Charge Details | Price Adjustment Details | Quality Information

Additional Information

Sales Order	**13782**	Line	**1.1**
Item	**AS18947**	Sub Total	**1,969.00**
Price List	**Corporate**	Discount	**<98.45>**
UOM	**Ea**	Charges	**15.00**
Ordered Quantity	**1**	Tax	**168.35**
Currency	**US dollar**	Line Total	**2,053.90**

Tax Details

Return to Top

Tax Code	Rate %	Amount
Location	9	168.35

Charge Details

Return to Top

Charge Name	Type	Rate%	Amt/Unit	Charged	Auto	Refundable	Reason
Freight Costs	FREIGHT	15	15	15	Y		
Transportation Rate	FTEPRICE	0		0	Y	Y	

Price Adjustment Details

Return to Top

Modifier Level Code	Modifier Number	Modifier Name	Type	Application Method	Rate	Adjusted Amt	Extended Adjusted Amt	Auto	Reason Code	Reason Text
LINE	D0007A	5% Discount by Order Type	Discount	Percent	5	-98.45	-98.45	Y		

Quality Information

Return to Top

To view quality plan for this line, click the view quality plan link View Quality Plan