

Integrating Oracle Application Express with BI Publisher

*An Oracle White Paper
April 2009*

Integrating Oracle Application Express with BI Publisher

Introduction	3
Web service Support in Application Express.....	4
Prerequisites	5
Flexible Web service API (flex_ws_api package).....	5
Oracle Business Intelligence Publisher Enterprise	5
Oracle Application Express 3.2.....	5
Building a New Web Application Interface to Bi Publisher	6
Create an Application with One Blank Page	6
Create Application Substitution Strings	6
Create Authentication Scheme.....	7
Create a Process to Call getFolderContents	10
Create Download Report Page	13
Modify Shared Folders Report to Link to Download Page	16
Conclusion.....	18

Integrating Oracle Application Express with BI Publisher

INTRODUCTION

Oracle Application Express – a feature of the Oracle Database – is a powerful and easy to use web application development platform. With Oracle Application Express, you can quickly develop and deploy applications in a matter of hours, often without writing a single line of code.

Oracle Business Intelligence Publisher (BI Publisher), formally XML Publisher, is an enterprise reporting solution to author, manage, and deliver all types of highly formatted documents eliminating the need for costly point solutions. End users can easily design report layouts using familiar desktop tools, dramatically reducing the time and cost needed to develop and maintain reports.¹

Like chocolate and peanut butter, Oracle Application Express and Oracle BI Publisher go well together. You can make reports and standalone queries available as high-fidelity downloads in an Application Express application through BI Publisher integration starting in Application Express 3.0.² This solution requires you to define your report within Application Express and also requires configuration of the Application Express instance.

What if you already have reports defined within BI Publisher and you want to make those available in an Application Express Application? You can do this easily by utilizing the PublicReportService³ Web service that is installed by default when you install BI Publisher.

Note: If you want to schedule a report for delivery to an ftp server or send as an attachment to an email, there is a good blog post⁴ on how to use the scheduleReport operation of the PublicReportService to do just that.

The PublicReportService has an operation called runReport which returns the actual report in the requested format with the binary data encoded in base64. It also has an operation called validateLogin which, given a username-password combination, will return true if the user is a valid BI Publisher user (and false if they are not). The last component needed is the operation called getFolderContents

¹ [BI Publisher Data Sheet](#)

² [Configure PDF Printing in Application Express](#)

³ [PublicReportService Documentation](#)

⁴ [Tyler Muth's Blog](#)

which, given a path, will return everything in that path including reports and other folders.

By combining these three operations, it is possible to build an application entirely in Application Express that will give a similar experience to logging in to the Web interface of BI Publisher. This paper will describe how to build an Application Express application that will allow you to authenticate as a BI Publisher user, browse the folders and reports that the user has access to, and finally retrieve the reports in multiple formats, all from within the application.

WEB SERVICE SUPPORT IN APPLICATION EXPRESS

Application Express has supported consuming Web services since version 1.5. This initial support provided a wizard interface to build forms and reports on a Web service given the location to a WSDL document. This only worked with RPC literal-style Web services, which were prevalent at the time. Starting with version 3.0, enhanced support for Web services has been provided including support for document-style Web services, support for basic authentication, as well as support for interacting with Web services that have an HTTPS (SSL) endpoint.

A Manual Web Reference⁵ feature was also introduced in Application Express version 3.0. With manual Web references, the developer supplies information about the Web service such as the endpoint, the request envelope, and the action of the operation. The response from the Web service can be stored in a CLOB column of an Application Express collection.⁶ The XML document in the collection can then be queried in a standard report region or particular nodes of the XML document can be parsed by a PL/SQL process and inserted into a table or used to populate session state in the application.

There are times where you may need to interact with Web services in an Application Express application but you cannot use either the WSDL based support or the Manual Web references feature. An example is a Web service that has binary parameters. Another example is when you need to call a Web service as part of an authentication scheme. You can still interact with the Web services described in these examples by using PL/SQL in Application Express.

This paper will describe interacting with the BI Publisher PublicReportService Web service entirely in PL/SQL because you will need to handle binary parameters as well as build an authentication scheme that interacts with the Web service. The flex_ws_api⁷ PL/SQL package will be used to help interact with the PublicReportService.

⁵ [Implementing Web Services in Application Express](#)

⁶ [Using Collections](#)

⁷ [Flexible Web Service API](#)

PREREQUISITES

Flexible Web service API (flex_ws_api package)

The flex_ws_api package was created specifically for Application Express to interact with Web services when WSDL based and manual Web references cannot be used. The API has a procedure and a function that can call a Web service and store the results in a collection or return the results in an xmltype⁸. The flex_ws_api package has functions to return specific nodes of the response from the Web service as a VARCHAR2 or CLOB data type. It also has functions to work with binary data. There is a function that takes a BLOB and returns a base64 encoded CLOB as well as a function to do the reverse.

To use the flex_ws_api package, you must download it and then compile it in the schema associated with your workspace. You can get the flex_ws_api package at the following URL:

<http://jastraub.blogspot.com/2008/06/flexible-web-service-api.html>

If you are using Oracle Database 11gR1 or later, you must enable your database user to make network callouts to the host where BI Publisher is installed. Examples of how you can enable all hosts, or simply just localhost, are available in the Application Express Installation Guide:

http://download.oracle.com/docs/cd/E14373_01/install.32/e13366/otn_install.htm#BEHGBHDF

Replace references to APEX_030200 with the schema where the flex_ws_api package is installed.

Oracle Business Intelligence Publisher Enterprise

The PublicReportService Web service API included with BI Publisher differs slightly from version 10.1.3.3.2 to 10.1.3.4.0. The processes in this paper are written in such a way that it can be used with any version from 10.1.3.3.2 to 10.1.3.4.0. You will need to have access to a BI Publisher instance along with credentials to login to the BI Publisher Web interface.

Oracle Application Express 3.2

You need to pass the username and password of the BI Publisher user with every Web service request. You do not want the password available in clear text so use the new session state encryption feature of Application Express 3.2.

⁸ [Oracle XML DB Developer's Guide](#)

BUILDING A NEW WEB APPLICATION INTERFACE TO BI PUBLISHER

The application being built can be used to authenticate to BI Publisher, browse folders and reports available to the authenticated user, and then download those reports directly.

Please note that this tutorial cannot be demonstrated on apex.oracle.com because it does not support external network call-outs. You will need to use your own instance to be able to execute the resulting application.

Create an Application with One Blank Page

Start by creating a new application with one blank page. To create an Application Express application:

1. Login to your Application Express instance.
2. Select **Application Builder**
3. Click **Create >**
4. Supply the information as required by the Create Application wizard
5. Add one blank page on the Pages step of the wizard
6. Once you have created the application, if your environment requires a proxy server to reach pages on the Internet, supply the proxy server in the Proxy Server field of the Application Definition (Shared Components > Definition - under Application)

Create Application Substitution Strings

Your application should be portable so that you can configure it to work with any BI Publisher instance. To accomplish this, create substitution strings for the BI Publisher host, port, version, and namespace. To create an application substitution:

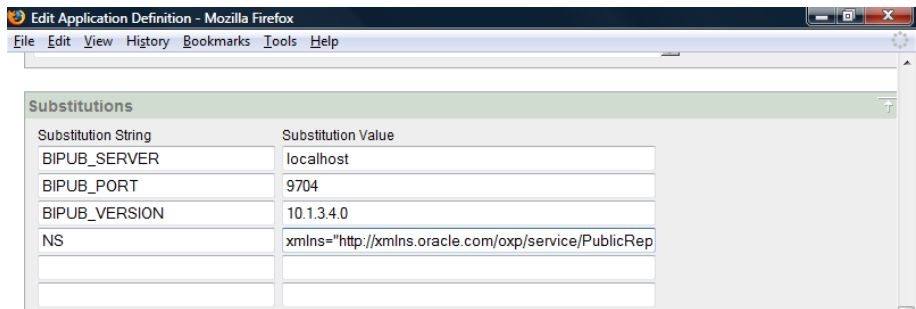
1. From within your application, navigate to Shared Components
2. Select **Definition** (under Application)
3. Click the **Substitutions** quick link (at the top of the page to focus on just the Substitutions region)
4. Enter **BIPUB_SERVER** in the first Substitution String field
5. Enter the host (without port number) of the BI Publisher instance you have access to. For example, localhost.
6. Repeat steps 1 – 5 above to add the additional substitutions:
 - a. **BIPUB_PORT**, example 9704
 - b. **BIPUB_VERSION**, example 10.1.3.4.0

c. **NS**, example

xmlns=http://xmlns.oracle.com/oxp/service/PublicReportService

(leave blank if not 10.1.3.4.0)

7. Click **Apply Changes**



Create Authentication Scheme

Once the shell of the application has been created, you need to create a function that calls the PublicReportService validateLogin operation. This function will be used as the authentication function for the custom authentication scheme for the application. Compile the function in the following code listing in the schema associated with your workspace (you can use either the SQL Workshop SQL Commands feature or Oracle SQL Developer).

```
create or replace function validate_login(
    p_username in varchar2,
    p_password in varchar2) return boolean
is
    l_validate      varchar2(48);
    l_env           clob;
    l_xml           xmltype;
    l_ns            varchar2(4000) default null;
    l_url           varchar2(4000);
begin

    l_env := '<soapenv:Envelope
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:pub="http://xmlns.oracle.com/oxp/service/PublicReportService">
    <soapenv:Header/>
    <soapenv:Body>
        <pub:validateLogin
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
            <userID
xsi:type="xsd:string">' || p_username || '</userID>
            <password
xsi:type="xsd:string">' || p_password || '</password>
        </pub:validateLogin>
    </soapenv:Body>
</soapenv:Envelope>';

    l_url := 'http://' || v('BIPUB_SERVER');

    if v('BIPUB_PORT') is not null then
```

```

        l_url := l_url||':'||v('BIPUB_PORT');
    end if;

    l_url := l_url||'/xmlpserver/services/PublicReportService';

    l_xml := flex_ws_api.make_request(
        p_url => l_url,
        p_envelope => l_env);

    l_validate :=
flex_ws_api.parse_xml(l_xml,'//validateLoginReturn/text()',v('NS'));

    if l_validate = 'true' then
        return true;
    else
        return false;
    end if;

exception when others then
    return false;
end validate_login;
/
show errors

```

Now, create the custom authentication scheme and switch the scheme that the application currently uses to this new scheme. To create the custom authentication scheme:

1. Click **Shared Components** from the Application Builder home page
2. Click **Authentication Schemes** under Security
3. Click the **Create >** button
4. Select **From scratch** and click **Next >**
5. Enter **bipublisher** for Name and click **Create Scheme**
6. Click the **BIPUBLISHER** scheme to edit it
7. Select **101** (the login page) for Session Not Valid Page under Page Session Management and remove -BUILTIN- from Session Not Valid URL text area
8. Enter the following PL/SQL in the Authentication Function text area under Login Processing:

```

return validate_login;

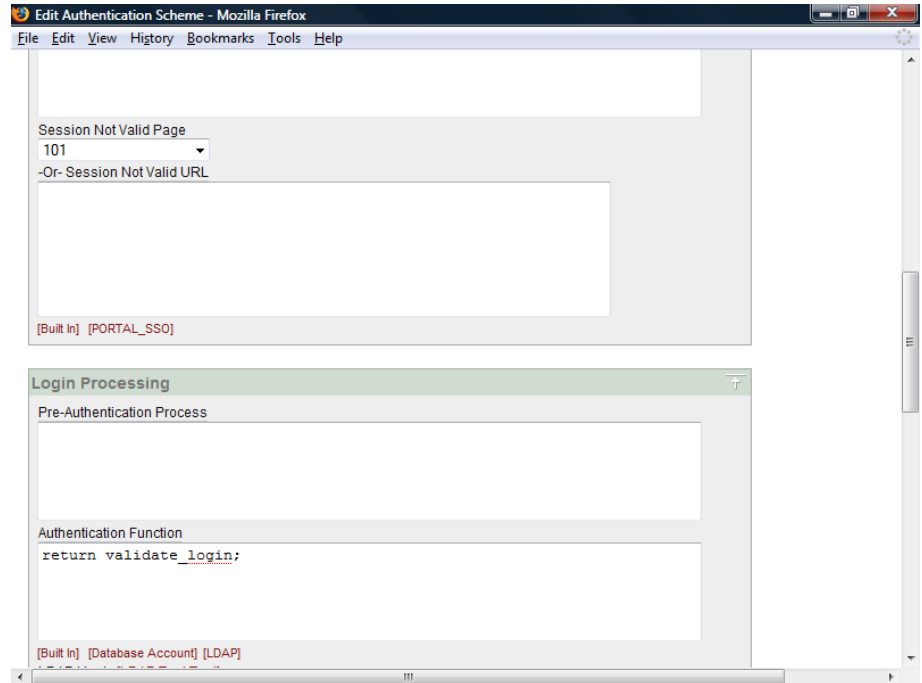
```
9. Enter the following in the Logout URL text area:

```

wwv_flow_custom_auth_std.logout?p_this_flow=&APP_ID.&amp;p_next_flow_page_sess=&APP_ID.:1

```
10. Click **Apply Changes**
11. Click the **make current** link (under status) to the right of the BIPUBLISHER authentication scheme

12. Click **Make Current**



Before testing the new authentication scheme you need to make a couple of changes to the login page. The application needs to be able to reference the username and password in subsequent calls to the PublicReportService. Change the login page to not clear session state and then change the password item to encrypt the value. To make the changes to the login page:

1. Click **page 101** from the Application Builder home page to edit it
2. Click **Clear Page(s) Cache** under Page Processing Processes
3. Click **Delete**
4. Click **P101_PASSWORD** under the Items area under Page Rendering
5. For Display As, select **Password (submits when Enter pressed)**
6. Under Security, for Store value encrypted in session state, select **Yes**
7. Click **Apply Changes**


Test the application to see if you can login. Return to the Application Builder home page and click Run. Login as a BI Publisher user. (By default, there is an Administrator/Administrator user.)

Next, create a page to call getFolderContents operation and write a report on the results.


Create a Process to Call getFolderContents

You create a PL/SQL before header process to call the getFolderContents and store the results in a collection. First, create an item and computation to hold the current path.

To create the item:


1. Click **Page 1** from the Application Builder home page to edit it
2. Click the create icon  in the Items area under Page Rendering
3. Select **Hidden** for the Item Type and click **Next >**
4. Leave **Hidden and Protected** with the default value and click **Next >**
5. Enter **P1_SHARED_PATH** for Item Name
6. Select **Page 1** for Region and click **Next >**
7. Accept the default and click **Next >**
8. Click **Create Item**

To create the computation:

1. Click the create icon  in the Computations area under Page Rendering
2. Select **Item on This Page** for Location and click **Next >**
3. Select **P1_SHARED_PATH** for Compute Item
4. Select **PL/SQL Function Body** for Computation Type and click **Next >**
5. Enter the following PL/SQL in the Computation Text area:

```
if :P1_SHARED_PATH is null then
    return '/';
else
    return
    wwv_flow_utilities.url_decode2(:P1_SHARED_PATH);
end if;
```
6. Click **Next >**
7. Click **Create**

To create the process:

1. Click the create icon  in the Processes area under Page Rendering
2. Select **PL/SQL** for process category and click **Next >**
3. Enter **call getFolderContents** in the Name field
4. Ensure that **On Load – Before Header** is selected for Point and click **Next >**

5. Enter the following PL/SQL in the text area and click **Create Process**:

```
declare
  l_env clob;
  l_url          varchar2(4000);
begin

  l_env := '<soapenv:Envelope
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:pub="http://xmlns.oracle.com/oxp/service/PublicReportService">

  <soapenv:Header/>
  <soapenv:Body>
    <pub:getFolderContents
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
      <folderAbsolutePath
xsi:type="xsd:string">' || :P1_SHARED_PATH || '</folderAbsolutePath>
      <userID
xsi:type="xsd:string">' || :APP_USER || '</userID>
      <password
xsi:type="xsd:string">' || :P101_PASSWORD || '</password>
    </pub:getFolderContents>
  </soapenv:Body>
</soapenv:Envelope>';

  l_url := 'http://' || v('BIPUB_SERVER');

  if v('BIPUB_PORT') is not null then
    l_url := l_url || ':' || v('BIPUB_PORT');
  end if;

  l_url :=
l_url || '/xmlpserver/services/PublicReportService';

  flex_ws_api.make_request(
    p_url => l_url,
    p_envelope => l_env,
    p_collection_name => 'SHARED_FOLDERS');


end;
```

You can hide the Page 1 region that was created when the application was created because it is no longer necessary. You do not want to delete it because it contains the P1_SHARED_PATH item. To hide the Page 1 region:

1. Click **Page 1** in the Regions area under Page Rendering
2. Navigate to the Conditions section
3. Select **Never** for Condition Type and click **Apply Changes**

Lastly, add a report region based on the response from the call to getFolderContents. You write the report against the collection called SHARED_FOLDERS and use XML DB functions and procedures to extract the

XML. The report links to the same page and sets the P1_SHARED_PATH item to the path of the folder that is clicked. To create the report region:

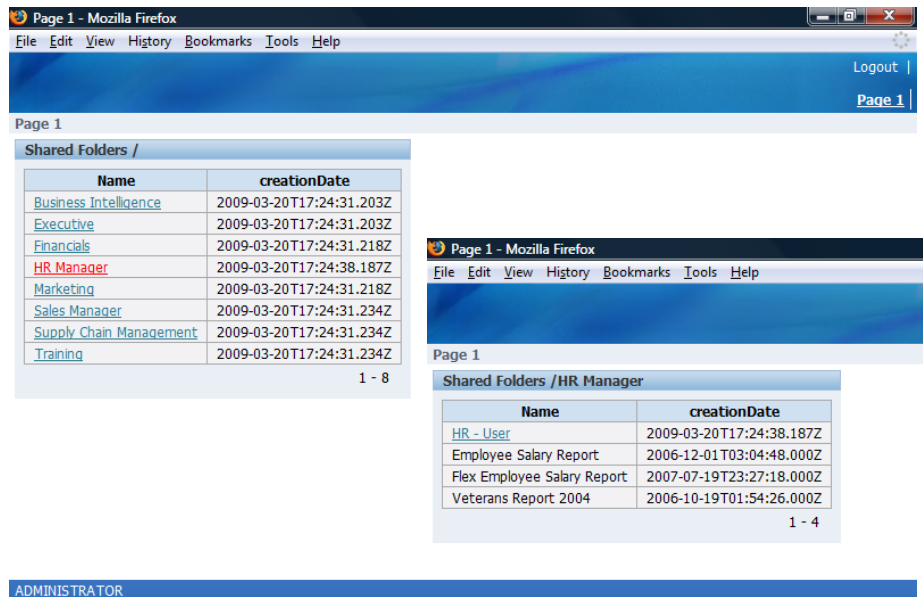
1. Click the create icon  in the Regions area
2. Select **Report** for region type and click **Next >**
3. Select **SQL Report** for Report Implementation and click **Next >**
4. Enter **Shared Folders & P1_SHARED_PATH.** for Title and click **Next >**
5. Enter the following SQL in the SQL Query text area:

```
select decode(extractValue(value(t), '/*/type', v('NS')),
'Report', extractValue(value(t), '/*/displayName', v('NS'))
,
'

```

6. Click **Create Region**

Run the page, logging in again if necessary, and observe the results. Click on a folder name to test that you can view the contents of the folders.





Create Download Report Page

You need to create a page to download the report. A before header process on the page calls the runReport operation of PublicReportService and gets the report back in base64 encoded binary data. The flex_ws_api package is used to convert the base64 character data into a blob and then the process prompts the browser to download the file. This requires three hidden items so the path to the report, format, and filename can be passed to the page and used in the process. Because the path to the filename is likely to have characters that need to be encoded in a URL prior to being passed, a computation needs to be created for the path item to decode the proper value.


To create the page:

1. Click **Create Page** > on the Application Builder home page
2. Select **Blank Page** for page type and click **Next** >
3. Enter **2** for Page Number and click **Next** >
4. Enter **Download Report** for Name and click **Next** >
5. Select **No** on the Tabs page and click **Next** >
6. Click **Finish**
7. Click **Edit Page**

To create the items:

1. First you create a region for the items to belong to. Click the create icon  in the Regions area.
2. Select **HTML** for region type and click **Next** >
3. Select **HTML** for HTML region container type and click **Next** >
4. Enter **Download** for Title and click **Create**
5. Click the create icon  in the Items area
6. Select **Hidden** for Item Type and click **Next** >
7. Select **Hidden and Protected** for Item Type and click **Next** >
8. Enter **P2_ABS_PATH** for Item Name and click **Next** >
9. Accept the defaults and click **Next** >
10. Click **Create Item**
11. Repeat steps 5 – 10 to create items named **P2_FORMAT** and **P2_FILENAME**


To create the computation to decode the path:

1. Click the create icon  in the Computations area under Page Rendering
2. Select **Item on This Page** for Location and click **Next** >

3. Select **P2_ABS_PATH** for Compute Item
4. Select **PL/SQL Function Body** for Computation Type and click **Next >**
5. Enter the following in the Computation text area:


```
return wwv_flow_utilities.url_decode2(:P2_ABS_PATH);
```
6. Click **Next >**
7. Click **Create**

Finally, create the before header process:

1. Click the create icon  in the Processes area under Page Rendering
2. Select **PL/SQL** for process category and click **Next >**
3. Enter **Download** for Name
4. Ensure that **On Load – Before Header** is selected for Point and click **Next >**
5. Enter the following PL/SQL in the text area and click **Create Process:**

```
declare
  l_mime      varchar2(48);
  l_name      varchar2(4000);
  l_env       clob;
  l_base64    clob;
  l_blob      blob;
  l_size      number;
  l_xml       xmltype;
  l_ns        varchar2(4000) default null;
  l_path      varchar(255) default '//multiRef';
  l_url       varchar2(4000);
begin

  l_env := '<soapenv:Envelope
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:pub="http://xmlns.oracle.com/oxp/service/PublicReportService"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">
  <soapenv:Header/>
  <soapenv:Body>
    <pub:runReport
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
      <reportRequest xsi:type="pub:ReportRequest">
        <attributeFormat
xsi:type="xsd:string">'||:P2_FORMAT||'</attributeFormat>
        <parameterNameValues
xsi:type="pub:ArrayOfParamNameValue"
soapenc:arrayType="pub:ParamNameValue[]" />
          <reportAbsolutePath
xsi:type="xsd:string">'||:P2_ABS_PATH||'</reportAbsolutePath>' ;
        if v('BIPUB_VERSION') = '10.1.3.4.0' then
          l_env := l_env||'<sizeOfDataChunkDownload>-
1</sizeOfDataChunkDownload>' ;
```

```

        l_path := '//runReportReturn';
    end if;
    l_env := l_env || '</reportRequest>'
        <userID
xsi:type="xsd:string">' || :APP_USER || '</userID>'
        <password
xsi:type="xsd:string">' || :P101_PASSWORD || '</password>'
        </pub:runReport>
    </soapenv:Body>
</soapenv:Envelope>';

    l_url := 'http://' || v('BIPUB_SERVER');

    if v('BIPUB_PORT') is not null then
        l_url := l_url || ':' || v('BIPUB_PORT');
    end if;

    l_url :=
l_url || '/xmlpserver/services/PublicReportService';

    l_xml := flex_ws_api.make_request(
        p_url => l_url,
        p_envelope => l_env );

    if l_xml.existsNode('//faultstring') = 1 then
        raise_application_error(-
20001,l_xml.extract('//faultstring/text()').getStringVal
());
    end if;

    l_ns := v('NS');

    l_mime :=
flex_ws_api.parse_xml(l_xml,l_path || '/reportContentType/
text()',l_ns);

    l_name := :P2_FILENAME;

    l_base64 :=
flex_ws_api.parse_xml_clob(l_xml,l_path || '/reportBytes/t
ext()',l_ns);

    l_blob := flex_ws_api.clobbase64blob(l_base64);

    l_size := dbms_lob.getlength(l_blob);

    http.init;

    owa_util.mime_header( nvl(l_mime,'application/octet'),
FALSE );
    http.p('Content-length: ' || l_size);
    http.p('Content-Disposition: attachment;
filename="' || replace(replace(l_name,chr(10),null),chr(13
),null) || '"');
    owa_util.http_header_close;
    wpg_docload.download_file( l_blob );

    apex_application.g_unrecoverable_error := true;

end;

```

6. Click **Create Process**

Modify Shared Folders Report to Link to Download Page

You now need to modify the Shared Folders report to link to the newly created download page and populate the hidden items you created on that page. You will also create a link in the region footer to return to the root folder.

To modify the Shared Folders report:

1. Navigate to the definition of page 1
2. Click **Shared Folders &P1_SHARED_PATH.** in the Regions Section
3. Replace the query in the Region Source text area with the following:

```
select decode(extractValue(value(t), '/*/type', v('NS'))),
'Report', extractValue(value(t), '/*/displayName', v('NS'))
,
'<a
href="f?p=&APP_ID.:1:&SESSION.:::P1_SHARED_PATH:|wwv_flow_utilities.url_encode2(extractValue(value(t), '/*/absolutePath', v('NS')))|'|>'||extractValue(value(t), '/*/displayName', v('NS'))|'|</a>') "Name"
, extractValue(value(t), '/*/creationDate', v('NS'))
"creationDate"
, decode(extractValue(value(t), '/*/type', v('NS'))
, 'Folder', '&nbsp;')
,
'<a
href="f?p=&APP_ID.:2:&SESSION.:::P2_ABS_PATH,P2_FORMAT,P2_FILENAME:|wwv_flow_utilities.url_encode2(extractValue(value(t), '/*/absolutePath', v('NS')))|'|,pdf,|'|extractValue(value(t), '/*/displayName', v('NS'))|'|.pdf">[PDF]</a> '
||'| <a
href="f?p=&APP_ID.:2:&SESSION.:::P2_ABS_PATH,P2_FORMAT,P2_FILENAME:|wwv_flow_utilities.url_encode2(extractValue(value(t), '/*/absolutePath', v('NS')))|'|,html,|'|extractValue(value(t), '/*/displayName', v('NS'))|'|.html">[HTML]</a> '
||'| <a
href="f?p=&APP_ID.:2:&SESSION.:::P2_ABS_PATH,P2_FORMAT,P2_FILENAME:|wwv_flow_utilities.url_encode2(extractValue(value(t), '/*/absolutePath', v('NS')))|'|,rtf,|'|extractValue(value(t), '/*/displayName', v('NS'))|'|.rtf">[Word]</a> '
||'| <a
href="f?p=&APP_ID.:2:&SESSION.:::P2_ABS_PATH,P2_FORMAT,P2_FILENAME:|wwv_flow_utilities.url_encode2(extractValue(value(t), '/*/absolutePath', v('NS')))|'|,excel,|'|extractValue(value(t), '/*/displayName', v('NS'))|'|.xls">[Excel]</a> '
||'| <a
href="f?p=&APP_ID.:2:&SESSION.:::P2_ABS_PATH,P2_FORMAT,P2_FILENAME:|wwv_flow_utilities.url_encode2(extractValue(value(t), '/*/absolutePath', v('NS')))|'|,ppt,|'|extractValue(value(t), '/*/displayName', v('NS'))|'|.ppt">[PPT]</a>') "Download"
from wwv_flow_collections c,

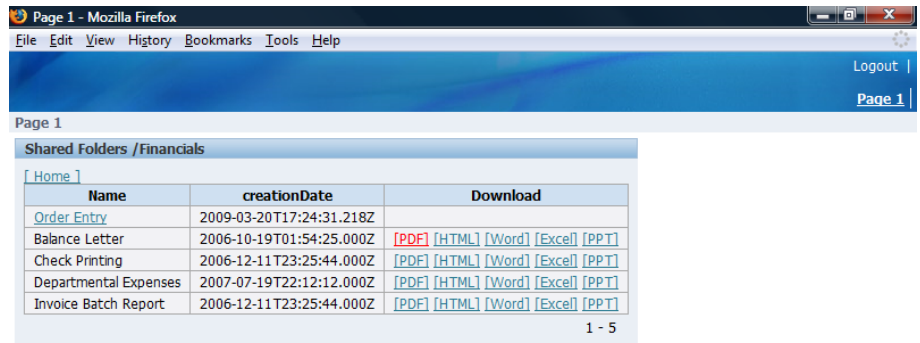
table(xmlsequence(extract(xmltype.createxml(c.clob001), decode(v('BIPUB_VERSION'), '10.1.3.4.0', '//getFolderContentsReturn', '//multiRef'), v('NS')))) t
where c.collection_name = 'SHARED_FOLDERS'
```

4. Navigate to the Region Header text area, under Header and Footer, and enter the following:

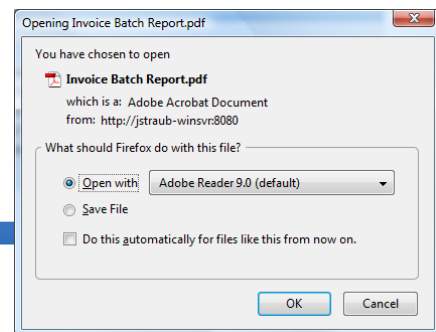
```
<a  
href=" f?p=&APP_ID.:1:&SESSION.:::P1_SHARED_PATH:%2F" > [  
Home ]</a>
```

5. Click **Apply Changes**

Run the application and navigate to a folder. Click one of the report download links. You should be prompted to open the file.



ADMINISTRATOR



**Oracle Application Express Free Trial
Service**

<http://apex.oracle.com>

Oracle Application Express Home

<http://apex.oracle.com/otn>

Oracle Application Express Studio

<http://apex.oracle.com/studio>

Oracle Application Express Forum on OTN

<http://apex.oracle.com/forums>

Oracle Application Express References

<http://apex.oracle.com/references>

CONCLUSION

Like chocolate and peanut butter, Application Express and BI Publisher go great together. By interacting with the validateLogin, getFolderContents, and runReport operations of the Oracle BI Publisher PublicReportService Web service you can make more use of your BI Publisher instance.

This paper described in detail how you can build an Application Express application that allows you to login, browse folders, and download reports from Oracle BI Publisher.



Integrating Oracle Application Express with BiPublisher

April 2009

Author: Jason Straub

Oracle Corporation
World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.

Worldwide Inquiries:
Phone: +1.650.506.7000
Fax: +1.650.506.7200
oracle.com

Copyright © 2009, Oracle. All rights reserved.

This document is provided for information purposes only and the contents hereof are subject to change without notice.

This document is not warranted to be error-free, nor subject to any other warranties or conditions, whether expressed orally or implied in law, including implied warranties and conditions of merchantability or fitness for a particular purpose. We specifically disclaim any liability with respect to this document and no contractual obligations are formed either directly or indirectly by this document. This document may not be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without our prior written permission.

Oracle, JD Edwards, and PeopleSoft are registered trademarks of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.