Oracle® Business Intelligence Publisher

Using Oracle BI Publisher Extension for Oracle JDeveloper

11*g* Release 1 (11.1.1)

E48204-01

March 2014

This document describes how to add a BI Publisher report to an Oracle Application Development Framework (ADF) application for Oracle Fusion Applications using Oracle JDeveloper.

It includes the following sections:

> **Note:** This document assumes familiarity with Oracle Application Development Framework (ADF) and Oracle JDeveloper. For more information about these see:
>
> - *Oracle Fusion Middleware Fusion Developer's Guide for Oracle Application Development Framework*
> - *Oracle JDeveloper 11g Online Help*

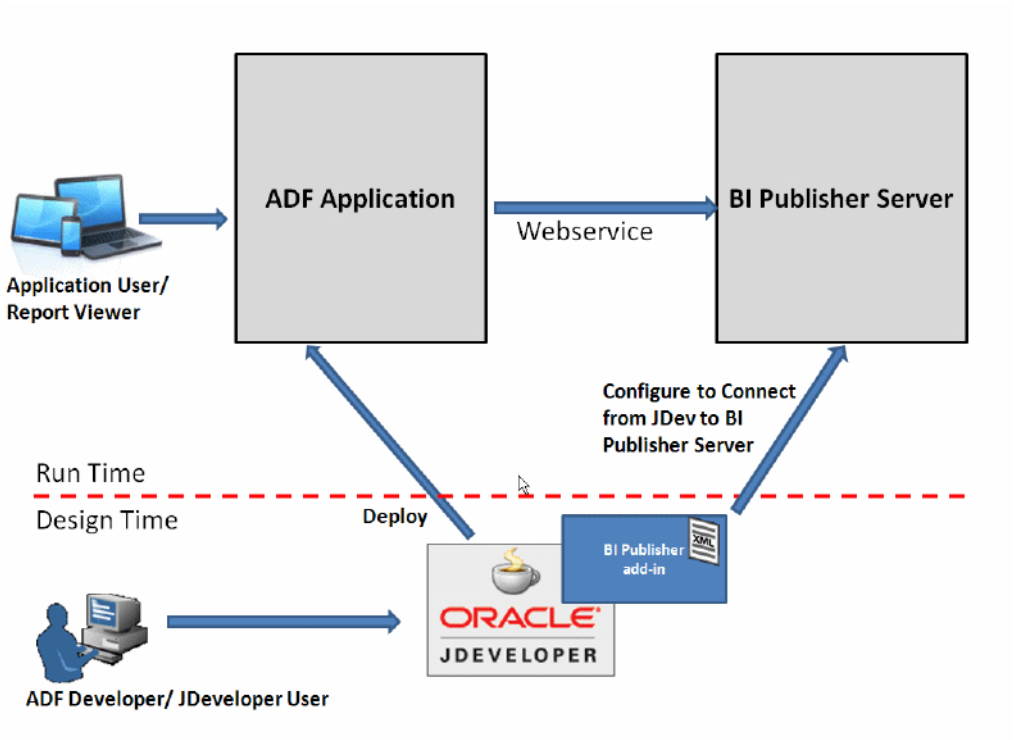
1 Including Oracle BI Publisher Reports in ADF Applications

BI Publisher provides an integration with Oracle Application Development Framework (ADF) that enables you to embed a BI Publisher report in an application (JSF) page for Oracle Fusion Applications. At runtime, your application sends a web service request to the BI Publisher server to run a report in the BI Publisher catalog and retrieve the output to display in your application page. The data for your report can be generated by the BI Publisher data engine, or you can push data from another source to the BI Publisher report formatting engine. The integration also supports passing parameters from your application page back to the BI Publisher server.

To facilitate development of your application, BI Publisher provides an extension to JDeveloper. The extension enables you to drag and drop a report region to your page and establish the connection between this region and the BI Publisher server. The extension also enables you to define aspects of the report format and to set properties for the region.

This integration is illustrated in Figure 1.

Figure 1 Design Time and Run-Time Interaction Between BI Publisher and ADF



As shown in the figure, during design time, the BI Publisher extension establishes the connection between your page and the BI Publisher server and enables you to set properties for your report region. After deploying your application, at run time, the ADF application uses a web service to run and retrieve the BI Publisher report back to your page.

The Oracle BI Publisher and ADF JDeveloper integration is available from the Oracle Fusion Applications Development Environment extension bundle. For more information about obtaining this bundle, see the *Oracle Fusion Applications Developer's Guide*.

2 Adding BI Publisher Content to an ADF Project

After you install the BI Publisher extension you can create a project that includes a BI Publisher report region. The following procedures describe how to add BI Publisher content to an ADF project:

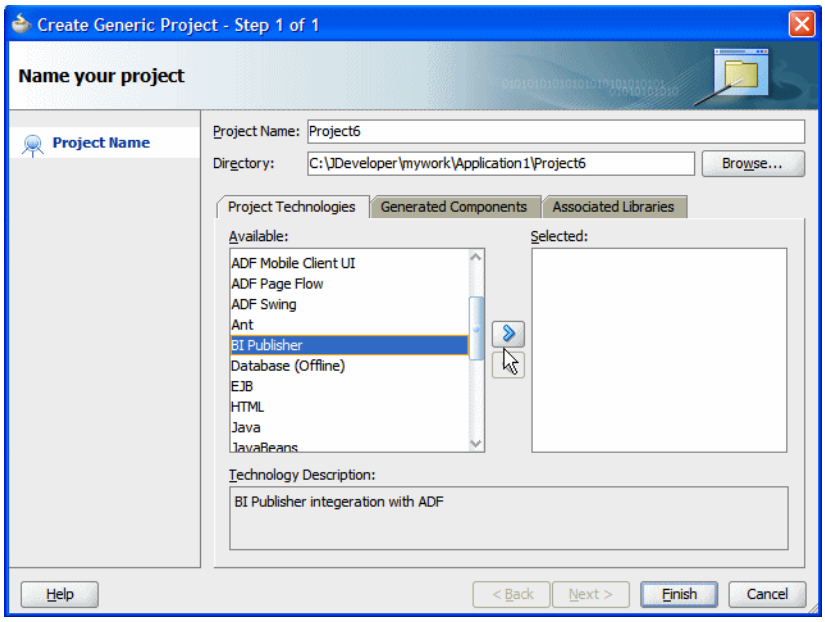
- Section 2.1, "Adding the BI Publisher Technology Scope"
- Section 2.2, "Adding a BI Publisher Region to the JSF Page"
- Section 2.3, "Configuring Connection to the BI Publisher Server"
- Section 2.4, "Setting Properties for the BI Publisher Region"

2.1 Adding the BI Publisher Technology Scope

To display the BI Publisher options, add the BI Publisher technology scope to your project.

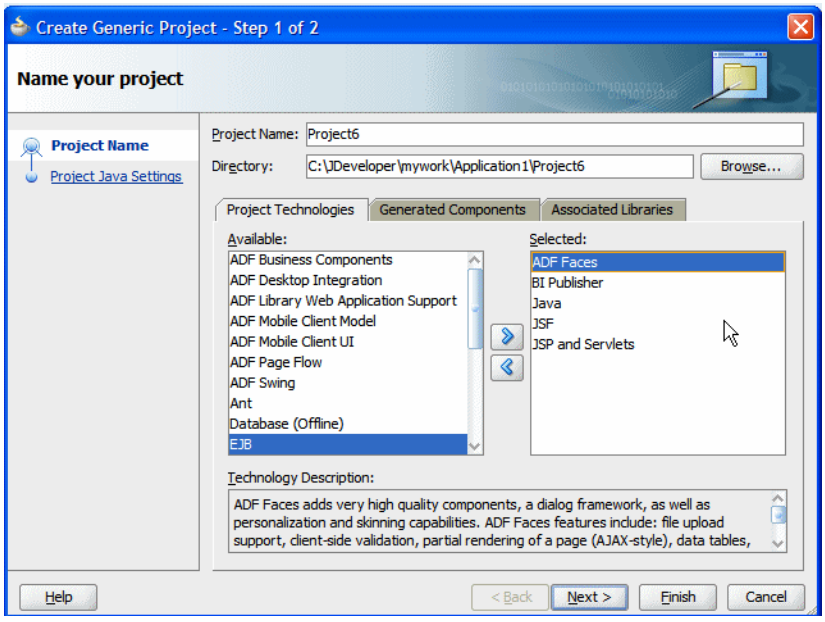
1. In Oracle JDeveloper, go to the Projects Pane and right-click the project to which you want to add the technology scopes and select Project Properties.
2. Select Technology Scope.
3. In the Available Technologies list, select BI Publisher as shown in Figure 2:

Figure 2 Selecting the BI Publisher Technology Scope



4. Click the right shuttle button to add BI Publisher and its dependent technologies to your project. Dependent technologies are moved together. Figure 3 shows the results of this action.

Figure 3 BI Publisher and Dependent Technologies



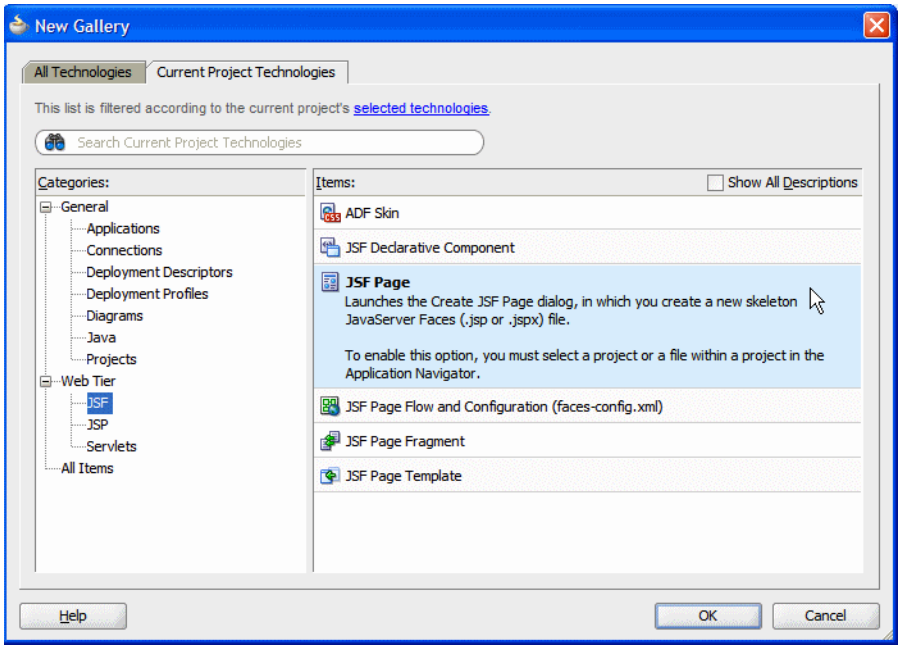
5. Click **Finish**.

2.2 Adding a BI Publisher Region to the JSF Page

The BI Publisher region must reside on a JSF page. To add a JSF page to your project:

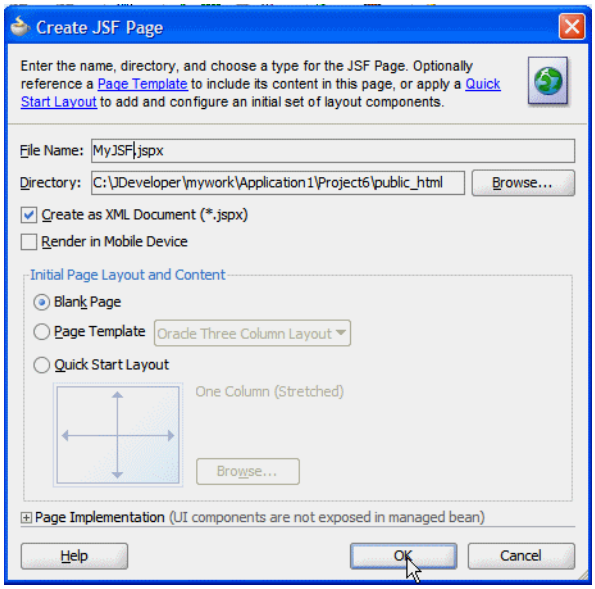
1. Right-click your project and select New.
2. From the Gallery dialog, on the Categories pane, under Web Tier, select JSF.
3. From the items presented in the right pane, select JSF Page, as shown in Figure 4.

Figure 4 Selecting a JSF Page



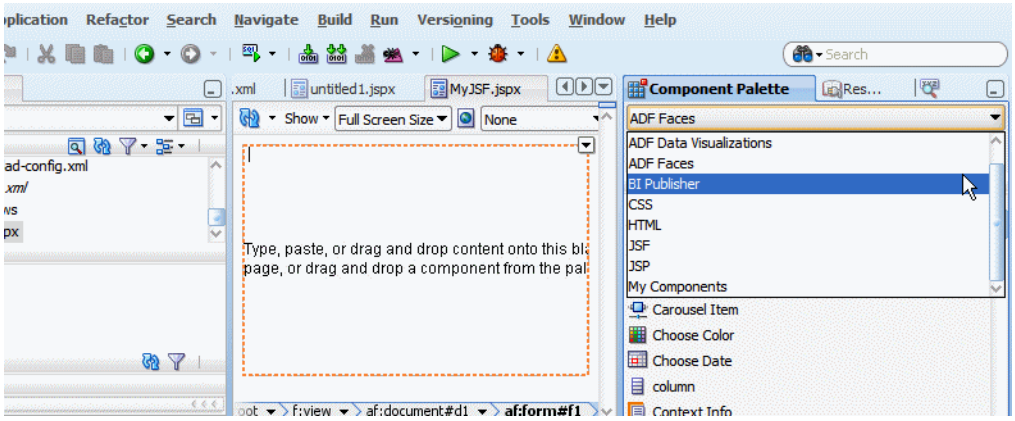
4. On the Create JSF Page dialog, enter a name for your JSF page. Oracle recommends selecting Create as XML Document to create an XML-based JSP document (extension `.jspx`).

Figure 5 Create JSF Page Dialog



5. From the Component Palette list, select BI Publisher, as shown in Figure 6.

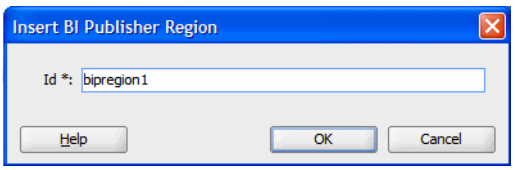
Figure 6 Component Palette List



6. Drag and drop the BI Publisher Region component from the palette to the page.
7. Once the BI Publisher Region is dropped to the JSPX page, the **Insert BI Publisher Region** dialog, shown in Figure 7, prompts you to enter a region ID for the BI Publisher Region.

   Note: The region ID must be unique and must not include the underscore "_" character.

Figure 7 BI Publisher Region Dialog

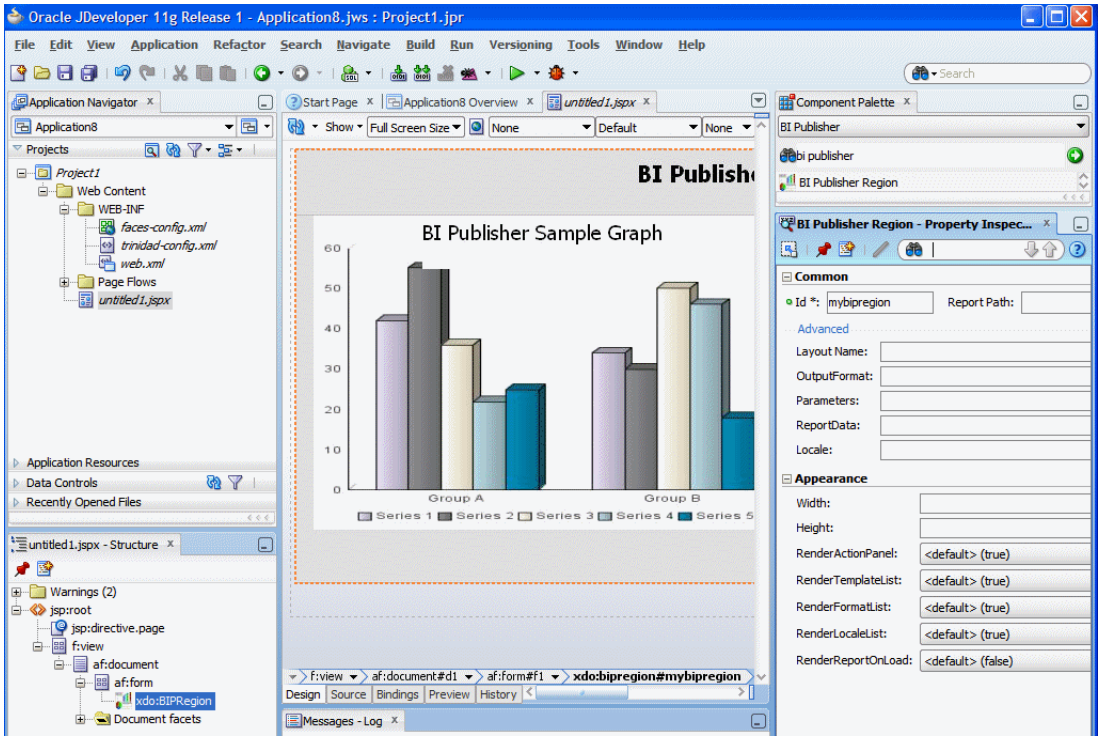


Click OK. When you insert the BI Publisher region notice the following:

- A placeholder image displays in the BI Publisher region of your page.
- The BI Publisher configuration file, `/WEB-INF/xmlp-client-config.xml` is created. Click **Refresh** on the Projects pane if it does not display immediately.
- The Property Inspector presents the BI Publisher region properties when the BI Publisher region is selected.

Figure 8 shows these items:

Figure 8 Report Image, Configuration File, and Property Inspector



8. In the Property Inspector, under the **Appearance** region, enter the report region Width and Height in px to specify the size of the report region in your application page. For example 1000 px and 800 px. No defaults are assigned for height and width, so ensure that you set these fields.

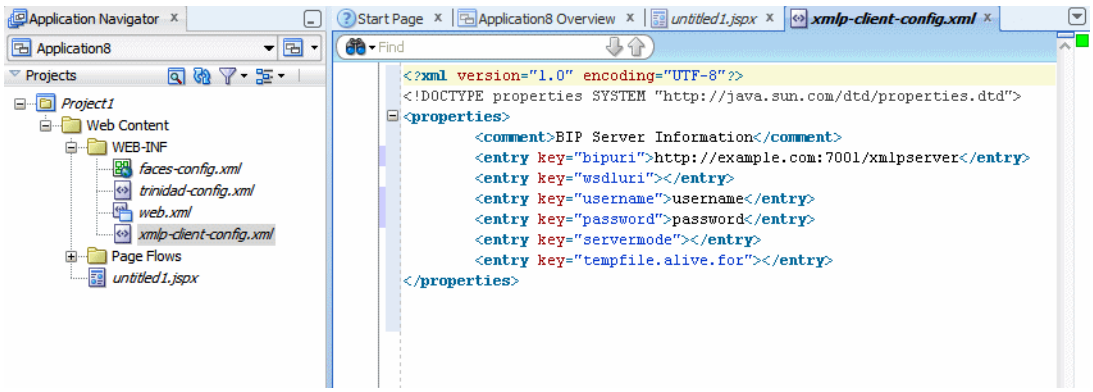
2.3 Configuring Connection to the BI Publisher Server

To establish a connection to the BI Publisher server, update the `xmlp-client-config.xml` file under Project > Web Content > WEB-INF with the connection information as follows:

1. Double-click **xmlp-client-config.xml** to open the file for editing.

   Figure 9 shows the sample **xmlp-client-config.xml** file.

Figure 9 Entering the Connection Keys



2. Update the properties (keys) as shown in the following example. Add the properties not found in the default file.

   The sample **xmlp-client-config.xml** file:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE properties SYSTEM "http://java.sun.com/dtd/properties.dtd">
<properties>
        <comment>BIP Server Information</comment>
        <entry key="bipuri">http://example.com:7001/xmlpserver</entry>
        <entry key="username">username</entry>
</properties>
```

   - bipuri - enter the URL for the BI Publisher server. This may be behind a firewall. For example: `http://hostname.com:7001/xmlpserver`
   - username - enter the secure user name to connect to BI Publisher. This user is the authenticated user used to connect from your application to BI Publisher.
   - password- for the BI Publisher user name entered above

2.4 Setting Properties for the BI Publisher Region

The following properties are available in the Property Inspector:

Table 1 Common Properties

Property	Description
Id	The unique ID for the BI Publisher region. You assign this ID when you insert the BI Publisher region to your page. The ID must not contain spaces.
Report Path	The path to the report in the BI Publisher catalog. Reports have the extension ".xdo". Begin the path from the first level beneath "Shared Folders," but do not include "Shared Folders." For example: "/Samples/1.+Overview/Balance+Letter+Report.xdo"
Rendered	The control to show or hide the BI Publisher region when the page is rendered. The default is true (show).

Table 2 Advanced Properties

Property	Description
Layout Name	The layout from the report definition to apply to the report data. For example, "My Layout". If you do not specify a value here the default layout from the report definition is used.
OutputFormat	Sets the default output format. See Table 4 for the list of valid values. If you do not specify a value here the default output format from the report definition is used.
Parameters	To pass parameters from your application page to the BI Publisher report, use this property to specify the backing bean that captures the parameter values. Use the Expression Builder to populate this field. For example: #{UIBackingBean.parameters} See Section 4, "Passing Parameters from the Application Page" for information.

Table 2 (Cont.) Advanced Properties

Property	Description
ReportData	If your report uses a push data model, use this property to specify the backing bean that contains the method that defines the report data location. Use the Expression Builder to populate this field. For example: #{UIBackingBean.reportData} See Section 5, "Using a Push Data Model" for more information.
Locale	Enter a default locale format using the ISO language code-country code combination, for example en-US.

Table 3 Appearance Properties

Property	Description
Width	Sets the report region width. Enter the value in pixels, for example, 1000 px.
Height	Sets the report region height. Enter the value in pixels, for example, 800 px.
RenderActionPanel	Specifies whether to show or hide the report viewer action panel. The default is true. Note that if you set this to false, you must set renderReportOnLoad to true.
RenderFormatList	Specifies whether to show or hide the output Format List. The default is true.
RenderLocaleList	Specifies whether to show or hide the Locale List. The default is true.
RenderReportOnLoad	When set to true, the report is generated when the ADF page is launched. The default is true.

Table 4 Valid Values for OutputFormat

Output Format	Value to Enter for OutputFormat Property	Template Types That Can Generate This Output Format
Interactive	N/A	Not supported
HTML	html	BI Publisher, RTF, XSL Stylesheet (FO)
PDF	pdf	BI Publisher, RTF, PDF, Flash, XSL Stylesheet (FO)
RTF	rtf	BI Publisher, RTF, XSL Stylesheet (FO)
Excel (mhtml)	excel	BI Publisher, RTF, Excel, XSL Stylesheet (FO)
Excel (html)	excel2000	BI Publisher, RTF, Excel, XSL Stylesheet (FO)
Excel (*.xlsx)	xlsx	BI Publisher, RTF, XSL Stylesheet (FO)
PowerPoint (mhtml)	ppt	BI Publisher, RTF, XSL Stylesheet (FO)
PowerPoint (.*pptx)	pptx	BI Publisher, RTF, XSL Stylesheet (FO)
MHTML	mhtml	BI Publisher, RTF, Flash, XSL Stylesheet (FO)
PDF/A	pdfa	BI Publisher, RTF, XSL Stylesheet (FO)
PDF/X	pdfx	BI Publisher, RTF, XSL Stylesheet (FO)

Table 4 (Cont.) Valid Values for OutputFormat

Output Format	Value to Enter for OutputFormat Property	Template Types That Can Generate This Output Format
Zipped PDFs	pdfz	BI Publisher, RTF, PDF, XSL Stylesheet (FO)
FO Formatted XML	xslfo	BI Publisher, RTF, XSL Stylesheet (FO)
Data (XML)	xml	BI Publisher, RTF, PDF, Excel, Flash, XSL Stylesheet (FO), Etext, XSL Stylesheet (HTML XML/Text)
Data (CSV)	csv	BI Publisher, RTF, PDF, Excel, Flash, XSL Stylesheet (FO), XSL Stylesheet (HTML XML/Text), Etext
XML	xml	XSL Stylesheet (HTML XML/Text)
Text	text	XSL Stylesheet (HTML XML/Text), Etext
Flash	flash	Flash

3 Deploying and Running the Application

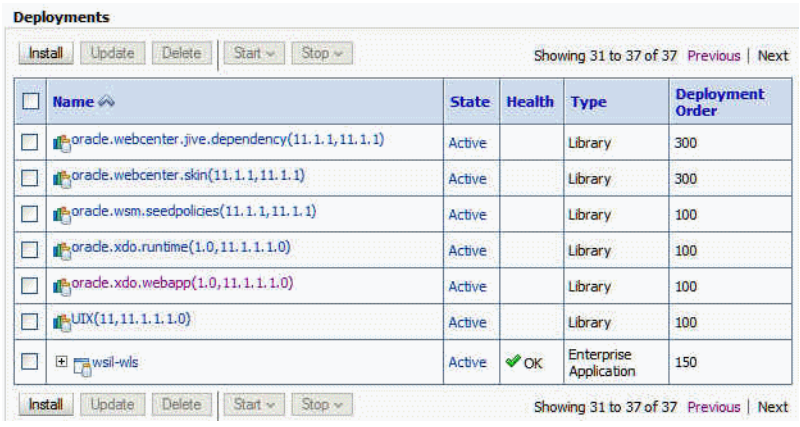
To deploy and run the application, first copy the required libraries to the client where JDeveloper is installed and then follow the deployment steps.

- Copy Libraries
- Deploy the Application

Copy Libraries

1. Ensure that Oracle Business Intelligence Publisher shared libraries are installed on your target WebLogic Server instance.
   - `oracle.xdo.runtime: $MW_HOME/jdeveloper/xdo/lib/xdoruntime.ear`
   - `oracle.xdo.webapp: $MW_HOME/jdeveloper/xdo/lib/xdowebapp.war`
2. When the shared libraries are installed, you can see them in the WebLogic Server console as shown in Figure 10.

Figure 10 Libraries Shown in WebLogic Server Console

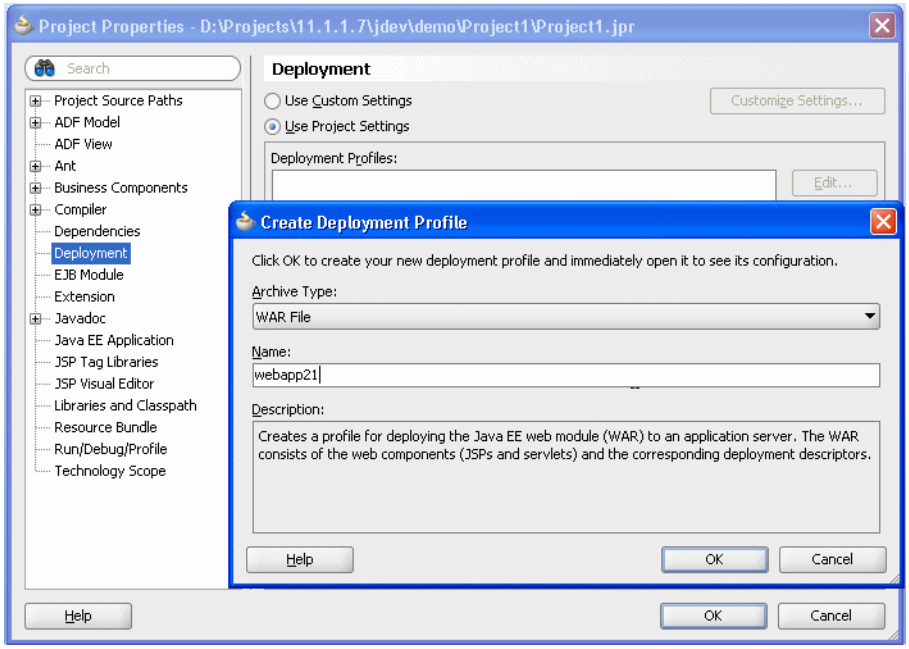


Deploy the Application

To deploy and run the application:

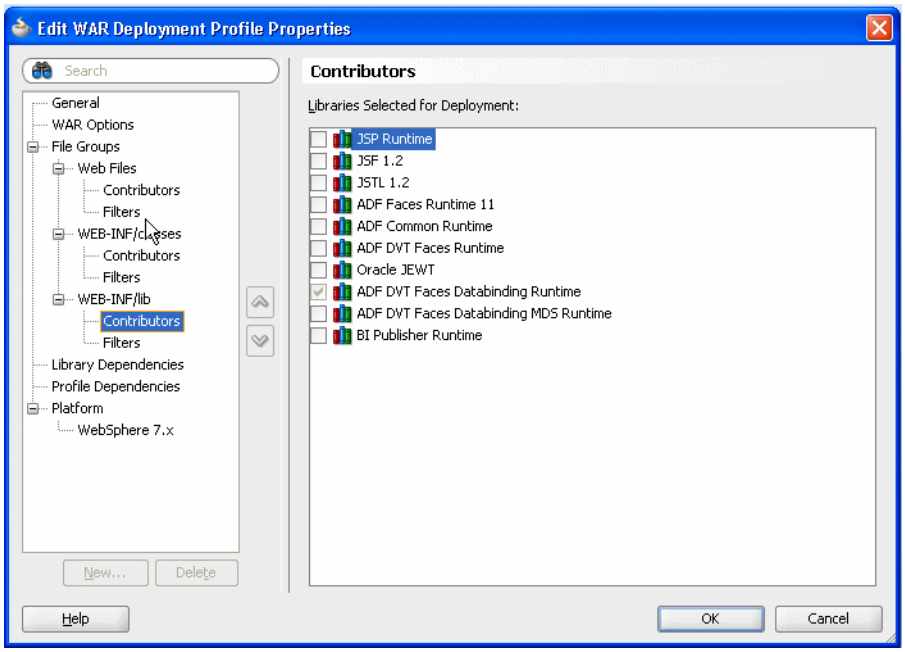
1. From Project Properties, click **Deployment**, click **New** and enter a value for the J2EE application name. Figure 10 shows the Create Deployment Profile dialog.

Figure 11 Create Deployment Dialog



2. Click **WEB-INF/lib**. BI Publisher Runtime does not have to be included so long as **weblogic.xml** and **weblogic-application.xml** are properly configured. Figure 12 shows libraries selected for deployment.

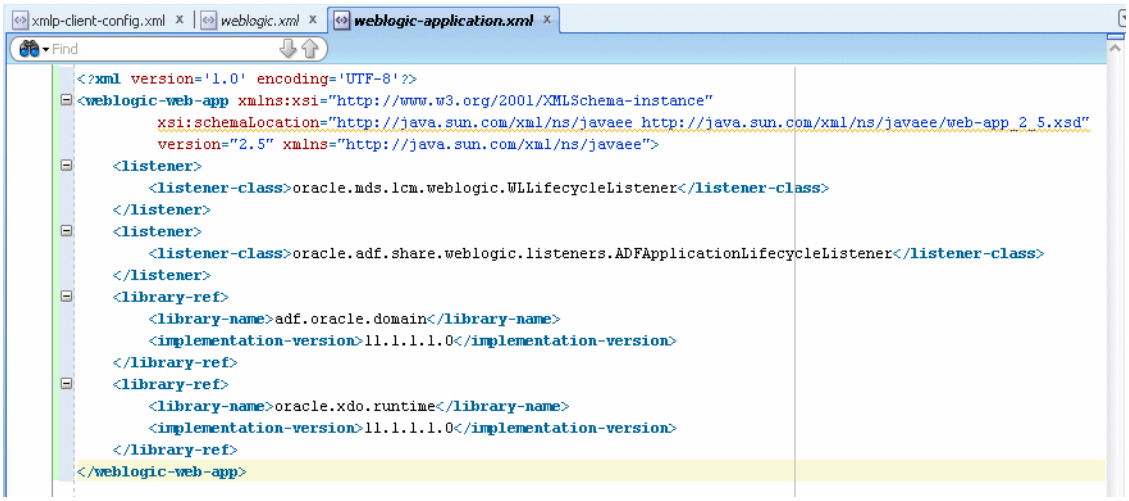
Figure 12 Libraries Selected for Deployment



3. Add a shared library reference to `weblogic.xml` and `weblogic-application.xml` as shown in Figure 13 and Figure 14.

Figure 13 Library Entry in weblogic.xml

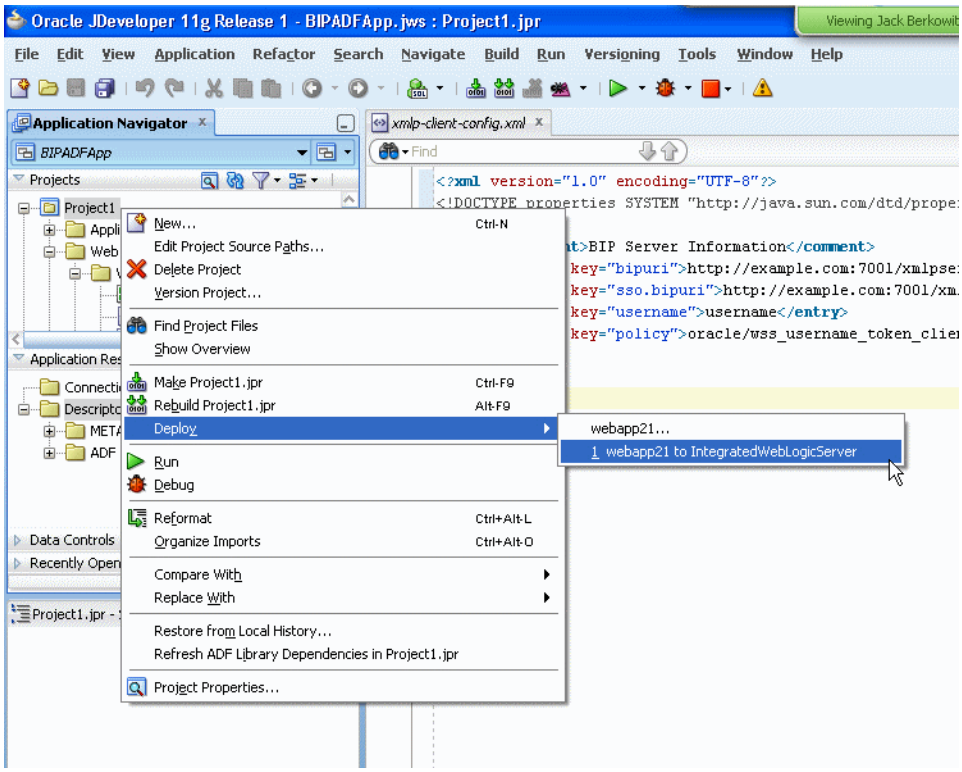
Figure 14 Example weblogic-application.xml



4. From the JDeveloper menu, select **Run > Start Server Instance**. Do not run the JSPX page directly.

5. Right-click the application and select **Deploy - (your BIP ADF Application name) - to IntegratedWLSConnection** as shown in Figure 15. Make sure the BIP ADF Application deployment profile has your J2EE application name **(webapp21** in this example) selected on Application Assembly.

Figure 15 Deploying an Application to a Connection

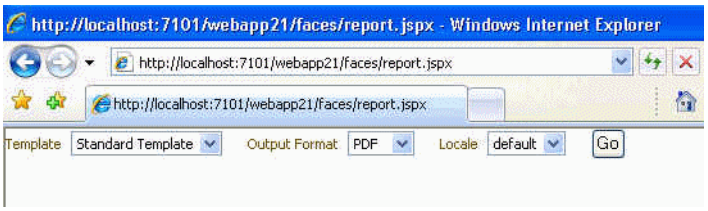


The deployment completed message displays in the Log window.

6. Open your browser and enter `http://<hostname>:<port>/j2eeWebAppName/faces/report.jspx` in the address field. When the page launches, the contents display according to the settings you defined for RenderActionPanel, RenderFormatList, RenderLocaleList, and RenderReportOnLoad.

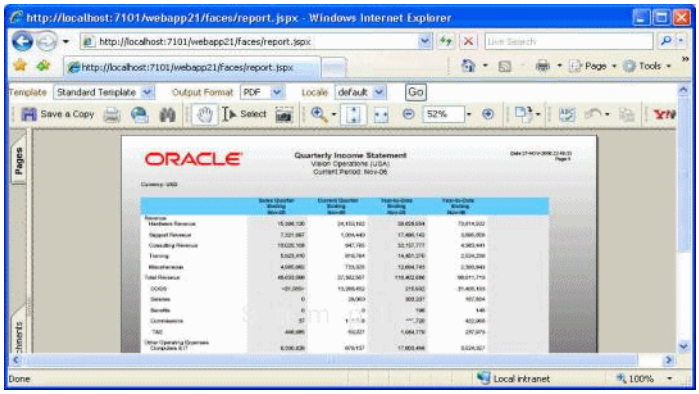
   Figure 16 shows an example display when RenderActionPanel, RenderFormatList, and RenderLocaleList are set to true and RenderReportOnLoad is set to false. The Template list, Output Format list, and Locale list display and can be updated by the user. To render the report, the user must click Go.

Figure 16 Displayed Fields



7. Click **Go**. Your report displays, as shown in Figure 17.

Figure 17 Generated Report



4 Passing Parameters from the Application Page

You can pass parameters entered from your application page to the report using a backing bean. The BIP region tag must define the parameters property and bind it to the backing bean method as shown:

```
<?xml version='1.0' encoding='UTF-8'?>
<jsp:root xmlns:jsp="http://java.sun.com/JSP/Page" version="2.1"
          xmlns:f="http://java.sun.com/jsf/core"
          xmlns:h="http://java.sun.com/jsf/html"
          xmlns:af="http://xmlns.oracle.com/adf/faces/rich"
          xmlns:xdo="http://xmlns.oracle.com/xdo/faces">
  <jsp:directive.page contentType="text/html;charset=UTF-8"/>
  <f:view>
    <af:document id="d1">
      <af:form id="f1">
        <p>
          <af:inputText label="Employee ID" id="it1"
                        binding="#{UIBackingBean.empId}"/>
        </p>
        <p>
          ?[34m|
        </p>
        <p>
          <af:separator id="s1"/>
        </p>
          <xdo:BIPRegion id="bipregion1" reportId="/Samples/Overview/Balance+Letter+Report.xdo"
                        width="1000px" height="800px"
                        parameters="#{UIBackingBean.parameters}" reportData=""/>
        <f:facet name="second"/>
      </af:form>
    </af:document>
  </f:view>
</jsp:root>
```

Specify the backing bean in the Parameters property of the Property Inspector. See Table 2, " Advanced Properties" for more information.

5 Using a Push Data Model

Some reports require XML data from a source other than a BI Publisher data model. For this requirement, you can push data to the BI Publisher server to use as input for your report. For this scenario, store the data in a location that is accessible by your application and create a backing bean that defines the getter method to

retrieve the report data from its stored location. You still define the Report Path, Layout Name and other properties for the report as defined on the BI Publisher server.

The `BIPRegion` tag must define the `reportData` property for the tag, and bind it to the backing bean method. The following sample code shows this binding:

```
<?xml version='1.0' encoding='UTF-8'?>
<jsp:root xmlns:jsp="http://java.sun.com/JSP/Page" version="2.1"
          xmlns:f="http://java.sun.com/jsf/core"
          xmlns:h="http://java.sun.com/jsf/html"
          xmlns:af="http://xmlns.oracle.com/adf/faces/rich"
          xmlns:xdo="http://xmlns.oracle.com/xdo/faces">
  <jsp:directive.page contentType="text/html;charset=UTF-8"/>
  <f:view>
    <af:document id="d1">
      <af:form id="f1">
        <xdo:BIPRegion id="id3"
                       reportId="/Samples/Overview/Balance+Letter+Report.xdo"
                       width="1000px" height="800px"
                       reportData="#{UIBackingBean.reportData}"/>
      </af:form>
    </af:document>
  </f:view>
</jsp:root>
```

The following code sample shows the `UIBackingBean` class referenced in the previous sample:

```
package fusionApp;
import java.io.RandomAccessFile;
import java.util.Hashtable;
import java.util.Properties;
import oracle.adf.view.rich.component.rich.input.RichInputText;
public class UIBeackingBean {
    private RichInputText empId;
    private Properties mParameters;
    private byte[] reportData;

    public UIBeackingBean() {
        mParameters = new Properties();
        reportData = null;
    }

    public void setEmpId(RichInputText empId) {
        this.empId = empId;
    }

    public RichInputText getEmpId() {
        return empId;
    }

    public Hashtable getParameters() {
        if(empId != null && empId.getValue() != null)
        {
            mParameters.put("userid", new String[]{empId.getValue().toString()});

            String[] values = (String[])mParameters.get("userid");
            System.out.println("getParameters() is called : " + values[0]);
        }
        return mParameters;
    }
```

```
    public byte[] getReportData() {

        String dataFile = "/tmp/reportData.xml";
        try
        {
        RandomAccessFile raf = new RandomAccessFile(dataFile, "r");
        reportData= new byte[(int)raf.length()];


        raf.read(reportData);
        raf.close();
        //write this to temp
        java.io.FileOutputStream outStream = new
 java.io.FileOutputStream("/tmp/output_reportData");
        outStream.write(reportData);
        outStream.close();

        } catch (Exception e) {
            System.out.println("Error reading file : " + e.getMessage());
        }
        return reportData;

    }
}
```

To define a push data model:

1. Create a backing bean class that defines the getter method that retrieves your data from its stored location.
2. In JDeveloper, create the BIP Region for your page and specify all properties for the report that you wish to run on the BI Publisher server (Report Path, Layout Name, Output Format, and so on).
3. In the Property Inspector, for the ReportData property, enter the expression that defines the variable to pass to the backing bean method to call the appropriate getter method to retrieve the report data from its stored location. Use the Expression Builder to build the unified expression language (EL) syntax.

6 Conditionally Required Settings

Depending on the type of report you are running you may need to make other settings for your reports to run as expected.

This section contains the following topics:

6.1 Setting the MIME Types for Your Report Output Type

To ensure that all the output types can be successfully generated from your page, add the MIME-type mappings to the `web.xml` file. Typically you only need to add the MIME-type mappings for Excel output.

To add the MIME-type mappings:

1. Double-click the Project `web.xml` file to open it for editing.
2. Expand the MIME mappings region.
3. Click Add.
4. In the Property Inspector, enter the following:

- extension - (Required) Enter the file name extension of the document type you want to map to a particular MIME type for your web application. For example, pdf. This field corresponds to the <extension> tag of the <mime-mapping> subelement.
- mime-type - (Required) Enter the document MIME type that you want to map to the specified file name extension. For example, application/pdf for Adobe's Portable Document Format. This field corresponds to the <mime-type> tag of the <mime-mapping> subelement. Note, the official registry of Internet MIME types is managed by the Internet Assigned Numbers Authority (IANA) at www.iana.org.

The following table shows the required MIME types to generate the Microsoft Excel output options supported by BI Publisher:

Extension	Mime-Type
xls	application/vnd.ms-excel
xlsx	application/vnd.openxmlformatsofficedocument.spreadsheetml.sheet
xlsm	application/vnd.ms-excel.sheet.macroEnabled.12
mhtml	message/rfc822

7 Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc.

Access to Oracle Support

Oracle customers have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

Oracle Business Intelligence Pubisher Using the Oracle Business Intelligence Publisher Extension for Oracle JDeveloper , 11*g* Release 1 (11.1.1)
E48204-01