Oracle® Business Intelligence Publisher

Administrator's and Developer's Guide

Release 10.1.3.4

Part No. E12188-01

August 2008

ORACLE

Oracle Business Intelligence Publisher Administrator's and Developer's Guide, Release 10.1.3.4

Part No. E12188-01

Primary Author: Leslie Studdard

Contributing Author: Ahmed Ali, Hisaki Danjo, Tim Dexter, Mike Donohue, Klaus Fabian, Chiang Guo, Incheol Kang, Kazuko Kawahara, Hide Kojima, Hok-Min Lie, Nikos Psomas, Kei Saito, Pradeep Sharma, Ashish Shrivastava, Elise Tung-Loo, Shinji Yoshida

Contents

12 Using the Delivery Manager APIs

13 Setting Up an After Report Trigger

A Configuration File Reference

Index

Preface

Intended Audience

Welcome to Release 10.1.3.4 of the *Oracle Business Intelligence Publisher Administrator's and Developer's Guide.*

This guide is intended for administrators and developers. This guide contains the describes the following for administrators:

- Setting up BI Publisher security and define users and roles
- Setting up data sources
- Setting up integrations with other Oracle products
- Configuring system-level properties
- Configuring delivery options
- Performing system maintenance

This guide contains the following information for developers:

- Using the BI Publisher public Web service
- Using the BI Publisher APIs
- Setting up an after-report trigger

See Related Information Sources on page ix for more Oracle product information.

TTY Relay Access to Oracle Support Services

To reach AT&T Customer Assistants, dial 711 or 1.800.855.2880. An AT&T Customer Assistant will relay information between the customer and Oracle Support Services at

1.800.223.1711. Complete instructions for using the AT&T relay services are available at http://www.consumer.att.com/relay/tty/standard2.html. After the AT&T Customer Assistant contacts Oracle Support Services, an Oracle Support Services engineer will handle technical issues and provide customer support according to the Oracle service request process.

Documentation Accessibility

Our goal is to make Oracle products, services, and supporting documentation accessible to all users, including users that are disabled. To that end, our documentation includes features that make information available to users of assistive technology. This documentation is available in HTML format, and contains markup to facilitate access by the disabled community. Accessibility standards will continue to evolve over time, and Oracle is actively engaged with other market-leading technology vendors to address technical obstacles so that our documentation can be accessible to all of our customers. For more information, visit the Oracle Accessibility Program Web site at http://www.oracle.com/accessibility/.

Accessibility of Code Examples in Documentation

Screen readers may not always correctly read the code examples in this document. The conventions for writing code require that closing braces should appear on an otherwise empty line; however, some screen readers may not always read a line of text that consists solely of a bracket or brace.

Accessibility of Links to External Web Sites in Documentation

This documentation may contain links to Web sites of other companies or organizations that Oracle does not own or control. Oracle neither evaluates nor makes any representations regarding the accessibility of these Web sites.

Structure

Related Information Sources

To download free release notes, installation documentation, white papers, or other collateral, please visit the Oracle Technology Network (OTN). You must register online before using OTN; registration is free and can be done

at

http://www.oracle.com/technetwork/community/join/overview/index.html

If you already have a username and password for OTN, then you can go directly to the documentation section of the OTN Web site at

http://www.oracle.com/technology/documentation/

Information specifically related to BI Publisher can be found at:

http://www.oracle.com/technology/documentation/bi_pub.html

Other guides in the Oracle Business Intelligence Publisher documentation set include:

Oracle Business Intelligence Publisher Report Designer's Guide

Oracle Business Intelligence Publisher Installation Guide

Oracle Business Intelligence Publisher Certification Guide

Oracle Business Intelligence Publisher Release Notes

Oracle Business Intelligence Publisher Java API Reference Guide

All of these guides are available from:

http://www.oracle.com/technology/documentation/bi_pub.html

If your installation is integrated with the Oracle Business Intelligence Enterprise Edition, please see the Oracle Business Intelligence Enterprise Edition documentation available from:

http://www.oracle.com/technology/documentation/bi_ee.html

Part 1

Administering BI Publisher

1

Introduction to Oracle BI Publisher Administration

Introduction

Oracle Business Intelligence Publisher (BI Publisher) offers you the most efficient, scalable reporting solution available for complex, distributed environments. It provides a central architecture for generating and delivering information to employees, customers, and suppliers—both securely and in the right format. Oracle BI Publisher reduces the high costs associated with the development, customization and maintenance of business documents, while increasing the efficiency of reports management.

Administering BI Publisher requires setting up and maintaining the following system components:

- BI Publisher security
- Data source connections
- Integrations with other Oracle products
- Report delivery options
- Runtime configuration settings
- BI Publisher Scheduler configurations
- System maintenance options

About the Security Model Options

BI Publisher offers the following security options:

- BI Publisher Security

  Use BI Publisher's Users and Roles paradigm to control access to reports and data sources.

- Integration with your LDAP server

  Set up the BI Publisher roles in your LDAP server then configure BI Publisher to integrate with it.

- Oracle E-Business Suite

  Upload a DBC file to recognize your Oracle E-Business Suite users. .

- Oracle BI Server

  Set up the BI Publisher roles in your BI Server Administration tool then configure BI Publisher to integrate with it.

- Oracle Database

  Set up the BI Publisher roles in your Oracle Database and then configure BI Publisher to integrate with it.

- Oracle's Hyperion Common Shared Services

  If you plan to run BI Publisher inside of Oracle Enterprise Performance Management Workspace, you must use this security model.

For more information on the supported security models, see Defining a Security Model, page 2-1.

About Integrations with Other Oracle Products

Oracle BI Publisher enables you to integrate with the following Oracle products:

- Oracle BI Presentation Services

  This integration enables you to use Oracle BI Answers requests as data sources for your reports.

- Oracle BI Discoverer

  This integration enables you to use Oracle BI Discoverer Worksheets as data sources for your reports.

- Hyperion Workspace and Shared Services

  This integration enables you to access BI Publisher from within the Oracle Enterprise Performance Management Workspace.

- Oracle Smart Space Client

This integration enables you to set up a link to the Oracle Smart Space download page. This will enable all BI Publisher users to easily access the Oracle Smart Space client download page from their BI Publisher session.

For more information on supported integrations, see Setting Up Integrations, page 5-1
.

About the Data Source Connections

BI Publisher reports rely on XML data. BI Publisher supports a variety of data sources from which you can supply XML data. The data can come from any of the following sources:

- Database connection

  BI Publisher supports direct JDBC connections and connections through a JNDI pool

- HTTP XML feed
- Web services
- OLAP cube
- An existing XML file that is stored in an accessible location

If you have integrated your system with Oracle Business Intelligence and Oracle Discoverer you can also take advantage of the following data sources:

- Oracle BI Answers Request
- Oracle BI Discoverer worksheet

For more information on setting up data source connections, see Setting Up Data Sources, page 4-1.

About Report Delivery Options

The BI Publisher delivery manager enables you to set up connections to support the following delivery channels:

- Print
- Fax
- Email
- HTTP notification

- FTP
- WebDAV

For more information on setting up the delivery options, see Setting Up Delivery Options, page 7-1.

About Setting Runtime Configuration Properties

Use the Runtime Configuration page to enable configuration settings for your system. The properties include settings that

- Control the processing for different output types
- Define bursting options
- Enable digital signature
- Tune for scalability and performance
- Define font mappings

For more information on setting configuration properties and font mappings, see Setting Runtime Properties, page 8-1.

About the System Maintenance Options

BI Publisher administration also includes a set of system maintenance settings and tasks. These are:

- Defining your reports repository
- Setting general properties for the BI Publisher server, including caching and debug level
- Configuring the database for your Scheduler schema
- Refreshing the server metadata

For more information on these tasks and settings, see Setting System Maintenance Options, page 6-1.

Flow of Tasks for First Time Setup of BI Publisher

If you are setting up BI Publisher for the first time, following is the recommended flow of tasks:

- Define a Local Superuser

Set up this Superuser to ensure access to all administrative functions in case of problems with the current security setup. For more information, see Defining a Local Superuser, page 2-2.

- Determine your security model
- Set up your chosen security model and test
- Set up your data sources and test
- Set up integration services and test
- Set up your scheduler and test
- Set up your delivery options and test

2

Defining a Security Model

This chapter covers the following topics:

- Security Model Overview
- Understanding BI Publisher's Users and Roles
- Considerations When Deleting a User
- Integrating with LDAP
- Setting Up Oracle Single Sign-On
- Integrating with Oracle E-Business Suite Security
- Integrating with Oracle BI Server Security
- Integrating with Oracle Database Security
- Implementing a Digital Signature

Security Model Overview

BI Publisher offers the following security options:

- BI Publisher Security

  Use BI Publisher's Users and Roles paradigm to control access to reports and data sources. See Understanding BI Publisher's Users and Roles, page 2-3.

- LDAP

  Set up the BI Publisher roles in your LDAP server then configure BI Publisher to integrate with it. See Integrating with LDAP, page 2-7.

- Oracle E-Business Suite

  Upload a DBC file to recognize your Oracle E-Business Suite users. See Integrating with E-Business Suite Security, page 2-24.

- Oracle BI Server

  Set up the BI Publisher roles in your BI Server Administration tool then configure BI Publisher to integrate with it. See Integrating with Oracle BI Server Security, page 2-26.

- Oracle Database

  Set up the BI Publisher roles in your Oracle Database and then configure BI Publisher to integrate with it. See Integrating with Oracle Database Security, page 2-27.

- Oracle's Hyperion Common Shared Services

  If you plan to run BI Publisher inside of Oracle Enterprise Performance Management Workspace, you must use this security model. This option is enabled only after you have completed the integration steps with Workspace. For more information , see Integrating with Oracle Enterprise Performance Management Workspace, page 5-5.

Defining a Local Superuser

BI Publisher allows you to define an administration Superuser. Using the Superuser credentials you can directly access the BI Publisher server administrative functions without logging in through the defined security model.

Set up this Superuser to ensure access to all administrative functions in case of problems with the current security setup.

1. Select the **Admin** tab.
2. Under **Security Center** select **Security Configuration**.
3. Under **Local Superuser**, select the box and enter the credentials for the Superuser.

Allowing Guest Access

BI Publisher allows you to set up a public access folder. Any user can access the reports in this folder without entering credentials.

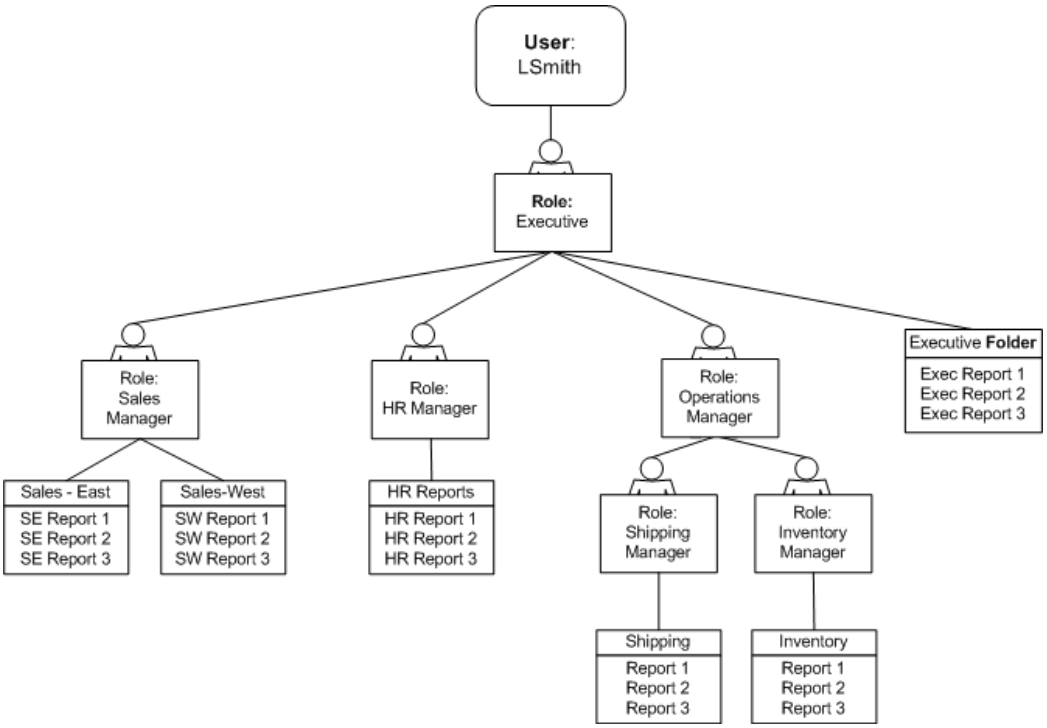
1. Under Shared Folders, create the folder to which you want to grant public access.
2. Select the **Admin** tab.
3. Under **Security Center** select **Security Configuration**.
4. Under **Guest Access**, select **Allow Guest Access**.
5. Enter the name of the folder that you created for public access.

6. Restart the BI Publisher application.

The next time you access BI Publisher, you will see the Guest button. Users can select this button and view the reports in your chosen guest folder without presenting credentials.

Understanding BI Publisher's Users and Roles

A user is granted one or multiple **Roles**. A **Role** defines a user's access to Folders and functions. A role can be assigned one or multiple **Folders** to which access is granted. Reports are contained within the folders. You can create a hierarchy of roles by assigning roles to other roles. In this way the privileges of multiple roles can roll up to higher level roles. The following graphic shows an example of the hierarchy structure of User, Role, and Folder:



Functional Roles

BI Publisher provides five functional roles to grant access to specific tasks within the application. Assign these roles to users based on their need to perform the associated tasks. These roles cannot be updated or deleted.

The following table shows the privileges granted to each functional role:

Role	Privileges
no roles assigned	View (online reports only) Export History
BI Publisher Excel Analyzer	View Export History Grants access to the Excel Analyzer
BI Publisher Online Analyzer	View Export History (public reports only) Grants access to the Analyzer
BI Publisher Scheduler	View Export History Schedule
BI Publisher Template Designer	View Export History Enables log on from Template Builder
BI Publisher Developer	View Export History Edit Configure Folder and Report Tasks Enables log on from the Template Builder

Role	Privileges
BI Publisher Administrator	View
	Export
	Edit
	Schedule
	History
	Configure
	Folder and Report Tasks
	Excel Analyzer
	Online Analyzer
	Admin tab and all administration tasks
	Enables log on from the Template Builder

Setting Up Users and Roles

There are two options for setting up users and roles:

- Set up users and roles in the BI Publisher Enterprise Security Center

  For this option, follow the instructions in this section.

- Integrate BI Publisher Enterprise with an existing LDAP server

  For this option, See Integrating with LDAP, page 2-7.

Create a Role:

1. From the **Security Center**, select **Roles and Permissions**; this will invoke the **Security Center** page. Here you can see the list of existing roles and permissions.
2. Select **Create Role**.
3. Enter a **Role Name** and **Description** and select **Apply**.
4. Grant access to data sources for the role. See Setting Up Data Sources, page 4-1.

Add a User:

1. From the **Security Center**, select **Users**. This will invoke the **Security Center Users**

page. Here you can see the list of existing users.

2. Select **Create User**.

3. Add the **User Name** and **Password** for the user.

Update a User:

1. From the **Security Center**, select **Users**. This will invoke the **Security Center Users** page. Here you can see the list of existing users.

2. Select the user name. You can update both the user name and the password.

Add a Role to a User:

1. From the **Security Center**, select **Users**. This will invoke the **Security Center Users** page. Here you can see the list of existing users.

2. Select the **Assign Roles** icon for the user.

3. From the **Assign Roles** page, select the role from the **Available Roles** list and then select the **Move** shuttle button to move the role to the **Assigned Roles** list. When done assigning all roles, select **Apply**.

Add a Folder to a Role:

1. From the **Security Center**, select **Roles and Permissions**; this will invoke the **Security Center** page. Here you can see the list of existing roles and permissions.

2. Select the **Add Folders** icon.

3. Select the desired folder from the **Available Folders** list and use the **Move** shuttle button to move it to the **Allowed Folders** list.

   Note that the folders are presented as the directory structure is set up in your system. Selecting the top level folder will grant access to all subfolders. Selecting just the subfolder entry will allow access only to the subfolder.

Add a Data Source to a Role

1. From the **Security Center**, select **Roles and Permissions**; this will invoke the **Security Center** page. Here you can see the list of existing roles and permissions.

2. Select the **Add Data Sources** icon for the Role.

3. Move selections from the **Available Data Sources** list to the **Allowed Data Sources** list.

Users with this role will only be allowed to run reports that access data sources on the Allowed Data Sources list.

Add a Role to a Role:

1. From the **Security Center**, select **Roles and Permissions**; this will invoke the **Security Center** page. Here you can see the list of existing roles and permissions.
2. Select the **Add Roles** icon for the Role.
3. Select the desired role from the **Available Roles** list and use the **Move** shuttle button to move it to the **Included Roles**.

Considerations When Deleting a User

When you delete a user in any security model (built-in, LDAP, E-Business Suite, or BI Server), ensure that you delete the user folder from the repository. If you are logged in as an Administrator, the user folders are located on the Reports page under *Users*/*<username>*. If the individual user folder is not deleted and a new user is created with the same user name, then the new user will have access to the contents of the existing user folder.

Integrating with LDAP

BI Publisher can be integrated with your LDAP server to manage users and report access. Create the users and roles within your LDAP server, then configure the BI Publisher server to access your LDAP server.

In the BI Publisher security center module, assign folders to those roles. When a user logs into the server they will have access to those folders and reports assigned to the LDAP roles.

Integrating the BI Publisher server with Oracle LDAP consists of three main tasks:

1. Set up users and roles
2. Configure BI Publisher to recognize your LDAP server
3. Assign report folders and data sources to roles

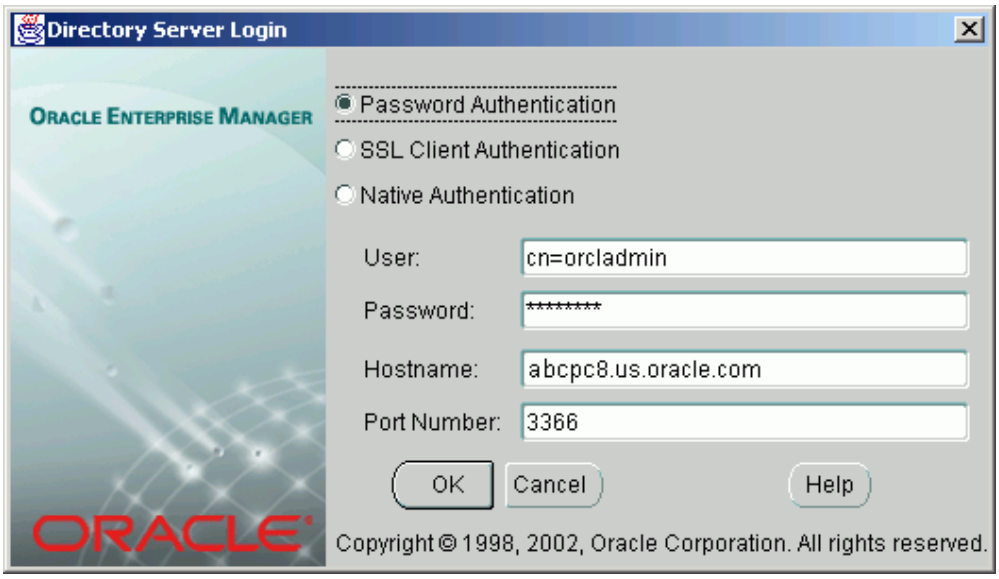
These tasks can be performed through the Oracle Internet Directory (OID) Web UI, or through the client application. Each method is described in detail.

Using the Client Application

Set Up Users and Roles

1. Use the Enterprise Security Manager login to access your LDAP Server.

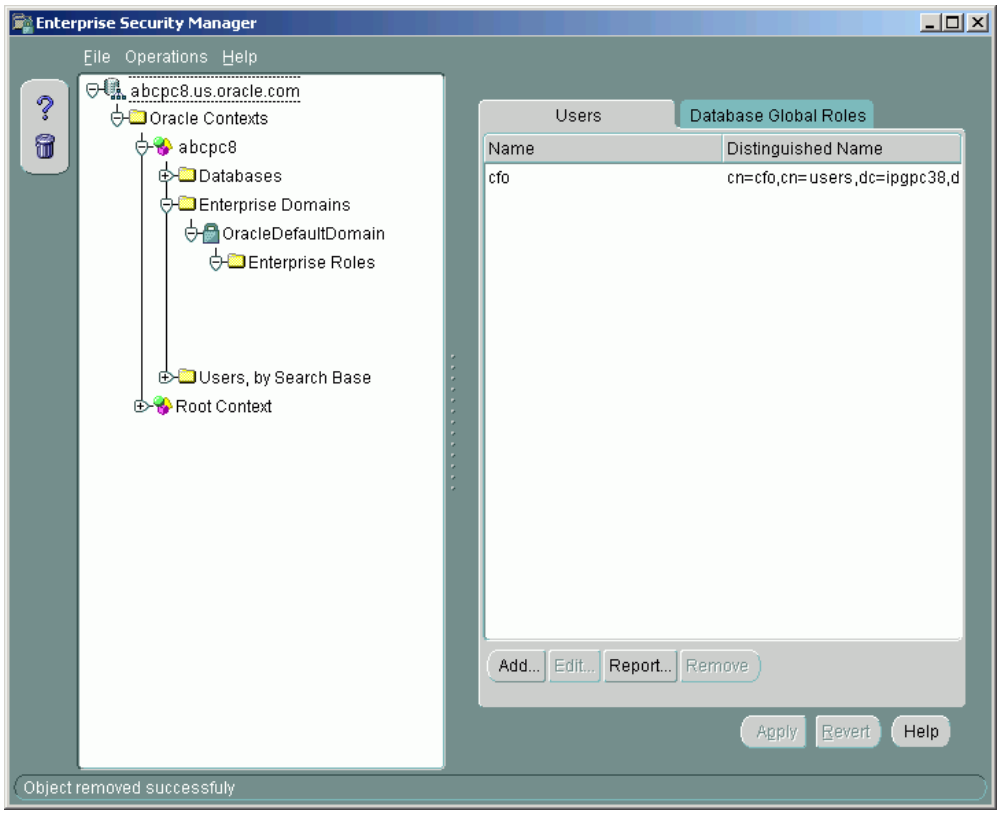
   An example Directory Server Login screen is shown in the following figure:



2. Create Roles.

   Navigate to the Enterprise Roles node under the OracleDefaultDomain node.

   A sample Enterprise Security Manager screen is shown in the following figure:

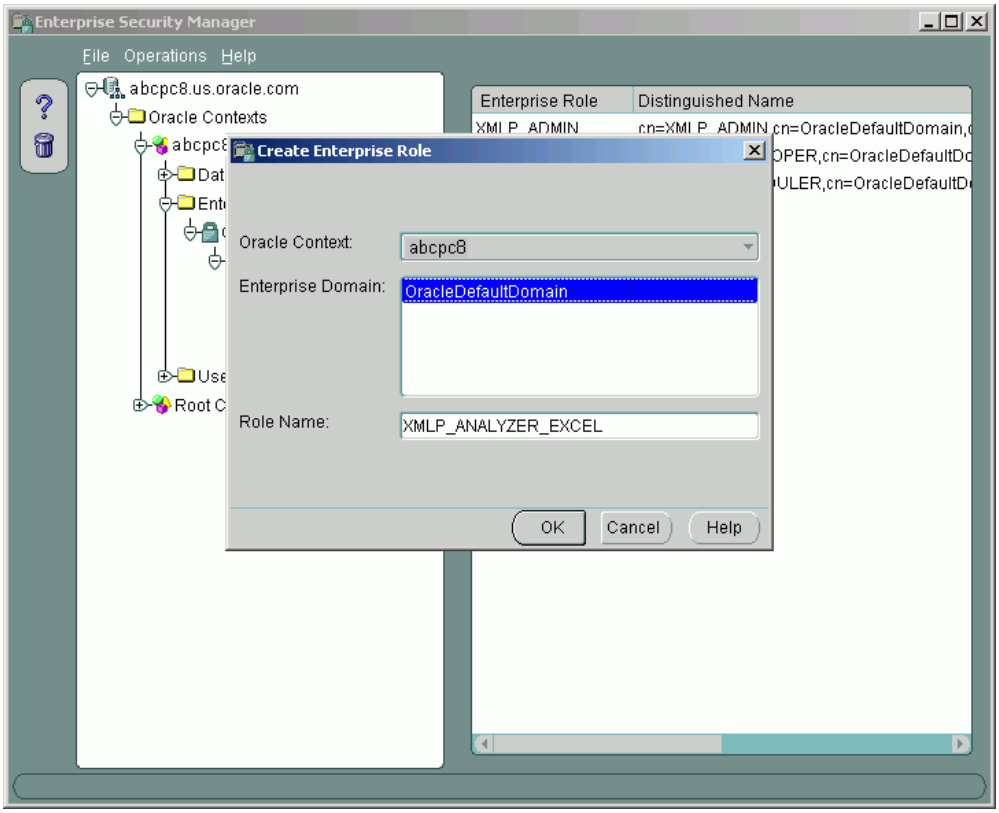


3. To create a role, select the **Enterprise Roles** node, then select **Create Enterprise Role** from the **Operations** menu.

   You must create the following roles to integrate with BI Publisher: See Understanding Users and Roles, page 2-3 for full descriptions of the required functional roles.

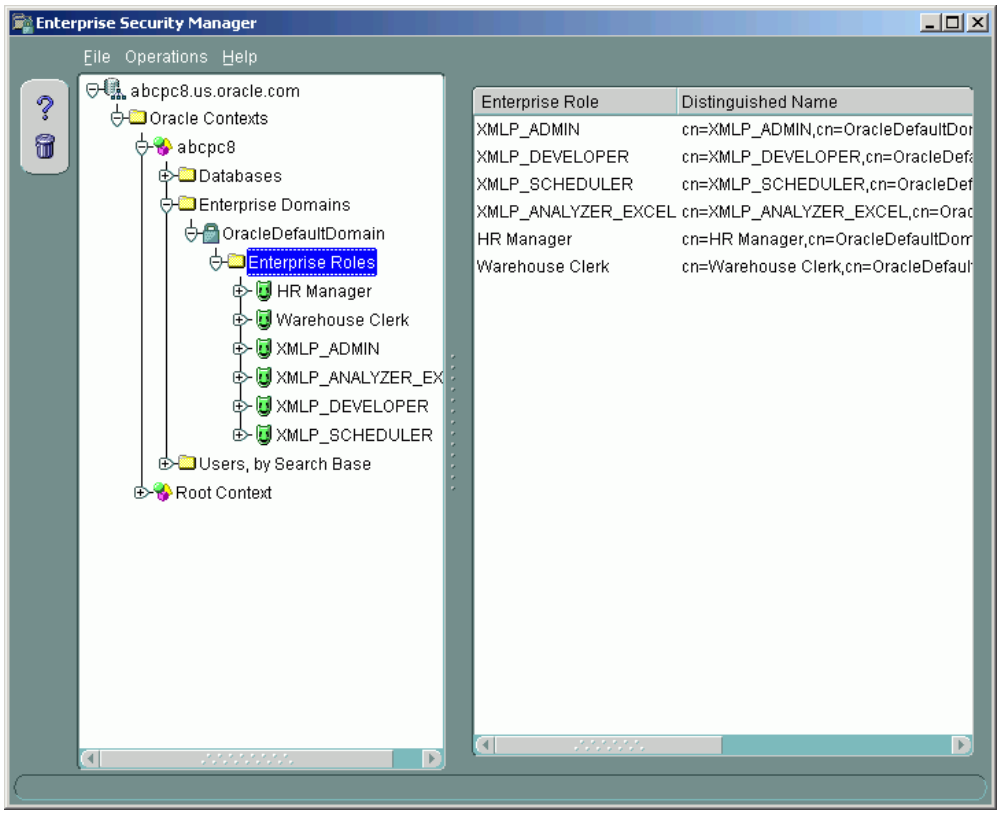
   - XMLP_ADMIN – this is the administrator role for the BI Publisher server.
   - XMLP_DEVELOPER – allows users to build reports in the system.
   - XMLP_SCHEDULER – allows users to schedule reports.
   - XMLP_ANALYZER_EXCEL – allows users to use the Excel analysis feature.
   - XMLP_ANALYZER_ONLINE – allows users to use the online analysis feature.
   - XMLP_TEMPLATE_DESIGNER - allows users to connect to the BI Publisher server from the Template Builder and to upload and download templates.

   The following figure shows a sample **Create Enterprise Role** dialog:



4. Create other functional roles as required by your implementation, for example: HR Manager, Warehouse Clerk, or Sales Manager.

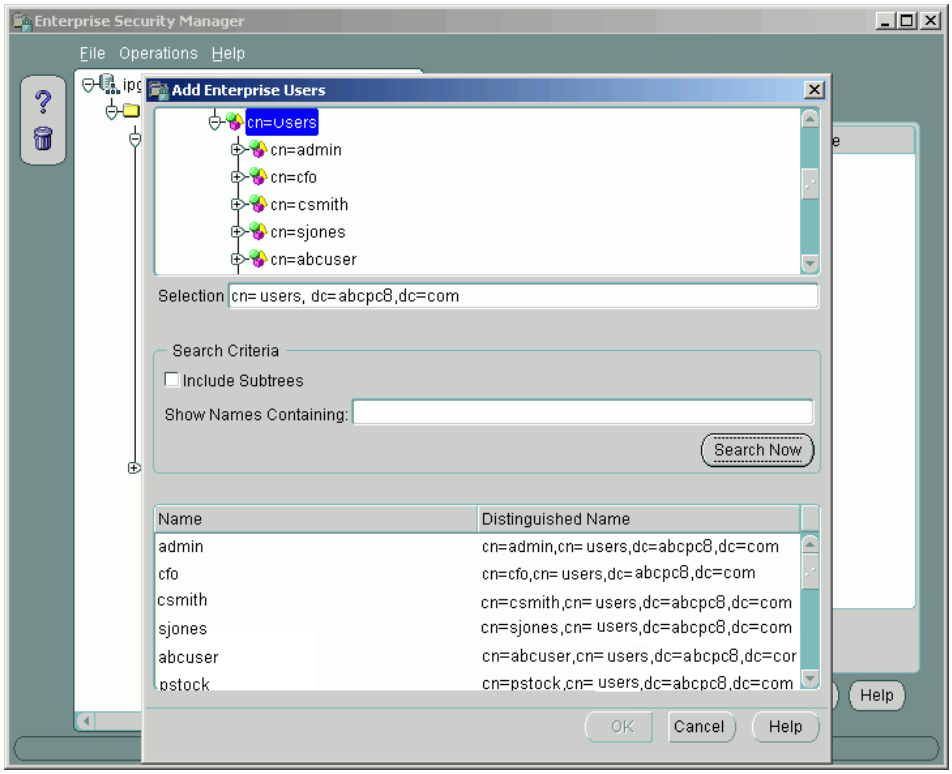
   The following figure shows an example Enterprise Security Manager screen with multiple roles defined:



5. Assign roles to users.

   - Select the role you wish to add a user to.
   - Select **Add**.
   - Navigate to the **Users** node and select **Search** to find users.

     The following figure shows an example **Add Enterprise Users** dialog:



- Select the user(s) you wish to add to the role and select **OK**.

  This action will close the dialog. Select **Apply** on the main form to save your changes.

  Now if you expand the Users node under the role, the new users will appear.

Configure the BI Publisher Server to Recognize Your LDAP Server

To configure the BI Publisher server to recognize your LDAP server, update the Security properties in the BI Publisher Admin interface as follows:

1. Navigate to the Security Configuration page: Select the **Admin** tab. Under **Security Center** select **Security Configuration**.

2. Scroll down to the **Security Model** region. Select LDAP for the Security Model.

3. Enter the following:

   - URL

     For example: `ldap://ldap.server.com:389/`

   - Administrator Username

     For example: `orcladmin`

- Administrator Password:

  For example: `welcome`

- Distinguished Name for Users

  For example: `cn=users,dc=server,dc=com`

  > **Important:** The distinguished name values are case-sensitive and must match the settings in your LDAP server.

- Distinguished Name for Groups

  For example: `cn=Groups, dc=us,dc=oracle,dc=com`

  The default value is `cn=OracleDefaultDomain,cn=OracleDBSecurity,cn=Products,cn=OracleContext,dc=example,dc=com`

- Group Search Filter

  The default values is `(&(objectclass=groupofuniquenames)(cn=*))`

- Group Attribute Name

  The default value is `cn`

- Group Member Attribute Name

  The default value is `uniquemember`

- Group Description Attribute Name

  The default value is `description`

- JNDI Context Factory Class

  The default value is `com.sun.jndi.ldap.LdapCtxFactory`

- Group Retrieval Page Size

  Setting this values enables support of the LDAPv3 control extension for simple paging of search results. By default, pagination is not used. This value determines the number of results to return on a page (for example, 200). Your LDAP server must support control type 1.2.840.113556.1.4.319 to support this feature, such as OID 10.1.4. See your LDAP server documentation for information on support of this control type.

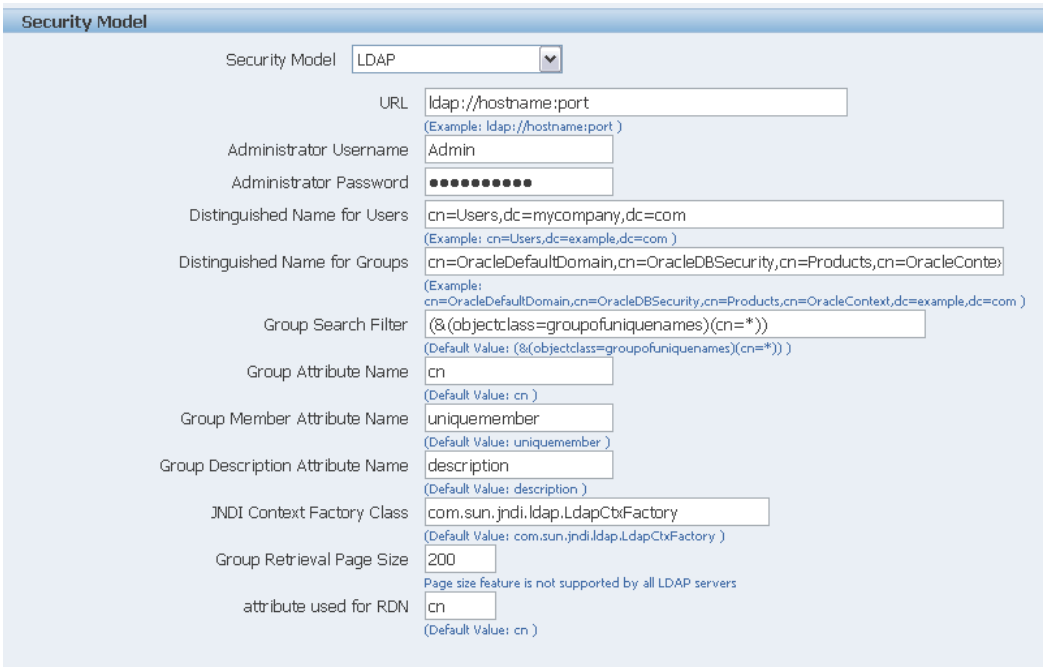
  For more information about LDAP pagination and the required control type, see the article: RFC 2696 - LDAP Control Extension for Simple Paged Results Manipulation (http://www.faqs.org/rfcs/rfc2696.html) .

- attribute used for RDN

  Enter the attribute that supplies the value for the Relative Distinguished Name. This value defaults to `cn`.

  Important: You must restart the server for changes to the security model to take effect.

The following figure shows a sample of the LDAP security model entry fields from the Security Configuration page:



Assign Folders and Data Sources to Roles

1. Log in with an Administrator role.

2. Navigate to the **Admin** tab. From **Security Center** select **Roles and Permissions**.

   You will see the roles you created and assigned in the security manager application. Note the following:

   - The XMLP_X roles are not shown because these are controlled through the LDAP interface.

   - The Users tab is no longer available under the Security Center because users are now managed through your LDAP interface.

   - Roles are not updateable in the BI Publisher interface, with the exceptions of

adding folders and adding data sources.

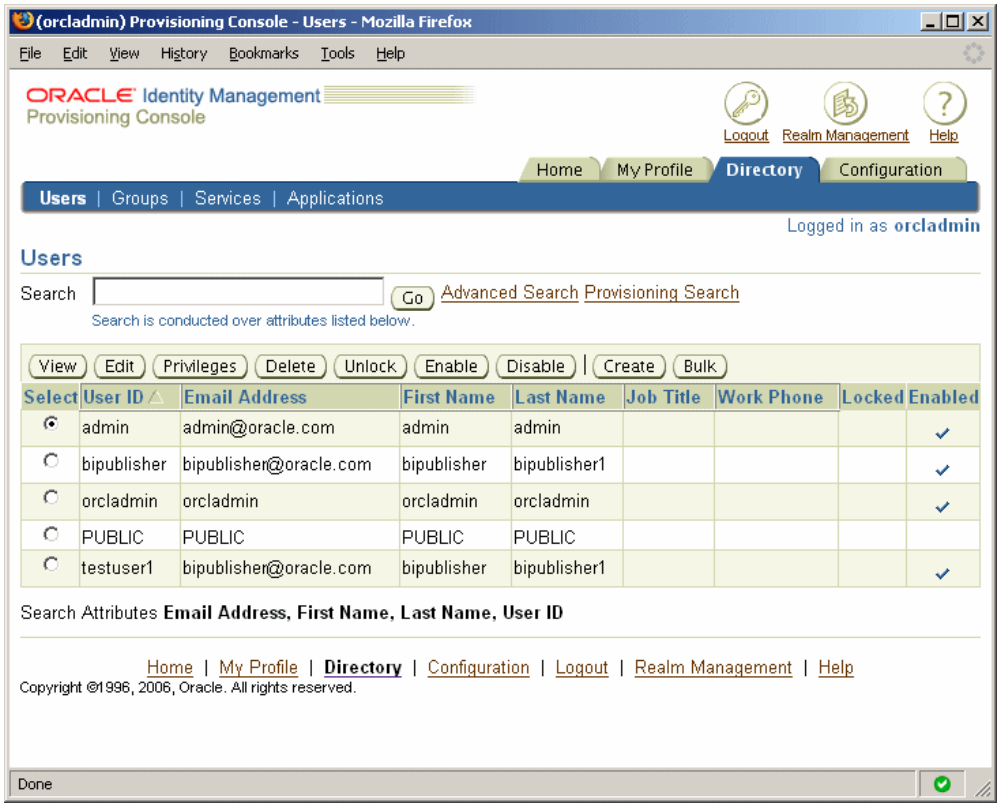
3. Select **Add Folders** to add folders to a particular role using the tree shuttle. Select **Add Data Sources** to add BI Publisher data sources to the role. A role must be assigned access to a data source to run reports from that data source.

Users can now log in using their LDAP username/password and will have access to reports in the folders assigned to their roles set up in LDAP.

Using the OID Web UI

1. Log in to OID. The URL is typically `http://(AS host):(AS port)/oiddas/`

2. Create users for BI Publisher. Select the **Directory** tab, then the **Users** subtab, and then click the **Create** button.

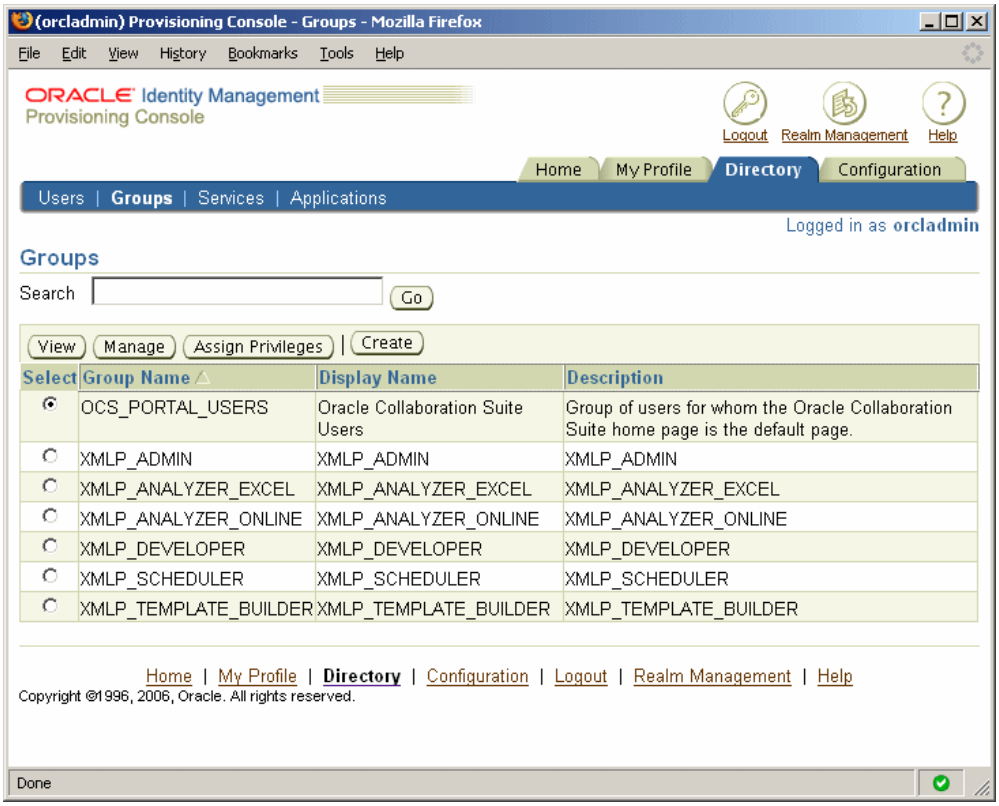


3. Create the following roles to integrate with BI Publisher: See Understanding Users and Roles, page 2-3 for full descriptions of the required functional roles.

   - XMLP_ADMIN – this is the administrator role for the BI Publisher server.
   - XMLP_DEVELOPER – allows users to build reports in the system.

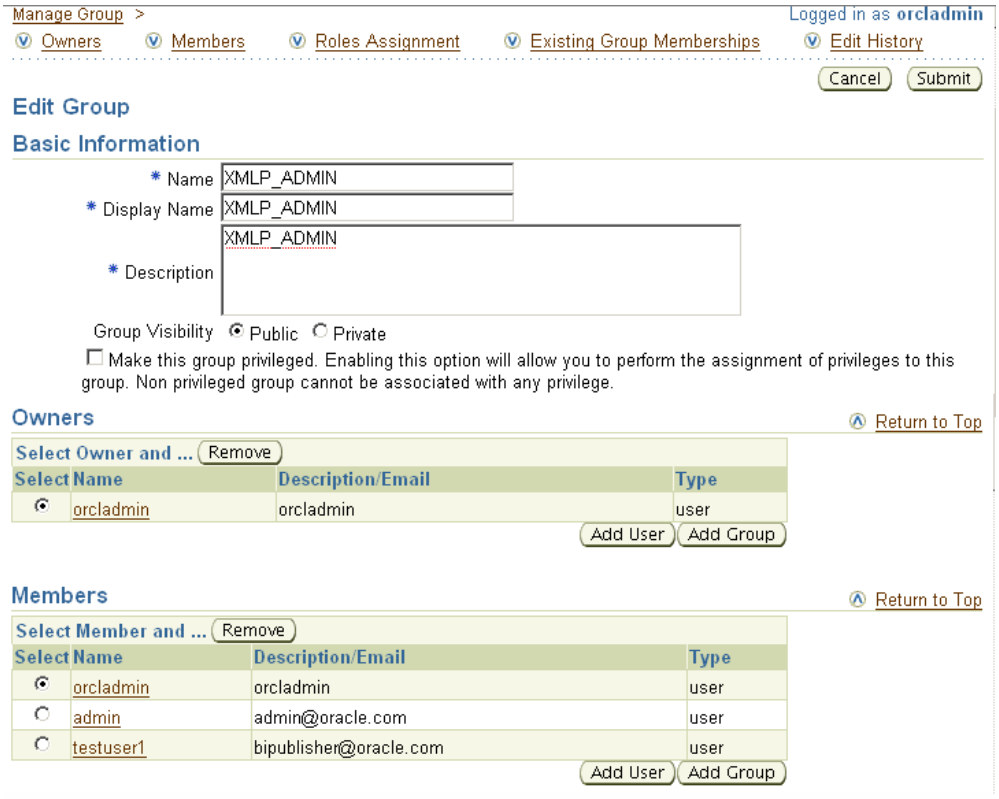
- XMLP_SCHEDULER – allows users to schedule reports.
- XMLP_ANALYZER_EXCEL – allows users to use the Excel analysis feature.
- XMLP_ANALYZER_ONLINE – allows users to use the online analysis feature.
- XMLP_TEMPLATE_DESIGNER - allows users to connect to the BI Publisher server from the Template Builder and to upload and download templates.

To create the Group, select the **Groups** subtab, then click **Create**.

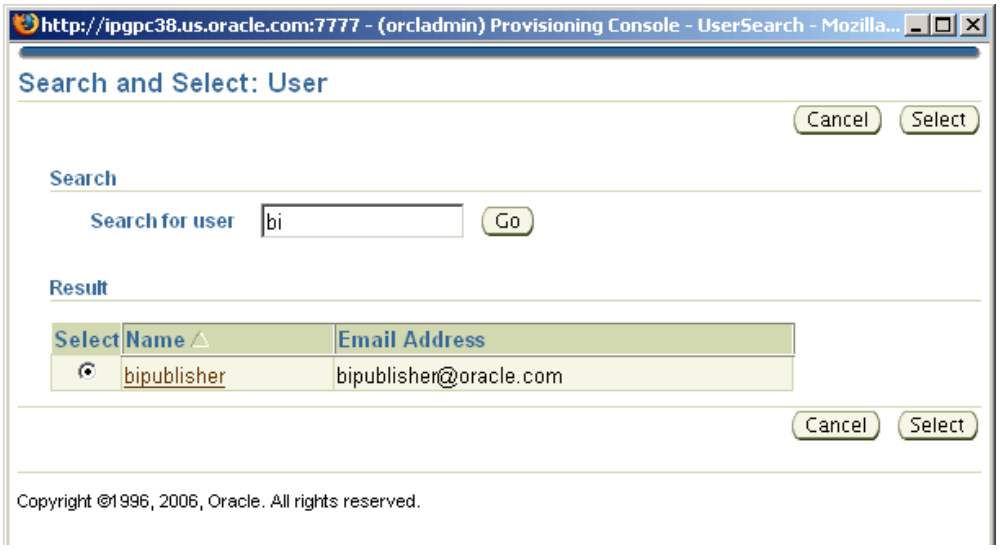


4. Assign users to the group.

   Select each group and click **Manage**, then click **Edit**.



5. Click the **Add User** button to add users to the Group.



Setting Up Oracle Single Sign-On

This section describes how to set up Oracle Single Sign-On with Oracle 10g Application

Server (OracleAS). These guidelines are written based on the Oracle 10g Application Server 10.1.3 release.

Prerequisites

- OracleAS 10g Infrastructure installation (including SSO server)
- The BI Publisher xmlpserver is set up with Oracle Internet Directory (OID) LDAP server.

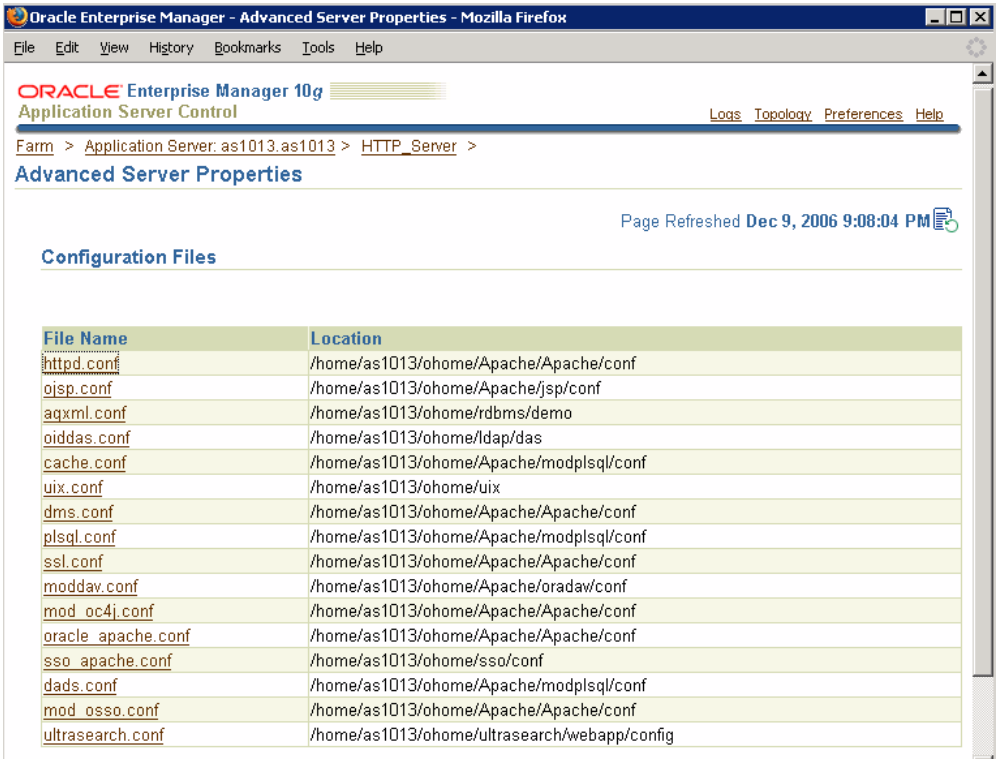
  Note: If you want to set up the BI Publisher server on a different server, that server must also be OracleAS 10g and must be registered in the main OracleAS 10g Infrastructure. This can be done by installing the new OracleAS 10g J2EE and Web Cache. The procedure for this installation is as follows (for AS 10.1.3):

  - Run the AS installer
  - Choose Oracle Application Server 10g 10.1.3
  - Choose J2EE and Web Cache
  - Follow the installer instructions. In the OID section, point to the master OracleAS 10g Infrastructure installation on your main server.

Setup Procedure

1. Create a BI Publisher Local Superuser. Before performing any security updates, you must set up a BI Publisher Local Superuser to ensure access to BI Publisher regardless of your selected security configuration. See Defining a Local Superuser, page 2-2 for more information.
2. Modify the application server configuration file. Navigate to Application Server Control (ASC). Choose **HTTP Server** and then choose **Advanced Server Properties**.

   The following figure shows a sample Advanced Server Properties page:



Select `mod_osso.conf` to open the file for editing. To protect the server, add a new "Location" directive as follows:

```
<!-- Protect xmlpserver -->
<Location /xmlpserver>
      require valid-user
      AuthType Basic
</Location>
```

3. To allow Web service communication between BI Publisher and its client components (the Template Builder and the Excel Analyzer) you must make additional modifications to the `mod_osso.conf` file. To open up the xmlpserver to allow these Web services, enter the following directives:

```
<Location /xmlpserver/services/>
  require valid-user
     AuthType Basic
     Allow from All
     Satisfy any
</Location>

<Location /xmlpserver/report_service/>
  require valid-user
     AuthType Basic
     Allow from All
     Satisfy any
</Location>

Location /xmlpserver/ReportTemplateService.xls/>
  require valid-user
     AuthType Basic
     Allow from All
     Satisfy any
</Location>
```

4. (Optional) To allow access to the Guest Folder in BI Publisher for users not signed on through SSO, you must make an additional modification to the `mod_osso.conf` file to allow traffic to the Guest folder without checking the SSO token. To do this, add the following directive:

```
<Location /xmlpserver/Guest/>
 require valid-user
     AuthType Basic
     Allow from All
     Satisfy any
</Location>
```

5. For integration with Oracle BI Presentation Services, you must disable SSO for Web services between the BI Presentation Services server and the BI Publisher server. If you made this entry when performing Step 8, you do not need to repeat this setup.

   To open up the xmlpserver to allow the Web service, enter the following directive in the `mod_osso.conf` file:

```
<Location /xmlpserver/services/>
 require valid-user
     AuthType Basic
     Allow from All
     Satisfy any
</Location>
```

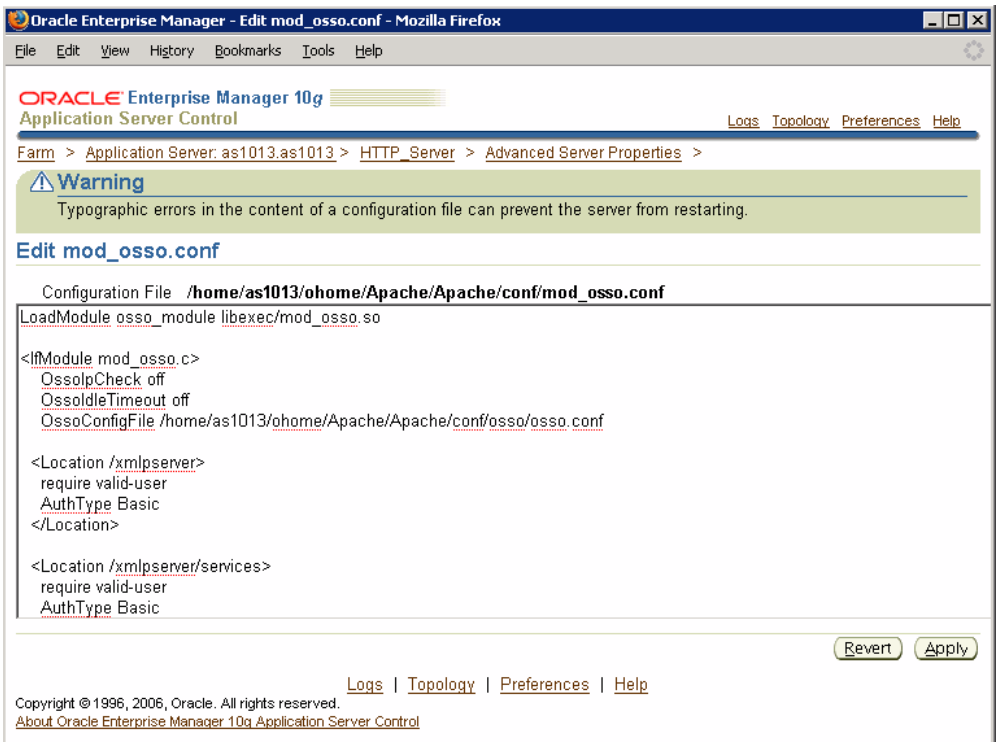
   You must make a similar entry to open the BI Presentation Services server. For more information on required configuration for BI Publisher Enterprise and Oracle BI Presentation services, see the *Oracle Business Intelligence Enterprise Edition Deployment Guide*.

   A sample `mod_osso.conf` file with the entries discussed in this section is shown below:

```
LoadModule osso_module libexec/mod_osso.so

 <IfModule mod_osso.c>
     OssoIpCheck off
     OssoIdleTimeout off
     OssoConfigFile
/home/as1013/ohome/Apache/Apache/conf/osso/osso.conf

   <Location /xmlpserver>
     require valid-user
     AuthType Basic
   </Location>


<Location /xmlpserver/services/>
  require valid-user
     AuthType Basic
     Allow from All
     Satisfy any
</Location>

<Location /xmlpserver/report_service/>
  require valid-user
     AuthType Basic
     Allow from All
     Satisfy any
</Location>

Location /xmlpserver/ReportTemplateService.xls/>
  require valid-user
     AuthType Basic
     Allow from All
     Satisfy any
</Location>

<Location /xmlpserver/Guest/>
 require valid-user
     AuthType Basic
     Allow from All
     Satisfy any
</Location>
 #
 # Insert Protected Resources: (see Notes below for how to protect
resources)
 #


 #______-
 #
 # Notes
 #
 #______-
 #
 # 1. Here's what you need to add to protect a resource,
 #    e.g. <ApacheServerRoot>/htdocs/private:
 #
 #      <Location /private>
 #      require valid-user
 #      AuthType Basic
 #      </Location>
 #
```

```
</IfModule>

 #
 # If you would like to have short hostnames redirected to
 # fully qualified hostnames to allow clients that need
 # authentication via mod_osso to be able to enter short
 # hostnames into their browsers uncomment out the following
 # lines
 #
 #PerlModule Apache::ShortHostnameRedirect
 #PerlHeaderParserHandler Apache::ShortHostnameRedirect
```

A sample of edit page is shown in the following figure:



6. Restart the HTTP server.

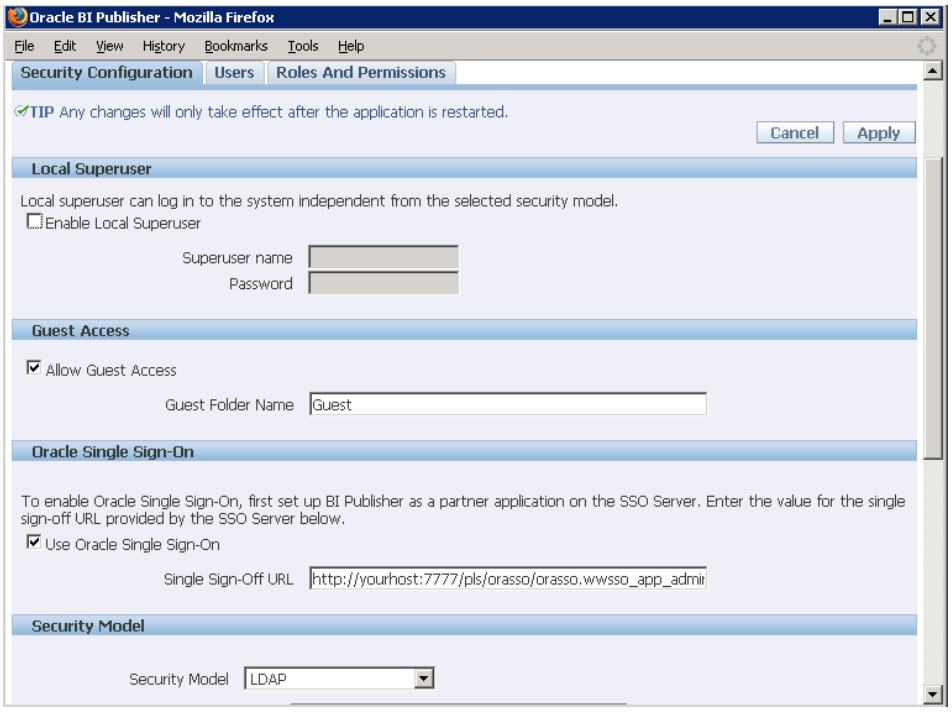
7. Set up the Single Sign-Off URL on the BI Publisher Security Configuration page.

   From the Admin tab, select Security Configuration. Enter the following in the Oracle Single Sign-On region:

   - Select **Use Oracle Single Sign-On**

   - Enter the Single Sign-Off URL with the value you wrote down in the preceding step.

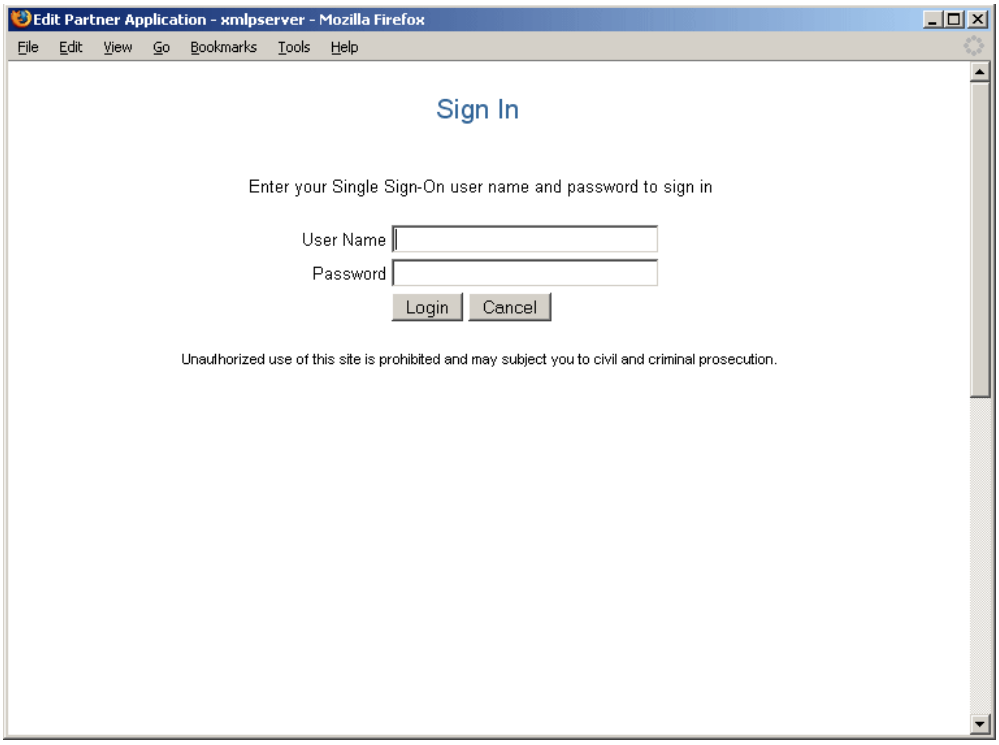
     A sample BI Publisher Security Configuration page is shown in the following figure:



8. Restart the application through the Application Server Control page.

9. Enter the URL to access the BI Publisher Enterprise application, and you will be redirected to the SSO login page.

   A sample SSO login page is shown in the following figure:



Integrating with Oracle E-Business Suite Security

BI Publisher can leverage your E-Business Suite security to enable your E-Business Suite users to log in to BI Publisher using their E-Business Suite credentials. When you integrate with the E-Business Suite security, your E-Business Suite responsibilities become available as roles in the BI Publisher security center. You can then associate BI Publisher report folders to the imported roles/responsibilities to allow access as you would using the BI Publisher native security. See Understanding Users and Roles, page 2-3.

> **Note:** In this release your users will not be able to access or execute reports stored on the E-Business Suite instance.

Setting Up the E-Business Suite Security in BI Publisher

Upload the dbc File

1. In the Oracle E-Business Suite, log in as a System Administrator and create the following responsibilities to correspond to the BI Publisher functional roles:
   - XMLP_ADMIN – this is the administrator role for the BI Publisher server.
   - XMLP_DEVELOPER – allows users to build reports in the system.

- XMLP_SCHEDULER – allows users to schedule reports.
- XMLP_ANALYZER_EXCEL – allows users to use the Excel analysis feature.
- XMLP_ANALYZER_ONLINE – allows users to use the online analysis feature.
- XMLP_TEMPLATE_DESIGNER - allows users to connect to the BI Publisher server from the Template Builder and to upload and download templates.

2. Add the new BI Publisher responsibilities to the appropriate Users.

   Note: Ensure to assign at least one user to the XMLP_ADMIN group.

3. Log in to BI Publisher Enterprise. From the Admin tab, select **Security Configuration**.
4. In the Security Model section of the page, select Oracle E-Business Suite from the list.
5. Load your dbc file from the E-Business Suite instance. This is typically located under the $FND_SECURE directory. If you do not have access to this file, contact your E-Business Suite system administrator. This file specifies how BI Publisher should access the E-Business Suite instance.
6. It is recommended that you create a local super user for the system to allow you to access the Administrator pages once the changes take effect. Select the **Enable Local Superuser** check box and enter a username and password for the user under the Local Superuser section of the Security Configuration tab.
7. Restart the BI Publisher server for the security changes to take effect.

Once you restart the system, all your E-Business Suite responsibilities will be visible as roles in the BI Publisher security center. Add folders to the E-Business Suite roles.

Add Folders to the E-Business Suite Roles

1. From the Admin tab select **Roles and Permissions**.
2. All of the responsibilities from your E-Business Suite instance will display as available roles.
3. Find the responsibility (role) that you wish to attach folders to and select **Add Folders**.

Now when EBS users log in using their EBS credentials they will have access to the folders and reports that have been attached to their responsibilities.

Integrating with Oracle BI Server Security

BI Publisher offers integration with Oracle BI Server security so that you can administer the BI Publisher users through the BI Server Administration tool. To accomplish this you must define the BI Publisher functional roles within the Oracle BI Server Administration tool, assign users to these groups, and then specify Oracle BI Security as your security model in the BI Publisher Admin interface.

> **Note:** For information on setting up Oracle BI security, see the *Oracle Business Intelligence Server Administration Guide*.

1. In the BI Server Administration tool, create the following groups to correspond to the BI Publisher functional roles:
   - XMLP_ADMIN – this is the administrator role for the BI Publisher server.
   - XMLP_DEVELOPER – allows users to build reports in the system.
   - XMLP_SCHEDULER – allows users to schedule reports.
   - XMLP_ANALYZER_EXCEL – allows users to use the Excel analysis feature.
   - XMLP_ANALYZER_ONLINE – allows users to use the online analysis feature.
   - XMLP_TEMPLATE_DESIGNER - allows users to connect to the BI Publisher server from the Template Builder and to upload and download templates.
2. Add the appropriate users to the BI Publisher groups in the BI Server Administration tool.

   > **Note:** Ensure to assign at least one user to the XMLP_ADMIN group.

3. In the BI Publisher Enterprise application, log in with Administration privileges. From the Admin tab select **Security Configuration**.
4. In the **Security Model** section of the page, select Oracle BI Server from the list. Provide the following connection information for the BI Server:
   - JDBC **Connection String** - example: `jdbc:oraclebi://host:port/`

     Note that if your Oracle BI Server is SSL-enabled, you must copy the keystore to the BI Publisher server and provide it in the connection string.

     If your Oracle BI servers are set up in a clustered configuration, the connection string must use the appropriate syntax. See Adding the Oracle BI Server as a

JDBC Data Source, page 4-4 for a description of the required syntax.

An example connection string for a clustered, SSL-enabled instance follows:

```
jdbc:oraclebi://mycompanyserver.com:9706/PrimaryCCS=BIdb01
;PrimaryCCSPort=9706;ssl=true;sslKeystorefilename=c:\mycom
pany\SSL\OracleBI\sslc\javahost.keystore;sslKeystorepasswo
rd=admin;trustanyserver=true;
```

For more information on SSL and on clustered configurations, see the *Oracle Business Intelligence Enterprise Edition Deployment Guide.*

- **Administrator Username** and **Administrator Password**
- **Database Driver Class** - example: `oracle.bi.jdbc.AnaJdbcDriver`

5. It is recommended that you create a local super user for the system to allow you to access the Administrator pages once the changes take effect. Select the **Enable Local Superuser** check box and enter a username and password for the user under the Local Superuser section of the Security Configuration tab.
6. Restart the BI Publisher server for the security changes to take effect.

Add Folders to the Oracle BI Server Roles

1. Log in to BI Publisher as a user with the XMLP_ADMIN role.
2. From the Admin tab select **Roles and Permissions**.
3. All of the groups from your Oracle BI instance will display as available roles.
4. Find the group (role) that you wish to attach folders to and select **Add Folders**.

Integrating with Oracle Database Security

BI Publisher offers integration with Oracle Database security so that you can administer the BI Publisher users with your Oracle Database users. To accomplish this you must define the BI Publisher functional roles in your Oracle Database, assign the roles to your Oracle Database users, and then specify Oracle Database as your security model in the BI Publisher Admin page.

> **Note:** For information on setting up Oracle DatabaseI security, see the *Oracle Database Security Guide*.

1. In your Oracle Database, create the following roles to correspond to the BI Publisher functional roles:
   - XMLP_ADMIN – this is the administrator role for the BI Publisher server.

- XMLP_DEVELOPER – allows users to build reports in the system.
- XMLP_SCHEDULER – allows users to schedule reports.
- XMLP_ANALYZER_EXCEL – allows users to use the Excel analysis feature.
- XMLP_ANALYZER_ONLINE – allows users to use the online analysis feature.
- XMLP_TEMPLATE_DESIGNER - allows users to connect to the BI Publisher server from the Template Builder and to upload and download templates.

2. Assign these roles to the appropriate Database users. You may also want to create additional reporting roles that you can utilize when setting up your report privileges on the BI Publisher side. For example, you might create a role called "HUMAN_RESOURCES_MANAGER" that you can assign a Human Resources Folder of reports to. You can then assign that role to any user requiring access to the Human Resources reports.
3. Ensure to assign the XMLP_ADMIN role to a user with administration privileges, such as SYSTEM.
4. In the BI Publisher application, log in with Administration privileges. From the Admin tab select **Security Configuration**.
5. In the **Security Model** section of the page, select Oracle Database from the list. Provide the following connection information:
   - JDBC **Connection String** - example: `jdbc:oracle:thin:@mycompany.com:1521:orcl`
   - **Administrator Username** and **Administrator Password** - note that the user that you enter must have privileges to access data from the `dba_users/_roles/_role_privs tables`.
   - **Database Driver Class** - example: `oracle.jdbc.driver.OracleDriver`
6. It is recommended that you create a local super user for the system to allow you to access the Administrator pages once the changes take effect. Select the **Enable Local Superuser** check box and enter a username and password for the user under the Local Superuser section of the Security Configuration tab.
7. Restart the BI Publisher server for the security changes to take effect.
8. After you restart the server, you can log in with the Oracle Database user to which you assigned the XMLP_ADMIN role. You will now see all your Oracle Database users and roles.

9. Assign report folders to the Oracle Database roles so that users can access reports.

Implementing a Digital Signature

Oracle BI Publisher supports digital signatures on PDF output documents. Digital signatures enable you to verify the authenticity of the documents you send and receive. Oracle BI Publisher can access your digital ID file from a central, secure location and at runtime sign the PDF output with the digital ID. The digital signature verifies the signer's identity and ensures that the document has not been altered after it was signed.

For additional information on digital signatures, see the following sources:

- Digital ID Introduction by Verisign
  - http://www.verisign.com/support/tlc/per/whitepaper.htm
  - http://www.verisign.com/stellent/groups/public/documents/gui des/005326.pdf
- Digital Signature by Adobe
  - http://www.adobe.com/security/digsig.html
- Digital Signatures in PDF and Acrobat
  - http://www.acrobatusers.com/articles/2006/07/digital_ signatures/index.php

Prerequisite

Before you can implement digital signatures with Oracle BI Publisher output documents, you need the following:

A digital ID obtained from a public certificate authority or from a private/internal certificate authority (if for internal use only). You must copy the digital ID file to a secure location of the file system on the server that is accessible by the BI Publisher server.

Limitations

Use of digital signatures with Oracle BI Publisher output documents has the following limitations:

- Only a single digital ID can be registered with BI Publisher. Future releases will support multiple digital IDs.
- Only reports submitted through BI Publisher's Schedule Report interface can include the digital signature.

- The digital signature is enabled at the report level; therefore, multiple templates assigned to the same report share the digital signature properties.

Procedure for Implementing a Digital Signature

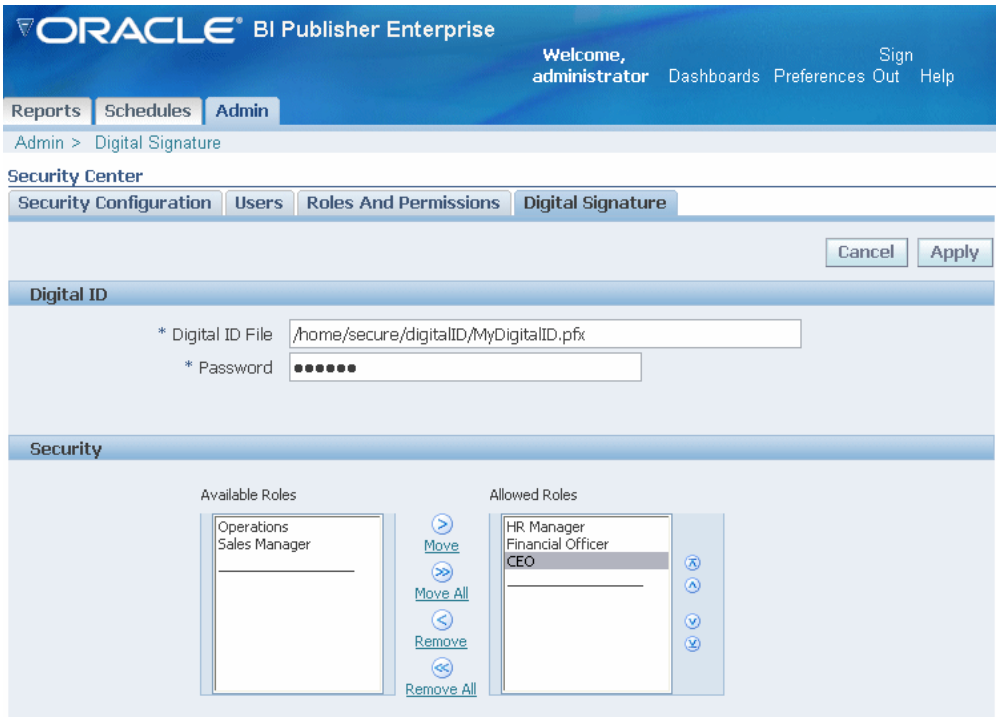
The following steps provide an overview of the tasks required to set up and sign your output PDF documents with a digital signature:

1. Register the digital ID in the BI Publisher Admin page.
2. Specify the Roles that are authorized to sign documents.
3. (Optional for PDF templates) Add a signature field to the PDF template in which to place the digital signature at runtime. See Adding or Designating a Field for Digital Signature, *Oracle BI Publisher Report Designer's Guide* for instructions on designating a specific field in a PDF template for the digital signature.
4. Enable Digital Signature for the report in the Runtime Configuration, page 8-6 properties and specify the position to place the digital signature on the completed document. This can be a signature field (for PDF templates), general location (top left, top center, or top right), or you can specify x and y coordinates.
5. Log in to BI Publisher as a user with an authorized role and submit the report through the BI Publisher scheduler, choosing PDF output. When the report completes it will be signed with your digital ID in the specified location of the document.

Registering Your Digital Signature ID and Assigning Authorized Roles

Currently, BI Publisher supports the identification of a single digital ID file only. Register the digital ID in the BI Publisher Admin page as follows:

1. Log in to BI Publisher with Administrator credentials.
2. Select the **Admin** tab, and then from the **Security Center** list select **Digital Signature**.
3. On the Digital Signature subtab, enter the file path to the digital ID file and enter the password for the digital ID.
4. Enable the Roles that will have the authority to sign documents with this digital ID. Use the shuttle buttons to move **Available Roles** to the **Allowed Roles** list.
5. Click **Apply**. The following figure shows the Digital Signature subtab:


ORACLE BI Publisher Enterprise
Welcome, administrator
Dashboards
Preferences
Sign Out
Help
Reports
Schedules
Admin
Admin > Digital Signature
Security Center
Security Configuration
Users
Roles And Permissions
Digital Signature
Cancel
Apply
Digital ID
* Digital ID File
/home/secure/digitalID/MyDigitalID.pfx
* Password
Security
Available Roles
Operations
Sales Manager
Move
Move All
Remove
Remove All
Allowed Roles
HR Manager
Financial Officer
CEO

3

Using the Admin Functions

This chapter covers the following topics:

- Overview
- Managing Reports and Folders

Overview

Use the Admin interface to set up the following:

- Data Sources, page 4-1
- Security Center Options

  Note: For the description of the Security Center options, see Defining a Security Model, page 2-1.

- System Maintenance Options, including Server and Scheduler Configuration, page 6-1
- Delivery Options, page 7-1
- Runtime Configuration, page 8-1
- Integration with

  BI Presentation Services, page 5-1

  Oracle BI Discoverer, page 5-2

  Hyperion Workspace and Shared Services, page 5-5

  Oracle Smart Space Client, page 5-13

Managing Reports and Folders

Administration View

If you are assigned the BI Publisher Administrator role your Reports home will display an additional Users folder. This folder will contain all the "My Folders" folders of all the users in your system. Each user "My Folders" folder is named according to the User name. The Administrator can see and update all the user folders.

Folder and Report Tasks

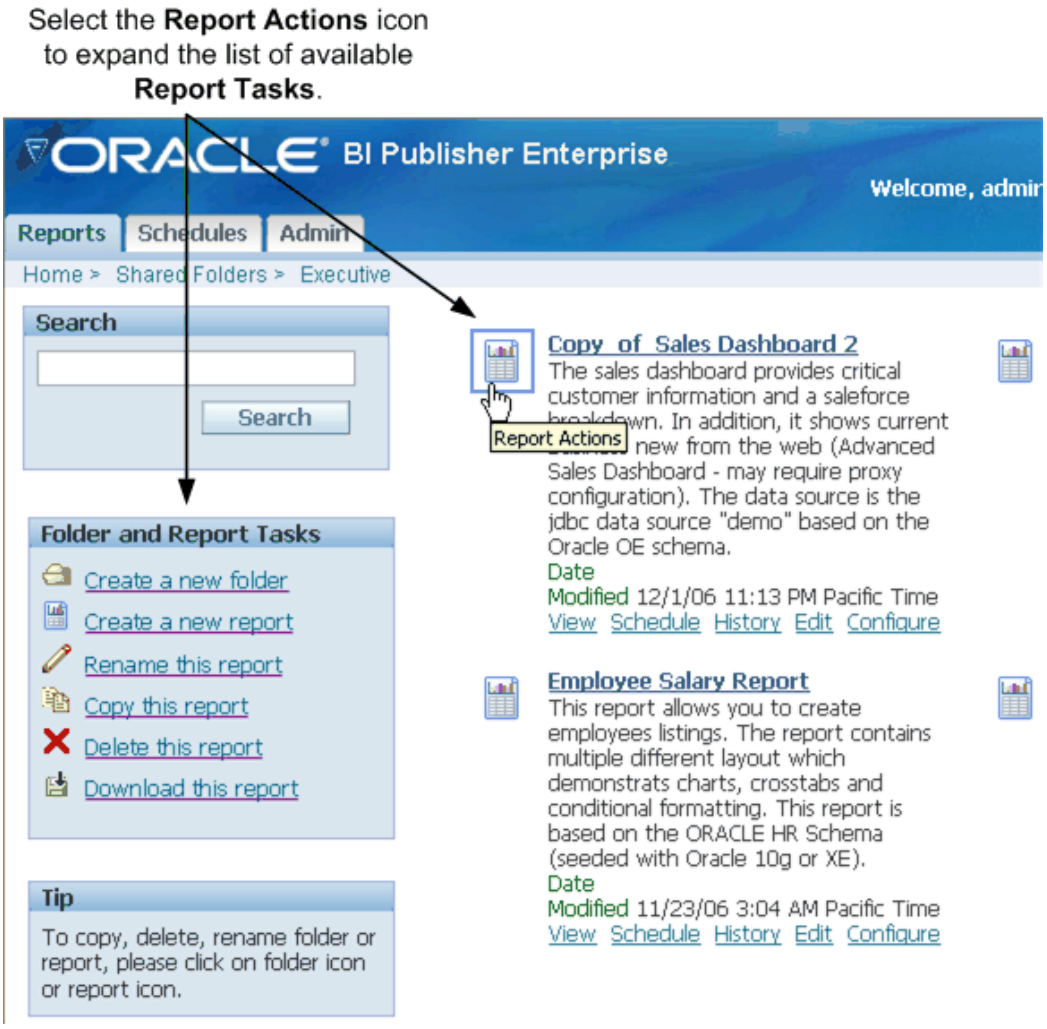
The Folder and Report Tasks menu is available to users assigned the BI Publisher Administrator role or the BI Publisher Developer Role.

Report Tasks

Select the Report Actions icon to expand the menu of available report tasks. Report Tasks include:

- Rename this report
- Copy this report
- Delete this report
- Download this report

> **Note:** The Report Actions menu can be viewed only by the BI Publisher Administrator role and the BI Publisher Developer role.



Rename a Report

1. Navigate to the report folder.
2. Select the **Report Actions** icon.
3. Select **Rename this report** from the **Folder and Report Tasks** region.
4. Enter the new name in the text box.

Upload a Report

You can upload a report definition directory as a zip file. The directory must include the report definition file (.xdo file). The zip file must have the same name as the .xdo file contained in the directory. You may also include template files (rtf, pdf, excel, and xsl-fo

files), translation files (.xlf), PDF mapping files (.map files), and sample XML data files (.xml).

1. Navigate to the report folder.
2. Select the **Report Actions** icon.
3. Select **Upload a Report** from the **Folder and Report Tasks** region.
4. Browse to locate the report, then select **Upload**.

Copy a Report

1. Navigate to the report folder.
2. Select the **Report Actions** icon.
3. Select **Copy this report** from the **Folder and Report Tasks** region to copy the report to the clipboard.
4. Navigate to the location you want to copy the report to and select **Paste from clipboard**.

Download a Report

1. Navigate to the report folder.
2. Select the **Report Actions** icon.
3. Select **Download this report** from the **Folder and Report Tasks** region. You will be prompted to save the report definition zip file.

The report definition directory is downloaded to the specified location as a zip file. The report definition directory contains all the files associated with the report, this may include:

- The report definition file (.xdo file)
- All template files (rtf, pdf, and xls files)
- PDF mapping files (.map file)
- Sample data files (xml file)
- Translation files (.xlf files)

Delete a Report

Note: You must be assigned the BI Publisher Developer role or BI Publisher Administration Role to delete reports.

1. Navigate to the report folder.
2. Select the **Report Actions** icon.
3. Select **Delete this report** from the **Folder and Report Tasks** region.

Folder Tasks

Select the Folder Actions icon to expand the menu of available folder tasks. Folder Tasks include:

- Create a new folder
- Rename this folder
- Copy this folder
- Delete this folder

Note: The Report Actions menu can be viewed only by the BI Publisher Administrator role and the BI Publisher Developer role.

Create a New Folder

You can create folders in My Folder, or navigate to the Shared folder in which you want the new folder to reside.

1. From the **Folder and Report Tasks** menu, select **Create a new folder**
2. Enter the Folder Name in the text box that launches, and select **Create**.

Rename a Folder

1. Select the Folder icon to view all **Folder Actions** in the **Folder and Report Tasks** menu.
2. Select **Rename this folder**.
3. Enter the new report name in the text box that launches, and select **Rename**.

Delete a Folder

1. Select the Folder icon to view all **Folder Actions** in the **Folder and Report Tasks** menu.
2. Select **Delete this folder**.
3. Confirm the deletion at the prompt.

4

Setting Up Data Sources

Setting Up Data Sources

BI Publisher supports a variety of data sources from which you can supply XML data. The data can come from a database, an HTTP XML feed, a Web service, an Oracle BI Answers request, an Oracle BI Discoverer worksheet, an OLAP cube, or a previously generated XML file.

This section describes how to set up the following:

- Connections to your database via a direct JDBC connection, page 4-3
- Connections to your database via a JNDI connection pool , page 4-6
- Connections to specific database types configured for Online Analytical Processing (OLAP), page 4-7
- Files, page 4-8

  The files data source option enables you to define a directory to which BI Publisher can connect. You can then place XML files in this directory to use as data input to your reports.

> **Note:** Connections to an HTTP XML feed or a Web service are set up when you define the data model for your report (see Defining the Data Model for Your Report, *Oracle Business Intelligence Publisher Report Designer's Guide*). Connections to Oracle BI Discoverer and Oracle BI Presentation Services are set up through the integration screens (see Setting Up Integrations, page 5-1).

When you set up data sources, you also define security for the data source by selecting which BI Publisher roles can access the data source.

This security mechanism is intended for use with the BI Publisher Developer role to

restrict developer use of data sources. For example, suppose you have two data sources: a database containing Financials data and a database containing Human Resources data. The Financials developers should only have access to the Financials data. You can create a role called Financials Developer and assign it the BI Publisher Developer role. You can then assign the Financials Developer to the Financials data source. When the user assigned this role logs in to create reports, he can only see the Financials data source.

By default, BI Publisher Administrators can access all data sources.

If you have not set up the user roles yet, you can assign data sources to a role from the Create Role interface. See Understanding Users and Roles, page 2-3 for more information.

About Proxy Authentication

BI Publisher supports proxy authentication for connections to the following data sources:

- Oracle 10*g* database
- Oracle 11*g* database
- Oracle BI Server

For direct data source connections via JDBC and connections via a JNDI connection pool, BI Publisher enables you to select "Use Proxy Authentication". When you select Use Proxy Authentication, BI Publisher passes the user name and password of the individual user (as logged into BI Publisher) to the data source and thus preserves the client identity and privileges when the BI Publisher server connects to the data source. For more information on Proxy Authentication in Oracle databases, refer to *Oracle Database Security Guide 10g* or the *Oracle Database Security Guide 11g*.

Note that for connections to the Oracle BI Server, Proxy Authentication is required. In this case, proxy authentication is handled by the Oracle BI Server, therefore the underlying database can be any database that is supported by the Oracle BI Server.

Prerequisites

❒ The JDBC driver for your selected database must be available to BI Publisher. If you are using an Oracle database or one of the DataDirect drivers provided by BI Publisher, then the drivers will be installed in the correct location and there is no further setup required.

For all other database and driver selections, you have the following options to make the native JDBC drivers available to BI Publisher:

- (Recommended) For OC4J deployments, you can copy the native driver JAR files to the following location: `<oc4j_home>/j2ee/home/applib`

- (Recommended) Configure your Web application server to register the JDBC drivers. See your application server documentation for information on registering third-party JDBC drivers.

- Copy the native driver JAR files to the following location:
  `xmlpserver/WEB-INF/lib`

  If you choose this option, you will be required to repeat this step each time you redeploy the BI Publisher server for an upgrade or other reason.

- Add the JDBC drivers to your application server's Java CLASSPATH.

Setting Up a JDBC Connection to Your Data Source:

1. From the Admin page select **JDBC Connection**. This will display the list of existing JDBC connections.

2. Select the **Add Data Source** button.

3. Enter the following fields for the new connection:

   - **Data Source Name** - enter a display name for the data source. This name will appear in the Data Source selection list in the Report Editor.

   - **Driver Type** - select your database type from the list. When you select a driver type, BI Publisher will automatically display the appropriate Database Driver Class as well as provide the appropriate Connection String format for your selected database.

   - **Database Driver Class** - This will be automatically entered based on your selection for Driver Type. You can update this field if desired.

     For example: `oracle.jdbc.OracleDriver` or

     `com.microsoft.jdbc.sqlserver.SQLServerDriver`

   - **Connection String** - enter the database connect string.

     When you select the driver type, this field will automatically display the appropriate connection string format for your database type.

     For an Oracle database the connect string will have the following format:

     `jdbc:oracle:thin@server:port:sid`

     For example:
     `jdbc:oracle:thin@myserver.mycompany.com:1521:prod`

     For a Microsoft SQL Server, the connect string will have the following format:

     `jdbc:sqlserver://server`

     For example:

```
jdbc:sqlserver://myserver.mycompany.com
```

- **User Name** - enter the user name required to access the data source on the database.
- **Password** - enter the password associated with the user name for access to the data source on the database.
- **Use Proxy Authentication** - select this box to enable Proxy Authentication. This supported for Oracle 10*g* or Oracle 11*g* deployments only.

4. Select **Test Connection**. If the test is successful, the confirmation message, "Connection established successfully" will display. If connection error occurs, the message "Could not establish connection," will display.
5. Define security for this data source. Use the shuttle buttons to move roles from the **Available Roles** list to the **Allowed Roles** list. Only users assigned the roles on the Allowed Roles list will be able to create reports from this data source.

Setting Up a JDBC Connection to the Oracle BI Server:

Note: If you included BI Publisher Enterprise in your Oracle BI Enterprise Edition installation, this data source will be automatically added. You must configure the settings appropriately for your deployment.

To add the Oracle BI Enterprise Edition server as a JDBC data source, follow the guidelines in Adding a JDBC Data Source, page 4-3.

Note that if your Oracle BI Server is SSL-enabled, you must copy the keystore to the BI Publisher server and provide it in the connection string. If your Oracle BI servers are set up in a clustered configuration, the connection string must use the appropriate syntax described in this section.

The entries for Database Driver Class and Connection String must be as follows:

Database Driver Class: `oracle.bi.jdbc.AnaJdbcDriver`

Connection String: The appropriate connection string depends on your specific deployment. Clustered and SSL-enabled deployments require specific parameters to construct the URL. For example, if your Oracle BI Server is SSL-enabled, you must copy the keystore to the BI Publisher server and provide it in the connection string. For more information on SSL and on clustered configurations, see the *Oracle Business Intelligence Enterprise Edition Deployment Guide*.

The URL for the connection string requires the following format:

```
<URL>:= <Prefix>: [//<Host>:<Port>/][<Property Name>=<Property Value>;]*
```

where

<Prefix>: is the string jdbc:oraclebi

<Host>: is the hostname of the analytics server. It can be an IP Address or hostname. The default is localhost.

<Port> is he port number that the server is listening on. The default is 9703.

```
<Property Name>:=
<Catalog>|<User>|<Password>|<SSL>|<SSLKeyStoreFileName>
|<SSLKeyStorePassword>|<TrustAnyServer>|<TrustStoreFileName
>|<TrustStorePassword>|<LogLevel>|<LogFilePath>|<PrimaryCCS>|<PrimaryCCS
Port>| <SecondaryCCS>|<SecondaryCCSPort>
```

Valid property values are:

<Catalog> - can be any catalog name that is available on the server. If the catalog is not specified, then it will default to the default catalog specified by the server. If the catalog name is not found in the server, it will still use the default catalog and issue a warning during connect.

<User> - specifies the username for the BI Server. The default is "Administrator".

<Password>- specifies the password for the BI Server for the username. The password will be encrypted using 3DES.

<SSL>True|False - default is False. Specifies if the JDBC driver will use SSL or not. If true, the driver will check if SSLKeyStoreFileName is readable; if not, it will issue an error message.

<SSLKeyStoreFileName> - the name of the file that store the SSL Keys. This file must exist in the local file system and be readable by the driver.

<SSLKeyStorePassword> - the password to open the file pointed to by SSLKeyStoreFileName.

<TrustAnyServer> - True | False - the default is False. If SSL is set to "True" the property specifies whether to check the trust store for the server. If TrustAnyServer is set to "False", the driver will verify that TrustStoreFileName is readable.

<TrustStoreFileName> - if TrustAnyServer is set to false, this property is required to specify the trust store file name.

<TrustStorePassword> - if TrustAnyServer and TrustStoreFileName are specified, this property specifies the password to open up the file specified by TrustStoreFileName.

<LogLevel> - specify the log level. Valid values are

SEVERE | WARNING | INFO | CONFIG | FINE | FINER | FINEST

<LogFilePath> - specifies the file path of the desired logging destination. Default is %TEMP% on windows, $TMP on UNIX. Driver needs to have write permission on the file. It will create a new entry marked as _0, _1 if the same file name already exists.

<PrimaryCCS> -(For clustered configurations) specifies the primary CCS machine

name instead of using the "host" to connect. If this property is specified, the "host" property value is ignored. The jdbc driver will try to connect to the CCS to obtain the load-balanced machine. Default is localhost.

`<PrimaryCCSPort>` - specifies the primary CCS port number running on the PrimaryCCS machine. Default is 9706.

`<SecondaryCCS>` - specifies the secondary CCS machine name instead of using the " host" to connect. If this property is specified, then the jdbc driver will try to connect to the CCS to obtain the load-balanced machine. Default is localhost.

`<SecondaryCCSPort>` - specifies the secondary CCS port number running on the secondary machine. Default is 9706.

Following is an example connection string for a clustered deployment with SSL enabled:

```
jdbc:oraclebi://machine01.domain:9706/PrimaryCCS=machine01;PrimaryCCSPor
t=9706;SecondaryCCS=machine02;SecondaryCCSPort=9706;user=admin;password=
welcome;ssl=true;sslKeystorefilename=c:\mycompany\OracleBI\sslc\javahost
.keystore;sslKeystorepassword=welcome;trustanyserver=true;
```

Use Proxy Authentication - select this box. If you included BI Publisher in your Business Intelligence Enterprise Edition installation, this box will be enabled by default.

Setting Up a Connection to a JNDI Connection Pool:

BI Publisher supports connecting to your JDBC data source via a connection pool. Using a connection pool increases efficiency by maintaining a cache of physical connections that can be reused. When a client closes a connection, the connection gets placed back into the pool so that another client can use it. A connection pool improves performance and scalability by allowing multiple clients to share a small number of physical connections. You set up the connection pool in your application server and access it via Java Naming and Directory Interface (JNDI).

After you set up the connection pool in your application server, enter the required fields in this page so that BI Publisher can utilize the pool to establish connections. For information on setting up a connection pool in OC4J, see the chapter "Data Sources" in the *Oracle Containers for J2EE Services Guide 10g*.

1. From the Admin page select **JNDI Connection**. This will display the list of existing JNDI connections.

2. Select the **Add Data Source** button.

3. Enter the following fields for the new connection:

   - **Data Source Name** - enter a display name for the data source. This name will appear in the Data Source selection list in the Report Editor.

   - **JNDI Name** - enter the JNDI location for the pool. For example, `jdbc/BIP10gSource`.

- **Use Proxy Authentication** - select this box to enable Proxy Authentication. This is supported for Oracle 10*g* or Oracle 11*g* deployments only.

4. If you would like to test the connection, select **Test Connection**. If the test is successful, the confirmation message, "Connection established successfully" will display. If connection error occurs, the message "Could not establish connection," will display.

5. Define security for this data source. Use the shuttle buttons to move roles from the **Available Roles** list to the **Allowed Roles** list. Only users assigned the roles on the Allowed Roles list will be able to create reports from this the data source.

Setting Up a Connection to an OLAP Data Source:

BI Publisher supports connecting to several types of OLAP databases. Note that to connect to Microsoft SQL Server 2000 Analysis Services, BI Publisher must be installed on a supported Windows operating system.

1. From the Admin page select **OLAP Connection**. This will display the list of existing JNDI connections.

2. Select the **Add Data Source** button.

3. Enter the following fields for the new connection:

   - **Data Source Name** - enter a display name for the data source. This name will appear in the Data Source selection list in the Report Editor.

     OLAP Type - choose from the list of supported OLAP databases. When you choose the type, the OLAP Connection String field will update with the appropriate connection string format for your selection.

   - **OLAP Connection String** - enter the connection string for your OLAP database. Following are examples for each of the supported OLAP types:.

     - Oracle's Hyperion Essbase

       Format: [server name]

       Example: MyEssbaseServer

     - Microsoft SQL Server 2000 Analysis Services

       Format: Data Source=[server];Provider=msolap;Initial Catalog=[catalog]

       Example: Data Source=myMSServer;Provider=msolap;Initial Catalog=VideoStore

     - Microsoft SQL Server 2005 Analysis Services

Format: Data Source=[server];Provider=msolap.3;Initial Catalog=[catalog]

Example: Data Source=myMSServer;Provider=msolap.3;Initial Catalog=VideoStore

- SAP BW

  Format: ASHOST=[server] SYSNR=[system number] CLIENT=[client] LANG=[language]

  Example: ASHOST=166.21.57.44 SYSNR=01 CLIENT=800 LANG=EN

- **Username** and **Password** for the OLAP database.

4. If you would like to test the connection, select **Test Connection**. If the test is successful, the confirmation message, "Connection established successfully" will display. If connection error occurs, the message "Could not establish connection," will display.

5. Define security for this data source. Use the shuttle buttons to move roles from the **Available Roles** list to the **Allowed Roles** list. Only users assigned the roles on the Allowed Roles list will be able to create reports from this the data source.

Setting Up a Connection to a File Data Source:

1. From the Admin page select **File**. This will display the list of existing file sources.

2. Select the **Add Data Source** button.

3. Enter the following fields for the new data source:

   - **Data Source Name** - enter a display name for the data source. This name will appear in the Data Source selection list in the Report Editor.

   - **Path** - enter the full path to the top-level directory on your server.

4. Define security for this data source. Use the shuttle buttons to move roles from the **Available Roles** list to the **Allowed Roles** list. Only users assigned the roles on the Allowed Roles list will be able to create reports from this the data source.

Viewing or Updating a Data Source:

1. From the Admin page select the Data Source type to update.

2. Select the name of the connection to view or update. All fields are updateable.

3. Select **Apply** to apply any changes or **Cancel** to exit the update page.

5

Setting Up Integrations

Introduction

This chapter describes how to set up integration with the following:

- Oracle BI Presentation Services , page 5-1

  This integration enables you to use Oracle BI Answers requests as data sources for your reports. Once you have successfully set up the integration, Oracle BI Answers will be enabled as a data set type selection when building your Data Model in the Report Editor. The integration also enables connection to the Presentation Services catalog from the Template Builder Add-in for Microsoft Word.

- Oracle BI Discoverer, page 5-2

  This integration enables you to use Oracle BI Discoverer Worksheets as data sources for your reports. Once you have successfully set up the integration, Oracle BI Discoverer will be enabled as a data set type selection when building your Data Model in the Report Editor. The integration also enables connection to Discoverer from the Template Builder Add-in for Microsoft Word.

- Hyperion Workspace and Shared Services, page 5-5

  This integration enables you to access BI Publisher from within the Oracle Enterprise Performance Management Workspace.

- Oracle Smart Space Client, page 5-13

  This integration enables you to set up a link to the Oracle Smart Space download page. This will enable all BI Publisher users to easily access the Oracle Smart Space client download page from their BI Publisher session.

Setting Up Integration with Oracle BI Presentation Services

By setting up integration with Oracle BI Presentation Services you enable connection to

Oracle BI Answers requests as data sources for your reports.

The Oracle BI installer when installing BI Publisher, performs integration with Presentation Services if Presentation Services is also installed with BI Publisher. This means that the Oracle BI installer sets the Presentation Services hostname, port, url values in BI Publisher configuration file xmlp-server-config.xml. But the user still needs to set the Presentation Services username and password in BI Publisher configuration since those are not known at install time.

> **Note:** If you included BI Publisher Enterprise in your Oracle BI Enterprise Edition installation, the Oracle BI installer will set the Presentation Services hostname, port, and url values. However, you must manually enter the Presentation Services username and password here.

1. From the Admin page, under **Integration**, select **Oracle BI Presentation Services**.
2. Enter the following information about your BI Presentation Services server:
   - Server Protocol - select http or https
   - Server Version - select v4
   - Server - enter the server host name. For example: `server01.mycompany.com`
   - Port for the server
   - Administrator Username and Password
   - URL Suffix - default value is: `analytics/saw.dll`

     > **Note:** If your deployment is configured for SSO, ensure that this suffix matches the non-SSO application you set up to allow the Web service between the BI Publisher and BI Presentation Services servers. For example: `analyticsSOAP/saw.dll`. For more information, see Setting Up Oracle Single Sign-On, page 2-17.

   - Session time out in minutes

Setting Up Integration with Oracle BI Discoverer

Oracle BI Discoverer is an intuitive ad-hoc query, reporting, analysis, and Web publishing toolset that gives business users immediate access to information in databases. Using any standard Web browser, you have secure and immediate access to

data from both relational and multidimensional data sources. Oracle BI Discoverer provides a business view to hide the complexity of the underlying data structures, enabling you to focus on solving business problems.

Oracle BI Publisher can accept Oracle BI Discoverer worksheets as the data set for a report. This provides three key benefits to Discoverer users. Discoverer users will now be able to:

- Create highly formatted reports
- Schedule execution and Delivery of reports to wide range of destinations
- Generate reports in a greater range of formats

Oracle BI Publisher integrates with Oracle BI Discoverer via the Oracle BI Discoverer Web service. Use the BI Publisher Admin screen to enter the necessary integration information.

Prerequisites for Integration with Discoverer

The following are prerequisites for the integration with Discoverer:

- Oracle BI Discoverer 10.1.2.3 or BI Discoverer 10.1.2.2 has been installed with theBI Discoverer patch that includes the BI Discoverer Web Service.

  Note: The BI Discoverer (and patch) installation will install and deploy the Discoverer Web Service (discWS.jar) into the same Oracle Containers for J2EE (OC4J) that hosts Discoverer services. Once installed, the Discoverer Web service will be accessible from the following URL:

  `http://<1012AppServer>:<port>/discoverer/wsi`

- BI Discoverer has been configured to use Oracle Application Server Identity Management.

  Note: For information on how to configure BI Discoverer to use Oracle Application Server Identity Management see the *Oracle Business Intelligence Discoverer Configuration Guide*, topic: "Using Discoverer with Oracle Identity Management Infrastructure."

- BI Publisher has been configured to use Oracle Internet Directory (OID) LDAP.

  Note: For information on setting up OID LDAP for BI Publisher, see Integrating with LDAP, page 2-7.

- A dedicated user has been set up to access the BI Discoverer Web service from Oracle BI Publisher. This user must be assigned the "sr_users" role.

  Note: For more information on setting up users and roles, refer to "Security Role Mapping" in the *Oracle Application Server Containers for J2EE Security Guide 10g Release 2 (10.1.2)*.

- The Oracle database that holds the report data is registered in OID.

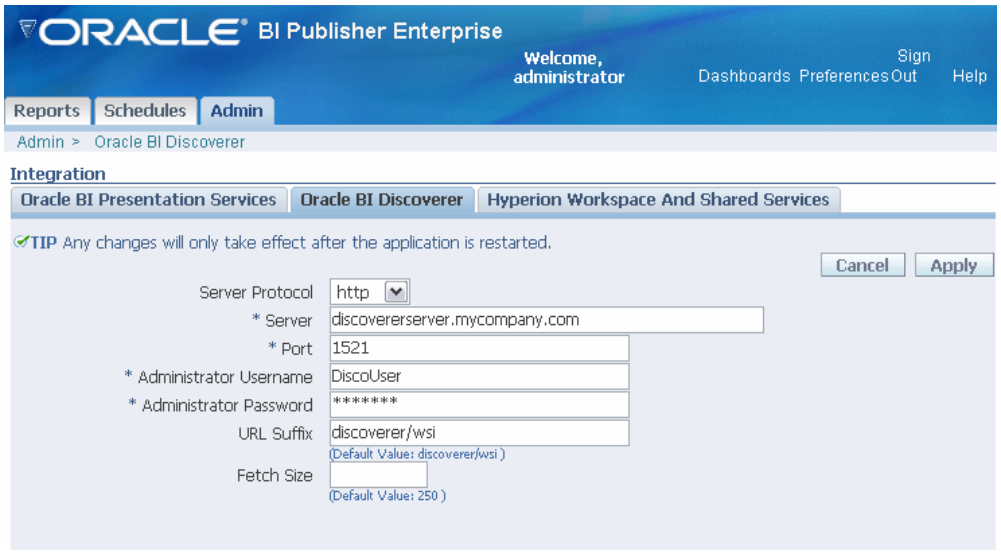
Limitations of the Integration with BI Discoverer

Following are limitations of the integration of BI Publisher with Discoverer:

- Only one BI Discoverer Web Service can be configured per Oracle BI Publisher server or cluster.
- An Oracle BI Publisher report can only contain one data set based on a BI Discoverer worksheet.
- The Discoverer Web service does not support Oracle Applications Single Sign-On (SSO), Oracle Applications ICX and non-SSO modes.

Configuring BI Publisher to Integrate with BI Discoverer

1. From the Admin page, under **Integration**, select **Oracle BI Discoverer**.
2. Enter the following fields:
   - Server Protocol - select HTTP or HTTPS.
   - Server - enter the server name on which you installed Oracle BI Discoverer (for example: myserver.company.com).
   - Port - enter the port used for the Oracle BI Discoverer server (for example: 7779)
   - Administrator Username and Administrator Password - enter the username and password of the dedicated user that you set up to access the BI Discoverer Web service from BI Publisher (see Prerequisites, page 5-3).
   - URL Suffix - enter `discoverer/wsi`.
   - Fetch size - enter the maximum number of rows to retrieve in one fetch from the database. Default is 250.



3. Restart the BI Publisher application.

Setting Up Integration with Oracle Enterprise Performance Management Workspace

This section will describe setting up the integration with Oracle Enterprise Performance Management Workspace. It includes the following topics:

- Integration with Oracle's Enterprise Performance Management Workspace Overview
- Prerequisites for Integrating with Workspace
- Configuring Oracle BI Publisher with Workspace

Integration with Oracle Enterprise Performance Management Workspace

Oracle Enterprise Performance Management Workspace (EPM Workspace) is a component of Oracle's Hyperion Foundation Services. It is the central Web interface for users to access all Performance Management content and tools. With its ease-of-use and flexibility, EPM Workspace provides users with a "windows-on-the-Web" experience.

Oracle BI Publisher can be integrated into the Oracle Enterprise Performance Management Workspace, Fusion Edition 11.1.1 release.

For more information about installing EPM Workspace, see *Oracle Hyperion Enterprise Performance Management System Installation and Configuration Guide*, Fusion Edition Release 11.1.1.

For tips on working with BI Publisher in the EPM Workspace environment, see Using

Oracle BI Publisher in Oracle Enterprise Performance Management Workspace., *Oracle Business Intelligence Publisher Report Designer's Guide*

Prerequisites for Integration with EPM Workspace

- Install Oracle BI Publisher version 10.1.3.4 or later.
- Install Oracle Enterprise Performance Management Workspace Fusion Edition 11.1.1 or later.
- Ensure the Workspace Registry Properties file is accessible by BI Publisher.

  BI Publisher uses the Registry Properties file (reg.properties) that is created in the Hyperion home directory to derive appropriate values for registration with EPM Workspace. This file must be located on the same computer as the BI Publisher server application.

  If you installed EPM Workspace on the same computer as BI Publisher, then when you ran the Oracle Hyperion Enterprise Performance Management (EPM) System Configurator, the reg. properties file was placed in the following location:

  <*Hyperion_Home*>\common\config\9.5.0.0\reg.properties

  If you did not install EPM Workspace and BI Publisher on the same computer, then you must copy the reg.properties file from the computer on which EPM Workspace is installed to the computer on which BI Publisher is installed, as follows:

  1. Create the required directory structure on the computer where BI Publisher is installed. The directory structure must mimic the structure of the Hyperion_Home directory where the reg.properties file is located. The directory structure must be:

     <Hyperion_Home>\common\config\9.5.0.0\

     For example, you can create the directory structure as follows:

     C:\Programs\Oracle\Hyperion\common\config\9.5.0.0\

  2. Copy the reg.properties file to the 9.5.0.0 directory.

     Using the preceding example, the resulting path will be:
     C:\Programs\Oracle\Hyperion\common\config\9.5.0.0\reg.properties

- If you installed Oracle EPM Shared Services and Oracle BI Publisher on different computers, you must update the hosts file located on the machine where you installed BI Publisher with the host name and IP address of the EPM Shared Services server.

  In Windows operating systems, the hosts file is typically located as follows:

  <*SystemRoot*>\system32\drivers\etc\hosts

  In Linux and Unix operating systems the hosts file is located as follows:

/etc/hosts

For example, if BI Publisher is located in domain publisher.us.mycompany.com and EPM Shared Services is located in domain hss.jp.mycompany.com, enter the Shared Services host name and IP address in the hosts file on the BI Publisher computer as follows:

```
147.132.42.18          hss.jp.mycompany.com
```

- Create a BI Publisher Local Superuser. The following procedure requires configuring BI Publisher to use the Hyperion Common Shared Services Security. Before performing any security updates, you must set up a BI Publisher Local Superuser to ensure access to BI Publisher regardless of your selected security configuration. See Defining a Local Superuser, page 2-2 for more information.

Configuring Oracle BI Publisher with EPM Workspace

To configure Oracle BI Publisher for use within EPM Workspace, perform the following tasks:

1. Register Oracle BI Publisher with Oracle's Hyperion Shared Services Registry.
2. Enable Hyperion Common Shared Services security in BI Publisher.
3. Configure the EPM Workspace Web server.
4. Provision BI Publisher users in EPM Workspace.

Register Oracle BI Publisher with EPM Shared Services and Enable Shared Services Security

Before you can use Oracle BI Publisher within Oracle EPM Workspace, you must register it with Oracle EPM Shared Services. After successful registration, BI Publisher is registered in EPM Workspace and corresponding Product, Application, Project, and Roles are seeded in Shared Services. Perform the following steps to register BI Publisher and enable Common Shared Services security:

1. Log in to Oracle BI Publisher with Administrator credentials.
2. Navigate to the Oracle BI Publisher Administration page by selecting the **Admin** tab
3. Under **Integration**, click the **Hyperion Workspace and Shared Services** link to launch the configuration page.
4. In the **Database Connection for Hyperion Workspace** region, enter the Hyperion home location where the reg.properties file is located. (Note that you must enter only the location of the Hyperion Home, not the full path to the reg.properties file.)

   For example: C:\Programs\Oracle\Hyperion

The reg.properties file must be located on the same computer as the BI Publisher server application. See Prerequisites for Integration with EPM Workspace, page 5-6 for more information.

5. Click **Load Properties**.

   This will populate the following fields based on the values in the reg.properties file that you pointed to in the previous step:

   - JDBC Connection String
   - Database Username
   - Database Driver Class

6. The fields in the Hyperion Workspace Registration region are defaulted from the servlet context. You can update these fields if you want to run BI Publisher in both SSL and non-SSL modes; or if you are using a load balancer, enter the server information for the load balancer.

   - BI Publisher Port
   - BI Publisher SSL Port
   - BI Publisher URL
   - BI Publisher SSL URL
   - EPM Workspace Application Server Type: select your application server type. Note that the list contains only the application servers that EPM Workspace supports. Valid values are:
     - Oracle Application Server 10g
     - IBM WebSphere Application Server V6.X
     - BEA WebLogic Server 9.2
     - Apache Tomcat 5.5x

7. In the **Hyperion Shared Services Registration** region, enter the following fields that describe your Shared Services installation:

   - HSS Server – specify the name of the computer where the Shared Services server is installed.
   - HSS Port – specify the Shared Services server port number on which the database listens.

- HSS Administrator Username – enter the name of the database user.

  Important: You must provision the HSS Administrator that you specify here with the BI Publisher Administrator role. Perform this step as part of the Provision Users in EPM Workspace, page 5-11 procedure.

- HSS Administrator Password – enter the password for the user you specified.

- Select **Use Hyperion CSS Security Model upon registration**. This will automatically change BI Publisher's security model to the Hyperion Common Shared Services security model upon restart. This will also be reflected on the BI Publisher Security Configuration page. Integration with EPM Workspace requires the use of the Hyperion Common Shared Services security model.

  Important: Before making any changes that impact security, ensure that you have set up a BI Publisher Local Superuser to ensure access to BI Publisher regardless of your selected security configuration. See Defining a Local Superuser, page 2-2 for more information.

  Note: If you do not select the checkbox here, you can manually change the security model from the BI Publisher Security Configuration page after successful registration and restart of the BI Publisher server application. If registration has not been completed, the Hyperion Common Shared Services selection will not be available from the BI Publisher Security Configuration page. Performing this step manually requires an additional restart of the BI Publisher server to make the changes to the Security Configuration page effective.

8. Click **Register**. If the registration process succeeds, you will receive a confirmation message, otherwise an error message will display.

9. Restart your BI Publisher server application for these changes to take effect.

Configure the EPM Workspace Web Server to Proxy BI Publisher Requests

Next modify the configuration of the EPM Workspace Web server so that requests for Oracle BI Publisher are routed through it.

To configure Oracle BI Publisher with EPM Workspace Web server, you can use EPM System Configurator to automatically configure them with the EPM Workspace Web server only if EPM Workspace and Oracle BI Publisher are deployed on a Web

application server of the same type (for example, when both EPM Workspace and Oracle BI Publisher Web application server are deployed on Oracle Application Server).

If you are running BI Publisher on a different type of Web server than the type of Web server running EPM Workspace, you must follow the steps under Manually Configuring the Web Server, page 5-10.

Using the EPM System Configurator to Configure the Web Server

To configure the Web server using the EPM System Configurator:

1. Launch the EPM System Configurator using one of the following methods:
   - From Microsoft Windows: From the Start menu, select Programs > Oracle EPM System > Foundation Services > EPM System Configurator.
   - Double-click configtool.bat from <*HYPERION_HOME*>/common/config/<*version_number*>
   - From a Windows command prompt, change to <*HYPERION_HOME* >/common/config/<*version_number*>, and then type:

     ```
     startconfigtool.bat -console
     ```

2. When you see "Select the products to configure", select Foundation, Workspace, and Web Server Configuration. Click **Next**.
3. On the Workspace Web Server plug-in pane, an entry for the BI Publisher server will display, showing the BI Publisher host, port, and context root. Select the box for the BI Publisher row as well as the boxes for all other products that need to be configured with the Web server whether you have previously configured them or not. When you have selected all products, click **Next**.
4. Click **Next** through the remaining screens, and then click **Finish**.

For information on using the EPM System Configurator, see the *Oracle Hyperion Enterprise Performance Management System Installation and Configuration Guide*.

Manually Configuring the Web Server

If BI Publisher is deployed on a different type of Web application server than the type of Web application server running EPM Workspace, back up the Web server's configuration files before running the EPM System Configurator to configure BI Publisher to work with EPM Workspace. After running the EPM System Configurator, manually restore the Web server's configuration files, then add the /xmlpserver context to the Web server's configuration files.

1. Back up the Web server configuration files. Navigate to

   <*HYPERION_HOME*>\common\httpServers\Apache\2.0.59\conf

   and back up the following files:

- HYSLTomcat-autogenerated.conf
- HYSLWorkers.properties
- httpd.conf

2. Use the EPM System Configurator to configure the Web server as described in Using the EPM System Configurator to Configure the Web Server, page 5-10.

3. Revert to the files you backed up in Step 1 by copying the backed up files back to <*HYPERION_HOME*>\common\httpServers\Apache\2.0.59\conf

4. Manually update the Web server's configuration file.

   For the Web server bundled with the EPM System, the configuration directives below will add the /xmlpserver context. For information on configuring other Web servers, see the *Oracle Hyperion Enterprise Performance Management System Manual Deployment Guide*.

   - Using a text editor, open httpd.conf.
   - Add the following directives, replacing <BIPUB_HOST> and <BIPUB_PORT> with the correct values:

```
#Change modules/mod_proxy.so as needed; e.g. libexec/mod_proxy on
UNIX
<IfModule !mod_proxy.c>
LoadModule proxy_module modules/mod_proxy.so
</IfModule>
<IfModule !mod_proxy_connect.c>
LoadModule proxy_connect_module modules/mod_proxy_connect.so
</IfModule>
<IfModule !mod_proxy_http.c>
LoadModule proxy_http_module modules/mod_proxy_http.so
</IfModule>
#ProxyRequests Off
# remove or comment out the ProxyRequests off directive
ProxyPass /xmlpserver http://<BIPUB_HOST>:<BIPUB_PORT>/xmlpserver
ProxyPassReverse /xmlpserver
http://<BIPUB_HOST>:<BIPUB_PORT>/xmlpserver
```

Provision Users in EPM Workspace

The final step is to provision users with the BI Publisher roles in Shared Services. Provisioning is the process of granting roles from EPM System applications to the users and groups that are available in the configured user directories. Provisioning is managed at the user or group level by Provisioning Managers or Shared Services Administrators. See *Oracle Hyperion Enterprise Performance Management System Security Administration Guide* for detailed information on how provisioning works.

Important: You must provision the HSS Administrator that you

specified in BI Publisher's Hyperion Workspace and Shared Services integration page with the BI Publisher Administrator role. Follow the procedure in this section to provision the HSS Administrator with the BI Publisher Administrator role.

To provision users or groups:

1. Launch the Shared Services Console using one of the following methods:
   - Using a browser, access the Shared Services Console URL.

     To launch Shared Services Console from a URL:

     Using a browser, access the following URL:

     `http://<server_name>:<port_number>/interop`

     In the URL, `<server_name>`indicates the name of the computer where the application server that hosts Shared Services is running and `<port_number>` indicates the server port that Shared Services is using; for example,

     `http://myserver:28080/interop`

     Note: Pop-up blockers may prevent User Management Console from opening.

   - On Windows, select Start, then All Programs, then Oracle EPM System, then Foundation Services, and then Shared Services Console.
   - Use an EPM System product interface.
2. On the Logon screen enter the same Administrator username and password that you entered in the BI Publisher Admin page for the Hyperion Shared Services Registration.
3. Locate the Shared Services node on which BI Publisher is registered.
4. Navigate to **User Directories** then **Native Directory** then **Groups**.
5. Query for the Group to which you want to grant BI Publisher roles; or create a new group (by right-clicking "Groups" in the tree).
6. Select and right-click the **Group**.
7. From the right-click menu, select **Provision**.
8. From the **Available Roles** list, expand the heading for BI Publisher, then BI Publisher Enterprise to see the list of available BI Publisher roles.

9. Use the shuttle buttons to move the desired roles from the **Available Roles** list to the **Selected Roles** list.

10. Save the changes. A Provision Summary will display to confirm your changes.

The next time you connect to BI Publisher as a user with BI Publisher Administrator privileges, you will see this newly created EPM Workspace group as a Role in BI Publisher's Security Center, Roles and Permissions tab. Use the **Roles and Permissions** tab to assign BI Publisher Folders and Data Sources to this group.

Note that BI Publisher fetches only the Shared Services Groups that have been assigned seeded BI Publisher roles. If the group does not have any seeded BI Publisher roles assigned to it, it is not displayed in BI Publisher's **Roles and Permissions** tab.

Once you have set up BI Publisher to use the Hyperion Common Shared Services Security model you will no longer be able to create users or roles in the BI Publisher Security Center. All users and roles must be created using Hyperion Shared Services. Use the BI Publisher Security Center to assign BI Publisher folders and data sources to the roles you created in Shared Services.

For more information on assigning folders and data sources to roles, see Add a Folder to a Role, page 2-6 and Add a Data Source to a Role, page 2-6.

Setting Up Integration with Oracle Smart Space Client

If you have integrated Oracle BI Publisher with Oracle Smart Space you can use this page to set up a link to the Oracle Smart Space client download page. This will enable all BI Publisher users to easily access the Oracle Smart Space client download page from their BI Publisher session.

For more information on setting up the integration between Oracle BI Publisher and Oracle Smart Space, see the *Oracle Smart Space, Fusion Edition User's Guide*.

To Set Up the Oracle Smart Space Client Download Page Link:

1. From the Admin page, under **Integration**, select **Oracle Smart Space Client**.

2. Enter the following fields:

   - **Server** - your Oracle Smart Space server name including domain.

     Example: mySmartSpaceserver.mycompany.com

   - **Port** - the port on which your Smart Space server listens.

3. Restart your BI Publisher application.

After you have successfully restarted BI Publisher, the **Oracle Smart Space Client** link will display in the header region of the BI Publisher user interface. All users can now

use this link to launch the Oracle Smart Space client download page where they can download the Oracle Smart Space client to install on their individual computers.

6

Setting System Maintenance Options

Setting System Maintenance Options

Under System Maintenance, you can perform the following administration tasks:

- Define Your Report Repository, page 6-1
- Set Server Configuration Options, page 6-3
- Configure the Scheduler, page 6-4
- Refresh Metadata

Refresh Metadata

If you copy reports as files or folders directly to the file system or XML database repository, you must refresh the metadata to make these reports available via the user interface.

Defining Your Report Repository

Defining Your Report Repository

The report repository can be set up in either the file system or the database.

Defining a File-Based Repository

BI Publisher will set a default file-based repository. The default is `${oracle.home}/xdo/repository`

If the "oracle.home" Java system property is set to a valid path, BI Publisher replaces ${oracle.home} with that value.

When BI Publisher is deployed on OC4J, ${oracle.home} is automatically set to a

directory path where OC4J is installed. For example, in an OC4J deployment in which the OC4J home is `C:\oc4j`, the BI Publisher repository is automatically set to:

```
C:\oc4j\xdo\repository
```

If you wish to choose another location for your repository, do so as follows:

1. Under Report Repository, select **File System** as the **Repository Type**.
2. Enter the absolute **Path**.

   For example: `/home/repository/xmlp`
3. Apply your changes and restart your BI Publisher application.

Defining a Database-Based Repository

> **Important:** If you stop and start or restart the database that contains your BI Publisher repository, you must restart you BI Publisher Enterprise server.

1. Under Report Repository, select **XML DB** as the **Repository Type**.
2. Enter the absolute **Path**. For example: `/public/Reports`
3. Select the **Connection Type**: JDBC or JNDI.

   > **Important:** Connection type JDBC is **not** recommended for the repository.

   - If you select JNDI, enter the JNDI connection pool **Name** (for example: jdbc/Oracle10gRepository).
   - If you select JDBC (not recommended), enter the following:
     - URL

       Example: `jdbc:oracle:thin:@rpts.mycompany.com:1525:ora10g`
     - Username
     - Password
     - Database Driver Class

       Example: `oracle.jdbc.driver.oracleDriver`

Setting Server Configuration Options

Use the Server Configuration tab to define

- General properties for the server
- Caching specifications

Defining General Properties

- **Debug Level** - Controls the amount of debug information generated by the system. It set to `Exception` (the default setting), only error information is generated. If set to `Debug`, all system output is generated.
- **Report Viewer Height** - sets the size of the report viewing frame in your browser. Enter a value in pixels. The default is 600.
- **Report Scalable Threshold** - sets the threshold at which data is cached on the disk. When the data volume is very large, caching the data will save memory, but will result in slower processing. Enter a value in kilobytes. The default is 10000000 (10 megabytes).
- **Output Formats** - select the formats that you want displayed to the user by default for every RTF template-based report. This server-level setting is overridden by the Output Format types selected in the report definition. See Adding Layouts to the Report Definition, *Oracle Business Intelligence Publisher Report Designer's Guide*.

Setting Cache Specifications

Set the following properties to configure the BI Publisher cache:

- **Cache Expiration** - Enter the expiration period for the dataset cache in minutes. The default is 30.

  For datasets returned by a SQL query, HTTP, or Oracle BI Answers, you have the option of caching the dataset returned by the query. The returned dataset will remain in cache for the period specified by this property. For more information about setting this property, see Defining the Data Model, *Oracle Business Intelligence Publisher Report Designer's Guide*.
- **Cache Size Limit** - Enter the maximum number of datasets to maintain in the cache. The default is 1000.
- **Maximum Cached Reports** - Enter the maximum number of reports to maintain in the cache. The default is 50.

Installing the Scheduler Schema

BI Publisher uses Quartz, an open-source job-scheduling system. To set up the BI Publisher scheduler you must install the scheduler tables to your database. To set up the scheduler database tables, use the BI Publisher Scheduler Configuration tab to define a JDBC connection to your database, then BI Publisher will install the tables to your database.

BI Publisher includes the Hyperion-branded DataDirect Connect for JDBC drivers that you can use to set up connection to your BI Publisher scheduler database. These drivers can be used as an alternative to the native JDBC drivers provided by your database vendor. When you choose a database for which a DataDirect driver is available, BI Publisher automatically enters the database driver class information in the setup screen for you. There is no additional setup required for the driver files.

If you choose to use the native driver for any database other than Oracle or if you choose to use a DataDirect driver not provided by BI Publisher, you must download, install, and configure the driver manually.

> **Important:** If you stop and start or restart the database that contains your BI Publisher Scheduler tables, you must restart you BI Publisher Enterprise server.

Recommendations for Using DataDirect Connect or Native Database Drivers

DataDirect Connect for JDBC drivers are provided for the following databases:

- IBM DB2 v8.1, v9.1
- Microsoft SQL Server 2000, 2005
- Sybase Adaptive Server Enterprise
- Oracle 9*i*, Oracle 10*g*, Oracle 11*g*,

The following table displays the driver recommendations for the supported scheduler databases:

Database	Native JDBC Driver	DataDirect JDBC Driver
Oracle 9*i*, Oracle 10*g*, Oracle 11*g*	Recommended	Supported
IBM DB2 v8.1, v9.1	Supported	Recommended

Database	Native JDBC Driver	DataDirect JDBC Driver
Microsoft SQL Server 2000, 2005	Supported	Recommended
Sybase Adaptive Server Enterprise	Supported	Recommended
MySQL 4.1.10a-NT, 5.0	Supported	Not Supplied

Set Up a User on Your Scheduler Database

To set up the connection to your scheduler database, you must ensure you have created a user on your selected database. BI Publisher will use this user to connect to the database. Depending on your database type, this user may require specific privileges. These are detailed in the database-specific sections below.

Entering Connection Information for Your Scheduler Database and Installing the Schema

Following are the general steps for setting up the Scheduler database. Please also refer to the subsequent section that is specific to your database.

1. Log in to BI Publisher with Administrator credentials and select the **Admin** tab.
2. Under **System Maintenance**, select **Scheduler Configuration**.
3. Enter the following fields for the Database Connection:
   - **Database Type** - select your database from the list. After you make a selection, the Database Driver Class field will automatically update with the recommended driver class.
   - **Connection String** - enter the connection string for your selected database. Sample strings are provided in the database-specific sections that follow.
   - **Username** and **Password** - enter the scheduler user you set up for your database. The user must have permissions to connect to the database and create tables. Other permissions may be required depending on your database type. Please see the appropriate database-specific section below.
   - **Database Driver Class** - when you select the database type this field is automatically updated with the recommended driver. If you wish to use another driver, specify it in this field.

Note: The Oracle database drivers and the DataDirect drivers are installed with BI Publisher and no further setup is required. Note that for other databases, even though the recommended native drivers are automatically populated in this field, additional setup is required to make the drivers available to BI Publisher.

4. Click **Test Connection** to ensure that BI Publisher can connect to the database. If the connection fails, ensure that you have entered the fields as shown and set up your database appropriately.

5. Click **Install Schema** to install the BI Publisher scheduler schema to your database.

Entering Connection Information for an Oracle Database

Prerequisite: Ensure that the database user you enter has "connect" or "create session" and "create table" privileges and that the user has been assigned a quota (otherwise the quota will be 0).

For example, the following sample creates the user "bipubsched":

```
SQL> CREATE USER bipubsched
   2  IDENTIFIED BY welcome
   3  DEFAULT TABLESPACE USERS
   4  TEMPORARY TABLESPACE TEMP
   5  QUOTA 20G ON USERS
   6  QUOTA 1M ON TEMP;

User created.

 SQL> GRANT CREATE SESSION TO bipuser; -- or "GRANT CONNECT TO bipuser;"

 Grant succeeded.

 SQL> grant create table to bipublisher;

 Grant succeeded.
```

Enter the following to use the Oracle native driver to connect to your Oracle database:

Field	Entry
Database Type:	Select Oracle 11*g*, Oracle 10*g*, or Oracle 9*i* from the list.

Field	Entry

Connection String:	Enter the following connection string parameters:
	`jdbc:oracle:thin:@<hostname>:<port>:<oracle SID>`
	For example:
	`jdbc:oracle:thin:@mydatabaseserver.com:1521:bipscheduler`

Database Driver Class:	oracle.jdbc.driver.OracleDriver

Entering Connection Information for IBM DB2

Prerequisite: Ensure that the user that you enter to configure the scheduler has been set up with a 32 K page size tablespace. If not, create the table and assign it to the user. The user must also have "Connect to database" and "Create tables" privileges.

Enter the following to use the DataDirect driver to connect to an IBM DB2 v8 or IBM DB2 v9 database:

Field	Entry
Database Type:	Select IBM DB2 v9 or IBM DB2 v8 from the list.

Connection String:	Enter the following connection string parameters:
	`jdbc:hyperion:db2://<hostname>:<port>;DatabaseName=<DATABASENAME>`
	For example:
	`jdbc:hyperion:db2://mydatabaseserver.com:1433;DatabaseName=bipscheduler`

Database Driver Class:	hyperion.jdbc.db2.DB2Driver

Entering Connection Information for Microsoft SQL Server

Prerequisite: Ensure that your Microsoft SQL Server is set up with mixed mode authentication. Also ensure that the user you enter to configure the scheduler has the "db_owner" role.

Enter the following to use the DataDirect driver to connect to a Microsoft SQL Server 2000 or 2005 database:

Field	Entry
Database Type:	Select Microsoft SQL Server 2000 or Microsoft SQL Server 2005 from the list.

Connection String:	Enter the following connection string parameters:
	`jdbc:hyperion:sqlserver://<hostname>:<port>;DatabaseName=<DATABASENAME>`
	For example:
	`jdbc:hyperion:sqlserver://mydatabaseserver.com:1433;DatabaseName=bipscheduler`

Database Driver Class:	hyperion.jdbc.sqlserver.SQLServerDriver

Entering Connection Information for a Sybase Adaptive Server Enterprise Database

Prerequisite: Ensure that you set the "ddl in tran" mode to true in the database. Consult the Sybase documentation or contact your database administrator for instruction how to enable this option.

Enter the following to use the DataDirect driver to connect to your Sybase Adaptive Server Enterprise database:

Field	Entry
Database Type:	Select Sybase Adaptive Server Enterprise from the list.

Connection String:	Enter the following connection string parameters:
	`jdbc:hyperion:sybase://<hostname>:<port>;DatabaseName=<DATABASENAME>`
	For example:
	`jdbc:hyperion:sybase://mydatabaseserver.com:4100;DatabaseName=bipscheduler`

Database Driver Class:	hyperion.jdbc.sybase.SybaseDriver

Entering Connection Information for a MySQL Database

Prerequisite: You must download the MySQL JDBC driver and register it with the application server where BI Publisher is running.

The MySQL JDBC driver can be downloaded from:
http://www.mysql.com/products/connector/j/

See your application server documentation for information on registering the driver.

Enter the following if your scheduler database is MySQL 4.1 or MySQL 5.0

Field	Entry
Database Type:	Select MySQL 4.1 or MySQL 5.0 from the list.

Connection String:	Enter the following connection string parameters:
	`jdbc:mysql://<hostname>:<port>/<DATABASENAME>`
	For example:
	`jdbc:mysql://mydatabaseserver.com:4100/bipscheduler`

Database Driver Class:	com.mysql.jdbc.driver

7

Setting Up Delivery Options

Introduction

Use the Delivery Options page to set up all your delivery channels that you wish to use with your BI Publisher application. This chapter includes the following procedures:

Setting Up Delivery Options

Set Delivery Configuration Options:

1. From the Admin page select **Delivery Configuration**.
2. Enter the following properties:
    - **SSL Certificate File** - if SSL is enabled for your installation, you can leave this field empty if you wish to use the default certificates built-in with BI Publisher. SSL will work with the default certificate if the server uses the certificate signed by a trusted certificate authority such as Verisign. This field is mandatory only if the user uses the SSL with a self-signed certificate. The self-signed certificate means the certificate is signed by a non-trusted certificate authority (usually the

user).

- **Email From Address** - enter the From address that you want to appear on email report deliveries from the BI Publisher server. The default value is `bipublisher-report@oracle.com`.

- **Delivery Notification Email From Address** - enter the From address that you want to appear on notifications delivered from the BI Publisher server. The default value is `bipublisher-notification@oracle.com`.

- **Notification Subject** - enter the subject line you want to appear on the email for each of the following delivery types: Success, Warning, and Failure.

Set Up Print or Fax Server:

Printing is only supported through Internet Printing Protocol (IPP). You must set up CUPS or Windows Print Server for IPP. See Print Server Setup, page 9-1.

About Printing PDF

PDF is a popular output format for business reports and is printable from viewer software such as Adobe Reader. However, some reports require printing directly from the report server. For example, paychecks and invoices are usually printed as scheduled batch jobs. Some newer printers with Postscript Level 3 compliant Raster Image Processing can natively support PDF documents, but there are still many printers in business use that only support Postscript Level 2 that cannot print PDF documents directly.

To print PDF documents directly from the BI Publisher server you have the following options:

- Select BI Publisher's PDF to Postscript filter. This can be enabled for Unix or Windows print servers.

- Configure a custom, or third-party filter.

After completing all other required fields for the print server, you will be able to schedule reports to print directly from the BI Publisher server to any printer in your system that supports PostScript Level 2.

To set up your printer or fax:

1. From the Admin page select **Printer** or **Fax**. Select **Add Server**.

2. Enter the required fields **Server Name** and **URI** for the new server.

   Important: You must enter a unique name for each server regardless of the type (printer, fax, email, WebDAV, or FTP).

3. Enter a **Filter** (optional).

   A filter enables you to call a conversion utility to convert PDF to Postscript or PDF to TIFF (for fax) for Windows-based print servers.

   BI Publisher includes a PDF to Postscript filter. This filter converts PDF to Postscript Level 2. Choose **PDF to Postscript** from the list if you wish to use BI Publisher's predefined filter. See About Printing PDF, page 7-2 for more information.

   About Custom Filters

   To specify a custom filter, pass the native OS command string with the 2 placeholders for the input and output filename, {infile} and {outfile}.

   This is useful especially if you are trying to call IPP printers directly or IPP printers on Microsoft Internet Information Service (IIS). Unlike CUPS, those print servers do not translate the print file to a format the printer can understand, therefore only limited document formats are supported. With the filter functionality, you can call any of the native OS commands to transform the document to the format that the target printer can understand.

   For example, to transform a PDF document to a Postscript format, enter the following PDF to PS command in the **Filter** field:

   ```
   pdftops {infile} {outfile}
   ```

   To call an HP LaserJet printer setup on a Microsoft IIS from Linux, you can set Ghostscript as a filter to transform the PDF document into the format that the HP LaserJet can understand. To do this, enter the following Ghostscript command in the **Filter** field:

   ```
   gs -q -dNOPAUSE -dBATCH -sDEVICE=laserjet -sOutputFile={outfile}
   {infile}
   ```

   For fax servers, you can use the filter to transform the file to Tag Image File Format (TIFF).

4. Optionally enter the following fields if appropriate:

   - Security fields: Username and Password, Authentication Type (None, Basic, Digest) and Encryption Type (None, SSL).

   - Proxy Server fields: Host, Port, User Name, Password, Authentication Type (None, Basic, Digest)

Set Up WebDAV Server:

1. From the Admin page select **WebDAV**. This will show the list of servers already added. Select **Add Server**.

2. Enter the **Name** and **Host** for the new server.

Important: You must enter a unique name for each server regardless of the type (printer, fax, email, WebDAV, or FTP).

3. Optionally enter the following fields if appropriate:
   - General fields: Port
   - Security fields: Authentication Type (None, Basic, Digest) and Encryption Type (None, SSL).
   - Proxy Server fields: Host, Port, User Name, Password, Authentication Type (None, Basic, Digest)

Set Up an HTTP Server:

You can register an application URL or postprocess HTTP URL as an HTTP server to send a notification request to after your report has completed. The HTTP notification sent by BI Publisher appends the job ID, report URL, and job status to the URL provided in this screen.

For more information about setting up an HTTP notification to integrate with a third-party application, see Setting Up an After Report Trigger, page 13-1. For information on adding the trigger to a scheduled report, see Scheduling a Report, *Oracle Business Intelligence Publisher Report Designer's Guide*.

1. From the Admin page select HTTP. This will show the list of servers already added. Select **Add Server**.
2. Enter a name for your server, and enter the URL. When your report finishes processing, BI Publisher will append the job ID, report URL, and job status to the URL provided here.
3. Enter the Security information, if required. If your server is password protected, enter the Username and Password. Select the Authentication Type: None, Basic, or Digest; and Encryption Type: None or SSL.
4. If the notification is to be sent through a proxy server, enter the fully qualified Host name, the Port, the Username and Password, and Authentication type of the proxy server.

Set Up Email or FTP Server:

1. From the Admin page select Email or FTP. This will show the list of servers already added. Select **Add Server**.
2. Enter the **Name** and **Host** for the Email or FTP server.

Important: You must enter a unique name for each server regardless of the type (printer, fax, email, WebDAV, or FTP).

3. Optionally enter the following fields if appropriate:
   - General fields: Port
   - Security fields: Username and Password.

Set Up Common Unix Printing System (CUPS) Server:

1. From the Admin page select CUPS. This will show the list of servers already added. Select **Add Server**.
2. Enter the **Server Name** and **Host** and **Port** for the CUPS server.

   For more information see Setting Up Cups, page 9-1.

8

Setting Runtime Properties

Setting Runtime Properties

The Runtime Configuration page enables you to set runtime properties at the server level. You can also set properties at the report level. If conflicting values are set for a property at each level, the report level will take precedence.

To set a property at the report level, select the report, and then select the **Configure** link. This will launch the Runtime Configuration page, displaying a column to enable update to the properties for the report and a column that displays the read-only values set for the server.

> **Note:** In versions prior to 10.1.3.2 the Runtime Configuration properties administered through this page were set in a configuration file. This file is still used as a fallback if values are not set through this interface. However, please note that the file is not updated when you update the Runtime Configuration Properties page. For details about the file, see Configuration File Reference, page A-1.

Bursting Properties

If you are running BI Publisher on a multiprocessor machine or even a machine with a dual core single processor, you may be able to achieve even higher bursting throughput using the multithreading functionality for bursting.

To enable multithreading for bursting, set "Enable multithreading" to true and set "Thread count" to a number greater than one up to the number of processors or cores present on the machine.

Note that if the report delivery channel is File System, there will not be any considerable performance gain using multithreading. For delivery destinations other then file delivery, you should notice the performance gain.

Due to other processes that might be running on your system you may need to

empirically determine what is the optimal setting for "Thread count." Try a series of tests by varying the setting "Thread count" to see what is optimal for your environment.

> **Important:** Leave these settings at the defaults if your system does not have multicore processors or more than one processor. Setting "Enable multithreading" to True and "Thread count" to a number greater than the number of cores on the machine will lead to higher CPU usage without any gain in performance.

Property Name	Internal Name	Default Value	Description
Enable multithreading	`bursting-multithreading-on`	false	Set to "true" to enable multithreading during bursting. This property is for use only when running BI Publisher on a machine with multiprocessors or dual core single processors.
Thread count	`bursting-thread-count`	2	If Enable multithreading is set to "true", enter the number of concurrent threads you wish to have active during bursting. Do not exceed the number of processors or cores on the machine. Note that setting this value to the maximum may not necessarily achieve the best performance for your system.

PDF Output Properties

The following properties are available for PDF output:

Property Name	Internal Name	Default Value	Description
Compress PDF output	`pdf-compression`	True	Specify "True" or "False" to control compression of the output PDF file.
Hide PDF viewer's menu bars	`pdf-hide-menubar`	False	Specify "True" to hide the viewer application's menu bar when the document is active. The menu bar option is only effective when using the Export button, which displays the output in a standalone Acrobat Reader application outside of the browser.

Property Name	Internal Name	Default Value	Description
Hide PDF viewer's tool bars	`pdf-hide-toolbar`	False	Specify "True" to hide the viewer application's toolbar when the document is active.
Replace smart quotes	`pdf-replace-smartquotes`	True	Set to "False" if you do not want curly quotes replaced with straight quotes in your PDF output.

PDF Security

Use the following properties to control the security settings for your output PDF documents:

Property Name	Internal Name	Default Value	Description
Enable PDF Security	`pdf-security`	False	If you specify "True," the output PDF file will be encrypted. You must also specify the following properties: • Open document password • Modify permissions password • Encryption Level
Open document password	`pdf-open-password`	N/A	This password will be required for opening the document. It will enable users to open the document only. This property is enabled only when "Enable PDF Security" is set to "True".
Modify permissions password	`pdf-permissions-password`	N/A	This password enables users to override the security setting. This property is effective only when "Enable PDF Security" is set to "True".

Property Name	Internal Name	Default Value	Description
Encryption level	`pdf-encryption-level`	0 - low	Specify the encryption level for the output PDF file. The possible values are: • 0: Low (40-bit RC4, Acrobat 3.0 or later) • 1: High (128-bit RC4, Acrobat 5.0 or later) This property is effective only when "Enable PDF Security" is set to "True". When Encryption level is set to 0, you can also set the following properties: • Disable printing • Disable document modification • Disable context copying, extraction, and accessibility • Disable adding or changing comments and form fields When Encryption level is set to 1, the following properties are available: • Enable text access for screen readers • Enable copying of text, images, and other content • Allowed change level • Allowed printing level
Disable document modification	`pdf-no-changing-the-document`	False	Permission available when "Encryption level" is set to 0. When set to "True", the PDF file cannot be edited.
Disable printing	`pdf-no-printing`	False	Permission available when "Encryption level" is set to 0. When set to "True", printing is disabled for the PDF file.

Property Name	Internal Name	Default Value	Description
Disable adding or changing comments and form fields	`pdf-no-accff`	False	Permission available when "Encryption level" is set to 0. When set to "True", the ability to add or change comments and form fields is disabled.
Disable context copying, extraction, and accessibility	`pdf-no-cceda`	False	Permission available when "Encryption level" is set to 0. When set to "True", the context copying, extraction, and accessibility features are disabled.
Enable text access for screen readers	`pdf-enable-accessibility`	True	Permission available when "Encryption level" is set to 1. When set to "True", text access for screen reader devices is enabled.
Enable copying of text, images, and other content	`pdf-enable-copying`	False	Permission available when "Encryption level" is set to 1. When set to "True", copying of text, images, and other content is enabled.
Allowed change level	`pdf-changes-allowed`	0	Permission available when "Encryption level" is set to 1. Valid Values are: • 0: none • 1: Allows inserting, deleting, and rotating pages • 2: Allows filling in form fields and signing • 3: Allows commenting, filling in form fields, and signing • 4: Allows all changes except extracting pages

Property Name	Internal Name	Default Value	Description
Allowed printing level	pdf-printing-allowed	0	Permission available when "Encryption level" is set to 1. Valid values are: • 0: None • 1: Low resolution (150 dpi) • 2: High resolution

PDF Digital Signature Properties

The following properties should only be set at the report level to enable digital signature for a report and to define the placement of the signature in the output PDF document. For more information on how to enable digital signature for your output PDF documents, see Implementing a Digital Signature, page 2-29.

Note that to implement digital signature for a report based on a PDF layout template or an RTF layout template, you must set the property **Enable Digital Signature** to "True" for the report.

You also must set the appropriate properties to place the digital signature in the desired location on your output report. Your choices for placement of the digital signature depend on the template type. The choices are as follows:

- (PDF only) Place the digital signature in a specific field by setting the **Existing signature field name** property.
- (RTF and PDF) Place the digital signature in a general location of the page (top left, top center, or top right) by setting the **Signature field location** property.
- (RTF and PDF) Place the digital signature in a specific location designated by x and y coordinates by setting the **Signature field x coordinate** and **Signature field y coordinate** properties.

  If you choose this option, you can also set **Signature field width** and **Signature field height** to define the size of the field in your document.

Note that if you enable digital signature, but do not set any location properties, the digital signature placement will default to the top left of the document.

Property Name	Internal Name	Default Value	Description
Enable Digital Signature	`signature-enable`	False	Set this to "True" to enable digital signature for the report.
Existing signature field name	`signature-field-name`	N/A	This property applies to PDF layout templates only. If your report is based on a PDF template, you can enter a field from the PDF template in which to place the digital signature. For more information on defining a field for the signature in a PDF template, see Adding or Designating a Field for a Digital Signature, *Oracle Business Intelligence Publisher Report Designer's Guide*.
Signature field location	`signature-field-location`	top-left	This property can apply to RTF or PDF layout templates. This property provides a list containing the following values: Top Left, Top Center, Top Right. Choose one of these general locations and BI Publisher will insert the digital signature to the output document, sized and positioned appropriately. If you choose to set this property, do not enter X and Y coordinates or width and height properties.
Signature field X coordinate	`signature-field-pos-x`	0	This property can apply to RTF or PDF layout templates. Using the left edge of the document as the zero point of the X axis, enter the position in points that you want the digital signature to be placed from the left. For example, if you want the digital signature to be placed horizontally in the middle of an 8.5 inch by 11 inch document (that is, 612 points in width and 792 points in height), enter 306.

Property Name	Internal Name	Default Value	Description
Signature field Y coordinate	`signature-field-pos-y`	0	This property can apply to RTF or PDF layout templates. Using the bottom edge of the document as the zero point of the Y axis, enter the position in points that you want the digital signature to be placed from the bottom. For example, if you want the digital signature to be placed vertically in the middle of an 8.5 inch by 11 inch document (that is, 612 points in width and 792 points in height), enter 396.
Signature field width	`signature-field-width`	0	Enter in points (72 points equal one inch) the desired width of the inserted digital signature field. This applies only if you are also setting the properties **Signature field x coordinate** and **Signature field Y coordinate**.
Signature field height	`signature-field-height`	0	Enter in points (72 points equal one inch) the desired height of the inserted digital signature field. This applies only if you are also setting the properties **Signature field x coordinate** and **Signature field Y coordinate**.

RTF Output

The following properties can be set to govern RTF output files:

Property Name	Internal Name	Default Value	Description
Enable change tracking	`rtf-track-changes`	False	Set to "True" to enable change tracking in the output RTF document.
Protect document for tracked changes	`rtf-protect-document-for-tracked-changes`	False	Set to "True" to protect the document for tracked changes.

Property Name	Internal Name	Default Value	Description
Default font	`rtf-output-default-font`	Arial:12	Use this property to define the font style and size in RTF output when no other font has been defined. This is particularly useful to control the sizing of empty table cells in generated reports. Enter the font name and size in the following format <FontName>:<size> for example: Arial:12. Note that the font you choose must be available to the BI Publisher processing engine at runtime. See Defining Font Mappings, page 8-15 for information on installing fonts for the BI Publisher server and also for the list of fonts predefined for BI Publisher.

HTML Output

The following properties can be set to govern HTML output files:

Property Name	Internal Name	Default Value	Description
Show header	`html-show-header`	True	Set to "False" to suppress the template header in HTML output.
Show footer	`html-show-footer`	True	Set to "False" to suppress the template footer in HTML output.
Replace smart quotes	`html-replace-smartquotes`	True	Set to "False" if you do not want curly quotes replaced with straight quotes in your HTML output.
Character set	`html-output-charset`	UTF-8	Specify the output HTML character set.
Make HTML output accessible	`make-accessible`	False	Specify true if you want to make the HTML output accessible.

Property Name	Internal Name	Default Value	Description
Use percentage width for table columns	`html-output-width-in-percentage`	True	Set this property to True to render table columns according to a percentage value of the total width of the table rather than as a value in points. This property is especially useful if your browser renders tables with extremely wide columns. Setting this property to True will improve readability of the tables.

FO Processing Properties

The following properties can be set to govern FO processing:

Property Name	Internal Name	Default Value	Description
Use BI Publisher's XSLT processor	`xslt-xdoparser`	True	Controls BI Publisher's parser usage. If set to False, XSLT will not be parsed.
Enable scalable feature of XSLT processor	`xslt-scalable`	False	Controls the scalable feature of the XDO parser. The property "Use BI Publisher's XSLT processor" must be set to "True" for this property to be effective.
Enable XSLT runtime optimization	xslt-runtime-optimization	True	When set to "True", the overall performance of the FO processor is increased and the size of the temporary FO files generated in the temp directory is significantly decreased. Note that for small reports (for example 1-2 pages) the increase in performance is not as marked. To further enhance performance when you set this property to True, it is recommended that you set the property **Extract attribute sets** to "False". See RTF Template Properties, page 8-12.

Property Name	Internal Name	Default Value	Description
Pages cached during processing	`system-cache-page-size`	50	This property is enabled only when you have specified a Temporary Directory (under General properties). During table of contents generation, the FO Processor caches the pages until the number of pages exceeds the value specified for this property. It then writes the pages to a file in the Temporary Directory.
Bidi language digit substitution type	`digit-substitution`	None	Valid values are "None" and "National". When set to "None", Eastern European numbers will be used. When set to "National", Hindi format (Arabic-Indic digits) will be used. This setting is effective only when the locale is Arabic, otherwise it is ignored.
Disable variable header support	`fo-prevent-variable-header`	False	If "True", prevents variable header support. Variable header support automatically extends the size of the header to accommodate the contents.
Add prefix to IDs when merging FO	`fo-merge-conflict-resolution`	False	When merging multiple XSL-FO inputs, the FO Processor automatically adds random prefixes to resolve conflicting IDs. Setting this property to "True" disables this feature.
Enable multithreading	`fo-multi-threads`	False	If you have a multiprocessor machine or a machine with a dual-core single processor, you may be able to achieve faster document generation by setting this option to True. See
Disable external references	`xdk-secure-io-mode`	True	A "True" setting (default) disallows the importing of secondary files such as subtemplates or other XML documents during XSL processing and XML parsing. This increases the security of your system. Set this to "False" if your report or template calls external files. See Notes on Enabling Multithreading, page 8-12.

Property Name	Internal Name	Default Value	Description
FO Parsing Buffer Size	`fo-chunk-size`	1000000	Sets the size of the buffer for the FO Processor. When the buffer is full, the elements from the buffer will be rendered in the report. Reports with large tables or crosstabs that require complex formatting and calculations may require a larger buffer to properly render those objects in the report. Increase the size of the buffer at the report level for these reports. Note that increasing this value will affect the memory consumption of your system.

Notes on "Enable Multithreading"

The amount of performance gain seen by enabling this setting will depend on how much the current system resources are utilized. On a system that has numerous users running and relatively high CPU utilizations, you will likely only see minor improvements after setting "Enable multithreading" to True. If the system is used by only a few users, or reports are scheduled sequentially one at a time, or the number of CPUs is more than the number of concurrent reports, then turning on multiple threads will speed up report generation.

Note that memory utilization is likely to increase once "Enable multithreading" is set to True.

> **Important:** If you are running BI Publisher on a single-core, one processor machine, leave these multithreading configuration settings at the default value of False.

RTF Template Properties

The following properties can be set to govern RTF templates:

Property Name	Internal Name	Default Value	Description
Extract attribute sets	`rtf-extract-attribute-sets`	Auto	The RTF processor will automatically extract attribute sets within the generated XSL-FO. The extracted sets are placed in an extra FO block, which can be referenced. This improves processing performance and reduces file size. Valid values are: • Enable - extract attribute sets for all templates and subtemplates • Auto - extract attribute sets for templates, but not subtemplates • Disable - do not extract attribute sets
Enable XPath rewriting	`rtf-rewrite-path`	True	When converting an RTF template to XSL-FO, the RTF processor will automatically rewrite the XML tag names to represent the full XPath notations. Set this property to "False" to disable this feature.
Characters used for checkbox	`rtf-checkbox-glyph`	Default value: Albany WT J;9746;9747/A	The BI Publisher default PDF output font does not include a glyph to represent a checkbox. If your template contains a checkbox, use this property to define a Unicode font for the representation of checkboxes in your PDF output. You must define the Unicode font number for the "checked" state and the Unicode font number for the "unchecked" state using the following syntax: `fontname;<` *`unicode font number for true value's glyph >;<unicode font number for false value's glyph>`* Example: Albany WT J;9746;9747/A Note that the font that you specify must be made available to BI Publisher at runtime.

PDF Form Template Properties

The following properties can be set to govern PDF templates:

Property Name	Internal Name	Default Value	Description
Remove PDF fields from output	`remove-pdf-fields`	False	Specify "true" to remove PDF fields from the output. When PDF fields are removed, data entered in the fields cannot be extracted. For more information, see Setting Fields as Updateable or Read Only, *Oracle Business Intelligence Publisher Report Designer's Guide*.
Set all fields as read only in output	`all-field-readonly`	True	By default, BI Publisher sets all fields in the output PDF of a PDF template to be read only. If you want to set all fields to be updateable, set this property to "false". For more information, see Setting Fields as Updateable or Read Only, *Oracle Business Intelligence Publisher Report Designer's Guide*.
Maintain each field's read only setting	`all-fields-readonly-asis`	False	Set this property to "true" if you want to maintain the "Read Only" setting of each field as defined in the PDF template. This property overrides the settings of "Set all fields as read only in output." For more information, see Setting Fields as Updateable or Read Only, *Oracle Business Intelligence Publisher Report Designer's Guide*.

Flash Template Properties

The following properties can be set to govern Flash templates:

Property Name	Internal Name	Default Value	Description
Page width of wrapper document	`flash-page-width`	792	Specify in points the width of the output PDF document. The default is 792, or 11 inches.

Property Name	Internal Name	Default Value	Description
Page height of wrapper document	`flash-page-height`	612	Specify in points the height of the output PDF document. The default is 612, or 8.5 inches
Start x position of Flash area in PDF	`flash-startx`	18	Using the left edge of the document as the 0 axis point, specify in points the beginning horizontal position of the Flash object in the PDF document. The default is 18, or .25 inch
Start y position of Flash area in PDF	`flash-starty`	18	Using the upper left corner of the document as the 0 axis point, specify in points the beginning vertical position of the Flash object in the PDF document. The default is 18, or .25 inch.
Width of Flash area	`flash-width`	Same as flash width in points in swf	Enter in points the width of the area in the document for the Flash object to occupy. The default is the width of the SWF object.
Height of Flash area	`flash-height`	Same as flash height in points in swf	Enter in points the height of the area in the document for the Flash object to occupy. The default is the height of the SWF object.

Defining Font Mappings

BI Publisher's Font Mapping feature enables you to map base fonts in RTF or PDF templates to target fonts to be used in the published document. Font Mappings can be specified at the site or report level. Font mapping is performed only for PDF PowerPoint output.

There are two types of font mappings:

- RTF Templates - for mapping fonts from RTF templates and XSL-FO templates to PDF and PowerPoint output fonts
- PDF Templates - for mapping fonts from PDF templates to different PDF output fonts.

Making Fonts Available to BI Publisher

BI Publisher provides a set of Type1 fonts and a set of TrueType fonts. You can select any of the fonts in these sets as a target font with no additional setup required. For a list of the predefined fonts see BI Publisher's Predefined Fonts, page 8-17.

The predefined fonts are located in `$JAVA_HOME/jre/lib/fonts`. If you wish to map to another font, you must place the font in this directory to make it available to BI Publisher at runtime. If your environment is clustered, you must place the font on every server.

Setting Font Mapping at the Site Level or Report Level

A font mapping can be defined at the site level or the report level:

- To set a mapping at the site level, select the Font Mappings link from the Admin page.
- To set a mapping at the report level, select the Configuration link for the report, then select the Font Mappings tab. These settings will apply to the selected report only.

The report-level settings will take precedence over the site-level settings.

Creating a Font Mapping

From the **Admin** page, under **Runtime Configuration**, select **Font Mappings**.

To create a Font Mapping

- Under RTF Templates or PDF Templates, select **Add Font Mapping**.
- Enter the following on the **Add Font Mapping** page:
  - Base Font - enter the font family that will be mapped to a new font. Example: Arial
  - Select the **Style**: Normal or Italic (Not applicable to PDF Template font mappings)
  - Select the **Weight**: Normal or Bold (Not applicable to PDF Template font mappings)
  - Select the **Target Font Type**: Type 1 or TrueType
  - Enter the **Target Font**

    If you selected TrueType, you can enter a specific numbered font in the collection. Enter the **TrueType Collection (TTC) Number** of the desired font.

For a list of the predefined fonts see BI Publisher's Predefined Fonts, page 8-17

BI Publisher's Predefined Fonts

BI Publisher provides a set of Type1 fonts and a set of TrueType fonts. You can select any of these fonts as a target font with no additional setup required.

The Type1 fonts are listed in the following table:

Type 1 Fonts

Number	Font Family	Style	Weight	Font Name
1	serif	normal	normal	Time-Roman
1	serif	normal	bold	Times-Bold
1	serif	italic	normal	Times-Italic
1	serif	italic	bold	Times-BoldItalic
2	sans-serif	normal	normal	Helvetica
2	sans-serif	normal	bold	Helvetica-Bold
2	sans-serif	italic	normal	Helvetica-Oblique
2	sans-serif	italic	bold	Helvetica-BoldObliq ue
3	monospace	normal	normal	Courier
3	monospace	normal	bold	Courier-Bold
3	monospace	italic	normal	Courier-Oblique
3	monospace	italic	bold	Courier-BoldOblique
4	Courier	normal	normal	Courier
4	Courier	normal	bold	Courier-Bold
4	Courier	italic	normal	Courier-Oblique

Number	Font Family	Style	Weight	Font Name
4	Courier	italic	bold	Courier-BoldOblique
5	Helvetica	normal	normal	Helvetica
5	Helvetica	normal	bold	Helvetica-Bold
5	Helvetica	italic	normal	Helvetica-Oblique
5	Helvetica	italic	bold	Helvetica-BoldObliq ue
6	Times	normal	normal	Times
6	Times	normal	bold	Times-Bold
6	Times	italic	normal	Times-Italic
6	Times	italic	bold	Times-BoldItalic
7	Symbol	normal	normal	Symbol
8	ZapfDingbats	normal	normal	ZapfDingbats

The TrueType fonts are listed in the following table. All TrueType fonts will be subsetted and embedded into PDF.

Number	Font Family Name	Style	Weight	Actual Font	Actual Font Type
1	Albany WT	normal	normal	ALBANYWT.ttf	TrueType (Latin1 only)
2	Albany WT J	normal	normal	ALBANWTJ.ttf	TrueType (Japanese flavor)
3	Albany WT K	normal	normal	ALBANWTK.ttf	TrueType (Korean flavor)

Number	Font Family Name	Style	Weight	Actual Font	Actual Font Type
4	Albany WT SC	normal	normal	ALBANWTS.ttf	TrueType (Simplified Chinese flavor)
5	Albany WT TC	normal	normal	ALBANWTT.ttf	TrueType (Traditional Chinese flavor)
6	Andale Duospace WT	normal	normal	ADUO.ttf	TrueType (Latin1 only, Fixed width)
6	Andale Duospace WT	bold	bold	ADUOB.ttf	TrueType (Latin1 only, Fixed width)
7	Andale Duospace WT J	normal	normal	ADUOJ.ttf	TrueType (Japanese flavor, Fixed width)
7	Andale Duospace WT J	bold	bold	ADUOJB.ttf	TrueType (Japanese flavor, Fixed width)
8	Andale Duospace WT K	normal	normal	ADUOK.ttf	TrueType (Korean flavor, Fixed width)
8	Andale Duospace WT K	bold	bold	ADUOKB.ttf	TrueType (Korean flavor, Fixed width)
9	Andale Duospace WT SC	normal	normal	ADUOSC.ttf	TrueType (Simplified Chinese flavor, Fixed width)
9	Andale Duospace WT SC	bold	bold	ADUOSCB.ttf	TrueType (Simplified Chinese flavor, Fixed width)

Number	Font Family Name	Style	Weight	Actual Font	Actual Font Type
10	Andale Duospace WT TC	normal	normal	ADUOTC.ttf	TrueType (Traditional Chinese flavor, Fixed width)
10	Andale Duospace WT TC	bold	bold	ADUOTCB.ttf	TrueType (Traditional Chinese flavor, Fixed width)

9

Setting Up Print Servers

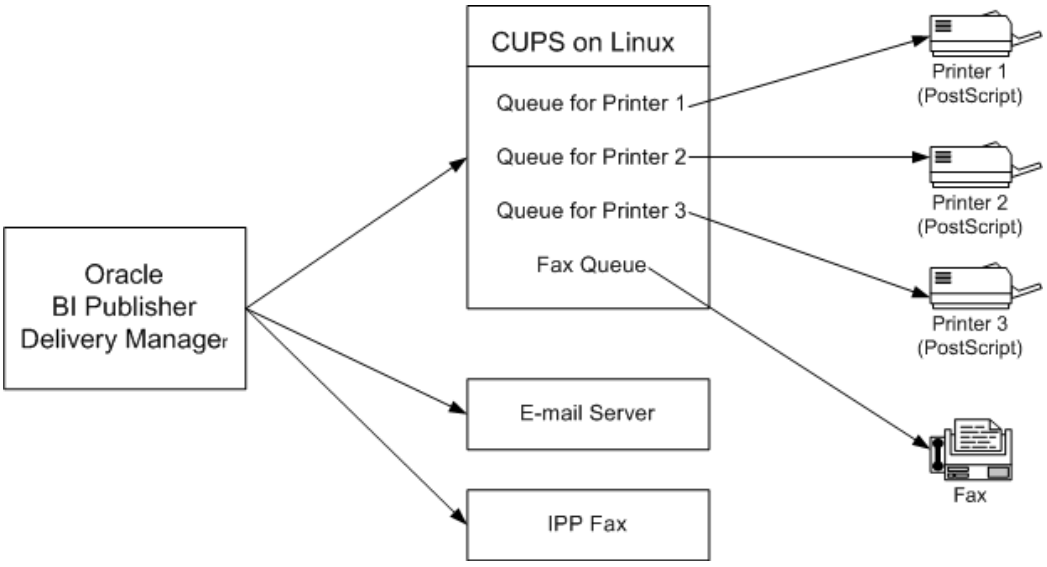
This chapter covers the following topics:

- Setting Up CUPS
- Windows XP Setup

Setting Up CUPS

The delivery manager requires Common UNIX Printing System (CUPS) to print and fax documents. This section describes how to set up CUPS for printing and faxing on RedHat Linux.

The following diagram shows the integration between BIPublisher and CUPS:



The following procedures describe how to add a printer or fax to CUPS and how to test your setup. For more information, see the *CUPS Software Administrators Manual* (http://www.cups.org/doc-1.1/sam.html) and the Redhat Advanced Server online help.

Prerequisites

- RedHat Advanced Server 3.0
- Fax Modem connected to the Linux box
- Development Tools for the RedHat Advanced Server installed
- CUPS (Installed by default when installing RedHat AS 3.0)

Setting Up a Printer on CUPS

The RedHat Advanced Server provides a configuration wizard to help you set up your printers. The RedHat process is summarized below:

Using the RedHat Printer Configuration Wizard:

1. Run "redhat-config-printer"

   While logged on as the root user, open a terminal and execute "redhat-config-printer". This invokes the **Printer configuration** window.

2. Select the **New** tab to launch the **Add a new print queue** wizard.

3. Follow the wizard prompts to:

   - Enter a queue name.
   - Select the queue type.

     Select "Networked_JetDirect" to set up a network printer. For this selection, you must also enter the following:

     - Printer - enter a hostname or IP address.
     - Port - enter a port.

     If the printer driver is installed in Microsoft Windows, the Printer and Port information is available from the Properties dialog for the printer (Settings > Printers and Faxes > (select printer) > File > Properties).

   - Select the printer model.

     If your printer supports PostScript, select the following:

     - Manufacturer: "Generic"
     - Model: "PostScript Printer"

- Review your selections and select "Apply" to create your new print queue.

4. Your new queue now displays in the Printer configuration window.

Test Your Printer on CUPS:

1. Launch a browser on RedHat and enter the following URL:

   http://localhost:631

2. Select the **Printers** tab. The printer you just created will be listed.

   To use your Windows browser to access this page, see Making CUPS Accessible from Other Machines, page 9-5.

3. Select **Print Test Page** to test your printer setup. If the test page does not print, repeat the configuration steps. Ensure that your printer type and model selections are correct.

Installing and Setting Up Fax for CUPS

This section describes how to install efax-0.9 software and configure it for CUPS.

Install the Fax Software:

1. Download efax-0.9 from one of the following locations:
   - http://www.cce.com/efax/download/
   - ftp://ftp.metalab.unc.edu/pub/Linux/apps/serialcomm/fax/efax-0.9.tar.gz

2. Extract the files to a working directory using the following commands:
   - gunzip efax-0.9.tar.gz
   - tar xvf efax-0.9.tar

3. Compile and install using the following commands (refer to the Readme for more information):
   - make
   - make install

   Note: You must have `make` and `gcc` installed in your RedHat AS.

4. Test the fax.

Enter the following command:

```
fax send <fax_number><tiff file>
```

For example:

```
fax send 1234567 test.tiff
```

The fax is successful if you get the return code:

```
done, returning 0 (success)
```

5. Download fax4CUPs. It is available from the following site:
   - http://www.gnu.org/directory/productivity/special/fax4CUPS.html
6. Install fax4CUPS as follows:
   - Extract the tar file to a temporary directory
   - Change the directory: cd fax4CUPS-1.23
   - Open the INSTALL file and follow all steps.
7. Restart CUPS using the following command:

   ```
   /etc/rc.d/init.d/cups restart
   ```

Setting Up a Fax on CUPS:

1. Launch a browser and go to the following URL: `http://localhost:631/admin`
2. Enter the admin username and password in the dialog that launches.
3. From the **Admin** page, select **Add Printer**.
4. Add a Fax queue as follows:

   In the **Add New Printer** region, enter the following fields:
   - Name - enter a meaningful name for the, such as "efaxserver". This will be referred to as "ipp://serverName:631/printers/efaxserver".
   - Location - optional.
   - Description - optional.
5. Select a device for the fax queue.

   Select "Faxmodem (efax on /dev/modem)". In some cases, "/dev/ttySxx" will be shown instead.
6. Select a model for the fax queue.

Select "efax". You can also select either "HylaFAX" or "mgetty-fax" if these have been installed.

7. Select the driver for the fax queue.

   Select "efax (en)".

8. Verify that the new fax queue appears on the CUPS Admin Web page.

9. Text the fax on CUPS.

   Enter the following command to test the fax:

   ```
   /usr/bin/lp -d <printer name> -t <phone#> test.pdf
   ```

   Example:

   ```
   /usr/bin/lp -d efax1 -t 5556231 myfax.pdf
   ```

Making CUPS Accessible from Other Machines

By default, CUPS does not allow access from other network machines. However, it can be configured to allow access, as follows:

1. Open a CUPS configuration file using the following command:

   ```
   Open /etc/cups/cupsd.conf
   ```

2. Add a "Listen" instruction.

   - Scroll to the bottom of the configuration file where the other Listen instructions are declared.
   - Copy "Listen 127.0.0.1:631" and paste it above or below the original.
   - Replace "127.0.0.1" with the Linux server's IP address.

3. Configure each printer.

   - In the configuration file, locate:

     ```
     <Location /printers/your_printer_queue>
     ```

   - Comment the instruction "Deny From All".

     Example:

     ```
     # Deny From All
     ```

   - Change "Allow from 127.0.0.1" to "Allow from All"
   - Repeat for all printer or fax queues that you wan to make accessible.

4. Save the configuration file and restart CUPS.

- Use the following command to stop CUPS:

  ```
  /etc/rc.d/init.d/cups stop
  ```

- Use the following command to start CUPS:

  ```
  /etc/rc.d/init.d/cups start
  ```

5. Test the accessibility from other machines.

   Launch a browser from another machine and enter one of the following URLs to ensure that the CUPS web page can be accessed:

   - `http://linux_server_name:631`
   - `http://linux_ip_address:631`

Windows XP Setup

This section describes how to set up Internet Printing Protocol (IPP) on a Windows XP server.

Prerequisite:

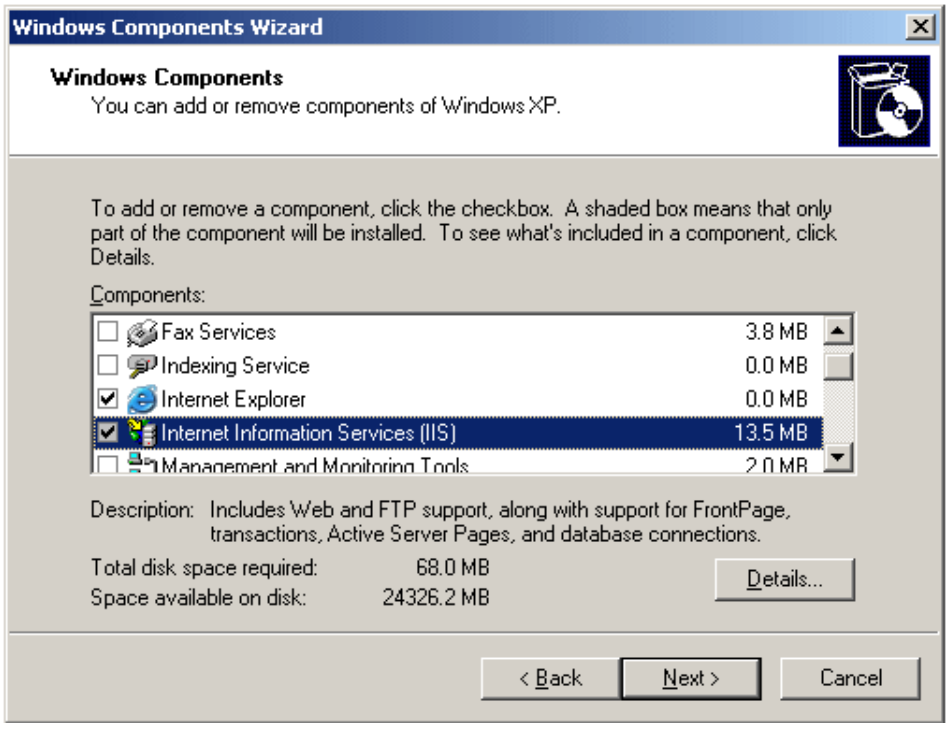
Microsoft Windows XP

Setting Up IPP Printers on Windows XP Professional

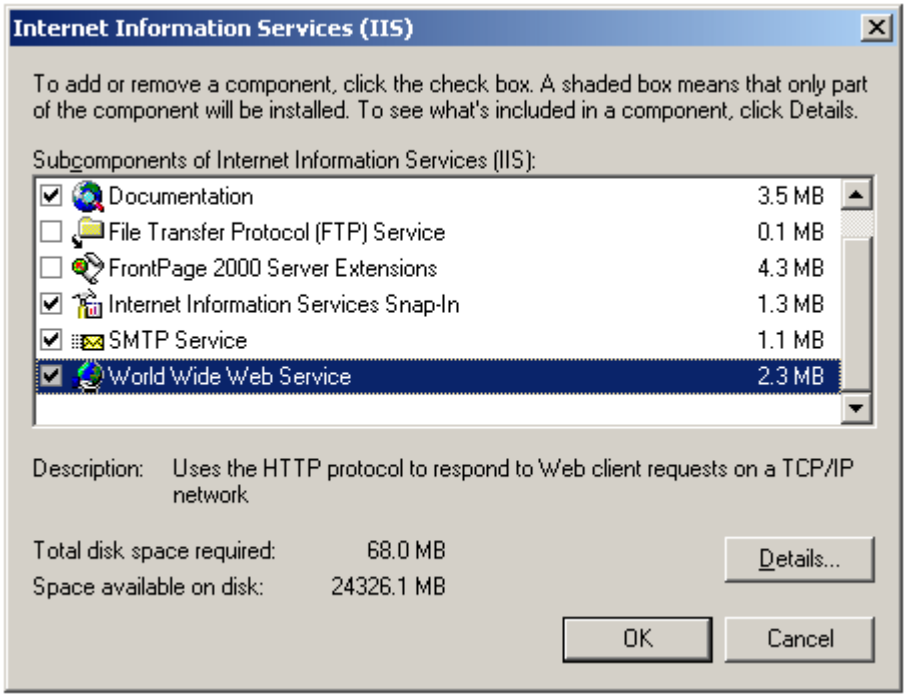
1. Install Internet Information Services (IIS).

   - Open the Control Panel. Select **Add or Remove Programs**, then **Add/Remove Windows Components**.

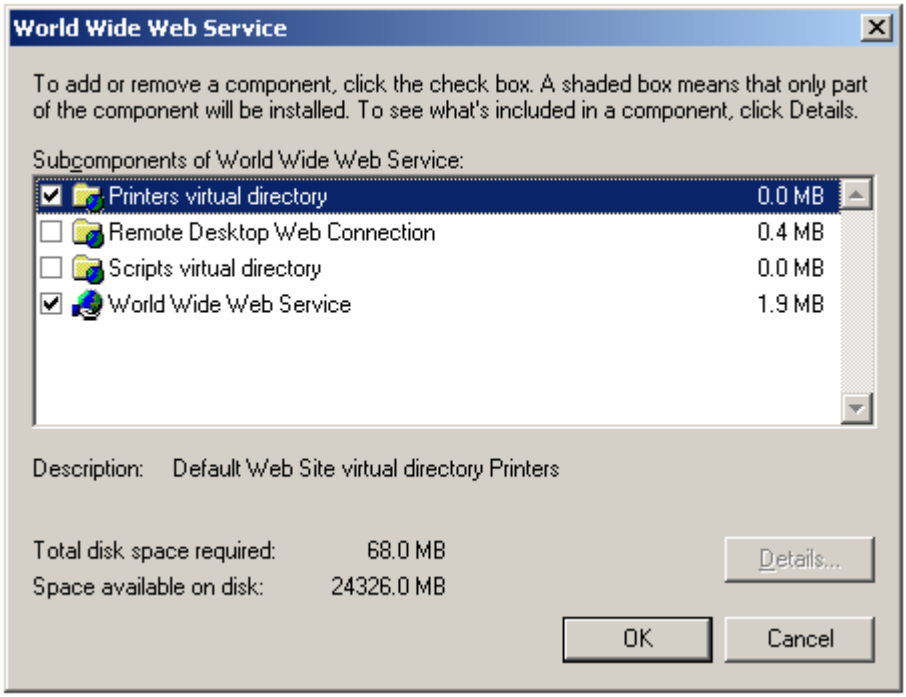
   - Select the check box for Internet Information Services (IIS) from the list of available Windows Components (shown in the following figure).



- With IIS highlighted, select **Details**. Ensure that **World Wide Web Service** is selected (shown in the following figure).



- With **World Wide Web Service** highlighted, select **Details**. Ensure that **Printers virtual directory** is selected (shown in the following figure).



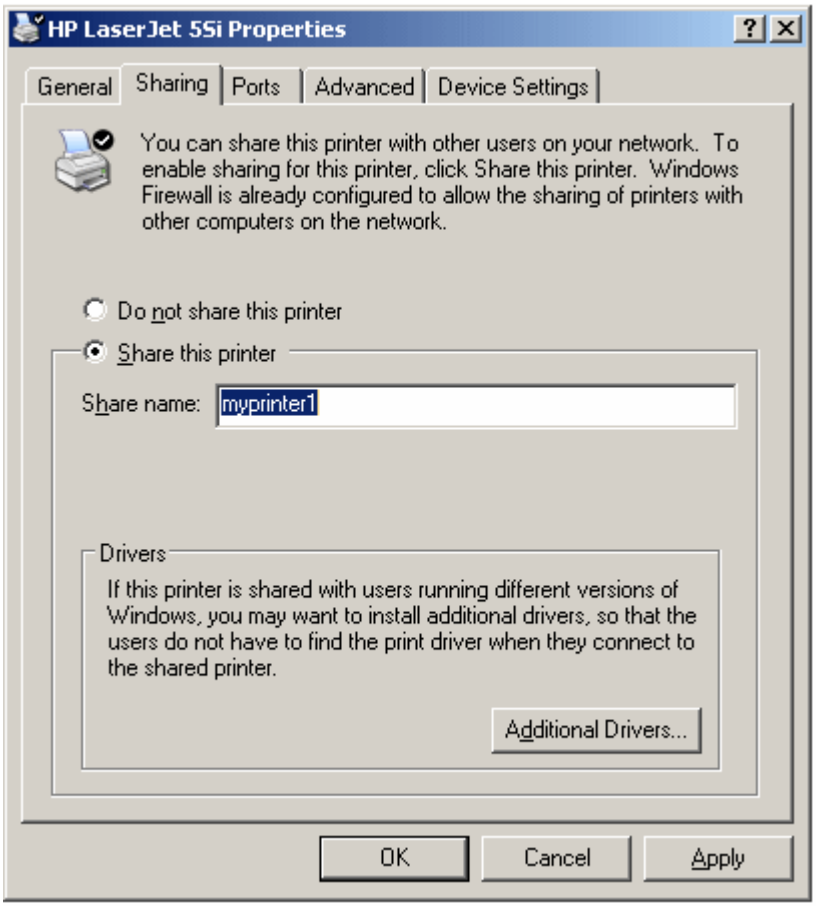
2. Restart Windows XP.

3. Share the printers.

   - From the **Start** menu, select **Settings**, then **Printers and Faxes**.

   - Right-click the printer icon and select **Sharing**.

   - In the printer **Properties** dialog, select **Share this printer** and assign a **Share name** (for example: myprinter1). An example is shown in the following figure.

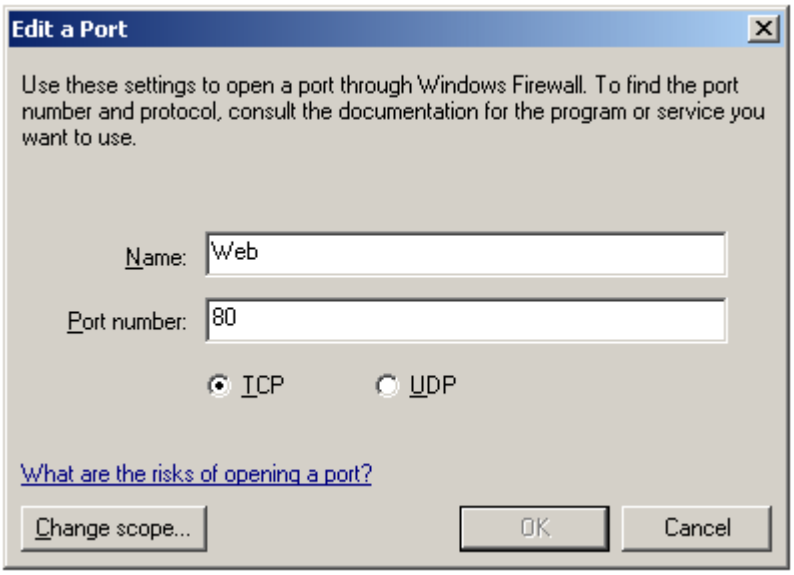


4. Configure the Windows Firewall to open a port to your XMLP Server:
   - From the **Start** menu, select **Settings**, then **Control Panel**.
   - From the **Control Panel**, select **Windows Firewall**.
   - From the **Windows Firewall** dialog, select the **Exceptions** tab.
   - Create an entry in the list of Programs and Services as follows:
     1. Select **Add Port**.
     2. Enter the **Name**: for example, Web

        Enter the **Port number**: for example, 80

        Select **TCP**.

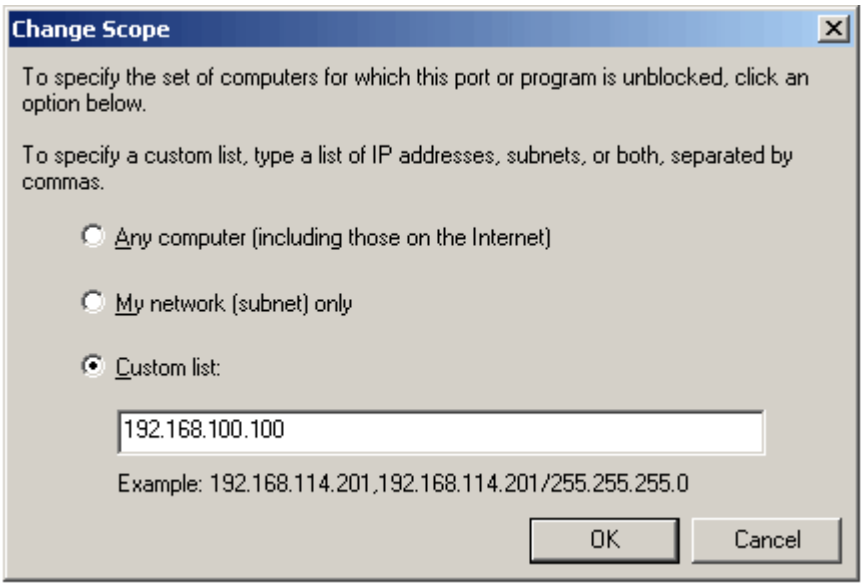
        An example is shown in the following figure:



- To allow access from a specific IP address only:
  1. Select your entry, then select **Edit**.
  2. From the **Edit a Program** dialog, select **Change Scope**.
  3. Choose **Custom list**, and enter the IP address of the XMLP Server.

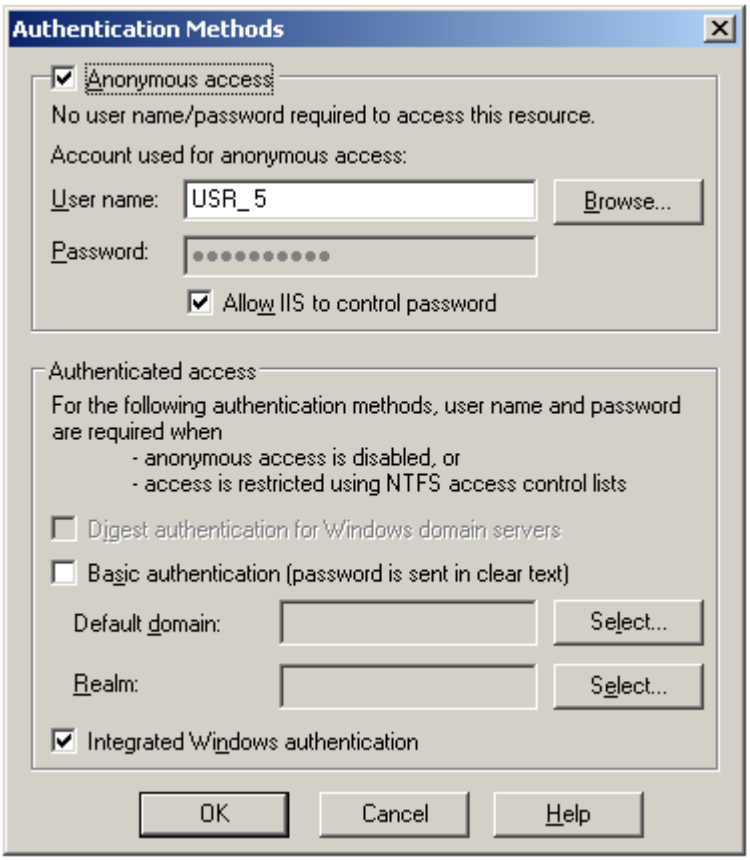
     Note that if you use the HTTP proxy server in BI Publisher Server, you must enter the IP address of the proxy server.

     An example is shown in the following figure.



5. Change the virtual directory security setting.

   - From the **Control Panel**, select **Administrative Tools**, then **Internet Information Service**.
   - Navigate the Internet Information Service directory hierarchy as follows: Internet Information Services > [your server name] > Web Sites > Default Web Site > Printers. Right-click **Printers** and choose **Properties**.
   - From the **Printers Properties** dialog, select the **Directory Security** tab.
   - In the **Anonymous access and authentication control** region, select **Edit**.
   - In the **Authentication Methods** dialog, select the **Anonymous access** check box. An example is shown in the following figure.



6. Open a browser in a remote machine and enter the following URL: `http://<your server name>/printers`

   You will see the list of shared printers.

Part 2

Developer's Guide

10

Using the BI Publisher Web Services

Overview

PublicReportService is the Web service that wraps the Oracle BI Publisher public APIs. The following general categories of operations are supported:

- Operations for validation of privileges
- Operations to get information about reports and the repository
- Operations to run reports
- Operations to manage scheduled reports and history of scheduled reports
- Operations to create and manage reports
- Operations to get information about the report server

> **Note:** For more information about Web services, see http://ws.apache.org/axis/java/user-guide.html

WSDL Definition for PublicReportService

After you have installed or deployed Oracle BI Publisher, there is a unique URL associated with PublicReportService:

```
http://<host>:<port>/xmlpserver/services/PublicReportService?wsdl
```

Enter this URL in your browser, substituting in the correct host and port number, to display the full Web Service Description Language definition for the publicly supported BI Publisher Web service.

Oracle BI Publisher Web Service Data Types

This section contains the following topics:

- Base Data Types
- XML-to-Java Data Type Mappings
- Complex Types

Base Data Types

Oracle BI Publisher Web services use the following base data types:

Base Type	Description	Example
xsd:boolean	Boolean	true, false
xsd:dateTime	Date and Time	2007-10-26T21:32:52
xsd:int	Integer	23
xsd:string	String	/Home/Shared/HR Reports/Salary Report
xsd:base64Binary	64 bit binary	A document, for example a PDF or HTML report

XML-to-Java Data Type Mappings

BI Publisher Web Services use document/literal formats. The mapping between Web service XML schema data types and Java data types depends on the SOAP development environment. The following table shows mappings for the Oracle JDeveloper environment:

Base Type	Oracle JDeveloper
xsd:boolean	java.lang.Boolean
xsd:dateTime	java.util.Date

Base Type	Oracle JDeveloper
xsd:int	java.lang.Integer
xsd:string	java.lang.String
xsd:base64Binary	java.lang.Byte

Complex Types

Oracle BI Publisher Web services define and use the following complex types:

AccessDeniedException

```
<complexType name="AccessDeniedException">
 <sequence />
</complexType>
```

Description: message returned when the user credentials do not have adequate privileges to complete an operation.

ArrayOfItemData

```
<complexType name="ArrayOfItemData">
   <complexContent>
      <restriction base="soapenc:Array">
      <attribute ref="soapenc:arrayType"
wsdl:arrayType="impl:ItemData[]" />
      </restriction>
   </complexContent>
</complexType>
```

Description: an array of ItemData - objects contained in the report repository.

ArrayOfParamNameValue

```
<complexType name="ArrayOfParamNameValue">
   <complexContent>
      <restriction base="soapenc:Array">
      <attribute ref="soapenc:arrayType"
wsdl:arrayType="impl:ParamNameValue[]" />
      </restriction>
   </complexContent>
</complexType>
```

Description: an array of ParamNameValues.

ArrayOf_xsd_string

```
<complexType name="ArrayOf_xsd_string">
  <complexContent>
    <restriction base="soapenc:Array">
    <attribute ref="soapenc:arrayType" wsdl:arrayType="xsd:string[]"/>
    </restriction>
  </complexContent>
</complexType>
```

Description: an array of strings used in other complex types.

DeliveryRequest

```
<complexType name="DeliveryRequest">
  <sequence>
    <element name="emailOption" nillable="true"
type="impl:EMailDeliveryOption"/>
    <element name="faxOption" nillable="true"
type="impl:FaxDeliveryOption"/>
    <element name="ftpOption" nillable="true"
type="impl:FTPDeliveryOption"/>
    <element name="localOption" nillable="true"
type="impl:LocalDeliveryOption"/>
    <element name="printOption" nillable="true"
type="impl:PrintDeliveryOption"/>
    <element name="webDAVOption" nillable="true"
type="impl:WebDAVDeliveryOption"/>
  </sequence>
</complexType>
```

Description: an object that provides the options to deliver a report to multiple destinations. This type is used in the following complex type structures:

JobInfo, page 10-10

ScheduleRequest, page 10-19

Parameter	Type	Description
emailOption	EMailDeliveryOption	See EMailDeliveryOption, page 10-5
faxOption	FaxDeliveryOption	See FaxDeliveryOption, page 10-5
ftpOption	FTPDeliveryOption	See FTPDeliveryOption, page 10-6
localOption	LocalDeliveryOption	See LocalDeliveryOption, page 10-12
printOption	PrintDeliveryOption	See PrintDeliveryOption, page 10-13
webDAVOption	WebDAVDeliveryOption	See WebDAVDeliveryOption, page 10-21

EMailDeliveryOption

```
<complexType name="EMailDeliveryOption">
  <sequence>
    <element name="emailBCC" nillable="true" type="xsd:string"/>
    <element name="emailBody" nillable="true" type="xsd:string"/>
    <element name="emailCC" nillable="true" type="xsd:string"/>
    <element name="emailFrom" nillable="false" type="xsd:string"/>
    <element name="emailReplyTo" nillable="true" type="xsd:string"/>
    <element name="emailSubject" nillable="false" type="xsd:string"/>
    <element name="emailTo" nillable="false" type="xsd:string"/>
  </sequence>
</complexType>
```

Description: the options to set for e-mail delivery of a report.

This type is used in the following complex type structure: DeliveryRequest, page 10-4.

Parameter	Type	Description
emailBCC	string	The e-mail addresses of to receive blind copies of the e-mail.
emailBody	string	A text string that will appear as the body of the e-mail.
emailCC	string	The e-mail addresses to receive copies of the e-mail.
emailFrom	string	The e-mail address that will appear as the From address.
emailReplyTo	string	The e-mail address to appear in the Reply-to field.
emailSubject	string	The subject line of the e-mail.
emailTo	string	The addresses to which to send the e-mail.

FaxDeliveryOption

```
<complexType name="FaxDeliveryOption">
  <sequence>
    <element name="faxNumber" nillable="false" type="xsd:string"/>
    <element name="faxServer" nillable="false" type="xsd:string"/>
  </sequence>
</complexType>
```

Description: the options to set for facsimile (fax) delivery of a report.

This type is used in the following complex type structure: DeliveryRequest, page 10-4

Parameter	Type	Description
faxNumber	string	The number to which to send the fax. For example: 916505069560
faxServer	string	The fax server to which to send the fax. For example: ipp://mycupsserver:631/printers/fax2

FTPDeliveryOption

```
<complexType name="FTPDeliveryOption">
  <sequence>
    <element name="ftpServerName" nillable="false" type="xsd:string"/>
    <element name="ftpUserName" nillable="false" type="xsd:string"/>
    <element name="ftpUserPassword" nillable="false" type="xsd:string"/>
    <element name="remoteFile" nillable="false" type="xsd:string"/>
    <element name="sftpOption" type="xsd:boolean"/>
  </sequence>
</complexType>
```

Description: the options to set for FTP delivery of a report.

This type is used in the following complex type structure: DeliveryRequest, page 10-4

Parameter	Type	Description
ftpServerName	string	The FTP server name. For example myftpserver.mycompany.com
ftpUserName	string	A user name for the FTP server.
ftpUserPassword	string	The password for the user entered.
remoteFile	string	The name to assign the file on the server. For example: report.pdf.
sftpOption	boolean	True indicates to use the secure FTP option.

InvalidParametersException

```
<complexType name="InvalidParametersException">
 <sequence />
</complexType>
```

Description: message returned when invalid parameters are supplied for a Web service operation.

ItemData

```
<complexType name="ItemData">
  <sequence>
    <element name="absolutePath" nillable="true" type="xsd:string"/>
    <element name="creationDate" nillable="true" type="xsd:dateTime"/>
    <element name="displayName" nillable="true" type="xsd:string"/>
    <element name="fileName" nillable="true" type="xsd:string"/>
    <element name="lastModified" nillable="true" type="xsd:dateTime"/>
    <element name="lastModifier" nillable="true" type="xsd:string"/>
    <element name="owner" nillable="true" type="xsd:string"/>
    <element name="parentAbsolutePath" nillable="true"
type="xsd:string"/>
    <element name="type" nillable="true" type="xsd:string" />
  </sequence>
</complexType>
```

Description: description of an object contained in the report repository.

Parameter	Type	Description
absolutePath	string	The path to the report object. For example: /HR Manager/ HR Reports/Employee Listing.xdo
creationDate	dateTime	The creation date of the report object.
displayName	string	The display name for the report object. For example: Employee Listing
fileName	string	The file name for the report object. For example Employee Listing.xdo
lastModified	dateTime	The last modified date for the report object.
lastModifier	sring	The user name of the last person to modify the report.
owner	string	The user name of the owner of the report.
parentAbsolutePath	string	The absolute path of the parent folder. For example, "/HR Manager/HR Reports" is the parentAbsolutePath for the report having the absolute path "/HR Manager/ HR Reports/Employee Listing.xdo".
type	string	The item type. Possible values are: "report" or "folder".

JobHistoryInfo

```
<complexType name="JobHistoryInfo">
  <sequence>
     <element name="createdDate" nillable="true" type="xsd:dateTime"/>
     <element name="documentDataAvailable" type="xsd:boolean"/>
     <element name="documentDataContentType" nillable="true"
type="xsd:string"/>
     <element name="documentOutput" nillable="true"
type="xsd:base64Binary"/>
     <element name="jobID" type="xsd:int"/>
     <element name="jobMessage" nillable="true" type="xsd:string"/>
     <element name="jobName" nillable="true" type="xsd:string"/>
     <element name="outputID" type="xsd:int"/>
     <element name="reportData" nillable="true"
type="xsd:base64Binary"/>
     <element name="reportDataAvailable" type="xsd:boolean"/>
     <element name="reportDataContentType" nillable="true"
type="xsd:string"/>
     <element name="reportName" nillable="true" type="xsd:string"/>
     <element name="reportURL" nillable="true" type="xsd:string"/>
     <element name="status" nillable="true" type="xsd:string"/>
     <element name="statusDetail" nillable="true" type="xsd:string"/>
     <element name="userName" nillable="true" type="xsd:string"/>
     <element name="username" nillable="true" type="xsd:string"/>
    </sequence>
</complexType>
```

Description: lists parameters describing the job history.

Parameter	Type	Description
createdDate	dateTime	The date the report job was created.
documentDataAvailable	boolean	True indicates that the user selected the "Save Output" option when the report was scheduled.
documentDataConentType	string	The content type of the generated document. Possible values are: "text/html;charset=UTF-8" "text/plain;charset=UTF-8" "application/pdf" "application/vnd.ms-powerpoint" "application/vnd.ms-excel" "application/msword" "application/x-shockwave-flash" "text/xml" "message/rfc822"

Parameter	Type	Description
documentOutput	base64binary	The output document.
jobID	integer	The identification number assigned to the job by BI Publisher.
jobMessage	string	When a scheduled report is executed, the scheduler generates a plain-text message associated with the scheduled job. This is the server message containing detailed information. For example, {pstatus=success, job_tz=America/Dawson, job_cal=GREGORIAN, job_locale=en_US, dstatus=success, dstatus4=nstarted, dstatus3=nstarted, dstatus2=nstarted, dstatus1=nstarted, dstatus0=nstarted} ... Or upon failure, gives you some error message as well – {pstatus=success, job_tz=Asia/Chongqing, job_cal=GREGORIAN, job_locale=zh_CN, dstatus=success, dstatus4=nstarted, dstatus3=nstarted, dstatus2=nstarted, dstatus1=nstarted, dstatus0=failed, ...
jobName	string	The user-assigned job name.
outputID	integer	The identification of the report in history. One scheduled JobID can be associated with multiple outputIDs or historyIDs. This is because one scheduled report can be executed or republished multiple times.
reportData	base64binary	The XML data generated by the report data model.
reportDataAvailable	boolean	True indicates that the user selected "Save data for republish" when the report was scheduled.
reportDataContentType	string	The content type value of the XML data that BI Publisher will use to generate the report. The value is "text/xml".
reportName	string	The name of the report.
reportURL	string	The report absolute path URL For example: "/HR Manager/HR Reports/Employee Listing.xdo".
status	string	Valid values are: "Completed", "Error", "Running", "Scheduled", "Suspended" and "Unknown"

Parameter	Type	Description
statusDetail	string	Detailed status information from the BI Publisher server.
username	string	The user who scheduled the report. For example, "Administrator".

JobInfo

```
<complexType name="JobInfo">
   <sequence>
     <element name="burstingOption" type="xsd:boolean"/>
     <element name="createdDate" nillable="true" type="xsd:dateTime"/>
     <element name="deliveryDescription" nillable="true"
type="xsd:string"/>
     <element name="deliveryParameters" nillable="true"
type="impl:DeliveryRequest"/>
     <element name="endDate" nillable="true" type="xsd:dateTime"/>
     <element name="jobGroup" nillable="true" type="xsd:string"/>
     <element name="jobID" type="xsd:int"/>
     <element name="jobName" nillable="true" type="xsd:string"/>
     <element name="locale" nillable="true" type="xsd:string"/>
     <element name="reportName" nillable="true" type="xsd:string"/>
     <element name="reportParameters" nillable="true"
type="impl:ArrayOfParamNameValue"/>
     <element name="reportSet" type="xsd:boolean"/>
     <element name="reportURL" nillable="true" type="xsd:string"/>
     <element name="runType" nillable="true" type="xsd:string"/>
     <element name="scheduleDescription" nillable="true"
type="xsd:string"/>
     <element name="scheduleParameters" nillable="true"
type="xsd:string"/>
     <element name="startDate" nillable="true" type="xsd:dateTime"/>
     <element name="status" nillable="true" type="xsd:string"/>
     <element name="username" nillable="true" type="xsd:string"/>
    </sequence>
</complexType>
```

Description: the collection of information about a job request.

Parameter	Type	Description
burstingOption	boolean	True indicates the bursting option is on.
createdDate	dateTime	The date the job was created.
deliveryDescription	string	Description of the scheduled delivery. In the current implementation this parameter returns null.

Parameter	Type	Description
deliveryParameters	DeliveryRequest	See DeliveryRequest, page 10-4
endDate	dateTime	The date the job ended.
jobGroup	string	In the current implementation this parameter returns the userID. The "jobGroup" refers to the group of scheduled requests that belong to a particular user, as when you query for a particular user's scheduled requests.
jobID	integer	The numeric identification assigned by BI Publisher to the job request.
jobName	string	The user-assigned job name.
locale	string	The user-selected locale for the scheduled report.
reportName	string	The name of the report.
reportParameters	ArrayOfParamNameValue	See ArrayOfParamNameValue, page 10-3
reportSet	boolean	True indicates the report is a member of a report set. In the current implementation this will always return false.
reportURL	string	The report absolute path URL, for example: /HR Manager/Employee Reports/Employee Salary Report.xdo.
runType	string	Possible values are: "Run Once Only" or "Run Recurring".
scheduleDescription	string	The description for a report schedule. In the current implementation this parameter returns null.
scheduleParameters	string	See ScheduleRequest, page 10-19.
startDate	dateTime	The date the schedule is to start.

Parameter	Type	Description
status	string	The status of the scheduled job request. Valid values are: Canceled", "Done", "Scheduled", "Suspended", or "Unknown".
username	string	The username of the user submitting the job request.

JobStatus

```
<complexType name="JobStatus">
    <sequence>
     <element name="jobID" nillable="true" type="xsd:string"/>
     <element name="jobStatus" nillable="true" type="xsd:string"/>
     <element name="message" nillable="true" type="xsd:string"/>
    </sequence>
   </complexType>
```

Description: describes the scheduled job status.

Parameter	Type	Description
jobID	string	The numeric identification assigned by BI Publisher to the job request.
jobStatus	string	The status of the job request. Possible values are: "Completed", "Error" "Running", "Scheduled", "Suspended", "Unknown"
message	string	BIP scheduler server message string.

LocalDeliveryOption

```
<complexType name="LocalDeliveryOption">
  <sequence>
    <element name="destination" nillable="false" type="xsd:string"/>
  </sequence>
</complexType>
```

Description: the options to set for delivery of a report to the BI Publisher repository.

This type is used in the following complex type structure: DeliveryRequest, page 10-4

Parameter	Type	Description
destination	string	The file path to the BI Publisher repository on the local server.

OperationFailedException

```
<complexType name="OperationFailedException">
  <sequence />
</complexType>
```

Description: the message returned when an operation fails.

ParamNameValue

```
<complexType name="ParamNameValue">
   <sequence>
      <element name="multiValuesAllowed" type="xsd:boolean" />
      <element name="name" nillable="true" type="xsd:string" />
      <element name="values" nillable="true"
type="impl:ArrayOf_xsd_string" />
   </sequence>
</complexType>
```

Description: a parameter name and array of parameter values.

Parameter	Type	Description
multiValuesAllowed	boolean	True indicates that a parameter may contain multiple values.
name	string	The parameter name.
values	ArrayOf_xsd_string	See ArrayOf_xsd_string, page 10-4

PrintDeliveryOption

```
<complexType name="PrintDeliveryOption">
  <sequence>
    <element name="printNumberOfCopy" nillable="true"
type="xsd:string"/>
    <element name="printRange" nillable="true" type="xsd:string"/>
    <element name="printSide" nillable="true" type="xsd:string"/>
    <element name="printTray" nillable="true" type="xsd:string"/>
     <element name="printerName" nillable="false" type="xsd:string"/>
  </sequence>
</complexType>
```

Description: the options to set for printer delivery of a report.

This type is used in the following complex type structure: DeliveryRequest, page 10-4

Parameter	Type	Description
printNumberOfCopy	string	The number of copies to print.
printRange	string	A range of pages to print. Separate multiple ranges with a comma. For example: 1,3-5,8-10
printSide	string	Valid values are "Single sided", "Double Sided Long Edge (Duplex)", and "Double Sided Short Edge (Tumble)"
printTray	string	Valid values are "Default", "Tray 1", "Tray 2", and "Tray 3"
printerName	string	The name of the printer to which to send the report.

ReportDataChunk

```
<complexType name="ReportDataChunk">
   <sequence>
      <element name="reportDataChunk" nillable="true"
type="xsd:base64Binary"/>
      <element name="reportDataFileID" nillable="true"
type="xsd:string"/>
      <element name="reportDataOffset" type="xsd:long"/>
   </sequence>
</complexType>
```

Parameter	Type	Description
reportDataChunk	xsd64binary	The report data.
reportDataFileID	string	The report file identifier.
reportDataOffset	long	The offset at which the data portion of the chunk begins.

ReportDefinition

```
<complexType name="ReportDefinition">
   <sequence>
      <element name="autoRun" type="xsd:boolean"/>
      <element name="cacheDocument" type="xsd:boolean"/>
      <element name="defaultOutputFormat" nillable="true"
type="xsd:string" />
      <element name="defaultTemplateId" nillable="true"
type="xsd:string" />
      <element name="diagnostics" type="xsd:boolean"/>
      <element name="listOfTemplateFormatsLabelValues" nillable="true"
type="impl:ArrayOfTemplateFormatsLabelValues"/>
      <element name="onLine" type="xsd:boolean"/>
      <element name="openLinkInNewWindow" type="xsd:boolean"/>
      <element name="parameterColumns" type="xsd:int"/>
      <element name="parameterNames" nillable="true"
        type="impl:ArrayOf_xsd_string" />
      <element name="reportParameterNameValues" nillable="true"
        type="impl:ArrayOfParamNameValue" />
      <element name="reportDefnTitle" nillable="true"
type="xsd:string"/>
      <element name="reportDescription" nillable="true"
type="xsd:string"/>
      <element name="reportName" nillable="true" type="xsd:string"/>
      <element name="reportParameterNameValues" nillable="true"
type="impl:ArrayOfParamNameValue"/>
      <element name="reportType" nillable="true" type="xsd:string"/>
      <element name="showControls" type="xsd:boolean"/>
      <element name="showReportLinks" type="xsd:boolean"/>
      <element name="templateIds" nillable="true"
type="impl:ArrayOf_xsd_string" />
      <element name="useExcelProcessor" type="xsd:boolean"/>
   </sequence>
</complexType>
```

Description: object returned from `getReportDefinition`.

Parameter	Type	Description
autoRun	boolean	True indicates that the report property Auto Run is turned on.
cacheDocument	boolean	True indicates that the report property Enable document cache is turned on.
defaultOutputFormat	string	The default output format. Valid values are: pdf, rtf, html, flash, csv, data, mhtml, excel, excel2000, and powerpoint.
defaultTemplateId	string	The default template identified for the report.

Parameter	Type	Description
diagnostics	boolean	True indicates that diagnostics have been turned on for the report.
listOfTemplateFormatsLabelValues	ArrayOfTemplateFormatsLabelValues	Passes the list of template format labels via the ArrayOfTemplateFormatsLabelValues structure.
onLine	boolean	True indicates the property "Run report online" is turned on for the report.
openLinkInNewWindow	boolean	True indicates the property "Open Links in New Window" is turned on for the report.
parameterColumns	integer	The value of the report property "Parameters per line."
parameterNames	ArrayOf_xsd_string	Passes the parameter names defined for the report via the ArrayOf_xsd_string, page 10-4 structure.
reportParameterNameValues	ArrayOfParamNameValue	Passes the report name-value pairs via the ArrayOfParamNameValue, page 10-3 structure.
reportType	string	This currently returns null.
showControls	boolean	True indicates the report property "Show controls" has been turned on.
showReportLinks	boolean	True indicates the report property "Show report links" has been turned on.
templateIds	ArrayOf_xsd_string	Passes the layout names of the report templates via the ArrayOf_xsd_string, page 10-4 structure.
useExcelProcessor	boolean	True indicates that the report property "Disable Client Access from Analyzer for Excel" has not been turned on.

ReportRequest

```
<complexType name="ReportRequest">
   <sequence>
      <element name="attributeFormat" nillable="true"
type="xsd:string"/>
      <element name="attributeLocale" nillable="true"
type="xsd:string"/>
      <element name="attributeTemplate" nillable="true"
type="xsd:string"/>
      <element name="flattenXML" type="xsd:boolean"/>
      <element name="parameterNameValues" nillable="true"
type="impl:ArrayOfParamNameValue"/>
       <element name="reportAbsolutePath" nillable="false"
type="xsd:string"/>
       <element name="reportData" nillable="true"
type="xsd:base64Binary"/>
      <element name="reportDataFileName" nillable="true"
type="xsd:string"/>
      <element name="sizeOfDataChunkDownload" type="xsd:int"/>
   </sequence>
</complexType>
```

Description: the collection of settings needed to run a report. Note that allowable values for `attributeFormat` will vary according to the type of template used (for example, PDF templates can only generate PDF output.)

Parameter	Type	Description
attributeFormat	string	The output format of the requested report. Valid values are: pdf, rtf, html, excel, excel2000, mhtml, csv, data, flash, and powerpoint.
attributeLocale	string	The locale selection for the report. Example: fr-FR
attributeTemplate	string	The template to apply to the report. For example: Employeelisting.rtf.
flattenXML	boolean	True indicates that the XML is to be flattened. This flag is used for the Analyzer for Microsoft Excel because Excel requires XML data structure to be flattened.
parameterNameValues	ArrayOfParamNameValue	The parameter name-value pairs to be used in the submission of this report request, passed via the ArrayOfParamNameValue, page 10-3 structure.

Parameter	Type	Description
reportAbsolutePath	string	The absolute path to the report in the BI Publisher repository. For example: /HR Manager/HR Reports/Employee Listing.xdo.
reportData	base64binary	If you are providing the data directly for the report use this element to pass the data.
reportDataFileName	string	If using a data file to create the report, enter the data file name.
sizeOfDataChunkDownload	integer	If you set flattenXML to true, or if you do not want to chunk the data, set this parameter to -1 to return all data back to the client.

ReportResponse

```
<complexType name="ReportResponse">
   <sequence>
      <element name="reportBytes" nillable="true"
type="xsd:base64Binary" />
      <element name="reportContentType" nillable="true"
type="xsd:string" />
      <element name="reportLocale" nillable="true" type="xsd:string" />
   </sequence>
</complexType>
```

Description: the document (in whatever file format `attributeFormat` is set to in `ReportRequest`), document type, and document locale that is returned from the `runReport` operation.

Parameter	Type	Description
reportBytes	base64binary	The report binary data output.

Parameter	Type	Description
reportContentType	string	The report content type. Possible values are: "text/html;charset=UTF-8" "text/plain;charset=UTF-8" "application/pdf" "application/vnd.ms-powerpoint" "application/vnd.ms-excel" "application/msword" "application/x-shockwave-flash" "text/xml" "message/rfc822"
reportLocale	string	The locale selected for the report. For example: fr-FR

ScheduleRequest

```
<complexType name="ScheduleRequest">
  <sequence>
    <element name="cronExpression" nillable="true" type="xsd:string"/>
    <element name="deliveryRequest" nillable="false"
type="impl:DeliveryRequest"/>
    <element name="endDate" nillable="true" type="xsd:dateTime"/>
    <element name="jobCalendar" nillable="true" type="xsd:string"/>
    <element name="jobLocale" nillable="true" type="xsd:string"/>
    <element name="jobTZ" nillable="true" type="xsd:string"/>
    <element name="notificationTo" nillable="true" type="xsd:string"/>
    <element name="notifyWhenFailed" type="xsd:boolean" />
    <element name="notifyWhenSuccess" type="xsd:boolean" />
    <element name="notifyWhenWarning" type="xsd:boolean" />
    <element name="repeatCount" type="xsd:int"/>
    <element name="repeatInterval" type="xsd:int"/>
    <element name="reportRequest" nillable="false"
type="impl:ReportRequest"/>
    <element name="saveDataOption" type="xsd:boolean" />
    <element name="saveOutputOption" type="xsd:boolean" />
    <element name="scheduleBurstingOption" type="xsd:boolean" />
    <element name="schedulePublicOption" type="xsd:boolean" />
    <element name="startDate" nillable="true" type="xsd:dateTime"/>
    <element name="useUTF8Option" type="xsd:boolean" />
    <element name="userJobName" nillable="true" type="xsd:string"/>
  </sequence>
</complexType>
```

Description: the options to schedule a report.

Parameter	Type	Description
cronExpression	string	The Cron expression that defines a recursive schedule.
deliveryRequest	DeliveryRequest	Required. Contains the delivery information for the scheduled request via the DeliveryRequest structure.
endDate	dateTime	The end date of the schedule.
jobCalendar	string	The report formatting calendar to use for the scheduled requests. Valid values are: Gregorian, Arabic Hijrah, English Hijrah, Japanese Imperial, Tha Buddha, and ROC Official.
jobLocale	string	The locale to use for the scheduled requests. Example: fr-FR
jobTZ	string	The time zone to use for the scheduled requests.
notificationTo	string	E-mail addresses to send notifications to.
notifyWhenFailed	boolean	True indicates to send a notification when the job request fails.
notifyWhenSuccess	boolean	True indicates to send a notification when the job request succeeds.
notifyWhenWarning	boolean	True indicates to send a notification when the job completes with a warning.
repeatCount	integer	The number of times to repeat the schedule. For the recursive scheduling of a report, startDate must not be null, and repeatCount repeatInterval should be greater than 0 for any meaningful schedule. The endDate can be null.
repeatInterval	integer	The interval between two scheduled jobs in seconds.

Parameter	Type	Description
reportRequest	ReportRequest	Information about the request included via the ReportRequest structure, page 10-17.
saveDataOption	boolean	True indicates that the report data from the scheduled request run will be saved.
saveOutputOption	boolean	True indicates that the report output from the scheduled request run will be saved.
scheduleBurstingOption	boolean	True indicates that the scheduled requests will be burst.
schedulePublicOption	boolean	True indicates that the scheduled requests are to be made public.
startDate	dateTime	The date on which the schedule starts.
useUTF8Option	boolean	True indicates that the Use UTF8 option is enabled.
userJobName	string	The user-entered name for the scheduled job.

WebDAVDeliveryOption

```
<complexType name="WebDAVDeliveryOption">
  <sequence>
    <element name="deliveryAuthType" nillable="true" type="xsd:string"/>
    <element name="deliveryAuthTypeBasic" nillable="true"
type="xsd:string"/>
    <element name="deliveryAuthTypeDigest" nillable="true"
type="xsd:string"/>
    <element name="password" nillable="true" type="xsd:string"/>
    <element name="remoteFilePath" nillable="false" type="xsd:string"/>
    <element name="server" nillable="false" type="xsd:string"/>
    <element name="userName" nillable="true" type="xsd:string"/>
  </sequence>
</complexType>
```

Parameter	Type	Description
deliveryAuthType	string	Authentication type. Valid values are: None, Basic, Digest

Parameter	Type	Description
deliveryAuthTypeBasic	string	Element to describe the basic authentication type.
deliveryAuthTypeDigest	string	Element to describe the digest authentication type.
password	string	If a proxy server has been set up, the password required to access the proxy server.
remoteFilePath	string	The path to directory on the remote server to which to deliver the report file.
server	string	The webDAV server name. For example "myserver".
userName	string	If a proxy server has been set up, the user name required to access the proxy server.

Description: the options to set for Web-based Distributed Authoring and Versioning (WebDAV) delivery of a report.

This type is used in the following complex type structure: DeliveryRequest, page 10-4

BI Publisher Web Service Operations

The operations available in the Oracle BI Publisher Web services can be described by the following categories:

- Operations for validation of privileges
- Operations to get information about reports and the repository
- Operations to run reports
- Operations to manage scheduled reports and history of scheduled reports
- Operations to create and manage reports
- Operations to get information about the report server

This section describes the input message, output message, possible exception messages

and also provides sample code for each operation.

Operations for Validation of Privileges

Operations for validation of privileges include:

- validateLogin
- hasReportAccess

validateLogin

Use validateLogin to validate that a UserID and Password have the privilege to access the Oracle BI Publisher report server.

Input Message: validateLoginRequest

Parameter	Type	Nillable	Description
userID	string	no	The BI Publisher user name.
password	string	no	The password for the entered user name.

Return Message: validateLoginResponse

Parameter	Type	Nillable	Description
validateLoginReturn	boolean	n/a	True indicates the login ID has the requested privilege.

hasReportAccess

Use hasReportAccess to validate that a UserID and Password have the privilege to access a specific report.

Input Message: hasReportAccessRequest

Parameter	Type	Nillable	Description
reportAbsolutePath	string	no	The directory path to the report. Example: /HR Manager/Employee Reports/Employee Listing.xdo
userID	string	no	The BI Publisher user name.
password	string	no	The password for the entered user name.

Return Message: hasReportAccessResponse

Parameter	Type	Nillable	Description
hasReportAccessReturn	boolean	n/a	True indicates the user has the requested privilege.

Possible Exceptions

- InvalidParametersException

See Messages for Errors, page 10-47 for descriptions of the possible exceptions.

Operations to Get Information About Reports and the Repository

Operations to get information about reports and the repository include:

- getFolderContents
- getReportDefinition
- getReportParameters

getFolderContents

Use getFolderContents to get all of the items in a folder. This will return all the reports and folders contained in the specified folder. You can then use these items to determine what reports you might want to execute or what folders you may want to further search to identify a report.

Input Message: getFolderContentsRequest

Parameter	Type	Nillable	Description
folderAbsolutePath	string	no	The path to the folder for which you wish to retrieve the contents. For example: /HR Manager/Employee Reports/
userID	string	no	The BI Publisher user name.
password	string	no	The password for the entered user name.

Return Message: getFolderContentsResponse

Parameter	Type	Nillable	Description
getFolderContentsReturn	ItemData	n/a	See ItemData, page 10-7

Possible Exceptions

- InvalidParametersException
- AccessDeniedException
- OperationFailedException

See Messages for Errors, page 10-47 for descriptions of the possible exceptions.

Usage Notes

- If you do not pass a value for "folderAbsolutePath" (for example: <folderAbsolutePath xsi:type="xsd:string"></folderAbsolutePath>) the contents of "Shared Folders" and "My Folders" for the user entered will be returned.
- If you pass "$HOME" as "folderAbsolutePath" (for example: <folderAbsolutePath xsi:type="xsd:string">$HOME</folderAbsolutePath>), all contents under the Users folder will be returned.

getReportDefinition

Use getReportDefinition to get information about a report such as default template, output type and a listing of template IDs. Once you have a list of template IDs you may wish to generate a report with a template other than the default.

Input Message: getReportDefinitionRequest

Parameter	Type	Nillable	Description
reportAbsolutePath	string	no	The path to the report for which you wish to retrieve the report definition. For example: /HR Manager/Employee Reports/Employee Listing.xdo
userID	string	no	The BI Publisher user name.
password	string	no	The password for the entered user name.

Return Message: getReportDefinitionResponse

Parameter	Type	Nillable	Description
getReportDefinitionReturn	ReportDefinition	n/a	See ReportDefinition, page 10-15

Possible Exceptions

- InvalidParametersException
- AccessDeniedException
- OperationFailedException

See Messages for Errors, page 10-47 for descriptions of the possible exceptions.

getReportParameters

Use getReportParameters to get an array of report parameters and their default values. Once you have the list of parameters you can set parameter values before running or scheduling a report.

Input Message: getReportParametersRequest

Parameter	Type	Nillable	Description
reportRequest	ReportRequest	n/a	See ReportRequest, page 10-17

Parameter	Type	Nillable	Description
userID	string	no	The BI Publisher user name.
password	string	no	The password for the entered user name.

Return Message: getReportParametersResponse

Parameter	Type	Nillable	Description
getReportParametersReturn	ParamNameValue	n/a	See ParamNameValue, page 10-13

Possible Exceptions

- InvalidParametersException
- AccessDeniedException
- OperationFailedException

See Messages for Errors, page 10-47 for descriptions of the possible exceptions.

Operations to Run Reports

Operations to run reports include:

- runReport
- uploadReportDataChunk
- downloadReportDataChunk

runReport

Use runReport to execute a report and return a generated document. Note that the document is returned in the specified file format.

Input Message: runReportRequest

Parameter	Type	Nillable	Description
reportRequest	ReportRequest	n/a	See ReportRequest, page 10-17
userID	string	no	The BI Publisher user name.
password	string	no	The password for the entered user name.

Return Message: runReportResponse

Parameter	Type	Nillable	Description
runReportReturn	ReportResponse	n/a	See ReportResponse, page 10-18

Possible Exceptions

- InvalidParametersException
- AccessDeniedException
- OperationFailedException

See Messages for Errors, page 10-47 for descriptions of the possible exceptions.

uploadReportDataChunk

Input Message: uploadReportDataChunkRequest

Parameter	Type	Nillable	Description
fileID	string	no	In the first call, you do not need to provide the fileID, after the successful uploading of the first chunk of XML Data, it will return a fileID, for examle: filename. On your subsequent calls, you can supply the same fileID in order to append the subsequent data chunks to the same file.

Parameter	Type	Nillable	Description
reportDataChunk	base64binary	n/a	The XML data.

Return Message: uploadReportDataChunkResponse

Parameter	Type	Nillable	Description
uploadReportDataChunkReturn	string	no	The return response.

downloadReportDataChunk

Input Message: downloadReportDataChunkRequest

Parameter	Type	Nillable	Description
fileID	string	no	In the first call, you do not need to provide the fileID, after the successful uploading of the first chunk of XML Data, it will return a fileID, e.g. filename. On your subsequent calls, you can supply the same fileID in order to append the subsequent data chunks to the same file.
beginIdx	integer	N/A	The offset of
size	integer	N/A	The size of the file in kilobytes.

Return Message: downloadReportDataChunkResponse

Parameter	Type	Nillable	Description
downloadReportDataChunkReturn	ReportDataChunk	no	See ReportDataChunk , page 10-14

Operations to Manage Scheduled Reports and History of Scheduled Reports

Operations to manage scheduled reports and history of scheduled reports include:

- scheduleReport
- suspendScheduledReport
- resumeScheduledReport
- deleteScheduledReport
- getScheduledReportInfo
- getScheduledReportHistoryInfo
- deleteScheduledReportHistory
- getScheduledReportStatus

scheduleReport

Use scheduleReport to schedule a report for execution and delivery to either printer, fax, email, WebDAV, ftp or simply save in the report repository. Jobs can be scheduled to run immediately, once, or on a recurring pattern and can have an end date to stop the recurrence. This operation returns JobID upon successfully scheduling the report job.

Input Message: scheduleReportRequest

Parameter	Type	Nillable	Description
scheduleRequest	ScheduleRequest	n/a	See ScheduleRequest, page 10-19
userID	string	no	The BI Publisher user name.
password	string	no	The password for the entered user name.

Return Message: scheduleReportResponse

Parameter	Type	Nillable	Description
scheduleReportReturn	string	no	The return response.

Possible Exceptions

- InvalidParametersException
- AccessDeniedException
- OperationFailedException

See Messages for Errors, page 10-47 for descriptions of the possible exceptions.

suspendScheduledReport

Use suspendScheduledReport to suspend a scheduled report.

Input Message: suspendScheduledReportRequest

Parameter	Type	Nillable	Description
scheduledJobID	string	no	The job ID of the scheduled report.
userID	string	no	The BI Publisher user name.
password	string	no	The password for the entered user name.

Return Message: suspendScheduledReportResponse

Parameter	Type	Nillable	Description
suspendScheduledReportReturn	boolean	n/a	True indicates the report is suspended.

Possible Exceptions

- InvalidParametersException
- AccessDeniedException

- OperationFailedException

See Messages for Errors, page 10-47 for descriptions of the possible exceptions.

resumeScheduledReport

Use resumeScheduledReport to resume a suspended scheduled report.

Input Message: resumeScheduledReportRequest

Parameter	Type	Nillable	Description
scheduledJobID	string	no	The job ID of the scheduled job.
userID	string	no	The BI Publisher user name.
password	string	no	The password for the entered user name.

Return Message: resumeScheduledReportResponse

Parameter	Type	Nillable	Description
resumeScheduledReportReturn	boolean	n/a	True indicates the schedule has been resumed.

Possible Exceptions

- InvalidParametersException
- AccessDeniedException
- OperationFailedException

See Messages for Errors, page 10-47 for descriptions of the possible exceptions.

deleteScheduledReport

Use deleteScheduledReport to delete a scheduled report.

Input Message: deleteScheduledReportRequest

Parameter	Type	Nillable	Description
scheduledJobID	string	no	The Job ID of the scheduled report.
userID	string	no	The BI Publisher user name.
password	string	no	The password for the entered user name.

Return Message: deleteScheduledReportResponse

Parameter	Type	Nillable	Description
deleteScheduledReportReturn	boolean	n/a	True indicates that the schedule has been deleted.

Possible Exceptions

- InvalidParametersException
- AccessDeniedException
- OperationFailedException

See Messages for Errors, page 10-47 for descriptions of the possible exceptions.

getScheduledReportInfo

Use getScheduledReportInfo to get information about a scheduled report.

Input Message: getScheduledReportInfoRequest

Parameter	Type	Nillable	Description
scheduledJobID	string	no	The Job ID of the scheduled report.
userID	string	no	The BI Publisher user name.

Parameter	Type	Nillable	Description
password	string	no	The password for the entered user name.
viewByFilter	string	no	A selected "view by" category. Valid values are: "My History", "Public History", or "All".

Return Message: getScheduledReportInfoResponse

Parameter	Type	Nillable	Description
getScheduledReportInfoReturn	JobInfo	n/a	See JobInfo, page 10-10

Possible Exceptions

- InvalidParametersException
- AccessDeniedException
- OperationFailedException

See Messages for Errors, page 10-47 for descriptions of the possible exceptions.

getScheduledReportHistoryInfo

Use getScheduledReportHistoryInfo to get historical information about a scheduled report.

Input Message: getScheduledReportHistoryInfoRequest

Parameter	Type	Nillable	Description
scheduledJobID	string	no	The job ID of the scheduled report.
userID	string	no	The BI Publisher user name.
password	string	no	The password for the entered user name.

Parameter	Type	Nillable	Description
viewByFilter	string	no	A selected "view by" category. Valid values are: "My History", "Public History", or "All".
DownloadReport	boolean	no	True indicates to download the report.

Return Message: getScheduledReportHistoryInfoResponse

Parameter	Type	Nillable	Description
getScheduledReportHistoryInfoReturn	JobHistoryInfo	n/a	See JobHistoryInfo, page 10-8

Possible Exceptions

- InvalidParametersException
- AccessDeniedException
- OperationFailedException

See Messages for Errors, page 10-47 for descriptions of the possible exceptions.

deleteScheduledReportHistory

Use deleteScheduledReportHistory to delete the historical information about a scheduled report.

Input Message: deleteScheduledReportHistoryRequest

Parameter	Type	Nillable	Description
scheduledJobID	string	no	The job ID of the scheduled report.
userID	string	no	The BI Publisher user name.
password	string	no	The password for the entered user name.

Return Message: deleteScheduledReportHistoryResponse

Parameter	Type	Nillable	Description
deleteScheduledReportHistoryReturn	boolean	n/a	True indicates the report was deleted from history.

Possible Exceptions

- InvalidParametersException
- AccessDeniedException
- OperationFailedException

See Messages for Errors, page 10-47 for descriptions of the possible exceptions.

getScheduledReportStatus

Use getScheduledReportStatus to get status information about a scheduled report.

Input Message: getScheduledReportStatusRequest

Parameter	Type	Nillable	Description
scheduledJobID	string	no	The job ID of the scheduled report.
userID	string	no	The BI Publisher user name.
password	string	no	The password for the entered user name.

Return Message: getScheduledReportStatusResponse

Parameter	Type	Nillable	Description
getScheduledReportStatusReturn	JobStatus	N/A	See JobStatus, page 10-12

Possible Exceptions

- InvalidParametersException

- AccessDeniedException
- OperationFailedException

See Messages for Errors, page 10-47 for descriptions of the possible exceptions.

Operations to Create and Manage Reports

Operations to create and manage reports include:

- createReport
- deleteReport
- updateReportDefinition
- createReportFolder
- uploadTemplateForReport
- removeTemplateForReport
- removeTemplateForReport
- uploadReport
- downloadReport

createReport

Use createReport to create a new report in the BI Publisher Reports Repository with an empty report definition.

Input Message: createReportRequest

Parameter	Type	Nillable	Description
reportName	string	no	The report name to create with the suffix ".xdo". For example, "myreport.xdo".
reportAbsolutePathURL	string	no	The path to the folder in which to place the created report. For example: xmlp/Reports/financials
templateFileName	string	yes	The file name of the template to apply.

Parameter	Type	Nillable	Description
templateData	base64Binary	yes	If you are including the template directly, the template data.
XLIFFFileName	string	yes	The file name of the XLIFF file to apply.
XLIFFData	base64binary	yes	If you are including the XLIFF file directly, the XLIFF data.
userID	string	no	The BI Publisher user name.
password	string	no	The password for the entered user name.

Return Message: createReportResponse

Parameter	Type	Nillable	Description
createReportReturn	string	no	Returns the report path to the newly created report. Note that the syntax for the report path is: <folder_path>/<report_name>/report_name.xdo. For example: xmlp/Reports/financials/my report/myreport.xdo where xmlp/Reports/financials is the folder in which the report is stored and "my report" is the report container, not a folder.

Possible Exceptions

- InvalidParametersException
- AccessDeniedException
- OperationFailedException

See Messages for Errors, page 10-47 for descriptions of the possible exceptions.

deleteReport

This API deletes a report folder containing the report definition file and any other supporting report files. Either provide the reportAbsolutePath ending with ".xdo", or provide the report folder name.

Input Message: deleteReportRequest

Parameter	Type	Nillable	Description
reportAbsolutePath	string	no	The absolute path to either the report folder or the report .xdo file. For example: C:\OraHome\xmlp\XMLP\Reports\Executive\Revenue by Region\Revenue by Region.xdo
userID	string	no	A valid BI Publisher Administrator user name.
password	string	no	The password for the BI Publisher Administrator user name.

Return Message: deleteReportResponse

Parameter	Type	Nillable	Description
deleteReportReturn	boolean	n/a	True indicates the report was deleted.

Possible Exceptions

- InvalidParametersException
- AccessDeniedException
- OperationFailedException

See Messages for Errors, page 10-47 for descriptions of the possible exceptions.

updateReportDefinition

The updateReportDefinition updates attributes of the report definition file (.xdo) and then writes the file back to the BI Publisher Reports Repository.

Input Message: updateReportDefinitionRequest

Parameter	Type	Nillable	Description
reportAbsPath	string	no	The path to the report file. For example: C:\OraHome\xmlp\XMLP\Reports\Executive\Revenue by Region\Revenue by Region.xdo
newReportDefn	ReportDefinition	no	see ReportDefinition, page 10-15
userID	string	no	A valid BI Publisher Administrator user name.
password	string	no	The password for the BI Publisher Administrator user name.

Return Message: updateReportDefinitionResponse

Parameter	Type	Nillable	Description
updateReportDefinitionReturn	boolean	no	True indicates the report definition was updated.

Possible Exceptions

- InvalidParametersException
- AccessDeniedException
- OperationFailedException

See Messages for Errors, page 10-47 for descriptions of the possible exceptions.

createReportFolder

Use createReportFolder to create a new folder in the BI Publisher Reports Repository.

Input Message: createReportFolderRequest

Parameter	Type	Nillable	Description
folderAbsolutePath	string	no	The absolute path to the folder the folder you wish to create in the BI Publisher repository.
userID	string	no	The BI Publisher user name.
password	string	no	The password for the entered user name.

Return Message: createReportFolderResponse

Parameter	Type	Nillable	Description
createReportFolderReturn	string	no	True indicates the report folder was created.

Possible Exceptions

- InvalidParametersException
- AccessDeniedException
- OperationFailedException

See Messages for Errors, page 10-47 for descriptions of the possible exceptions.

uploadTemplateForReport

Use uploadTemplateForReport to upload a template to the BI Publisher Reports Repository.

Input Message: uploadTemplateForReportRequest

Parameter	Type	Nillable	Description
reportAbsolutePath	string	no	The path to the report for which you want to upload a template.

Parameter	Type	Nillable	Description
templateFileName	string	no	The file name of the template.
templateData	base64Binary	no	The template data.
userID	string	no	The BI Publisher user name.
password	string	no	The password for the entered user name.

Return Message: uploadTemplateForReportResponse

Parameter	Type	Nillable	Description
uploadTemplateForReport Return	boolean	n/a	True indicates the template was uploaded successfully.

Possible Exceptions

- InvalidParametersException
- AccessDeniedException
- OperationFailedException

See Messages for Errors, page 10-47 for descriptions of the possible exceptions.

removeTemplateForReport

Use removeTemplateForReport to delete a template from the BI Publisher Reports Repository.

Input Message: removeTemplateForReportRequest

Parameter	Type	Nillable	Description
reportAbsolutePath	string	no	The path to the report directory.
templateFileName	string	no	The name of the template file to delete.

Parameter	Type	Nillable	Description
userID	string	no	The BI Publisher user name.
password	string	no	The password for the entered user name.

Return Message: removeTemplateForReportResponse

Parameter	Type	Nillable	Description
removeTemplateForReport Return	boolean	n/a	True indicates the template was removed.

Possible Exceptions

- InvalidParametersException
- AccessDeniedException
- OperationFailedException

See Messages for Errors, page 10-47 for descriptions of the possible exceptions.

uploadReport

Use uploadReport to upload a complete report definition file to the BI Publisher Reports Repository. The uploaded report must be in zipped file format. The API will first create a new report using reportAbsolutePathURL: /Report Folder/reportName.xdo It will then unzip the zipped data to extract the reportName.xdo file and then unzip any other data included with the file.

Input Message: uploadReportRequest

Parameter	Type	Nillable	Description
reportName	string	no	The report name, for example: reportname.xdo
reportAbsolutePathURL	string	no	The path to the report, for example: /Report Folder/report name

Parameter	Type	Nillable	Description
reportZippedData	base64binary	no	The report data in zipped format.
userID	string	no	The BI Publisher user name.
password	string	no	The password for the entered user name.

Return Message: uploadReportResponse

Parameter	Type	Nillable	Description
uploadReportReturn	string	no	The response.

Possible Exceptions

- InvalidParametersException
- AccessDeniedException
- OperationFailedException

See Messages for Errors, page 10-47 for descriptions of the possible exceptions.

downloadReport

This API downloads the BI Publisher report definition directory into a zipped format. This includes the report .XDO file and any other files associated with this report.

Input Message: downloadReportRequest

Parameter	Type	Nillable	Description
reportAbsolutePath	string	no	The path to the report to download.
userID	string	no	The BI Publisher user name.
password	string	no	The password for the entered user name.

Return Message: downloadReportResponse

Parameter	Type	Nillable	Description
downloadReportReturn	base64binary	no	The zip file of the downloaded report folder.

Possible Exceptions

- InvalidParametersException
- AccessDeniedException
- OperationFailedException

See Messages for Errors, page 10-47 for descriptions of the possible exceptions.

Operations about the Report Server

Operations about the report server include:

- getSecurityModel
- getHTTPSessionInterval

getSecurityModel

This service will return the security model of the BI Publisher server. The supported settings are:

Input Message: getSecurityModelRequest

Parameter	Type	Nillable	Description
none			

Return Message: getSecurityModelResponse

Parameter	Type	Nillable	Description
getSecurityModelReturn	string	no	Possible return values are: • XDO - XDO_SECURITY_MODEL • FND - FND_SECURITY_MODEL • LDAP - LDAP_SECURITY_MODEL • BI_SERVER - BI_SERVER_SECURITY_MODEL • ORACLE_DB - ORACLE_DB_SECURITY_MODEL • HYPERION_CSS - CSS_SECURITY_MODEL

getBIPHTTPSessionInterval

Returns the BI Publisher server HTTP session interval in minutes.

Input Message: getBIPHTTPSessionIntervalRequest

Parameter	Type	Nillable	Description
none			

Return Message: getBIPHTTPSessionIntervaResponse

Parameter	Type	Nillable	Description
getBIPHTTPSessionIntervalReturn	int	no	The BI Publisher server HTTP session interval in minutes.

Messages for Errors

One of the following messages may be returned to any of the operations if there is an error that occurs in the execution of the operation.

OperationFailedException

```
<wsdl:message name="OperationFailedException">
   <wsdl:part name="fault" type="impl:OperationFailedException"/>
</wsdl:message>
```

AccessDeniedException

```
<wsdl:message name="AccessDeniedException">
   <wsdl:part name="fault" type="impl:AccessDeniedException"/>
</wsdl:message>
```

InvalidParametersException

```
<wsdl:message name="InvalidParametersException">
  <wsdl:part name="fault" type="impl:InvalidParametersException"/>
</wsdl:message>
```

Debugging Web Service Applications

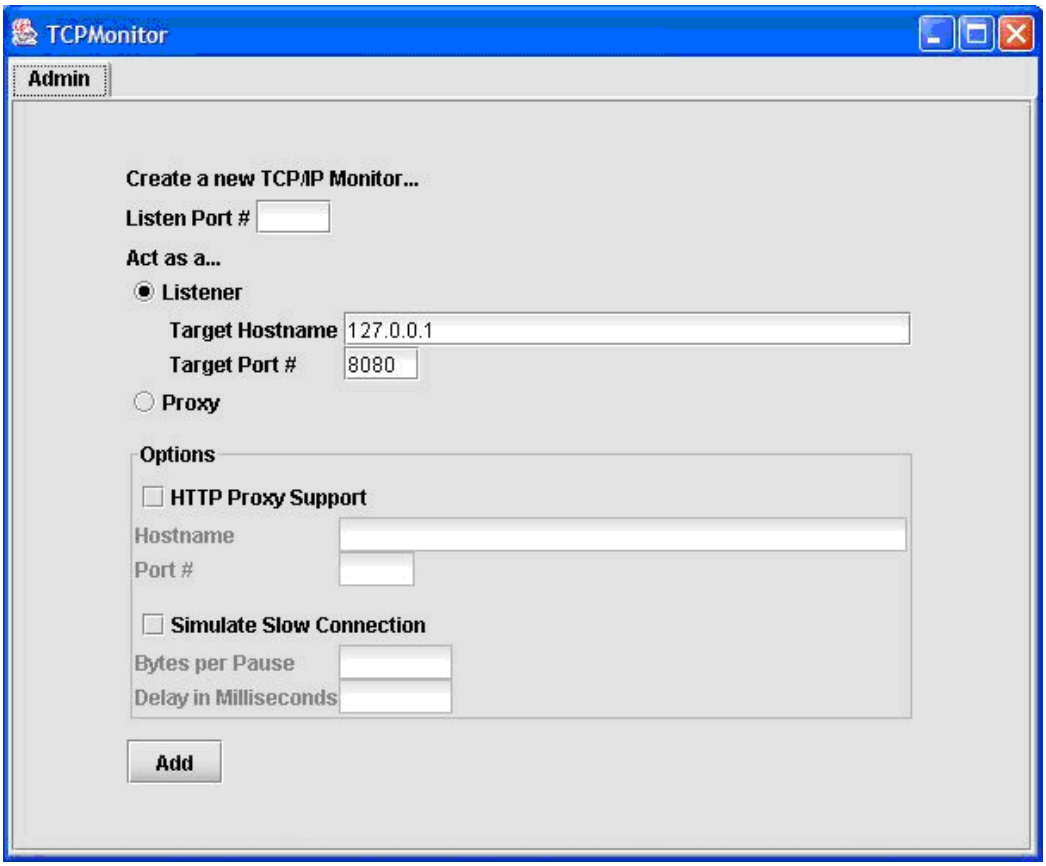
As a Web service developer you may need to see the SOAP request messages being used to invoke Web services along with the SOAP response to those request messages. Oracle BI Publisher bundles the Apache Axis SOAP Monitor utility to monitor the SOAP message flow without requiring any special configuration or restarting of the server.

The client sample code sends a SOAP request to listening port 8888, and then takes advantage of Axis TCP monitor to route SOAP request to port 8080, where the PublicReportService is running. This will be a useful debugging tool for the Web service developer.

To start Axis TCP Monitor (tcpmon), from the command line enter the following:

```
% java org.apache.axis.utils.tcpmon
```

This launches the TCP monitor window. A sample of the window without any of the optional arguments is shown in the following figure:



To use the program, select a local port that tcpmon will monitor for incoming connections, a target host where it will forward such connections, and the port number on the target machine which should be "tunneled" to. Then click Add. Another tab will then display for your new tunneled connection. From there, you will see the SOAP Request and Response message, which facilitates debugging.

To run tcpmonitor, enter the following:

```
set CLASSPATH=.
set CLASSPATH=%CLASSPATH%;%WEB_HOME%\PORTAL\WEB-INF\lib\axis.jar
set TCPMON_PORT=8081
if not %1.==. TCPMON_PORT=%1
java org.apache.axis.utils.tcpmon %TCPMON_PORT% %ADMINSERVER_HOSTNAME%
%ADMINSERVER_PORT%
```

11

Using the BI Publisher APIs

This chapter covers the following topics:

- Introduction
- BI Publisher Core APIs
- Prerequisites
- PDF Form Processing Engine
- RTF Processor Engine
- FO Processor Engine
- PDF Document Merger
- PDF Book Binder Processor
- Document Processor Engine
- Bursting Engine
- BI Publisher Properties
- Advanced Barcode Font Formatting Implementation

Introduction

This chapter is aimed at developers who wish to create programs or applications that interact with BI Publisher through its application programming interface. This information is meant to be used in conjunction with the Javadocs available with your installation files.

This section assumes the reader is familiar with Java programming, XML, and XSL technologies.

BI Publisher Core APIs

BI Publisher is made up of the following core API components:

- PDF Form Processing Engine

  Merges a PDF template with XML data (and optional metadata) to produce PDF document output.

- RTF Processor

  Converts an RTF template to XSL in preparation for input to the FO Engine.

- FO Engine

  Merges XSL and XML to produce any of the following output formats: Excel (HTML), PDF, RTF, or HTML.

- PDF Document Merger

  Provides optional postprocessing of PDF files to merge documents, add page numbering, and set watermarks.

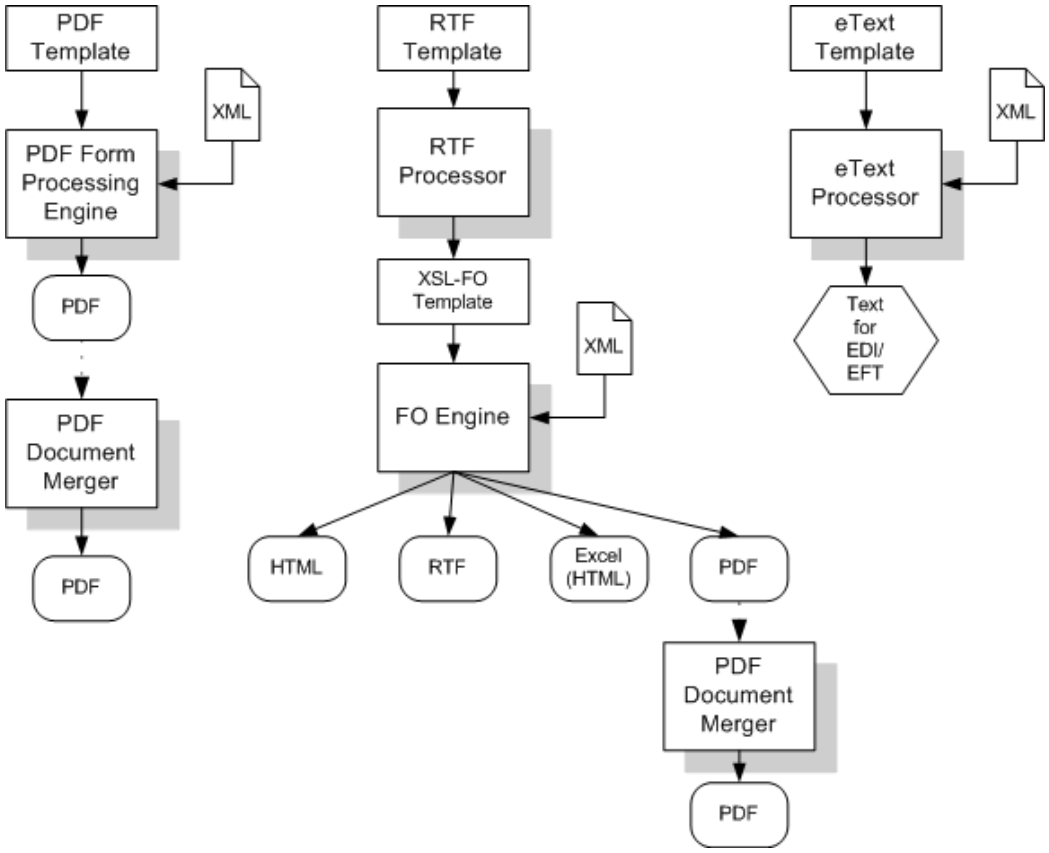
- eText Processor

  Converts RTF eText templates to XSL and merges the XSL with XML to produce text output for EDI and EFT transmissions.

- Document Processor (XML APIs)

  Provides batch processing functionality to access a single API or multiple APIs by passing a single XML file to specify template names, data sources, languages, output type, output names, and destinations.

The following diagram illustrates the template type and output type options for each core processing engine:



Prerequisites

In order to use the BI Publisher APIs you need the following JAR files in your class path:

- xdocore.jar - the core BI Publisher library
- aolj.jar - although this is an Oracle E-Business Suite library, it is required for standalone implementations as well
- i18nAPI_v3.jar - the i18n library used for localization functions
- xdoparser.jar - the scalable XML parser and XSLT 2.0 engine
- xmlparserv2-904.jar - the main XML parser/XSLT engine
- bipres.jar - a charting library
- bicmn.jar - a charting library
- jewt4.jar - a charting support library

- share.jar - a charting support library
- collections.jar - you only need this if you are working with the delivery APIs or bursting engine.

Obtaining the Libraries

If you are using Oracle JDeveloper, then the charting and XML Parser libraries will already be available to you. However, it is recommended that you create a directory with all of the required JAR files to use as a custom library in your project. This will help prevent unexpected errors after deployment.

You have the following options for obtaining the required libraries:

- Locate the xmlpserver.ear file in your installation. A typical path would be:

  C:\Ora_Home\xmlp\xmlpserver.ear

  Unpack this file to find the xmlpserver.war file. Unpack this file to find the WEB-INF directory. The lib directory under WEB-INF contains the JAR files. A sample path once both files are unpacked would be:
  C:\OraHome\xmlp\xmlpserver\xmlpserver\WEB-INF\lib

- Download the and install the Template Builder for Microsoft Word Add-in. The JAR files are packaged with the Template Builder in the jlib library under the install directory.

  A sample path to the jlib would be:

  C:\Program Files\Oracle\BI Publisher\BI Publisher Desktop\Template Builder for Word\jlib

  Note that the Template Builder does include the aolj.jar. You must use the versioninfo.jar instead.

PDF Form Processing Engine

The PDF Form Processing Engine creates a PDF document by merging a PDF template with an XML data file. This can be done using file names, streams, or an XML data string.

As input to the PDF Processing Engine you can optionally include an XML-based Template MetaInfo (.xtm) file. This is a supplemental template to define the placement of overflow data.

The FO Processing Engine also includes utilities to provide information about your PDF template. You can:

- Retrieve a list of field names from a PDF template

- Generate the XFDF data from the PDF template
- Convert XML data into XFDF using XSLT

Merging a PDF Template with XML Data

XML data can be merged with a PDF template to produce a PDF output document in three ways:

- Using input/output file names
- Using input/output streams
- Using an input XML data string

You can optionally include a metadata XML file to describe the placement of overflow data in your template.

Merging XML Data with a PDF Template Using Input/Output File Names

Input:

- Template file name (String)
- XML file name (String)
- Metadata XML file name (String)

Output:

- PDF file name (String)

Example

```
import oracle.apps.xdo.template.FormProcessor;
.
.
    FormProcessor fProcessor = new FormProcessor();

    fProcessor.setTemplate(args[0]);  // Input File (PDF) name
    fProcessor.setData(args[1]);     // Input XML data file name
    fProcessor.setOutput(args[2]);   // Output File (PDF) name
    fProcessor.setMetaInfo(args[3]);   // Metadata XML File name  You
can omit this setting when you do not use Metadata.

    fProcessor.process();
```

Merging XML Data with a PDF Template Using Input/Output Streams

Input:

- PDF Template (Input Stream)

- XML Data (Input Stream)
- Metadata XML Data (Input Stream)

Output:

- PDF (Output Stream)

Example

```
import java.io.*;
import oracle.apps.xdo.template.FormProcessor;
.
.
.
  FormProcessor fProcessor = new FormProcessor();

  FileInputStream fIs = new FileInputStream(originalFilePath); // Input
File
  FileInputStream fIs2 = new FileInputStream(dataFilePath); // Input
Data
  FileInputStream fIs3 = new FileInputStream(metaData); // Metadata XML
Data
  FileOutputStream fOs = new FileOutputStream(newFilePath); // Output
File

  fProcessor.setTemplate(fIs);
  fProcessor.setData(fIs2);   // Input Data
  fProcessor.setOutput(fOs);
  fProcessor.setMetaInfo(fIs3);
  fProcessor.process();

  fIs.close();
  fOs.close();
```

Merging an XML Data String with a PDF Template

Input:

- Template file name (String)
- XML data (String)
- Metadata XML file name (String)

Output:

- PDF file name (String)

Example

```
import oracle.apps.xdo.template.FormProcessor;
.
.
.
FormProcessor fProcessor = new FormProcessor();

fProcessor.setTemplate(originalFilePath);     // Input File (PDF) name
fProcessor.setDataString(xmlContents);       // Input XML string
fProcessor.setOutput(newFilePath);           // Output File (PDF) name
fProcessor.setMetaInfo(metaXml);        // Metadata XML File name    You
can omit this setting when you do not use Metadata.
fProcessor.process();
```

Retrieving a List of Field Names

Use the FormProcessor.getFieldNames() API to retrieve the field names from a PDF template. The API returns the field names into an Enumeration object.

Input:

- PDF Template

Output:

- Enumeration Object

Example

```
import java.util.Enumeration;
import oracle.apps.xdo.template.FormProcessor;
.
.
.
FormProcessor fProcessor = new FormProcessor();
fProcessor.setTemplate(filePath);        // Input File (PDF) name
Enumeration enum = fProcessor.getFieldNames();
while(enum.hasMoreElements()) {
  String formName = (String)enum.nextElement();
  System.out.println("name : " + formName + " , value : " +
fProcessor.getFieldValue(formName));
}
```

Generating XFDF Data

XML Forms Data Format (XFDF) is a format for representing forms data and annotations in a PDF document. XFDF is the XML version of Forms Data Format (FDF), a simplified version of PDF for representing forms data and annotations. Form fields in a PDF document include edit boxes, buttons, and radio buttons.

Use this class to generate XFDF data. When you create an instance of this class, an internal XFDF tree is initialized. Use append() methods to append a FIELD element to the XFDF tree by passing a String name-value pair. You can append data as many times as you want.

This class also allows you to append XML data by calling appendXML() methods. Note that you must set the appropriate XSL stylesheet by calling setStyleSheet() method before calling appendXML() methods. You can append XML data as many times as you want.

You can retrieve the internal XFDF document at any time by calling one of the following methods: toString(), toReader(), toInputStream(), or toXMLDocument().

The following is a sample of XFDF data:

Example

```
<?xml version="1.0" encoding="UTF-8"?>
<xfdf xmlns="http://ns.adobe.com/xfdf/" xml:space="preserve">
<fields>
 <field name="TITLE">
  <value>Purchase Order</value>
  </field>
 <field name="SUPPLIER_TITLE">
   <value>Supplie</value>
 </field>
  ...
 </fields>
```

The following code example shows how the API can be used:

Example

```
import oracle.apps.xdo.template.FormProcessor;
import oracle.apps.xdo.template.pdf.xfdf.XFDFObject;
.
.
.
FormProcessor fProcessor = new FormProcessor();
fProcessor.setTemplate(filePath);      // Input File (PDF) name
XFDFObject xfdfObject = new XFDFObject(fProcessor.getFieldInfo());
System.out.println(xfdfObject.toString());
```

Converting XML Data into XFDF Format Using XSLT

Use an XSL stylesheet to convert standard XML to the XFDF format. Following is an example of the conversion of sample XML data to XFDF:

Assume your starting XML has a ROWSET/ROW format as follows:

```
<ROWSET>
     <ROW num="0">
       <SUPPLIER>Supplier</SUPPLIER>
       <SUPPLIERNUMBER>Supplier Number</SUPPLIERNUMBER>
       <CURRCODE>Currency</CURRCODE>
     </ROW>
...
</ROWSET>
```

From this XML you want to generate the following XFDF format:

```
<fields>
  <field name="SUPPLIER1">
    <value>Supplier</value>
  </field>
  <field name="SUPPLIERNUMBER1">
    <value>Supplier Number</value>
  </field>
  <field name="CURRCODE1">
    <value>Currency</value>
  </field>
...
</fields>
```

The following XSLT will carry out the transformation:

```
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
<fields>
<xsl:apply-templates/>
</fields>
</xsl:template>
   <!-- Count how many ROWs(rows) are in the source XML file. -->
   <xsl:variable name="cnt" select="count(//row|//ROW)" />
   <!-- Try to match ROW (or row) element.
   <xsl:template match="ROW/*|row/*">
      <field>
         <!-- Set "name" attribute in "field" element. -->
         <xsl:attribute name="name">
            <!-- Set the name of the current element (column name)as a
value of the current name attribute. -->
            <xsl:value-of select="name(.)" />
            <!-- Add the number at the end of the name attribute value
if more than 1 rows found in the source XML file.-->
            <xsl:if test="$cnt > 1">
               <xsl:number count="ROW|row" level="single" format="1"/>
            </xsl:if>
         </xsl:attribute>
         <value>
            <!--Set the text data set in the current column data as a
text of the "value" element. -->
            <xsl:value-of select="." />
         </value>
      </field>
   </xsl:template>
</xsl:stylesheet>
```

You can then use the XFDFObject to convert XML to the XFDF format using an XSLT as follows:

Example

```
import java.io.*;
import oracle.apps.xdo.template.pdf.xfdf.XFDFObject;
.
.
.
XFDFObject xfdfObject  = new XFDFObject();

xfdfObject .setStylesheet(new BufferedInputStream(new
FileInputStream(xslPath)));   // XSL file name
xfdfObject .appendXML( new File(xmlPath1));   // XML data file name
xfdfObject .appendXML( new File(xmlPath2));   // XML data file name

System.out.print(xfdfObject .toString());
```

RTF Processor Engine

Generating XSL

The RTF processor engine takes an RTF template as input. The processor parses the template and creates an XSL-FO template. This can then be passed along with a data source (XML file) to the FO Engine to produce PDF, HTML, RTF, or Excel (HTML) output.

Use either input/output file names or input/output streams as shown in the following examples:

Generating XSL with Input/Output File Names

Input:

- RTF file name (String)

Output:

- XSL file name (String)

Example

```
import oracle.apps.xdo.template.FOProcessor;
.
.
.
  public static void main(String[] args)
  {
RTFProcessor rtfProcessor = new RTFProcessor(args[0]); //input template
rtfProcessor.setOutput(args[1]);  // output file
rtfProcessor.process();
    System.exit(0);
  }
```

Generating XSL with Input/Output Stream

Input:

- RTF (InputStream)

Output:

- XSL (OutputStream)

Example

```
import oracle.apps.xdo.template.FOProcessor;
.
.
.
  public static void main(String[] args)
  {
    FileInputStream   fIs  = new FileInputStream(args[0]);  //input
template
    FileOutputStream  fOs  = new FileOutputStream(args[1]); // output

    RTFProcessor rtfProcessor = new RTFProcessor(fIs);
    rtfProcessor.setOutput(fOs);
    rtfProcessor.process();
    // Closes inputStreams outputStream
    System.exit(0);
  }
```

FO Processor Engine

Generating Output from an XML File and an XSL File

The FO Processor Engine is BI Publisher's implementation of the W3C XSL-FO standard. It does not represent a complete implementation of every XSL-FO component. For a list of supported XSL-FO elements, see Supported XSL-FO Elements, *Oracle Business Intelligence Publisher Report Designer's Guide*.

The FO Processor can generate output in PDF, RTF, HTML, or Excel (HTML) from either of the following two inputs:

- Template (XSL) and Data (XML) combination
- FO object

Both input types can be passed as file names, streams, or in an array. Set the output format by setting the setOutputFormat method to one of the following:

- FORMAT_EXCEL
- FORMAT_HTML
- FORMAT_PDF
- FORMAT_RTF

An XSL-FO utility is also provided that creates XSL-FO from the following inputs:

- XSL file and XML file
- Two XML files and two XSL files
- Two XSL-FO files (merge)

The FO object output from the XSL-FO utility can then be used as input to the FO processor.

Major Features of the FO Processor

Bidirectional Text

BI Publisher utilizes the Unicode BiDi algorithm for BiDi layout. Based on specific values for the properties writing-mode, direction, and unicode bidi, the FO Processor supports the BiDi layout.

The writing-mode property defines how word order is supported in lines and order of lines in text. That is: right-to-left, top-to-bottom or left-to-right, top-to-bottom. The direction property determines how a string of text will be written: that is, in a specific direction, such as right-to-left or left-to-right. The unicode bidi controls and manages override behavior.

Font Fallback Mechanism

The FO Processor supports a two-level font fallback mechanism. This mechanism provides control over what default fonts to use when a specified font or glyph is not found. BI Publisher provides appropriate default fallback fonts automatically without requiring any configuration. BI Publisher also supports user-defined configuration files that specify the default fonts to use. For glyph fallback, the default mechanism will only replace the glyph and not the entire string.

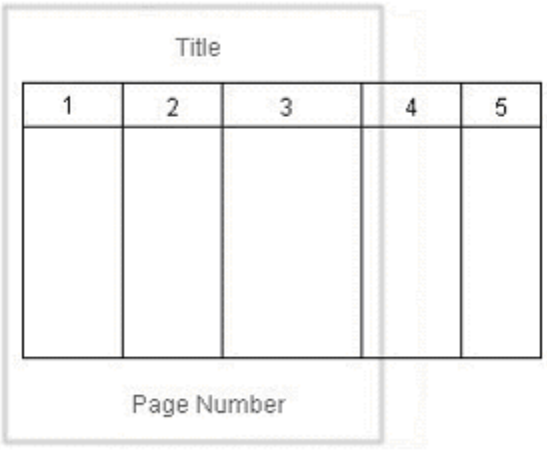
Variable Header and Footer

For headers and footers that require more space than what is defined in the template, the FO Processor extends the regions and reduces the body region by the difference between the value of the page header and footer and the value of the body region margin.

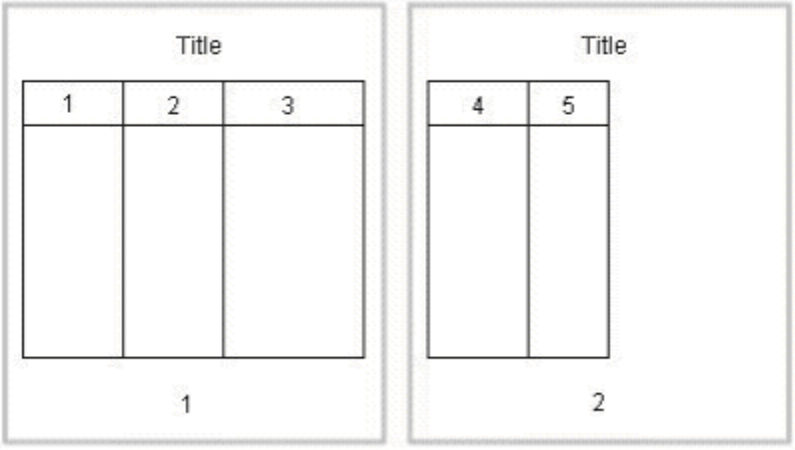
Horizontal Table Break

This feature supports a "Z style" of horizontal table break. The horizontal table break is not sensitive to column span, so that if the column-spanned cells exceed the page (or area width), the FO Processor splits it and does not apply any intelligent formatting to the split cell.

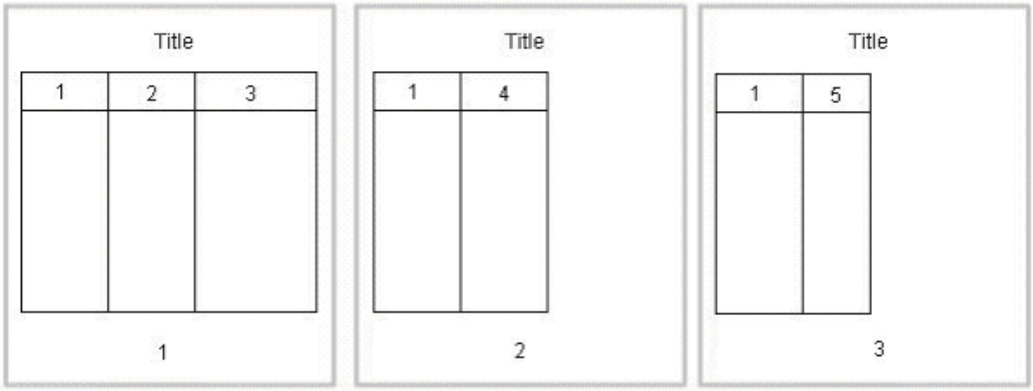
The following figure shows a table that is too wide to display on one page:



The following figure shows one option of how the horizontal table break will handle the wide table. In this example, a horizontal table break is inserted after the third column.



The following figure shows another option. The table breaks after the third column, but includes the first column with each new page.



Generating Output Using File Names

The following example shows how to use the FO Processor to create an output file using file names.

Input:

- XML file name (String)
- XSL file name (String)

Output:

- Output file name (String)

Example

```
import oracle.apps.xdo.template.FOProcessor;
.
.
.
  public static void main(String[] args)
  {

    FOProcessor processor = new FOProcessor();
    processor.setData(args[0]);     // set XML input file
    processor.setTemplate(args[1]); // set XSL input file
    processor.setOutput(args[2]);  //set output file
    processor.setOutputFormat(FOProcessor.FORMAT_PDF);
    // Start processing
    try
    {
       processor.generate();
    }
    catch (XDOException e)
    {
       e.printStackTrace();
       System.exit(1);
    }

    System.exit(0);
  }
```

Generating Output Using Streams

The processor can also be used with input/output streams as shown in the following example:

Input:

- XML data (InputStream)
- XSL data (InputStream)

Output:

- Output stream (OutputStream)

Example

```
import java.io.InputStream;
import java.io.OutputStream;
import oracle.apps.xdo.template.FOProcessor;
.
.
.
  public void runFOProcessor(InputStream xmlInputStream,
                             InputStream xslInputStream,
                             OutputStream pdfOutputStream)
  {

    FOProcessor processor = new FOProcessor();
    processor.setData(xmlInputStream);
    processor.setTemplate(xslInputStream);
    processor.setOutput(pdfOutputStream);
    // Set output format (for PDF generation)
    processor.setOutputFormat(FOProcessor.FORMAT_PDF);
    // Start processing
    try
    {
       processor.generate();
    }
    catch (XDOException e)
    {
       e.printStackTrace();
       System.exit(1);
    }

    System.exit(0);

  }
```

Generating Output from an Array of XSL Templates and XML Data

An array of data and template combinations can be processed to generate a single output file from the multiple inputs. The number of input data sources must match the number of templates that are to be applied to the data. For example, an input of File1.xml, File2.xml, File3.xml and File1.xsl, File2.xsl, and File3.xsl will produce a single File1_File2_File3.pdf.

Input:

- XML data (Array)
- XSL data (template) (Array)

Output:

- File Name (String)

Example

```
import java.io.InputStream;
import java.io.OutputStream;
import oracle.apps.xdo.template.FOProcessor;
.
.
.
   public static void main(String[] args)
  {

    String[] xmlInput = {"first.xml", "second.xml", "third.xml"};
    String[] xslInput = {"first.xsl", "second.xsl", "third.xsl"};


    FOProcessor processor = new FOProcessor();
    processor.setData(xmlInput);
    processor.setTemplate(xslInput);

    processor.setOutput("/tmp/output.pdf);          //set (PDF) output
file
    processor.setOutputFormat(FOProcessor.FORMAT_PDF);
processor.process();
    // Start processing
    try
    {
       processor.generate();
    }
    catch (XDOException e)
    {
       e.printStackTrace();
       System.exit(1);
    }

  }
```

Using the XSL-FO Utility

Use the XSL-FO Utility to create an XSL-FO output file from input XML and XSL files, or to merge two XSL-FO files. Output from this utility can be used to generate your final output. See Generating Output from an XSL-FO file, page 11-19.

Creating XSL-FO from an XML File and an XSL File

Input:

- XML file

- XSL file

Output:

- XSL-FO (InputStream)

Example

```
import oracle.apps.xdo.template.fo.util.FOUtility;
.
.
.
  public static void main(String[] args)
  {
    InputStream foStream;

    // creates XSL-FO InputStream from XML(arg[0])
    // and XSL(arg[1]) filepath String
    foStream = FOUtility.createFO(args[0], args[1]);
    if (mergedFOStream == null)
    {
      System.out.println("Merge failed.");
      System.exit(1);
    }

    System.exit(0);
  }
```

Creating XSL-FO from Two XML Files and Two XSL files

Input:

- XML File 1
- XML File 2
- XSL File 1
- XSL File 2

Output:

- XSL-FO (InputStream)

Example

```
import oracle.apps.xdo.template.fo.util.FOUtility;
.
.
.
  public static void main(String[] args)
  {
    InputStream firstFOStream, secondFOStream, mergedFOStream;
    InputStream[] input = InputStream[2];

    // creates XSL-FO from arguments
    firstFOStream = FOUtility.createFO(args[0], args[1]);

    // creates another XSL-FO from arguments
    secondFOStream = FOUtility.createFO(args[2], args[3]);

    // set each InputStream into the InputStream Array
    Array.set(input, 0, firstFOStream);
    Array.set(input, 1, secondFOStream);

    // merges two XSL-FOs
    mergedFOStream = FOUtility.mergeFOs(input);

    if (mergedFOStream == null)
    {
      System.out.println("Merge failed.");
      System.exit(1);
    }
    System.exit(0);
  }
```

Merging Two XSL-FO Files

Input:

- Two XSL-FO file names (Array)

Output:

- One XSL-FO (InputStream)

Example

```
import oracle.apps.xdo.template.fo.util.FOUtility;
.
.
.
  public static void main(String[] args)
  {
    InputStream mergedFOStream;

    // creates Array
    String[] input = {args[0], args[1]};

    // merges two FO files
    mergedFOStream = FOUtility.mergeFOs(input);
    if (mergedFOStream == null)
    {
      System.out.println("Merge failed.");
      System.exit(1);
    }
    System.exit(0);
  }
```

Generating Output from an FO file

The FO Processor can also be used to process an FO object to generate your final output. An FO object is the result of the application of an XSL-FO stylesheet to XML data. These objects can be generated from a third party application and fed as input to the FO Processor.

The processor is called using a similar method to those already described, but a template is not required as the formatting instructions are contained in the FO.

Generating Output Using File Names

Input:

- FO file name (String)

Output:

- PDF file name (String)

Example

```
import oracle.apps.xdo.template.FOProcessor;
.
.
.
  public static void main(String[] args) {

    FOProcessor processor = new FOProcessor();
    processor.setData(args[0]);     // set XSL-FO input file
    processor.setTemplate((String)null);
    processor.setOutput(args[2]);  //set (PDF) output file
    processor.setOutputFormat(FOProcessor.FORMAT_PDF);
    // Start processing
    try
    {
       processor.generate();
    }
    catch (XDOException e)
    {
       e.printStackTrace();
       System.exit(1);
    }

    System.exit(0);
  }
```

Generating Output Using Streams

Input:

- FO data (InputStream)

Output:

- Output (OutputStream)

Example

```
import java.io.InputStream;
import java.io.OutputStream;
import oracle.apps.xdo.template.FOProcessor;
.
.
.
  public void runFOProcessor(InputStream xmlfoInputStream,
                             OutputStream pdfOutputStream)
  {

    FOProcessor processor = new FOProcessor();
    processor.setData(xmlfoInputStream);
    processor.setTemplate((String)null);

    processor.setOutput(pdfOutputStream);
    // Set output format (for PDF generation)
    processor.setOutputFormat(FOProcessor.FORMAT_PDF);
    // Start processing
    try
    {
       processor.generate();
    }
    catch (XDOException e)
    {
       e.printStackTrace();
       System.exit(1);
    }
  }
```

Generating Output with an Array of FO Data

Pass multiple FO inputs as an array to generate a single output file. A template is not required, therefore set the members of the template array to null, as shown in the example.

Input:

- FO data (Array)

Output:

- Output File Name (String)

Example

```
import java.lang.reflect.Array;
import oracle.apps.xdo.template.FOProcessor;
.
.
.
  public static void main(String[] args)
  {

    String[] xmlInput = {"first.fo", "second.fo", "third.fo"};
    String[] xslInput = {null, null, null};   // null needs for xsl-fo
input

    FOProcessor processor = new FOProcessor();
    processor.setData(xmlInput);
    processor.setTemplate(xslInput);

    processor.setOutput("/tmp/output.pdf);          //set (PDF) output
file
    processor.setOutputFormat(FOProcessor.FORMAT_PDF);
processor.process();
    // Start processing
    try
    {
       processor.generate();
    }
    catch (XDOException e)
    {
       e.printStackTrace();
       System.exit(1);
    }

  }
```

PDF Document Merger

The PDF Document Merger class provides a set of utilities to manipulate PDF documents. Using these utilities, you can merge documents, add page numbering, set backgrounds, and add watermarks.

Merging PDF Documents

Many business documents are composed of several individual documents that need to be merged into a single final document. The PDFDocMerger class supports the merging of multiple documents to create a single PDF document. This can then be manipulated further to add page numbering, watermarks, or other background images.

Merging with Input/Output File Names

The following code demonstrates how to merge (concatenate) two PDF documents using physical files to generate a single output document.

Input:

- PDF_1 file name (String)
- PDF_2 file name (String)

Output:

- PDF file name (String)

Example

```
import java.io.*;
import oracle.apps.xdo.common.pdf.util.PDFDocMerger;
.
.
.
  public static void main(String[] args)
  {
    try
    {
      // Last argument is PDF file name for output
      int inputNumbers = args.length - 1;

      // Initialize inputStreams
      FileInputStream[] inputStreams = new
FileInputStream[inputNumbers];
      inputStreams[0] = new FileInputStream(args[0]);
      inputStreams[1] = new FileInputStream(args[1]);

      // Initialize outputStream
      FileOutputStream outputStream = new FileOutputStream(args[2]);

      // Initialize PDFDocMerger
      PDFDocMerger docMerger = new PDFDocMerger(inputStreams,
outputStream);

      // Merge PDF Documents and generates new PDF Document
      docMerger.mergePDFDocs();
      docMerger = null;

      // Closes inputStreams and outputStream
    }
    catch(Exception exc)
    {
      exc.printStackTrace();
    }
  }
```

Merging with Input/Output Streams

Input:

- PDF Documents (InputStream Array)

Output:

- PDF Document (OutputStream)

Example

```
import java.io.*;
import oracle.apps.xdo.common.pdf.util.PDFDocMerger;
.
.
.
  public boolean mergeDocs(InputStream[] inputStreams, OutputStream
outputStream)
  {
    try
    {
      // Initialize PDFDocMerger
      PDFDocMerger docMerger = new PDFDocMerger(inputStreams,
outputStream);

      // Merge PDF Documents and generates new PDF Document
      docMerger.mergePDFDocs();
      docMerger = null;

      return true;
    }
    catch(Exception exc)
    {
      exc.printStackTrace();
      return false;
    }
  }
```

Merging with Background to Place Page Numbering

The following code demonstrates how to merge two PDF documents using input streams to generate a single merged output stream.

To add page numbers:

1. Create a background PDF template document that includes a PDF form field in the position that you would like the page number to appear on the final output PDF document.

2. Name the form field `@pagenum@`.

3. Enter the number in the field from which to start the page numbering. If you do not enter a value in the field, the start page number defaults to 1.

Input:

- PDF Documents (InputStream Array)

- Background PDF Document (InputStream)

Output:

- PDF Document (OutputStream)

Example

```
import java.io.*;
import oracle.apps.xdo.common.pdf.util.PDFDocMerger;
.
.
.
 public static boolean mergeDocs(InputStream[] inputStreams, InputStream
backgroundStream, OutputStream outputStream)

 {
    try
    {
      // Initialize PDFDocMerger
      PDFDocMerger docMerger = new PDFDocMerger(inputStreams,
outputStream);

      // Set Background
      docMerger.setBackground(backgroundStream);

      // Merge PDF Documents and generates new PDF Document
      docMerger.mergePDFDocs();
      docMerger = null;

      return true;
    }
    catch(Exception exc)
    {
      exc.printStackTrace();
      return false;
    }
  }
```

Adding Page Numbers to Merged Documents

The FO Processor supports page numbering natively through the XSL-FO templates, but if you are merging multiple documents you must use this class to number the complete document from beginning to end.

The following code example places page numbers in a specific point on the page, formats the numbers, and sets the start value using the following methods:

- setPageNumberCoordinates (x, y) - sets the x and y coordinates for the page number position. The following example sets the coordinates to 300, 20.
- setPageNumberFontInfo (font name, size) - sets the font and size for the page number. If you do not call this method, the default "Helvetica", size 8 is used. The following example sets the font to "Courier", size 8.
- setPageNumberValue (n, n) - sets the start number and the page on which to begin numbering. If you do not call this method, the default values 1, 1 are used.

Input:

- PDF Documents (InputStream Arrary)

Output:

- PDF Document (OutputStream)

Example

```
import java.io.*;
import oracle.apps.xdo.common.pdf.util.PDFDocMerger;
.
.
.
  public boolean mergeDocs(InputStream[] inputStreams, OutputStream
outputStream)
  {
    try
    {
      // Initialize PDFDocMerger
      PDFDocMerger docMerger = new PDFDocMerger(inputStreams,
outputStream);

      // Calls several methods to specify Page Number

      // Calling setPageNumberCoordinates() method is necessary to set
Page Numbering
      // Please refer to javadoc for more information
      docMerger.setPageNumberCoordinates(300, 20);


      // If this method is not called, then the default font"(Helvetica,
8)" is used.
      docMerger.setPageNumberFontInfo("Courier", 8);

      // If this method is not called, then the default initial value
"(1, 1)" is used.
      docMerger.setPageNumberValue(1, 1);

      // Merge PDF Documents and generates new PDF Document
      docMerger.mergePDFDocs();
      docMerger = null;

      return true;
    }
    catch(Exception exc)
    {
      exc.printStackTrace();
      return false;
    }
  }
```

Setting a Text or Image Watermark

Some documents that are in a draft phase require that a watermark indicating "DRAFT"' be displayed throughout the document. Other documents might require a background image on the document. The following code sample shows how to use the PDFDocMerger class to set a watermark.

Setting a Text Watermark

Use the SetTextDefaultWatermark( ) method to set a text watermark with the following attributes:

- Text angle (in degrees): 55
- Color: light gray (0.9, 0.9, 0.9)
- Font: Helvetica
- Font Size: 100
- The start position is calculated based on the length of the text

Alternatively, use the SetTextWatermark( ) method to set each attribute separately. Use the SetTextWatermark() method as follows:

- SetTextWatermark ("Watermark Text", x, y) - declare the watermark text, and set the x and y coordinates of the start position. In the following example, the watermark text is "Draft" and the coordinates are 200f, 200f.
- setTextWatermarkAngle (n) - sets the angle of the watermark text. If this method is not called, 0 will be used.
- setTextWatermarkColor (R, G, B) - sets the RGB color. If this method is not called, light gray (0.9, 0.9, 0.9) will be used.
- setTextWatermarkFont ("font name", font size) - sets the font and size. If you do not call this method, Helvetica, 100 will be used.

The following example shows how to set these properties and then call the PDFDocMerger.

Input:

- PDF Documents (InputStream)

Output:

- PDF Document (OutputStream)

Example

```
import java.io.*;
import oracle.apps.xdo.common.pdf.util.PDFDocMerger;
.
.
.
  public boolean mergeDocs(InputStream inputStreams, OutputStream
outputStream)
  {
    try
    {
      // Initialize PDFDocMerger
      PDFDocMerger docMerger = new PDFDocMerger(inputStreams,
outputStream);

      // You can use setTextDefaultWatermark() without these detailed
setting
      docMerger.setTextWatermark("DRAFT", 200f, 200f); //set text and
place
      docMerger.setTextWatermarkAngle(80);                //set angle
      docMerger.setTextWatermarkColor(1.0f, 0.3f, 0.5f);  // set RGB
Color

      // Merge PDF Documents and generates new PDF Document
      docMerger.mergePDFDocs();
      docMerger = null;

      return true;
    }
    catch(Exception exc)
    {
      exc.printStackTrace();
      return false;
    }
  }
```

Setting Image Watermark

An image watermark can be set to cover the entire background of a document, or just to cover a specific area (for example, to display a logo). Specify the placement and size of the image using rectangular coordinates as follows:

```
float[ ] rct = {LowerLeft X, LowerLeft Y, UpperRight X,
UpperRight Y}
```

For example:

```
float[ ] rct = {100f, 100f, 200f, 200f}
```

The image will be sized to fit the rectangular area defined.

To use the actual image size, without sizing it, define the LowerLeft X and LowerLeft Y positions to define the placement and specify the UpperRight X and UpperRight Y coordinates as -1f. For example:

```
float[ ] rct = {100f, 100f, -1f, -1f}
```

Input:

- PDF Documents (InputStream)
- Image File (InputStream)

Output:

- PDF Document (OutputStream)

Example

```
import java.io.*;
import oracle.apps.xdo.common.pdf.util.PDFDocMerger;
.
.
.
  public boolean mergeDocs(InputStream inputStreams, OutputStream
outputStream, String imageFilePath)
  {
    try
    {
      // Initialize PDFDocMerger
      PDFDocMerger docMerger = new PDFDocMerger(inputStreams,
outputStream);

      FileInputStream wmStream = new FileInputStream(imageFilePath);
      float[] rct = {100f, 100f, -1f, -1f};
      pdfMerger.setImageWatermark(wmStream, rct);

      // Merge PDF Documents and generates new PDF Document
      docMerger.mergePDFDocs();
      docMerger = null;

      // Closes inputStreams
      return true;
    }
    catch(Exception exc)
    {
      exc.printStackTrace();
      return false;
    }
  }
```

PDF Book Binder Processor

The PDFBookBinder processor is useful for the merging of multiple PDF documents into a single document consisting of a hierarchy of chapters, sections, and subsections and a table of contents for the document. The processor also generates PDF style " bookmarks"; the outline structure is determined by the chapter and section hierarchy. The processor is extremely powerful allowing you complete control over the combined document.

Usage

The table of contents formatting and style is created through the use of an RTF template created in Microsoft Word. The chapters are passed into the program as separate PDF

files (one chapter, section, or subsection corresponds to one PDF file). Templates may also be specified at the chapter level for insertion of dynamic or static content, page numbering, and placement of hyperlinks within the document.

The templates can be in RTF or PDF format. RTF templates are more flexible by allowing you to leverage BI Publisher's support for dynamic content. PDF templates are much less flexible, making it difficult to achieve desirable effects such as the reflow of text areas when inserting page numbers and other types of dynamic content.

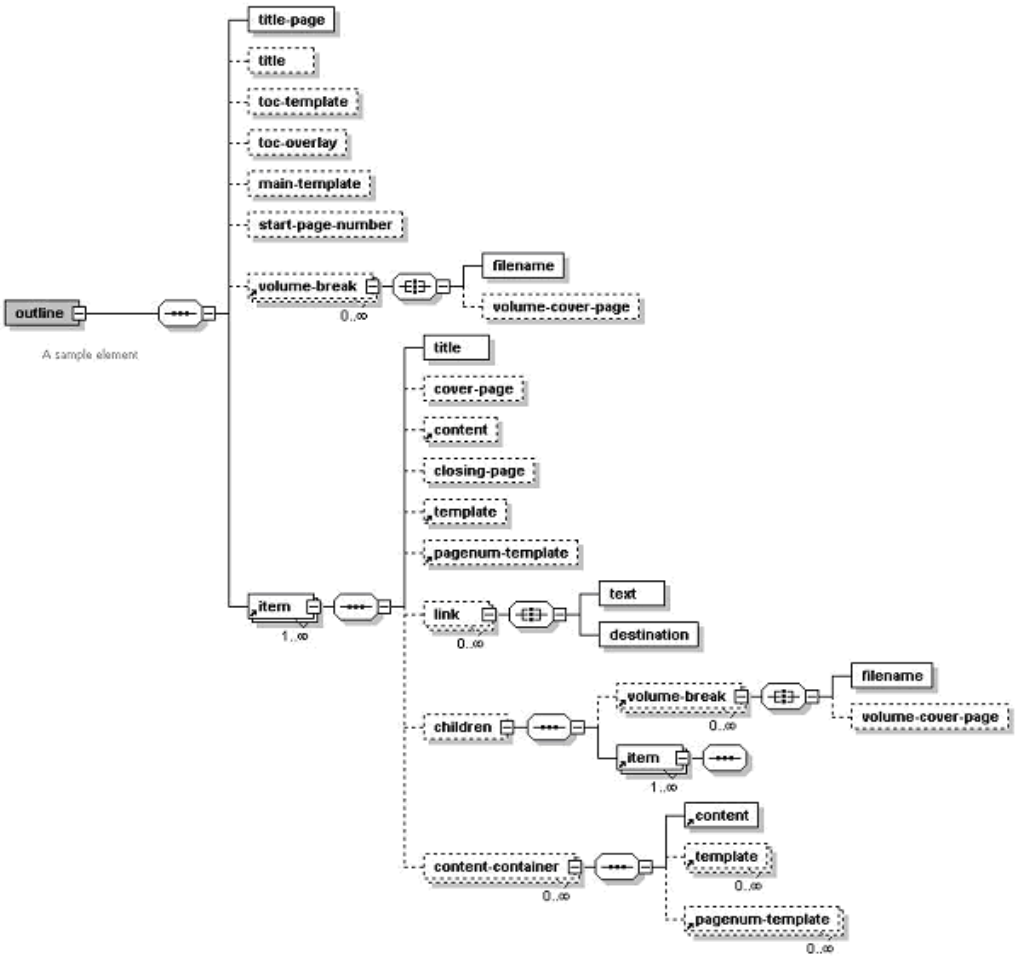
The templates can be rotated (at right angles) or be made transparent. A PDF template can also be specified at the book level, enabling the ability to specify global page numbering, or other content such as backgrounds and watermarks. A title page can also be passed in as a parameter, as well as cover and closing pages for each chapter or section.

XML Control File

The structure of the book's chapters, sections, and subsections is represented as XML and passed in as a command line parameter; or it can also be passed in at the API level. All of the chapter and section files, as well as all the templates files and their respective parameters, are specified inside this XML structure. Therefore, the only two required parameters are an XML file and a PDF output file.

You can also specify volume breaks inside the book structure. Specifying volume breaks will split the content up into separate output files for easier file and printer management.

The structure of the XML control file is represented in the following diagram:



To specify template and content file locations in your XML structure, you can specify a path relative to your local file system or you can specify a URL referring to the template or content location. Secure HTTP protocol is supported, as well as specially recognized BI Publisher protocols, such as:

- "xdo://" - used to specify BI Publisher Template Manager-specific data.
- "fnd://" - used to specify data located in the FND_LOBS table.
- "blob://" - used for specifying data in any user-defined BLOB table.

  The format for the "blob://" protocol is:

  ```
  blob://[table_name].[blob_column_name]/[pk_datatype]:[pk_name]=[pk_value]/../../..
  ```

Command Line Options

Following is an example of the command line usage:

```
java oracle.apps.template.pdf.book.PDFBookBinder [-debug <true or
false>] [-tmp <temp dir>] -xml <input xml> -pdf <output pdf>
```

where

`-xml <file>` is the file name of the input XML file containing the table of contents XML structure.

`-pdf <file>` is the final generated PDF output file.

`-tmp <directory>` is the temporary directory for better memory management. (This is optional, if not specified, the system environment variable "`java.io.tmpdir`" will be used.)

`-log <file>` sets the output log file (optional, default is `System.out`).

`-debug <true or false>` turns debugging off or on.

API Method Call

The following is an example of an API method call:

```
String xmlInputPath = "c:\\tmp\\toc.xml";
String pdfOutputPath = "c:\\tmp\\final_book.pdf";
PDFBookBinder bookBinder = new PDFBookBinder(xmlInputPath,
 pdfOutputPath);

bookBinder.setConfig(new Properties());
bookBinder.process();
```

Document Processor Engine

The Document Processor Engine provides batch processing functionality to access a single API or multiple APIs by passing a single XML instance document to specify template names, data sources, languages, output type, output names, and destinations.

This solution enables batch printing with BI Publisher, in which a single XML document can be used to define a set of invoices for customers, including the preferred output format and delivery channel for those customers. The XML format is very flexible allowing multiple documents to be created or a single master document.

This section:

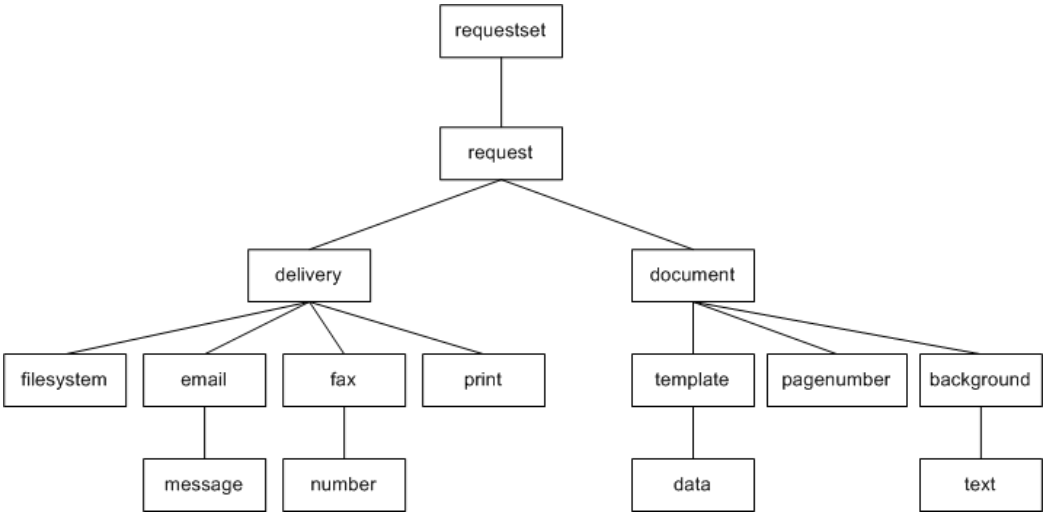
- Describes the hierarchy and elements of the Document Processor XML file
- Provides sample XML files to demonstrate specific processing options
- Provides example code to invoke the processors

Hierarchy and Elements of the Document Processor XML File

The Document Processor XML file has the following element hierarchy:

```
Requestset
   request
      delivery
         filesystem
         print
         fax
            number
         email
            message
      document
         background
            text
         pagenumber
         template
            data
```

This hierarchy is displayed in the following illustration:



The following table describes each of the elements:

Element	Attributes	Description
`requestset`	`xmlns` `version`	Root element must contain [`xmlns:xapi="http://xmlns.oracle.com/oxp/xapi/"`] block The `version` is not required, but defaults to "1.0".
`request`	N/A	Element that contains the data and template processing definitions.

Element	Attributes	Description
delivery	N/A	Defines where the generated output is sent.
document	output-type	Specify one output that can have several template elements. The output-type attribute is optional. Valid values are: pdf (Default) rtf html excel text
filesystem	output	Specify this element to save the output to the file system. Define the directory path in the output attribute.
print	• printer • server-alias	The print element can occur multiple times under delivery to print one document to several printers. Specify the printer attribute as a URI, such as: "ipp://myprintserver:631/printers/printername"
fax	• server • server-alias	Specify a URI in the server attribute, for example: "ipp://myfaxserver1:631/printers/myfaxmachine"
number		The number element can occur multiple times to list multiple fax numbers. Each element occurrence must contain only one number.

Element	Attributes	Description
email	• server • port • from • reply-to • server-alias	Specify the outgoing mail server (SMTP) in the server attribute. Specify the mail server port in the port attribute.
message	• to • cc • bcc • attachment • subject	The message element can be placed several times under the email element. You can specify character data in the message element. You can specify multiple e-mail addresses in the to, cc and bcc attributes separated by a comma. The attachment value is either true or false (default). If attachment is true, then a generated document will be attached when the e-mail is sent. The subject attribute is optional.
background	where	If the background text is required on a specific page, then set the where value to the page numbers required. The page index starts at 1. The default value is 0, which places the background on all pages.

Element	Attributes	Description
text	• title • default	Specify the watermark text in the title value. A default value of "yes" automatically draws the watermark with forward slash type. The default value is yes.
pagenumber	• initial-page-index • initial-value • x-pos • y-pos	The initial-page-index default value is 0. The initial-value default value is 1. "Helvetica" is used for the page number font. The x-pos provides lower left x position. The y-pos provides lower left y position.
template	• locale • location • type	Contains template information. Valid values for the type attribute are pdf rtf xsl-fo etext The default value is "pdf".

Element	Attributes	Description
data	location	Define the location attribute to specify the location of the data, or attach the actual XML data with subelements. The default value of location is "inline". It the location points to either an XML file or a URL, then the data should contain an XML declaration with the proper encoding. If the location attribute is not specified, the data element should contain the subelements for the actual data. This must not include an XML declaration.

XML File Samples

Following are sample XML files that show:

- Simple XML shape
- Defining two data sets
- Defining multiple templates and data
- Retrieving templates over HTTP
- Retrieving data over HTTP
- Generating more than one output
- Defining page numbers

Simple XML sample

The following sample is a simple example that shows the definition of one template (template1.pdf) and one data source (data1) to produce one output file (outfile.pdf) delivered to the file system:

Example

```
<?xml version="1.0" encoding="UTF-8" ?>
     <xapi:requestset xmlns:xapi="http://xmlns.oracle.com/oxp/xapi">
       <xapi:request>
         <xapi:delivery>
           <xapi:filesystem output="d:\tmp\outfile.pdf" />
         </xapi:delivery>
         <xapi:document output-type="pdf">
           <xapi:template type="pdf" location="d:\mywork\template1.pdf">
             <xapi:data>
               <field1>data1</field1>
             </xapi:data>
           </xapi:template>
         </xapi:document>
       </xapi:request>
    </xapi:requestset>
```

Defining two data sets

The following example shows how to define two data sources to merge with one template to produce one output file delivered to the file system:

Example

```
<?xml version="1.0" encoding="UTF-8"?>
<xapi:requestset xmlns:xapi="http://xmlns.oracle.com/oxp/xapi">
  <xapi:request>
    <xapi:delivery>
      <xapi:filesystem output="d:\tmp\outfile.pdf"/>
    </xapi:delivery>

    <xapi:document output-type="pdf">
      <xapi:template type="pdf"
                  location="d:\mywork\template1.pdf">
        <xapi:data>
          <field1>The first set of data</field1>
        </xapi:data>
        <xapi:data>
          <field1>The second set of data</field1>
        </xapi:data>
      </xapi:template>
    </xapi:document>
  </xapi:request>
</xapi:requestset>
```

Defining multiple templates and data

The following example builds on the previous examples by applying two data sources to one template and two data sources to a second template, and then merging the two into a single output file. Note that when merging documents, the `output-type` must be "pdf".

Example

```
<?xml version="1.0" encoding="UTF-8"?>
<xapi:requestset xmlns:xapi="http://xmlns.oracle.com/oxp/xapi">
  <xapi:request>
    <xapi:delivery>
      <xapi:filesystem output="d:\tmp\outfile3.pdf"/>
    </xapi:delivery>

    <xapi:document output-type="pdf">
      <xapi:template type="pdf"
                  location="d:\mywork\template1.pdf">
        <xapi:data>
          <field1>The first set of data</field1>
        </xapi:data>
        <xapi:data>
          <field1>The second set of data</field1>
        </xapi:data>
      </xapi:template>

      <xapi:template type="pdf"
                  location="d:\mywork\template2.pdf">
        <xapi:data>
          <field1>The third set of data</field1>
        </xapi:data>
        <xapi:data>
          <field1>The fourth set of data</field1>
        </xapi:data>
      </xapi:template>
    </xapi:document>
  </xapi:request>
</xapi:requestset>
```

Retrieving templates over HTTP

This sample is identical to the previous example, except in this case the two templates are retrieved over HTTP:

```
<?xml version="1.0" encoding="UTF-8"?>
<xapi:requestset xmlns:xapi="http://xmlns.oracle.com/oxp/xapi">
  <xapi:request>
    <xapi:delivery>
      <xapi:filesystem output="d:\temp\out4.pdf"/>
    </xapi:delivery>

    <xapi:document output-type="pdf">
      <xapi:template type="pdf"
       location="http://your.server:9999/templates/template1.pdf">
        <xapi:data>
          <field1>The first page data</field1>
        </xapi:data>
        <xapi:data>
          <field1>The second page data</field1>
        </xapi:data>
      </xapi:template>
      <xapi:template type="pdf"
       location="http://your.server:9999/templates/template2.pdf">
        <xapi:data>
          <field1>The third page data</field1>
        </xapi:data>
        <xapi:data>
          <field1>The fourth page data</field1>
        </xapi:data>
      </xapi:template>
    </xapi:document>
  </xapi:request>
</xapi:requestset>
```

Retrieving data over HTTP

This sample builds on the previous example and shows one template with two data sources, all retrieved via HTTP; and a second template retrieved via HTTP with its two data sources embedded in the XML:

Example

```
<?xml version="1.0" encoding="UTF-8"?>
<xapi:requestset xmlns:xapi="http://xmlns.oracle.com/oxp/xapi">
 <xapi:request>
  <xapi:delivery>
   <xapi:filesystem output="d:\temp\out5.pdf"/>
    </xapi:delivery>

    <xapi:document output-type="pdf">
     <xapi:template type="pdf"
      location="http://your.server:9999/templates/template1.pdf">
      <xapi:data location="http://your.server:9999/data/data_1.xml"/>
      <xapi:data location="http://your.server:9999/data/data_2.xml"/>
      </xapi:template>

      <xapi:template type="pdf"
       location="http://your.server:9999/templates/template2.pdf">
        <xapi:data>
          <field1>The third page data</field1>
        </xapi:data>
        <xapi:data>
          <field1>The fourth page data</field1>
        </xapi:data>
      </xapi:template>
    </xapi:document>
  </xapi:request>
</xapi:requestset>
```

Generating more than one output

The following sample shows the generation of two outputs: `out_1.pdf` and `out_2.pdf`. Note that a `request` element is defined for each output.

Example

```
<?xml version="1.0" encoding="UTF-8"?>
<xapi:requestset xmlns:xapi="http://xmlns.oracle.com/oxp/xapi">
  <xapi:request>
    <xapi:delivery>
      <xapi:filesystem output="d:\temp\out_1.pdf"/>
    </xapi:delivery>
    <xapi:document output-type="pdf">
      <xapi:template type="pdf"
                  location="d:\mywork\template1.pdf">
        <xapi:data>
          <field1>The first set of data</field1>
        </xapi:data>
        <xapi:data>
          <field1>The second set of data</field1>
        </xapi:data>
      </xapi:template>
    </xapi:document>
  </xapi:request>

  <xapi:request>
    <xapi:delivery>
      <xapi:filesystem output="d:\temp\out_2.pdf"/>
    </xapi:delivery>
    <xapi:document output-type="pdf">
      <xapi:template type="pdf"
                  location="d:mywork\template2.pdf">
        <xapi:data>
          <field1>The third set of data</field1>
        </xapi:data>
        <xapi:data>
          <field1>The fourth set of data</field1>
        </xapi:data>
      </xapi:template>
    </xapi:document>
  </xapi:request>

</xapi:requestset>
```

Defining page numbers

The following sample shows the use of the `pagenumber` element to define page numbers on a PDF output document. The first document that is generated will begin with an initial page number value of 1. The second output document will begin with an initial page number value of 3. The `pagenumber` element can reside anywhere within the `document` element tags.

Note that page numbering that is applied using the `pagenumber` element will not replace page numbers that are defined in the template.

```
<?xml version="1.0" encoding="UTF-8"?>
<xapi:requestset xmlns:xapi="http://xmlns.oracle.com/oxp/xapi">
  <xapi:request>
    <xapi:delivery>
      <xapi:filesystem output="d:\temp\out7-1.pdf"/>
    </xapi:delivery>
    <xapi:document output-type="pdf">
      <xapi:pagenumber initial-value="1" initial-page-index="1"
       x-pos="300" y-pos="20" />
      <xapi:template type="pdf"
        location="d:\mywork\template1.pdf">
        <xapi:data>
          <field1>The first page data</field1>
        </xapi:data>
        <xapi:data>
          <field1>The second page data</field1>
        </xapi:data>
      </xapi:template>
    </xapi:document>
  </xapi:request>

  <xapi:request>
    <xapi:delivery>
      <xapi:filesystem output="d:\temp\out7-2.pdf"/>
    </xapi:delivery>
    <xapi:document output-type="pdf">
      <xapi:template type="pdf"
           location="d:\mywork\template2.pdf">
        <xapi:data>
          <field1>The third page data</field1>
        </xapi:data>
        <xapi:data>
          <field1>The fourth page data</field1>
        </xapi:data>
      </xapi:template>
      <xapi:pagenumber initial-value="3" initial-page-index="1"
       x-pos="300" y-pos="20" />
    </xapi:document>
  </xapi:request>

</xapi:requestset>
```

Invoke Processors

The following code samples show how to invoke the document processor engine using an input file name and an input stream.

Invoke Processors with Input File Name

Input:

- Data file name (String)
- Directory for Temporary Files (String)

Example

```
import oracle.apps.xdo.batch.DocumentProcessor;
.
.
.
  public static void main(String[] args)
  {
.
.
.
    try
    {
      // dataFile --- File path of the Document Processor XML
      // tempDir  --- Temporary Directory path
      DocumentProcessor  docProcessor = new DocumentProcessor(dataFile,
tempDir);
      docProcessor.process();
    }
    catch(Exception e)
    {
 e.printStackTrace();
       System.exit(1);
    }
    System.exit(0);
  }
```

Invoke Processors with InputStream

Input:

- Data file (InputStream)
- Directory for Temporary Files (String)

Example

```
import oracle.apps.xdo.batch.DocumentProcessor;
import java.io.InputStream;
.
.
.
  public static void main(String[] args)
  {
.
.
.
    try
    {
      // dataFile --- File path of the Document Processor XML
      // tempDir  --- Temporary Directory path
      FileInputStream fIs = new FileInputStream(dataFile);

      DocumentProcessor  docProcessor = new DocumentProcessor(fIs,
tempDir);
      docProcessor.process();
      fIs.close();
    }
    catch(Exception e)
    {
 e.printStackTrace();
       System.exit(1);
    }
    System.exit(0);
  }
```

Bursting Engine

BI Publisher's bursting engine accepts a data stream and splits it based on multiple criteria, generates output based on a template, then delivers the individual documents through the delivery channel of choice. The engine provides a flexible range of possibilities for document generation and delivery. Example implementations include:

- Invoice generation and delivery based on customer-specific layouts and delivery preference
- Financial reporting to generate a master report of all cost centers, bursting out individual cost center reports to the appropriate manager
- Generation of payslips to all employees based on one extract and delivered via e-mail

Usage

The bursting engine is an extension of the Document Processor Engine, page 11-32 and has its own method be called to invoke it. The Document Processor XML structure has been extended to handle the new components required by the bursting engine. It

supports all of the delivery functionality that the Document Processor supports using the same format. It accepts the XML data to be burst and a control file that takes the Document Processor XML format (see Hierarchy and Elements of the Document Processor XML File, page 11-32).

Control File

The control file takes the same format as the Document Processor XML, page 11-32 with a few extensions:

- Use the attribute `select` under the `request` element to specify the element in the XML data that you wish to burst on.

  Example

  ```
  <xapi:request select="/EMPLOYEES/EMPLOYEE">
  ```

- Use the attribute `id` under the lowest level of the delivery structure (for example, for the delivery element `email`, the `id` attribute belongs to the `message` element. This assigns an ID to the delivery method to be referenced later in the XML file.

  Example

  ```
  <xapi:message id="123" to="jo.smith@company.com"
  ```

- Use the `delivery` attribute under the `document` element. This assigns the delivery method for the generated document as defined in the `id` attribute for the `delivery` element. You can specify multiple delivery channels separated by a comma.

  Example

  ```
  <xapi:document output-type="pdf" delivery="123">
  ```

- Use the `filter` attribute on the `template` element. Use this to apply a layout template based on a filter on your XML data.

  Example

  ```
  <xapi:template type="rtf" location="/usr/tmp/empGeneric.rtf">
  <xapi:template type="rtf" location="usr\tmp\empDet.rtf"
  filter=".//EMPLOYEE[ENAME='SMITH']"/>
  ```

  This will apply the empDet template only to those employees with the name "SMITH". All other employees will have the empGeneric template applied. This filter can use any XPATH expression to determine the rules for the template application.

Dynamic Delivery Destination

You can reference elements in the data to derive certain delivery attributes, such as an e-mail address or fax number. Enter the value for the attribute using the following form:

```
${ELEMENT}
```

where `ELEMENT` is the element name from the XML data that holds the value for the attribute.

For example:

```
<xapi:message id="123" to="${EMAIL}"/>
```

At runtime the value of the `to` attribute will be set to the value of the EMAIL element from the input XML file.

You can also set the value of an attribute by passing a parameter to API in a Properties object.

Dynamic Delivery Content

You can reference information in the XML data to be put into the delivery content. This takes the same format described above (that is, `${ELEMENT}`).

For example, suppose you wanted to burst a document to employees via e-mail and personalize the e-mail by using the employee's name in the subject line. Assuming the employee's name is held in an element called ENAME, you could use ${ENAME} to reference the employee's name in the control file as follows:

```
subject="Employee Details for ${ENAME}"
```

Sample Control File

The following sample control file shows an example control file to split data based on an EMPLOYEE element and send an e-mail to each employee with their own data. The sample file is annotated.

```
<?xml version="1.0" encoding="UTF-8"?>
<xapi:requestset xmlns:xapi="http://xmlns.oracle.com/oxp/xapi">
<xapi:request select="/EMPLOYEES/EMPLOYEE"><! -  This sets the bursting
element i.e., EMPLOYEE  - >
    <xapi:delivery>
      <xapi:email server="rgmamersmtp.oraclecorp.com" port="25"
       from="xmlpadmin1@oracle.com"  reply-to ="reply@oracle.com">
       <xapi:message id="123"  to="${EMAIL}"  cc="${EMAIL_ALL}"
        attachment="true" subject="Employee Details
        for ${ENAME}"> Mr. ${ENAME}, Please review the
        attached document</xapi:message><! - This assigns a delivery id
of '123'. It also sets the e-mail
     address of the employee and a cc copy to a parameter value
     EMAIL_ALL; this might be a manager's e-mail. The employee's
     name (ENAME) can also be used in the subject/body
     of the email.  - ></xapi:email>
   </xapi:delivery>
    <xapi:document output-type="pdf" delivery="123">
     <xapi:template type="rtf" location="/usr/tmp/empGeneric.rtf">
     <xapi:template type="rtf" location="/usr/tmp/empDet.rtf"
      filter=".//EMPLOYEE[ENAME='SMITH']" ><! -  Employees with the name
SMITH will have
      the empDet template applied  - >
      </xapi:template>
    </xapi:document>
  </xapi:request>
</xapi:requestset>
```

Multiple Bursting Options

The bursting engine can support multiple bursting criteria and delivery options. Assume you have a report that generates data for all employees with their manager's information. You can construct a control file that will:

- Burst the employee data to each employee
- Burst a report to each manager that contains the data about his employees

You can provide a different template for each bursting level. You can therefore generate the employee report based on one template and the summary manager's report based on a different template, but still use the same data set.

To achieve this multibursting result, you must add a second `request` element to the control file structure.

Multibursting Example

The following sample shows how to construct a control file that will burst on the EMPLOYEE level and the MANAGER level:

```
?xml version="1.0" encoding="UTF-8" ?>
<xapi:requestset xmlns:xapi="http://xmlns.oracle.com/oxp/xapi"><! -
First request to burst on employee  - >
 <xapi:request select="/EMPLOYEES/EMPLOYEE">
 <xapi:delivery>
 <xapi:email <<server details removed>> />
  <xapi:message id="123" <<message details removed>>
   </xapi:message>
  </xapi:email>
 <xapi:fax server="ipp://mycupsserver:631/printers/fax2">
  <xapi:number id="FAX1">916505069560</xapi:number>
  </xapi:fax>
  <xapi:print id="printer1"
   printer="ipp://mycupsserver:631/printers/printer1"
   copies="2" />
  </xapi:delivery>
 <xapi:document output-type="pdf" delivery="123">
  <xapi:template type="rtf" location="usr\tmp\empDet.rtf" />
  </xapi:document>
 </xapi:request><!Second request to burst on department  - >
<xapi:request select="/DATA/DEPT/MANAGER">
 <xapi:delivery>
 <xapi:email server="gmsmtp.oraclecorp.com" port=""
  from="XDOburstingTest@oracle.com" reply-to="reply@oracle.com">
  <xapi:message id="123" to="${MANAGER_EMAIL}"
   cc="${MANAGER_EMAIL}" attachment="true"
   subject="Department Summary for ${DEPTNO}">Please review
   the attached Department Summary for
   department ${DEPTNO}</xapi:message>
  </xapi:email>
  </xapi:delivery>
<xapi:document output-type="rtf" delivery="123">
  <xapi:template type="rtf"
   location="d:\burst_test\deptSummary.rtf" />
  </xapi:document>
  </xapi:request>
  </xapi:requestset>
```

Bursting Listeners

The bursting engine provides a listening interface that allows you to listen to the various stages of the bursting process. Following are the supported modes that you can subscribe to:

- `beforeProcess()` - before the bursting process starts.
- `afterProcess()` - after the bursting process completes.
- `beforeProcessRequest(int requestIndex)` - before the bursting request starts. This interface provides an assigned request ID for the current request.
- `afterProcessRequest(int requestIndex)` - after the bursting request has completed; provides the request ID for the current request.
- `beforeProcessDocument(int requestIndex,int documentIndex,String deliveryId)` - before the document generation starts;

provides the request ID and a document ID.

- `afterProcessDocument(int requestIndex,int documentIndex,Vector documentOutputs)` - after the document generation completes; provides the request ID and document ID, plus a Vector list of the document objects generated in the request.

- `beforeDocumentDelivery(int requestIndex,int documentIndex,String deliveryId)` - before the documents in the request are delivered; provides the request ID, the document ID, and a delivery ID.

- `afterDocumentDelivery(int requestIndex,int documentIndex,String deliveryId,Object deliveryObject,Vector attachments)` - after the document delivery completes; provides a request ID, document ID, and delivery ID, plus a Vector list of the documents delivered in the request.

You can subscribe to any of these interfaces in your calling Java class. The listeners are useful to determine if the processing of individual documents is proceeding successfully or to start another process based on the successful completion of a request.

Calling the Bursting API

To call the bursting API, instantiate an instance of DocumentProcessor class using on of the following formats:

```
DocumentProcessor(xmlCtrlInput, xmlDataInput, tmpDir)
```

where

`xmlCtrlInput` - is the control file for the bursting process. This can be a string reference to a file, an inputStream object, or a Reader object.

`xmlDataInput` - is the XML data to be burst. This can a string reference to a file, an inputStream object, or a Reader object.

`tmpDir` - is a temporary working directory. This takes the format of a String object. This is optional as long as the main BI Publisher temporary directory has been set.

Simple Example Java Class

The following is a sample Java class:

```
public class BurstingTest
{
  public BurstingTest()
  {
   try
   {
 DocumentProcessor dp = new DocumentProcessor
 ("\burst\burstCtrl.xml", "\\burst\\empData.xml","\\burst");
  dp.process();
 }
   }
   catch (Exception e)
   { System.out.println(e);

 public static void main(String[] args)
  {
    BurstingTest burst1 = new BurstingTest();
  }

}
```

Example Java Class with Listeners

To take advantage of the bursting listeners, add the interface to the class declaration and use the `registerListener` method. Then code for the listeners you want to subscribe to as follows:

```
public class BurstingTest implements BurstingListener
{
  public BurstingTest()
  {
   try
   {
 DocumentProcessor dp = new DocumentProcessor
 ("\burst\burstCtrl.xml", "\\burst\\empData.xml","\\burst");
 dp.registerListener(this);
  dp.process();
 }
   }
   catch (Exception e)
   { System.out.println(e);

 public static void main(String[] args)
  {
    BurstingTest burst1 = new BurstingTest();
}

public void beforeProcess(){
  System.out.println("Start of Bursting Process");
  }
public void afterProcess()
  {
  System.out.println("End of Bursting Process");
  }

public void beforeProcessRequest(int requestIndex)
  {
  System.out.println("Start of Process Request ID"+requestIndex);
  }
  public void afterProcessRequest(int requestIndex)
  {
   System.out.println("End of Process Request ID"+requestIndex ");

  }
  public void beforeProcessDocument(int requestIndex,int
   documentIndex)
  {
  System.out.println("Start of Process Document");
  System.out.println("   Request Index "+requestIndex);
  System.out.println("   Document Index " +documentIndex);

  }
  public void afterProcessDocument(int requestIndex,int
   documentIndex,
  Vector documentOutputs)
  {
     System.out.println("   ========End of Process Document");
     System.out.println("   Outputs :"+documentOutputs);
  }
  public void beforeDocumentDelivery(int requestIndex,int
   documentIndex,
  String deliveryId)
  {
  System.out.println("   ========Start of  Delivery");
  System.out.println("   Request Index "+requestIndex);
  System.out.println("   Document Index " +documentIndex);
  System.out.println("   DeliveryId  " +deliveryId);
  }
```

```
public void afterDocumentDelivery(int requestIndex,int documentIndex,
  String deliveryId,Object deliveryObject,Vector attachments)
  {
    System.out.println("   ========End of  Delivery");
    System.out.println("  Attachments : "+attachments);

   }

}
```

Passing a Parameter

To pass a parameter holding a value to be used in the control file for delivery, add the following code:

```
…
Properties prop= new Properties();
prop.put("user-variable:ADMIN_EMAIL","jo.smith@company.com");
dp.setConfig(prop);
dp.process();
…
```

Bursting Control File Examples

All of the examples in this section use the following XML data source:

```
<?xml version="1.0" encoding="UTF-8"?>
<DATA>
<DEPTS>
 <DEPT>
  <DEPTNO>20</DEPTNO>
  <NAME>Accounting</NAME>
  <MANAGER_EMAIL>tdexter@mycomp.com</MANAGER_EMAIL>
  <EMPLOYEES>
   <EMPLOYEE>
    <EMPNO>7369</EMPNO>
    <ENAME>SMITH</ENAME>
    <JOB>CLERK</JOB>
    <MGR>7902</MGR>
    <HIREDATE>1980-12-17T00:00:00.000-08:00</HIREDATE>
    <SAL>800</SAL>
    <DEPTNO>20</DEPTNO>
    <EMAIL>jsmith@mycomp.com</EMAIL>
   </EMPLOYEE>
   <EMPLOYEE>
    <EMPNO>7566</EMPNO>
    <ENAME>JONES</ENAME>
    <JOB>MANAGER</JOB>
    <MGR>7839</MGR>
    <HIREDATE>1981-04-02T00:00:00.000-08:00</HIREDATE>
    <SAL>2975</SAL>
    <DEPTNO>20</DEPTNO>
    <EMAIL>jjones@mycomp.com</EMAIL>
   </EMPLOYEE>
  </EMPLOYEES>
 </DEPT>
 <DEPT>
  <DEPTNO>30</DEPTNO>
  <NAME>Sales</NAME>
  <MANAGER_EMAIL>dsmith@mycomp.com</MANAGER_EMAIL>
  <EMPLOYEES>
   <EMPLOYEE>
    <EMPNO>7788</EMPNO>
    <ENAME>SCOTT</ENAME>
    <JOB>ANALYST</JOB>
    <MGR>7566</MGR>
    <HIREDATE>1982-12-09T00:00:00.000-08:00</HIREDATE>
    <SAL>3000</SAL>
    <DEPTNO>20</DEPTNO>
    <EMAIL>jscott@mycomp.com</EMAIL>
   </EMPLOYEE>
   <EMPLOYEE>
    <EMPNO>7876</EMPNO>
    <ENAME>ADAMS</ENAME>
    <JOB>CLERK</JOB>
    <MGR>7788</MGR>
    <HIREDATE>1983-01-12T00:00:00.000-08:00</HIREDATE>
    <SAL>1100</SAL>
    <EMAIL>jadams@mycomp.com</EMAIL>
   </EMPLOYEE>
  </EMPLOYEES>
 </DEPT>
</DEPTS>
</DATA>
```

Example 1 - Bursting Employee Data to Employees via E-mail

The following sample shows how to apply a template (empDet.rtf) to every employee's

data, generate a PDF document, and deliver the document to each employee via e-mail.

```
<?xml version="1.0" encoding="UTF-8"?>
<xapi:requestset xmlns:xapi="http://xmlns.oracle.com/oxp/xapi">
<xapi:request select="/DATA/DEPTS/DEPT/EMPLOYEES/EMPLOYEE"> <! -  Burst
on employee element  - >
    <xapi:delivery>
      <xapi:email server="my.smtp.server" port="25"
       from="xmlpadmin@mycomp.com" reply-to ="">
       <xapi:message id="123"  to="${EMAIL}"
<! -  Set the id for the delivery method  - ><! -  Use the employees
EMAIL element to email the document to
      the employee  - >cc="${ADMIN_EMAIL}"
<! -  Use the ADMIN_EMAIL parameter to CC the document
      to the administrator  - > attachment="true" subject="Employee
Details for ${ENAME}">
 Mr. ${ENAME}, Please review the attached document</xapi:message><! -
Embed the employees name into the email message  - >
</xapi:email>
   </xapi:delivery>
    <xapi:document output-type="pdf" delivery="123"><!Specify the
delivery method id to be used  - >
     <xapi:template type="rtf"
      location="\usr\empDet.rtf"></xapi:template>
    </xapi:document>
  </xapi:request>
</xapi:requestset>
```

Example 2 - Bursting Employee Data to Employees via Multiple Delivery Channels and Conditionally Using Layout Templates

This sample shows how to burst, check the employee name, and generate a PDF using the appropriate template. The documents will then be e-mailed and printed.

```
<?xml version="1.0" encoding="UTF-8"?>
<xapi:requestset xmlns:xapi="http://xmlns.oracle.com/oxp/xapi" >
 <xapi:globalData location="stream">
 </xapi:globalData >
  <xapi:request select="/DATA/DEPTS/DEPT/EMPLOYEES/EMPLOYEE">
    <xapi:delivery>
      <xapi:email server="my.smtp.server" port=""
       from="xmlpserver@oracle.com"
       reply-to ="reply@oracle.com">
     <xapi:message id="123"  to="${EMAIL}" cc="" attachment="true"
       subject="Employee Details for ${ENAME}"> Mr. ${ENAME},
       Please review the attached document</xapi:message>
</xapi:email>
<xapi:print id="printer1"
  printer="ipp://ipgpc1:631/printers/printer1" copies="2" /><! -  Add an
id for this delivery method i.e. printer1  - >     </xapi:delivery>
    <xapi:document output-type="pdf" delivery="printer1,123"><! -
Deliver to printer and email  - >     <xapi:template type="rtf"
location="/usr/empDetSmith.rtf"
      filter=".//EMPLOYEE[ENAME='SMITH']"><!- Specify template to be
used for employees called SMITH  - >
      </xapi:template>
     <xapi:template type="rtf" location="/usr/empSummary.rtf"><! -
Default template to be used  - >
      </xapi:template>
    </xapi:document>
  </xapi:request>
</xapi:requestset>
```

Example 3 - Bursting Employee Data to Employees and Their Manager

This sample shows how to burst an e-mail with a PDF attachment to all employees using the empDet template. It will also burst an employee summary PDF to the manager of each department via e-mail.

```
<?xml version="1.0" encoding="UTF-8"?>
<xapi:requestset xmlns:xapi="http://xmlns.oracle.com/oxp/xapi">
  <xapi:request select="/DATA/DEPTS/DEPT/EMPLOYEES/EMPLOYEE">
    <xapi:delivery>
      <xapi:email server="my.smtp.server" port=""
       from="xmlpserver@oracle.com" reply-to ="">
      <xapi:message id="123"  to="${EMAIL}" cc="${EMAIL}"
       attachment="true"
       subject="Employee Details for ${ENAME}"> Mr. ${ENAME},
       Please review the attached document</xapi:message>
</xapi:email>
    </xapi:delivery>
    <xapi:document output-type="pdf" delivery="123">
     <xapi:template type="rtf"
      location="/usr/empDet.rtf"</xapi:template>
    </xapi:document>
  </xapi:request>
<xapi:request select="/DATA/DEPTS/DEPT"><! -  Second request created to
burst the same dataset to the
      manager based on the DEPT element  - >
    <xapi:delivery>
       <xapi:email  server="my.smtp.server" port=""
        from="xmlpserver@oracle.com" reply-to ="">
<xapi:message id="456"  to="${MANAGER_EMAIL}"
 cc="${MANAGER_EMAIL}" attachment="true" subject="Department
 Summary for ${DEPTNO}"> Please review the attached
 Department Summary for department ${DEPTNO}</xapi:message>
</xapi:email>
    </xapi:delivery>
    <xapi:document output-type="rtf" delivery="456">
     <xapi:template type="rtf"
      location="\usr\deptSumm.rtf"></xapi:template>
  </xapi:document>
  </xapi:request>
</xapi:requestset>
```

BI Publisher Properties

The FO Processor supports PDF security and other properties that can be applied to your final documents. Security properties include making a document unprintable and applying password security to an encrypted document.

Other properties allow you to define font subsetting and embedding. If your template uses a font that would not normally be available to BI Publisher at runtime, you can use the font properties to specify the location of the font. At runtime BI Publisher will retrieve and use the font in the final document. For example, this property might be used for check printing for which a MICR font is used to generate the account and routing numbers on the checks.

See Setting Runtime Properties, page 8-1 for the full list of properties.

Setting Properties

The properties can be set in two ways:

- At runtime, specify the property as a Java Property object to pass to the FO Processor.
- Set the property in a configuration file.
- Set the property in the template (RTF templates only). See Setting Properties, *Oracle Business Intelligence Publisher Report Designer's Guide* in the RTF template for this method.

Passing Properties to the FO Engine

To pass a property as a Property object, set the name/value pair for the property prior to calling the FO Processor, as shown in the following example:

Input:

- XML file name (String)
- XSL file name (String)

Output:

- PDF file name (String)

Example

```
import oracle.apps.xdo.template.FOProcessor;
.
.
.
  public static void main(String[] args)
  {

    FOProcessor processor = new FOProcessor();
    processor.setData(args[0]);     // set XML input file
    processor.setTemplate(args[1]); // set XSL input file
    processor.setOutput(args[2]);  //set (PDF) output file
    processor.setOutputFormat(FOProcessor.FORMAT_PDF);
    Properties prop = new Properties();
    /* PDF Security control: */
    prop.put("pdf-security", "true");
    /* Permissions password: */
    prop.put("pdf-permissions-password", "abc");
    /* Encryption level: */
    prop.put("pdf-encription-level", "0");
    processor.setConfig(prop);
    // Start processing
    try
    {
       processor.generate();
    }
    catch (XDOException e)
    {
       e.printStackTrace();
       System.exit(1);
    }

    System.exit(0);
  }
```

Passing a Configuration File to the FO Processor

The following code shows an example of passing the location of a configuration file.

Input:

- XML file name (String)
- XSL file name (String)

Output:

- PDF file name (String)

```
import oracle.apps.xdo.template.FOProcessor;
.
.
.
  public static void main(String[] args)
  {
    FOProcessor processor = new FOProcessor();
    processor.setData(args[0]);     // set XML input file
   processor.setTemplate(args[1]); // set XSL input file
    processor.setOutput(args[2]);  //set (PDF) output file
    processor.setOutputFormat(FOProcessor.FORMAT_PDF);
    processor.setConfig("/tmp/xmlpconfig.xml");
    // Start processing
    try
    {
       processor.generate();
    }    catch (XDOException e)
    {       e.printStackTrace();
            System.exit(1);
    }
      System.exit(0);
  }
```

Passing Properties to the Document Processor

Input:

- Data file name (String)
- Directory for Temporary Files (String)

Output:

- PDF FIle

Example

```
import oracle.apps.xdo.batch.DocumentProcessor;
.
.
.
  public static void main(String[] args)
  {
.
.
.
    try
    {
      // dataFile --- File path of the Document Processor XML
      // tempDir  --- Temporary Directory path
      DocumentProcessor  docProcessor = new DocumentProcessor(dataFile,
tempDir);
      Properties prop = new Properties();
      /* PDF Security control: */
      prop.put("pdf-security", "true");
      /* Permissions password: */
      prop.put("pdf-permissions-password", "abc");
      /* encryption level: */
      prop.put("pdf-encription-level", "0");
      processor.setConfig(prop);
      docProcessor.process();
    }
    catch(Exception e)
    {
 e.printStackTrace();
       System.exit(1);
    }
    System.exit(0);
  }
```

Advanced Barcode Font Formatting Implementation

For the advanced formatting to work in the template, you must provide a Java class with the appropriate methods to format the data at runtime. Many font vendors offer the code with their fonts to carry out the formatting; these must be incorporated as methods into a class that is available to the BI Publisher formatting libraries at runtime. There are some specific interfaces that you must provide in the class for the library to call the correct method for encoding.

> **Note:** See Advanced Barcode Formatting, *Oracle Business Intelligence Publisher Report Designer's Guide* for the setup required in the RTF template.

You must implement the following three methods in this class:

```
/**
 * Return a unique ID for this bacode encoder
 * @return the id as a string
 */
  public String getVendorID();

/**
 * Return true if this encoder support a specific type of barcode
 * @param type the type of the barcode
 * @return true if supported
 */
  public boolean isSupported(String type);

/**
 * Encode a barcode string by given a specific type
 * @param data the original data for the barcode
 * @param type the type of the barcode
 * @return the formatted data
 */
  public String encode(String data, String type);
```

Place this class in the classpath for the middle tier JVM in which BI Publisher is running.

> **Note:** For E-Business Suite users, the class must be placed in the classpath for the middle tier and any concurrent nodes that are present.

If in the register-barcode-vendor command the `barcode_vendor_id` is not provided, BI Publisher will call the getVendorID() and use the result of the method as the ID for the vendor.

The following is an example class that supports the code128 a, b and c encodings:

> **Important:** The following code sample can be copied and pasted for use in your system. Note that due to publishing constraints you will need to correct line breaks and ensure that you delete quotes that display as "smart quotes" and replace them with simple quotes.

Example

```
package oracle.apps.xdo.template.rtf.util.barcoder;

import java.util.Hashtable;
import java.lang.reflect.Method;
import oracle.apps.xdo.template.rtf.util.XDOBarcodeEncoder;
import oracle.apps.xdo.common.log.Logger;
// This class name will be used in the register vendor
// field in the template.

public class BarcodeUtil  implements XDOBarcodeEncoder
// The class implements the XDOBarcodeEncoder interface
{
// This is the barcode vendor id that is used in the
// register vendor field and format-barcode fields
  public static final String BARCODE_VENDOR_ID = "XMLPBarVendor";
// The hashtable is used to store references to
// the encoding methods
  public static final Hashtable ENCODERS = new Hashtable(10);
// The BarcodeUtil class needs to be instantiated
  public static final BarcodeUtil mUtility = new BarcodeUtil();
// This is the main code that is executed in the class,
// it is loading the methods for the encoding into the hashtable.
// In this case we are loading the three code128 encoding
// methods we have created.
  static {
    try {
      Class[] clazz = new Class[] { "".getClass() };

      ENCODERS.put("code128a",mUtility.getClass().getMethod("code128a",
clazz));
      ENCODERS.put("code128b",mUtility.getClass().getMethod("code128b",
clazz));
      ENCODERS.put("code128c",mUtility.getClass().getMethod("code128c",
clazz));
     } catch (Exception e) {
// This is using the BI Publisher logging class to push
// errors to the XMLP log file.
      Logger.log(e,5);
    }
  }
```

```
// The getVendorID method is called from the template layer
// at runtime to ensure the correct encoding method are used
    public final String getVendorID()
    {
        return BARCODE_VENDOR_ID;
    }
//The isSupported method is called to ensure that the
// encoding method called from the template is actually
//  present in this class.
// If not then XMLP will report this in the log.
    public final boolean isSupported(String s)
    {
        if(s != null)
            return ENCODERS.containsKey(s.trim().toLowerCase());
        else
            return false;
    }

// The encode method is called to then call the appropriate
// encoding method, in this example the code128a/b/c methods.

   public final String encode(String s, String s1)
    {
        if(s != null && s1 != null)
        {
            try
            {
                Method method =
(Method)ENCODERS.get(s1.trim().toLowerCase());
                if(method != null)
                    return (String)method.invoke(this, new Object[] {
                        s
                    });
                else
                    return s;
            }
            catch(Exception exception)
            {
                  Logger.log(exception,5);
            }
            return s;
        } else
        {
            return s;
        }
    }

  /** This is the complete method for Code128a */

  public static final String code128a( String DataToEncode )
  {
    char C128_Start = (char)203;
    char C128_Stop = (char)206;
    String Printable_string = "";
    char CurrentChar;
    int CurrentValue=0;
    int weightedTotal=0;
    int CheckDigitValue=0;
    char C128_CheckDigit='w';

    DataToEncode = DataToEncode.trim();
```

```
weightedTotal = ((int)C128_Start) - 100;
    for( int i = 1; i <= DataToEncode.length(); i++ )
      {
 //get the value of each character
 CurrentChar = DataToEncode.charAt(i-1);
 if( ((int)CurrentChar) < 135 )
   CurrentValue = ((int)CurrentChar) - 32;
 if( ((int)CurrentChar) > 134 )
   CurrentValue = ((int)CurrentChar) - 100;

 CurrentValue = CurrentValue * i;
 weightedTotal = weightedTotal + CurrentValue;
      }
    //divide the WeightedTotal by 103 and get the remainder,//this is
the CheckDigitValue
    CheckDigitValue = weightedTotal % 103;
    if( (CheckDigitValue < 95) && (CheckDigitValue > 0) )
      C128_CheckDigit = (char)(CheckDigitValue + 32);
    if( CheckDigitValue > 94 )
      C128_CheckDigit = (char)(CheckDigitValue + 100);
    if( CheckDigitValue == 0 ){
      C128_CheckDigit = (char)194;
    }

    Printable_string = C128_Start + DataToEncode + C128_CheckDigit +
C128_Stop + " ";
    return Printable_string;
  }
```

```
/** This is the complete method for Code128b ***/

  public static final String code128b( String DataToEncode )
  {
    char C128_Start = (char)204;
    char C128_Stop = (char)206;
    String Printable_string = "";
    char CurrentChar;
    int CurrentValue=0;
    int weightedTotal=0;
    int CheckDigitValue=0;
    char C128_CheckDigit='w';

    DataToEncode = DataToEncode.trim();
    weightedTotal = ((int)C128_Start) - 100;
    for( int i = 1; i <= DataToEncode.length(); i++ )
      {
 //get the value of each character
 CurrentChar = DataToEncode.charAt(i-1);
 if( ((int)CurrentChar) < 135 )
   CurrentValue = ((int)CurrentChar) - 32;
 if( ((int)CurrentChar) > 134 )
   CurrentValue = ((int)CurrentChar) - 100;

 CurrentValue = CurrentValue * i;
 weightedTotal = weightedTotal + CurrentValue;
      }
    //divide the WeightedTotal by 103 and get the remainder,//this is
the CheckDigitValue
    CheckDigitValue = weightedTotal % 103;
    if( (CheckDigitValue < 95) && (CheckDigitValue > 0) )
      C128_CheckDigit = (char)(CheckDigitValue + 32);
    if( CheckDigitValue > 94 )
      C128_CheckDigit = (char)(CheckDigitValue + 100);
    if( CheckDigitValue == 0 ){
      C128_CheckDigit = (char)194;
    }

    Printable_string = C128_Start + DataToEncode + C128_CheckDigit +
C128_Stop + " ";
    return Printable_string;
  }


  /** This is the complete method for Code128c **/

  public static final String code128c( String s )
  {
    char C128_Start = (char)205;
    char C128_Stop = (char)206;
    String Printable_string = "";
    String DataToPrint = "";
    String OnlyCorrectData = "";
    int i=1;
    int CurrentChar=0;
    int CurrentValue=0;
    int weightedTotal=0;
    int CheckDigitValue=0;
    char C128_CheckDigit='w';
    DataToPrint = "";
    s = s.trim();
```

```
for(i = 1; i <= s.length(); i++ )
      {
 //Add only numbers to OnlyCorrectData string
 CurrentChar = (int)s.charAt(i-1);
 if((CurrentChar < 58) && (CurrentChar > 47))
   {
     OnlyCorrectData = OnlyCorrectData + (char)s.charAt(i-1);
   }
      }
    s = OnlyCorrectData;
    //Check for an even number of digits, add 0 if not even
    if( (s.length() % 2) == 1 )
      {
 s = "0" + s;
      }
    //<<<< Calculate Modulo 103 Check Digit and generate
    // DataToPrint >>>>//Set WeightedTotal to the Code 128 value of
// the start character
    weightedTotal = ((int)C128_Start) - 100;
    int WeightValue = 1;
    for( i = 1; i <= s.length(); i += 2 )
      {
 //Get the value of each number pair (ex: 5 and 6 = 5*10+6 =56) //And
assign the ASCII values to DataToPrint
 CurrentChar = ((((int)s.charAt(i-1))-48)*10) + (((int)s.charAt(i))-48);
 if((CurrentChar < 95) && (CurrentChar  > 0))
   DataToPrint = DataToPrint + (char)(CurrentChar + 32);
 if( CurrentChar > 94 )
   DataToPrint = DataToPrint + (char)(CurrentChar + 100);
 if( CurrentChar == 0)
   DataToPrint = DataToPrint + (char)194;
 //multiply by the weighting character
 //add the values together to get the weighted total
 weightedTotal = weightedTotal + (CurrentChar * WeightValue);
 WeightValue = WeightValue + 1;
      }
    //divide the WeightedTotal by 103 and get the remainder,//this is
the CheckDigitValue
    CheckDigitValue = weightedTotal % 103;
    if((CheckDigitValue < 95) && (CheckDigitValue > 0))
      C128_CheckDigit = (char)(CheckDigitValue + 32);
    if( CheckDigitValue > 94 )
      C128_CheckDigit = (char)(CheckDigitValue + 100);
    if( CheckDigitValue == 0 ){
      C128_CheckDigit = (char)194;
    }
    Printable_string = C128_Start + DataToPrint + C128_CheckDigit +
C128_Stop + " ";
    Logger.log(Printable_string,5);
    return Printable_string;
  }
}
```

Once you create the class and place it in the correct classpath, your template creators can start using it to format the data for barcodes. You must give them the following information to include in the template commands:

- The class name and path.

  In this example:

```
oracle.apps.xdo.template.rtf.util.barcoder.BarcodeUtil
```

- The barcode vendor ID you created.

  In this example: `XMLPBarVendor`

- The available encoding methods.

  In this example, code128a, code128b and code128c They can then use this information to sucessfully encode their data for barcode output.

They can then use this information to successfully encode their data for barcode output.

12

Using the Delivery Manager APIs

Introduction

The Delivery Manager is a set of Java APIs that allow you to control the delivery of your BI Publisher documents. Use the Delivery Manager to:

- Deliver documents through established delivery channels (e-mail, fax, printer, WebDAV, FTP, Secure FTP, AS2, or HTTP) or custom delivery channels
- Track the status of each delivery
- Redeliver documents

Using the Delivery Manager

To use the Delivery Manager follow these steps:

1. Create a DeliveryManager instance
2. Create a DeliveryRequest instance using the createRequest() method
3. Add the request properties (such as DeliveryRequest destination). Most properties require a String value. See the supported properties for each delivery channel for more information.
4. Set your document to the DeliveryRequest.
5. Call submit() to submit the delivery request.

One delivery request can handle one document and one destination. This facilitates monitoring and resubmission, if necessary.

DeliveryRequest allows you to set the documents in three ways as follows:

- Set InputStream of the document to DeliveryRequest. The DeliveryRequest will read the InputStream when you call submit() for the first time. The DeliveryRequest does not close the InputStream so you must ensure to close it.
- Set the file name of the document to DeliveryRequest.

The Delivery Manager supports streamlined delivery when you set the direct mode. See Direct and Buffering Modes, page 12-33.

The follow delivery channels are described in this document:

- E-mail
- Printer
- Fax
- WebDAV
- FTP
- Secure FTP
- HTTP
- AS2

Delivering Documents by e-Mail

The following sample demonstrates delivery via E-mail:

Example

```
// create delivery manager instance
     DeliveryManager dm = new DeliveryManager();
     // create a delivery request
     DeliveryRequest req =
dm.createRequest(DeliveryManager.TYPE_SMTP_EMAIL);

     // set email subject
     req.addProperty(DeliveryPropertyDefinitions.SMTP_SUBJECT, "test
mail");
     // set SMTP server host
     req.addProperty(
       DeliveryPropertyDefinitions.SMTP_HOST, "mysmtphost");
     // set the sender email address
     req.addProperty(DeliveryPropertyDefinitions.SMTP_FROM,
"myname@mydomain.com");
     // set the destination email address
     req.addProperty(
       DeliveryPropertyDefinitions.SMTP_TO_RECIPIENTS,
"user1@mydomain.com, user2@mydomain.com" );
     // set the content type of the email body
     req.addProperty(DeliveryPropertyDefinitions.SMTP_CONTENT_TYPE,
"application/pdf");
     // set the document file name appeared in the email
     req.addProperty(DeliveryPropertyDefinitions.SMTP_CONTENT_FILENAME,
"test.pdf");
     // set the document to deliver
     req.setDocument("/document/test.pdf");

     // submit the request
     req.submit();
     // close the request
     req.close();
```

The following table lists the supported properties:

Property	Description
SMTP_TO_RECIPIENTS	Required Enter multiple recipients separated by a comma (example: "user1@mydomain.com, user2@mydomain.com")
SMTP_CC_RECIPIENTS	Optional Enter multiple recipients separated by a comma.
SMTP_BCC_RECIPIENTS	Optional Enter multiple recipients separated by a comma.

Property	Description
SMTP_FROM	Required Enter the e-mail address of the sending party.
SMTP_REPLY_TO	Optional Enter the reply-to e-mail address.
SMTP_SUBJECT	Required Enter the subject of the e-mail.
SMTP_CHARACTER_ENCODING	Optional Default is "UTF-8".
SMTP_ATTACHMENT	Optional If you are including an attachment, enter the attachment object name.
SMTP_CONTENT_FILENAME	Required Enter the file name of the document (example: invoice.pdf)
SMTP_CONTENT_TYPE	Required Enter the MIME type.
SMTP_SMTP_HOST	Required Enter the SMTP host name.
SMTP_SMTP_PORT	Optional Enter the SMTP port. Default is 25.
SMTP_SMTP_USERNAME	Optional If the SMTP server requires authentication, enter your username for the server.

Property	Description
SMTP_SMTP_PASSWORD	Optional If the SMTP server requires authentication, enter the password for the username you entered.
SMTP_ATTACHMENT_FIRST	Optional If your e-mail contains an attachment and you want the attachment to appear first, enter "true". If you do not want the attachment to appear first, enter "false".

Defining Multiple Recipients

The e-mail delivery server channel supports multiple documents and multiple destinations per request. The following example demonstrates multiple TO and CC addresses:

Example

```
// set the TO email addresses
     req.addProperty(
       DeliveryPropertyDefinitions.SMTP_TO_RECIPIENTS,
        "user1@mydomain.com, user2@mydomain.com, user3@mydomain.com");

     // set the CC email addresses
     req.addProperty(
       DeliveryPropertyDefinitions.SMTP_CC_RECIPIENTS,
        "user4@mydomain.com, user5@mydomain.com, user6@mydomain.com");
```

Attaching Multiple Documents into One Request

Use the Attachment utility class (`oracle.apps.xdo.delivery.smtp.Attachment`) to attach multiple documents into one request. Sample usage is as follows:

Example

```
:
     :
   // create Attachment instance
   Attachment m = new Attachment();

   // add PDF attachment
   m.addAttachment(
       "/pdf_doc/invoice.pdf",      // file to deliver
       "invoice.pdf",               // file name as appears in email
       "application/pdf");          // content type

   // add RTF attachment
   m.addAttachment(
       "/rtf_doc/product.rtf",      // file to deliver
       "product.rtf",               // file name appears in the email
       "application/rtf");          // content type

   // add XML attachment
   m.addAttachment(
       "/xml_doc/data.xml",          // file to deliver
       "data.xml",                   // file name appears in the email
       "text/xml");                  // content type

   // If you want to attach HTML doucments, use addHtmlAttachment().
   // This method automatically resolves the image references
   // in your HTML document and attaches those images.
   m.addHtmlAttachment("/html_doc/invoice.html");

   // add the attachment to the request
   req.addProperty(DeliveryPropertyDefinitions.SMTP_ATTACHMENT, m);

     :
     :
```

Attaching HTML Documents

You can attach HTML documents into one request. If you have references to image files located in the local file system in your HTML document, the Attachment utility automatically attaches those image files also. The sample usage is as follows:

Example

```
Attachment m = new Attachment();
   m.addHtmlAttachment("/path/to/my.html");
     :
     :

   req.addProperty(DeliveryPropertyDefinitions.SMTP_ATTACHMENT, m);
```

Displaying the Attachment at the top of the e-mail

If you want to show your attachment at the top of the e-mail, set the property SMTP_ATTACHMENT_FIRST to "true". Sample usage is as follows.

Example

```
Attachment m = new Attachment();
   m.addHtmlAttachment("/path/to/my.html");
     :
     :
   req.addProperty(DeliveryPropertyDefinitions.SMTP_ATTACHMENT_FIRST,
"true");
     :
```

Using a String Object as the e-Mail Body

You can use a String object for the e-mail body. This may be useful if you want to include a message with your attached files. The following sample code will deliver the message "Please find the attached invoice." in the e-mail body and one PDF document "invoice.pdf" as an attachment.

```
// create delivery manager instance
   DeliveryManager dm = new DeliveryManager();
   // create a delivery request
   DeliveryRequest req =
dm.createRequest(DeliveryManager.TYPE_SMTP_EMAIL);

   // set email subject
   req.addProperty(DeliveryPropertyDefinitions.SMTP_SUBJECT, "Invoice");
   // set SMTP server host
   req.addProperty(
     DeliveryPropertyDefinitions.SMTP_HOST, "mysmtphost");
   // set the sender email address
   req.addProperty(DeliveryPropertyDefinitions.SMTP_FROM,
"myname@mydomain.com");
   // set the destination email address
   req.addProperty(
     DeliveryPropertyDefinitions.SMTP_TO_RECIPIENTS,
"user1@mydomain.com, user2@mydomain.com" );
   // set the document to deliver
   req.setDocument("Please find the attached invoice. ", "UTF-8");

   // create Attachment
   Attachment m = new Attachment();
   // add attachments
   m.addAttachment(
     "/pdf_doc/invoice.pdf",           // file to deliver
     "invoice.pdf",                    // file name appears in the
email
     "application/pdf");               // content type
   // add the attachment to the request
   req.addProperty(DeliveryPropertyDefinitions.SMTP_ATTACHMENT, m);

   // submit the request
   req.submit();
   // close the request
   req.close();

     :
     :
```

Using an HTML Document as the e-Mail Body

You can also use an HTML document for the e-mail body. The utility automatically

resolves the local image references in your HTML document and attaches those images also.

Sample usage is as follows:

```
// create delivery manager instance
   DeliveryManager dm = new DeliveryManager();
   // create a delivery request
   DeliveryRequest req =
dm.createRequest(DeliveryManager.TYPE_SMTP_EMAIL);

   // set email subject
   req.addProperty(DeliveryPropertyDefinitions.SMTP_SUBJECT, "Invoice");
   // set SMTP server host
   req.addProperty(
     DeliveryPropertyDefinitions.SMTP_HOST, "mysmtphost");
   // set the sender email address
   req.addProperty(DeliveryPropertyDefinitions.SMTP_FROM,
"myname@mydomain.com");
   // set the destination email address
   req.addProperty(
     DeliveryPropertyDefinitions.SMTP_TO_RECIPIENTS,
"user1@mydomain.com, user2@mydomain.com" );

   // set the content type of the email body
   req.addProperty(DeliveryPropertyDefinitions.SMTP_CONTENT_TYPE,
"text/html");
   // set the document file name appeared in the email
   req.addProperty(DeliveryPropertyDefinitions.SMTP_CONTENT_FILENAME,
"body.html");
   // set the document to deliver
   req.setDocument("/document/invoice.html");

   // submit the request
   req.submit();
   // close the request
   req.close();

     :
     :
```

Providing Username and Password for Authentication

If the SMTP server requires authentication, you can specify the username and password to the delivery request.

Example

```
:
   req.addProperty(DeliveryPropertyDefinitions.SMTP_USERNAME, "scott");
   req.addProperty(DeliveryPropertyDefinitions.SMTP_PASSWORD, "tiger");
     :
```

Delivering Your Document to a Printer

The Delivery Server supports Internet Printing Protocol (IPP) as defined in RFC 2910 and 2911 for the delivery of documents to IPP-supported printers or servers, such as CUPS.

Common Unix Printing System (CUPS) is a free, server-style, IPP-based software that can accept IPP requests and dispatch those requests to both IPP and non-IPP based devices, such as printers and fax machines. See http://www.cups.org/ for more information about CUPS. See Setting Up Cups, page 9-1for additional information about setting up CUPS in your system.

To print out your document with the IPP, you need to transform your document into the format that the target IPP printers or servers can understand before the delivery. For example, if the target printer is a Postscript printer, you must transform your document to Postscript format. Usually, printers do not natively understand PDF, RTF, Excel or Word document formats. The Delivery API itself does not provide the document format transformation functionality, but it does offer document filter support for this purpose. See Document Filter Support, page 12-35 for more information.

Following is a code sample for delivery to a printer:

Example

```
// create delivery manager instance
     DeliveryManager dm = new DeliveryManager();
     // create a delivery request
     DeliveryRequest req =
dm.createRequest(DeliveryManager.TYPE_IPP_PRINTER);

     // set IPP printer host
     req.addProperty(DeliveryPropertyDefinitions.IPP_HOST, "myhost");
     // set IPP printer port
     req.addProperty(DeliveryPropertyDefinitions.IPP_PORT, "631");
     // set IPP printer name
     req.addProperty(DeliveryPropertyDefinitions.IPP_PRINTER_NAME,
"/printers/myprinter");
     // set the document format
     req.addProperty(DeliveryPropertyDefinitions.IPP_DOCUMENT_FORMAT,
       DeliveryPropertyDefinitions.IPP_DOCUMENT_FORMAT_POSTSCRIPT);
     // set the document
     req.setDocument("/document/invoice.ps");

     // submit the request
     req.submit();
     // close the request
     req.close();
```

The following properties are supported. A string value is required for each property, unless otherwise noted. Note that printer-specific properties such as IPP_SIDES, IPP_COPIES and IPP_ORIENTATION depend on the printer capabilities. For example, if the target printer does not support duplex printing, the IPP_SIDES setting will have no effect.

Property	Description
IPP_HOST	Required Enter the host name.

Property	Description
IPP_PORT	Optional Default is 631.
IPP_PRINTER_NAME	Required Enter the name of the printer that is to receive the output. • If you use CUPS with the default setup, enter the printer name as `/printers/<printer-name>` • If you use the Microsoft Internet Information Service (IIS) with the default setup, enter the printer name as `/printers/<printer-name>/.printer`
IPP_AUTHTYPE	Optional Valid values for authentication type are: IPP_AUTHTYPE_NONE - no authentication (default) IPP_AUTHTYPE_BASIC - use HTTP basic authentication IPP_AUTHTYPE_DIGEST - use HTTP digest authentication
IPP_USERNAME	Optional Enter the username for HTTP authentication.
IPP_PASSWORD	Optional Enter the password for HTTP authentication.

Property	Description
IPP_ENCTYPE	Optional The encryption type can be set to either of the following: IPP_ENCTYPE_NONE - no encryption (default) IPP_ENCTYPE_SSL - use Secure Socket Layer
IPP_USE_FULL_URL	Optional Set to "true" to send the full URL for the HTTP request header. Valid values are "true" or "false" (default).
IPP_USE_CHUNKED_BODY	Optional Valid values are "true" (default) to use HTTP chunked transfer coding for the message body, or "false".
IPP_ATTRIBUTE_CHARSET	Optional Attribute character set of the IPP request. Default is "UTF-8".
IPP_NATURAL_LANGUAGE	Optional The natural language of the IPP request. Default is "en".
IPP_JOB_NAME	Optional Job name of the IPP request.
IPP_COPIES	Optional Define the number of copies to print (example: "1" , "5", "10"). Default is 1.

Property	Description
IPP_SIDES	Optional Enable two-sided printing. This setting will be ignored if the target printer does not support two-sided printing. Valid values are: • IPP_SIDES_ONE_SIDED - default • IPP_SIDES_TWO_SIDED_LONG_EDGE - prints both sides of paper for binding long edge. • IPP_SIDES_TWO_SIDED_SHORT_EDGE - prints both sides of paper for binding short edge. • IPP_SIDES_DUPLEX : Same as IPP_SIDES_TWO_SIDED_LONG_EDGE. • IPP_SIDES_TUMBLE : Same as IPP_SIDES_TWO_SIDED_SHORT_EDGE .
IPP_ORIENTATIONS	Optional Sets the paper orientation. This setting will be ignored if the target printer does not support orientation settings. Valid values are: IPP_ORIENTATIONS_PORTRAIT (default) IPP_ORIENTATIONS_LANDSCAPE
IPP_DOCUMENT_FORMAT	Optional The target printer must support the specified format. Valid values are: IPP_DOCUMENT_FORMAT_POSTSCRIPT IPP_DOCUMENT_FORMAT_PLAINTEXT IPP_DOCUMENT_FORMAT_PDF IPP_DOCUMENT_FORMAT_OCTETSTREAM (default)

Property	Description
IPP_MEDIA	You can choose either the paper size or the tray number. If you do not specify this option, the default media of the target printer will be used. It will be ignored if the target printer doesn't support the media option. Valid values are: • IPP_MEDIA_TRAY1 : Media on tray 1 • IPP_MEDIA_TRAY2 : Media on tray 2 • IPP_MEDIA_TRAY3 : Media on tray 3 • IPP_MEDIA_A3 : A3 Media • IPP_MEDIA_A4 : A4 Media • IPP_MEDIA_A5 : A5 Media • IPP_MEDIA_B4 : B4 Media • IPP_MEDIA_B5 : B5 Media
IPP_PAGE_RANGES	Specify page ranges to print. By default, all pages are printed. Example valid values are: • "3" : prints only page 3. • "2-5" : prints pages 2-5. • "1,3-5" : print page 1 and 3-5.

Printing over an HTTP Proxy Server

To deliver documents to IPP printers or fax machines over an HTTP proxy server, you may encounter delivery problems due to differences in the HTTP implementations between CUPS and the proxy servers. Setting the following two properties can resolve most of these problems:

- DeliveryPropertyDefinitions.IPP_USE_FULL_URL - set to "true"
- DeliveryPropertyDefinitions.IPP_USE_CHUNKED_BODY - set to "false"

If you use CUPS with the default setup, the typical property settings are as follows:

- IPP_HOST : <*host-name*>
- IPP_PORT : 631
- IPP_PRINTER_NAME : /printers/<*printer-name*>

If you use the Microsoft Internet Information Service (IIS) with the default setup, the typical property settings are as follows:

- IPP_HOST : <*host-name*>
- IPP_PORT : 80
- IPP_PRINTER_NAME : /printers/<*printer-name*>/.printer

Delivering Your Documents by a Fax Server

The delivery system supports the delivery of documents to fax modems configured on CUPS. You can configure fax modems on CUPS with efax (http://www.cce.com/efax/) and FAX4CUPS (http://www.gnu.org/directory/productivity/special/fax4CUPS.html).

Sample code for fax delivery is as follows:

Example

```
// create delivery manager instance
     DeliveryManager dm = new DeliveryManager();
     // create a delivery request
     DeliveryRequest req =
dm.createRequest(DeliveryManager.TYPE_IPP_FAX);

     // set IPP fax host
     req.addProperty(DeliveryPropertyDefinitions.IPP_HOST, "myhost");
     // set IPP fax port
     req.addProperty(DeliveryPropertyDefinitions.IPP_PORT, "631");
     // set IPP fax name
     req.addProperty(DeliveryPropertyDefinitions.IPP_PRINTER_NAME,
"/printers/myfax");
     // set the document format
     req.addProperty(DeliveryPropertyDefinitions.IPP_DOCUMENT_FORMAT,
"application/postscript");
     // set the phone number to send
     req.addProperty(DeliveryPropertyDefinitions.IPP_PHONE_NUMBER,
"9999999");
     // set the document
     req.setDocument("/document/invoice.pdf");

     // submit the request
     req.submit();
     // close the request
     req.close();
```

The supported properties are the same as those supported for printer documents, plus the following:

Property	Description
IPP_PHONE_NUMBER	Required Enter the fax number.

Delivering Your Documents to a WebDAV Server

The following is sample code for delivery to a Web-based Distributed Authoring and Versioning (WebDAV) server:

Example

```
// create delivery manager instance
     DeliveryManager dm = new DeliveryManager();
     // create a delivery request
     DeliveryRequest req =
dm.createRequest(DeliveryManager.TYPE_WEBDAV);

     // set document content type
     req.addProperty(DeliveryPropertyDefinitions.WEBDAV_CONTENT_TYPE,
"application/pdf");
     // set the WebDAV server hostname
     req.addProperty(DeliveryPropertyDefinitions.WEBDAV_HOST,
"mywebdavhost");
     // set the WebDAV server port number
     req.addProperty(DeliveryPropertyDefinitions.WEBDAV_PORT, "80");
     // set the target remote directory

req.addProperty(DeliveryPropertyDefinitions.WEBDAV_REMOTE_DIRECTORY,
"/content/");
     // set the remote filename
     req.addProperty(DeliveryPropertyDefinitions.WEBDAV_REMOTE_FILENAME,
"xdotest.pdf");

     // set username and password to access WebDAV server
     req.addProperty(DeliveryPropertyDefinitions.WEBDAV_USERNAME,
"xdo");
     req.addProperty(DeliveryPropertyDefinitions.WEBDAV_PASSWORD,
"xdo");
     // set the document
     req.setDocument("/document/test.pdf");

     // submit the request
     req.submit();
     // close the request
     req.close();
```

The following properties are supported. A String value is required for each, unless otherwise noted.

Property	Description
WEBDAV_CONTENT_TYPE	Required Enter the document content type (example: "application/pdf").
WEBDAV_HOST	Required Enter the server host name.
WEBDAV_PORT	Optional Enter the server port number. Default is 80.
WEBDAV_REMOTE_DIRECTORY	Required. Enter the remote directory name (example: "/myreports/").
WEBDAV_REMOTE_FILENAME	Required. Enter the remote file name.
WEBDAV_AUTHTYPE	Optional Valid values for authentication type are: WEBDAV_AUTHTYPE_NONE - no authentication (default) WEBDAV_AUTHTYPE_BASIC - use HTTP basic authentication WEBDAV_AUTHTYPE_DIGEST - use HTTP digest authentication
WEBDAV_USERNAME	Optional Enter the username for HTTP authentication.
WEBDAV_PASSWORD	Optional Enter the password for HTTP authentication.

Property	Description
WEBDAV_ENCTYPE	Optional Valid values for encryption type are: WEBDAV_ENCTYPE_NONE - no encryption (default) WEBDAV_ENCTYPE_SSL - use Secure Socket Layer
WEBDAV_USE_FULL_URL	Optional Set to "true" to send the full URL for the HTTP request header. Valid values are "true" or "false" (default).
WEBDAV_USE_CHUNKED_BODY	Optional Valid values are "true" (default) to use HTTP chunked transfer coding for the message body, or "false".
WEBDAV_URL_CHARACTER_ENCODING	Encoding of the URL. It will be used if you use non-ASCII characters in the URL. Set the Java-supported encoding string for the value.

Delivering Your Document Using FTP

The following is sample code for delivery to a FTP server:

Example

```
// create delivery manager instance
     DeliveryManager dm = new DeliveryManager();
     // create a delivery request
     DeliveryRequest req = dm.createRequest(DeliveryManager.TYPE_FTP);

     // set hostname of the FTP server
     req.addProperty(DeliveryPropertyDefinitions.FTP_HOST, "myftphost");
     // set port# of the FTP server
     req.addProperty(DeliveryPropertyDefinitions.FTP_PORT, "21");
     // set username and password to access WebDAV server
     req.addProperty(DeliveryPropertyDefinitions.FTP_USERNAME, "xdo");
     req.addProperty(DeliveryPropertyDefinitions.FTP_PASSWORD, "xdo");
     // set the remote directory that you want to send your document to
     req.addProperty(DeliveryPropertyDefinitions.FTP_REMOTE_DIRECTORY,
"pub");
     // set the remote file name
     req.addProperty(DeliveryPropertyDefinitions.FTP_REMOTE_FILENAME,
"test.pdf");
     // set the document
     req.setDocument("/document/test.pdf");

     // submit the request
     req.submit();
     // close the request
     req.close();
```

The following properties are supported. A String value is required unless otherwise noted.

Property	Description
FTP_HOST	Required Enter the server host name.
FTP_PORT	Optional Enter the server port number. Default is 21.
FTP_USERNAME	Required Enter the login user name to the FTP server.
FTP_PASSWORD	Required Enter the login password to the FTP server.
FTP_REMOTE_DIRECTORY	Required Enter the directory to which to deliver the document (example: /pub/)

Property	Description
FTP_REMOTE_FILENAME	Required Enter the document file name for the remote server.
FTP_BINARY_MODE	Optional Valid values are "true" (default) or "false".

Delivering Your Documents over Secure FTP

Secure FTP is the protocol based on the Secure Shell technology (ssh) and it is widely used to transfer files in a secure manner. Both Secure Shell and Secure FTP are defined by the Internet Engineering Task Force (IETF) and the specifications are available on their Web site: http://www.ietf.org. The delivery system supports the delivery of documents to secure FTP servers.

The following tables lists the supported properties. A string value is required for each property unless otherwise noted.

Example

```
// create delivery manager instance
     DeliveryManager dm = new DeliveryManager();
     // create a delivery request
     DeliveryRequest req = dm.createRequest(DeliveryManager.TYPE_SFTP);
     // set hostname of the SFTP server
     req.addProperty(DeliveryPropertyDefinitions.SFTP_HOST,
"mysftphost");
     // set username and password to access server
     req.addProperty(DeliveryPropertyDefinitions.SFTP_USERNAME,
"myname");
     req.addProperty(DeliveryPropertyDefinitions.SFTP_PASSWORD,
"mypassword");
     // set the remote directory that you want to send your document to
     req.addProperty(DeliveryPropertyDefinitions.SFTP_REMOTE_DIRECTORY,
"pub");
     // set the remote file name
     req.addProperty(DeliveryPropertyDefinitions.SFTP_REMOTE_FILENAME,
"test.pdf");
     // set the document
     req.setDocument("/document/test.pdf");

     // submit the request
     req.submit();
     // close the request
     req.close();
```

Property	Description
SFTP_HOST	Required Enter the target server host name.
SFTP_PORT	Optional Enter the target server SSH port number. Default is 22.
SFTP_USERNAME	Required Enter the login user name.
SFTP_PASSWORD	Required if you choose the SFTP_AUTH_TYPE_PASSWORD authentication type. Enter the login password.
SFTP_REMOTE_DIRECTORY	Enter the directory to which to deliver the document (example: /pub/). If no value is entered, the document will be delivered to the login directory.
SFTP_REMOTE_FILENAME	Required Enter the document file name on the remote server.
SFTP_AUTH_TYPE	Set either of the following: SFTP_AUTH_TYPE_PASSWORD (Default) Requires providing password at login. SFTP_AUTH_TYPE_PUBLIC_KEY - public key authorization type.
SFTP_PRIVATE_KEY_FILE	Enter the client private key file. Required if you choose SFTP_AUTH_TYPE_PUBLIC_KEY.
SFTP_PRIVATE_KEY_PASSWORD	Enter the client private key password. Required if you choose SFTP_AUTH_TYPE_PUBLIC_KEY.

Property	Description
SFTP_FILE_PERMISSION	Enter the permissions to set for the file being created. Default is 0755.

Authentication Modes

The secure FTP delivery supports two authentication modes: password authentication and public key authentication. Set the property SFTP_AUTH_TYPE to choose the mode. The default mode is password authentication.

```
:
        :
      // set public key auth type
      req.addProperty(DeliveryPropertyDefinitions.SFTP_AUTH_TYPE,

DeliveryPropertyDefinitions.SFTP_AUTH_TYPE_PUBLIC_KEY);
      // set username
      req.addProperty(DeliveryPropertyDefinitions.SFTP_USERNAME,
"myname");
      // set the client's private key file
      req.addProperty(DeliveryPropertyDefinitions.SFTP_PRIVATE_KEY_FILE,
             "/path/to/the/key");
      // set the client's private key password

req.addProperty(DeliveryPropertyDefinitions.SFTP_PRIVATE_KEY_PASSWORD,
"myPrivateKeyPass");
         :
         :
```

The password authentication mode requires the username and password to log in to the secure FTP server. The following example shows sample code:

Example

```
:
         :
     // set password auth type
      req.addProperty(DeliveryPropertyDefinitions.SFTP_AUTH_TYPE,

DeliveryPropertyDefinitions.SFTP_AUTH_TYPE_PASSWORD);
      // set username and password to access server
      req.addProperty(DeliveryPropertyDefinitions.SFTP_USERNAME,
"myname");
      req.addProperty(DeliveryPropertyDefinitions.SFTP_PASSWORD,
"mypassword");
         :
         :
```

The public key authorization mode requires the username, your private key and password for the private key. This is a more secure method than the password authentication. Note that in order to use the public key authentication mode, you must set up the public key in the ssh/secure FTP server in advance. The following example shows sample code:

```
:
         :
      // set public key auth type
      req.addProperty(DeliveryPropertyDefinitions.SFTP_AUTH_TYPE,

DeliveryPropertyDefinitions.SFTP_AUTH_TYPE_PUBLIC_KEY);
      // set username
      req.addProperty(DeliveryPropertyDefinitions.SFTP_USERNAME,
"myname");
      // set the client's private key file
      req.addProperty(DeliveryPropertyDefinitions.SFTP_PRIVATE_KEY_FILE,
                      "/path/to/the/key");
      // set the client's private key password

req.addProperty(DeliveryPropertyDefinitions.SFTP_PRIVATE_KEY_PASSWORD,
"myPrivateKeyPass");
         :
         :
```

Delivering Your Documents over HTTP

The Delivery Manager supports delivery of documents to HTTP servers. The following sample sends a document through the HTTP POST method. Note that the receiving HTTP server must be able to accept your custom HTTP request in advance (for example via a custom servlet or CGI program).

Example

```
// create delivery manager instance
     DeliveryManager dm = new DeliveryManager();
     // create a delivery request
     DeliveryRequest req = dm.createRequest(DeliveryManager.TYPE_HTTP);

     // set request method
     req.addProperty(DeliveryPropertyDefinitions.HTTP_METHOD,
DeliveryPropertyDefinitions.HTTP_METHOD_POST);
     // set document content type
     req.addProperty(DeliveryPropertyDefinitions.HTTP_CONTENT_TYPE,
"application/pdf");
     // set the HTTP server hostname
     req.addProperty(DeliveryPropertyDefinitions.HTTP_HOST, "myhost");
     // set the HTTP server port number
     req.addProperty(DeliveryPropertyDefinitions.HTTP_PORT, "80");
     // set the target remote directory
     req.addProperty(DeliveryPropertyDefinitions.HTTP_REMOTE_DIRECTORY,
"/servlet/");
     // set the remote filename (servlet class)
     req.addProperty(DeliveryPropertyDefinitions.HTTP_REMOTE_FILENAME,
"uploadDocument");

     // set the document
     req.setDocument("/document/test.pdf");

     // submit the request
     req.submit();
     // close the request
     req.close();
```

The following table lists the properties that are supported. A String value is required for

each property unless otherwise noted.

Property	Description
HTTP_METHOD	Optional Sets the HTTP request method. Valid values are: HTTP_METHOD_POST (Default) HTTP_METHOD_PUT
HTTP_CONTENT_TYPE	Optional The document content type (example: "application/pdf").
HTTP_HOST	Required Enter the server host name.
HTTP_PORT	Optional Enter the server port number. The default is 80.
HTTP_REMOTE_DIRECTORY	Required Enter the remote directory name (example: "/home/").
HTTP_REMOTE_FILENAME	Required Enter the file name to save the document as in the remote directory.
HTTP_AUTHTYPE	Optional Valid values for authentication type are: HTTP_AUTHTYPE_NONE - no authentication (default) HTTP_AUTHTYPE_BASIC - use basic HTTP authentication HTTP_AUTHTYPE_DIGEST - use digest HTTP authentication

Property	Description
HTTP_USERNAME	Optional If the server requires authentication, enter the username.
HTTP_PASSWORD	Optional If the server requires authentication, enter the password for the username.
HTTP_ENCTYPE	Optional Enter the encryption type: HTTP_ENCTYPE_NONE - no encryption (default) HTTP_ENCTYPE_SSL - use Secure Socket Layer
HTTP_USE_FULL_URL	Optional Set to "true" to send the full URL for the HTTP request header. Valid values are "true" or "false" (default).
HTTP_USE_CHUNKED_BODY	Optional Valid values are "true" (default) to use HTTP chunked transfer coding for the message body, or "false".
HTTP_TIMEOUT	Optional Enter a length of time in milliseconds after which to terminate the request if a connection is not made to the HTTP server. The default is 60000 (1 minute).
HTTP_URL_CHARACTER_ENCODING	Encoding of the URL. It will be used if you use non-ASCII characters in the URL. Set the Java-supported encoding string for the value.

Delivering Documents over AS2

AS2 is one of the standard protocols defined in the Electronic Data Interchange-Internet

Integration (EDI-INT). AS2 is based on HTTP and other internet standard technologies and is designed to exchange data over the internet in a secure manner. The AS2 specification is defined in RFC4130 (available at http://www.ietf.org/). The delivery system supports the delivery of documents to AS2 servers. Sample code is as follows:

Example

```
// create delivery manager instance
    DeliveryManager dm = new DeliveryManager();
    // create a delivery request
    DeliveryRequest req = dm.createRequest(DeliveryManager.TYPE_AS2);

    // set AS2 message properties
    req.addProperty(DeliveryPropertyDefinitions.AS2_FROM, "Me");
    req.addProperty(DeliveryPropertyDefinitions.AS2_TO, "You");
    req.addProperty(DeliveryPropertyDefinitions.AS2_SUBJECT, "My EDI
Message");
    req.addProperty(DeliveryPropertyDefinitions.AS2_CONTENT_TYPE,
"applications/EDIFACT");

    // set HTTP properties
    req.addProperty(DeliveryPropertyDefinitions.AS2_HTTP_HOST,
"as2hsot");

req.addProperty(DeliveryPropertyDefinitions.AS2_HTTP_REMOTE_DIRECTORY,
"/");

req.addProperty(DeliveryPropertyDefinitions.AS2_HTTP_REMOTE_FILENAME,
"as2");

    // set the document
    req.setDocument("/document/myEDIdoc");
    // submit the request
    DeliveryResponse res = req.submit();
    // close the request
    req.close();
```

The following table lists the supported properties. A string value is required for each property unless otherwise noted.

Property	Description
AS2_FROM	Required. Enter the AS2 message sender.
AS2_TO	Required. Enter the AS2 message recipient.
AS2_SUBJECT	Required. Enter the message subject.

Property	Description
AS2_MESSAGE_COMPRESSION	Default value is False. Enter True to compress the message.
AS2_MESSAGE_SIGNATURE	Default value is False. Enter True to sign the message.
AS2_MESSAGE_ENCRYPTION	Default value is False. Enter True to encrypt the message.
AS2_CONTENT_TYPE	Required. Enter the content type of the document. Valid values are: • application/EDIFACT • application/xml
AS2_ENC_ALGO	The AS2 encryption algorithm. Set one of the following: • AS2_ENC_ALGO_RC2_40 • AS2_ENC_ALGO_RC2_64 • AS2_ENC_ALGO_RC2_128 • AS2_ENC_ALGO_DES • AS2_ENC_ALGO_DES_EDE3 (Defau • AS2_ENC_ALGO_AES_128 • AS2_ENC_ALGO_AES_192 • AS2_ENC_ALGO_AES_256
AS2_DIGEST_ALGO	Enter the AS2 digest algorithm for signing the messages. Set either of the following: • AS2_DIGEST_ALGO_MD5 (Default) • AS2_DIGEST_ALGO_SHA1

Property	Description
AS2_ASYNC_ADDRESS	Enter the asynchronous address to which MDN notifications should be set.
AS2_ASYNC_EMAIL_SERVER_HOST	Enter the email server host for asynchronous email MDN.
AS2_ASYNC_EMAIL_SERVER_PORT	Enter the email server port for asynchronous email MDN.
AS2_ASYNC_EMAIL_SERVER_USERNAME	Enter the email server USERNAME for asynchronous email MDN.
AS2_ASYNC_EMAIL_SERVER_PASSWORD	Enter the email server PASSWORD for asynchronous email MDN.
AS2_ASYNC_EMAIL_SERVER_FOLDER_NAME	Enter the IMAP folder name for aynchronous email MDN.
AS2_SENDER_PKCS12_FILE	Location of the sender's PKCS12 (public/private key) file.
AS2_SENDER_PKCS12_PASSWORD	Password for the sender's PKCS12 (public/private key).
AS2_RECEIVER_CERTIFICATES_FILE	Location of the receiver's certificates file.
AS2_DELIVERY_RECEIPT_DIRECTORY	Directory to store the delivery receipts. This directory must be specified if you wish to receive delivery receipts.
AS2_HTTP_HOST	Required. Enter the server host name.
AS2_HTTP_PORT	Enter the server HTTP port number. The default is 80.
AS2_HTTP_REMOTE_DIRECTORY	Required. Enter the remote directory name. (Example: /home/)

Property	Description
AS2_HTTP_REMOTE_FILENAME	Required. Enter the remote file name.
AS2_HTTP_AUTHTYPE	Enter the HTTP authentication type. Valid values are: • AS2_HTTP_AUTHTYPE_NONE - no authentication (Default) • AS2_HTTP_AUTHTYPE_BASIC - Use HTTP basic authentication. • AS2_HTTP_AUTHTYPE_DIGEST - user HTTP digest authentication.
AS2_HTTP_USERNAME	Enter the username for HTTP authentication.
AS2_HTTP_PASSWORD	Enter the password for HTTP authentication.
AS2_HTTP_ENCTYPE	Set the encryption type. Valid values are: • AS2_HTTP_ENCTYPE_NONE - no encryption (default) • AS2_HTTP_ENCTYPE_SSL - use secure socket layer (SSL)
AS2_HTTP_TIMEOUT	Enter the time out allowance in milliseconds. Default is 60,000 (1 minute)
AS2_HTTP_PROXY_HOST	Required. Enter the proxy server host name.
AS2_HTTP_PROXY_PORT	Enter the proxy server port number. Default is 80.

Property	Description
AS2_HTTP_PROXY_AUTHTYPE	• AS2_HTTP_AUTHTYPE_NONE - no authentication (Default) • AS2_HTTP_AUTHTYPE_BASIC - Use HTTP basic authentication. • AS2_HTTP_AUTHTYPE_DIGEST - user HTTP digest authentication.
AS2_HTTP_PROXY_USERNAME	Enter the username for proxy authentication.
AS2_HTTP_PROXY_PASSWORD	Enter the password for HTTP proxy authentication.

Delivery Receipt

The AS2 server always issues an AS2 delivery receipt for each AS2 request. Set the AS2_DELIVERY_RECEIPT_DIRECTORY property to specify the location to store the delivery receipts. If you do not specify this directory, delivery receipts will be ignored. Example code for setting the delivery receipt directory is as follows:

```
:
         :
 // Set the delivery receipt directory

req.addProperty(DeliveryPropertyDefinitions.AS2_DELIVERY_RECEIPT_DIRECTO
RY, "/my/receipt/dir");
         :
         :
```

Synchrony

You can send either synchronous or asynchronous delivery requests to the AS2 servers. By default, the request is synchronous so that you can see the Message Disposition Notification (MDN) immediately in the DeliveryResponse.

If you set the AS2_ASYNC_ADDRESS to your request, the request will be asynchronous. You can specify either an HTTP URL or an e-mail address where the delivery receipt will be delivered after processing. You must set up the HTTP server or e-mail address to receive the delivery receipts.

The Delivery API can track down the asynchronous request if you specify the e-mail address for the AS2_ASYNC_ADDRESS. If you provide the e-mail account information to the Delivery API, the Delivery API will periodically check the e-mail account to obtain the delivery receipt. Sample code for this is as follows:

Example

```
:
         :
 // Set the email address - async request
 req.addProperty(DeliveryPropertyDefinitions.AS2_ASYNC_ADDRESS,
"async_target@acme.com");

 // Set the delivery receipt directory

req.addProperty(DeliveryPropertyDefinitions.AS2_DELIVERY_RECEIPT_DIRECTO
RY, "/my/receipt/dir");

 // Set the email server information where the delivery receipt will be
delivered to.
 req.addProperty(
     DeliveryPropertyDefinitions.AS2_ASYNC_EMAIL_SERVER_HOST,
"mail.acme.com");
 req.addProperty(
     DeliveryPropertyDefinitions.AS2_ASYNC_EMAIL_SERVER_USERNAME,
"async_target");
 req.addProperty(
     DeliveryPropertyDefinitions.AS2_ASYNC_EMAIL_SERVER_PASSWORD,
"mypassword");
 req.addProperty(
     DeliveryPropertyDefinitions.AS2_ASYNC_EMAIL_SERVER_FOLDER_NAME,
"inbox");

 // set the document
 req.setDocument("/document/myEDIdoc");

 // submit the request with the DeliveryResponseListener
 req.submit(myDeliveryListener);
        :
        :
```

Note that as shown in the preceding code, you must use the Delivery API's asynchronous delivery request mechanism to track down the asynchronous requests. See Asynchronous Delivery Requests, page 12-34 for more information.

Document Signing

The Delivery API allows you to sign a document for the secure transaction. This is based on the public key architecture, so you must set up the following:

- Sender side: sender's public/private keys

  Sender must have sender's public/private keys in a PKCS12 standard file. The file extension is .p12. Place that file in your local system where you want to run the Delivery API.

- Receiver side (AS2 server side) : sender's public key certificate

  The receiver must have the sender's public key certificate. Installing certificates on the AS2 server can vary depending on your server. Generally, you must copy the .der or .cer certificates to a particular location. Consult your AS2 server manual for the procedure.

Once you have completed the setup, you can sign your document by setting properties in the delivery request. Sample code for this is as follows:

```
:
        :
 // Signing the document
 req.addProperty(DeliveryPropertyDefinitions.AS2_MESSAGE_SIGNATURE,
"true");
 req.addProperty(DeliveryPropertyDefinitions.AS2_SENDER_PKCS12_FILE,
"/path/to/mykey.p12");
 req.addProperty(DeliveryPropertyDefinitions.AS2_SENDER_PKCS12_PASSWORD,
"welcome");
        :
        :
```

Document Encryption

The Delivery API allows you to encrypt documents for the secure transaction. This is based on the public key architecture, so you need to set up the following:

- Sender's side: Receiver's public key certificate

  The sender side must have the receiver's public key certificate file. The file extension is .der or .cer. Place that file in your local system where you want to run the Delivery API. Please consult the manual of your AS2 server for the procedure to obtain the AS2 server's public key certificate.

- Receiver's side (AS2 server side): Receiver's public/private keys

  The receiver side (AS2 Server) must have the receiver's public/private keys. Please consult the manual of your AS2 server for the procedure to set up keys.

Once set up, you can encrypt your document by setting properties in the delivery request. The sample code is as follows:

```
:
        :
 // Encrypting the document
 req.addProperty(DeliveryPropertyDefinitions.AS2_MESSAGE_ENCRYPTION,
"true");

req.addProperty(DeliveryPropertyDefinitions.AS2_RECEIVER_CERTIFICATES_FI
LE,  "/path/to/server-certificate.der");
        :
        :
```

Delivering Documents Using an External Command

The Delivery API supports the use of external, Operating System (OS) native commands to deliver documents.

Specify your OS native command with the `{file}` placeholder. At runtime, this placeholder will be replaced with the document file name.

The delivery status is determined by the exit value of the OS command. If the value is

'0', the request is marked successful.

Sample code is as follows:

```
// create delivery manager instance
     DeliveryManager dm = new DeliveryManager();
     // create a delivery request
     DeliveryRequest req =
dm.createRequest(DeliveryManager.TYPE_EXTERNAL);
     // set the OS native command for delivery

req.addProperty(ExternalDeliveryPropertyDefinitions.EXTERNAL_DELIVERY_CO
MMAND,
     "/usr/bin/lp -d myprinter {file}");
     // set the document
     req.setDocument("/document/test.pdf");

     // submit the request
     req.submit();
     // close the request
     req.close();
```

The following property is supported and defined in DeliveryPropertyDefinitions:

Property	Description
EXTERNAL_DELIVERY_COMMAND	Required. Enter the OS native command for delivery.

Delivering Documents to the Local File System

The Delivery API supports the delivery of documents to the local file system where the Delivery API runs. The command copies the file to the location you specify.

The following sample code copies the file /document/test.pdf to /destination/document.pdf.

```
// create delivery manager instance
     DeliveryManager dm = new DeliveryManager();
     // create a delivery request
     DeliveryRequest req = dm.createRequest(DeliveryManager.TYPE_LOCAL);
     // set the document destination in the local filesystem.

req.addProperty(ExternalDeliveryPropertyDefinitions.LOCAL_DESTINATION,
"/destination/document.pdf");
     // set the document to deliver.
     req.setDocument("/document/test.pdf");

     // submit the request
     req.submit();
     // close the request
     req.close();
```

The following property is supported and defined in DeliveryPropertyDefinitons:

Property	Description
LOCAL_DESTINATION	Required. Full path to the destination file name in the local file system.

Direct and Buffering Modes

The delivery system supports two modes: Direct mode and Buffering mode. Buffering Mode is the default.

Direct Mode

Direct Mode offers full, streamlined delivery processing. Documents are delivered to the connection streams that are directly connected to the destinations. This mode is fast, and uses less memory and disk space. It is recommended for online interactive processing.

To set the direct mode, set the BUFFERING_MODE property to "false". Following is a code sample:

Example

```
// create delivery manager instance
     DeliveryManager dm = new DeliveryManager();

     // create a delivery request
     DeliveryRequest req =
dm.createRequest(DeliveryManager.TYPE_IPP_PRINTER);

     // set the direct mode
     req.addProperty(DeliveryPropertyDefinitions.BUFFERING_MODE,
"false");
           :
           :
           :
```

This mode does not offer document redelivery. For redelivery requirements, use the buffering mode.

Buffering Mode

The buffering mode allows you to redeliver documents as many times as you want. The delivery system uses temporary files to buffer documents, if you specify a temporary directory (`ds-temp-dir`) in the delivery server configuration file. If you do not specify a temporary directory, the delivery system uses the temporary memory buffer. It is recommended that you define a temporary directory. For more information about the configuration file, see Configuration File Support, page 12-45.

You can explicitly clear the temporary file or buffer by calling `DeliveryRequest.close()` after finishing your delivery request.

Example

```
// create delivery manager instance
     DeliveryManager dm = new DeliveryManager();

     // create a delivery request
     DeliveryRequest req =
dm.createRequest(DeliveryManager.TYPE_IPP_PRINTER);

     // set buffering mode
     req.addProperty(DeliveryPropertyDefinitions.BUFFERING_MODE,
"true");
     req.addProperty(DeliveryPropertyDefinitions.TEMP_DIR, "/tmp");
           :
           :
           :
     // submit request
     req.submit();
           :
           :
     // submit request again
     req.submit();
           :
           :
     // close the request
     req.close();
```

Asynchronous Delivery Requests

The Delivery API provides the ability to run the delivery requests asynchronously by registering the callback functions.

You can create your own callback logic by implementing the DeliveryResponseListener interface. You must implement the resposeReceived() method. You can implement your logic in this method so that it will be called when the delivery request is finished. Sample code is as follows:

```
import oracle.apps.xdo.delivery.DeliveryResponseListener;

  class MyListener implements DeliveryResponseListener
  {

    public void responseReceived(DeliveryResponse pResponse)
    {
      // Show the status to the System.out
      System.out.println("Request done!");
      System.out.println("Request status id  : " +
pResponse.getStatus());
      System.out.println("Request status msg : " +
pResponse.getStatusMessage());
    }

  }
```

Once you implement the callback, you can pass your callback when you call the submit() method of your DeliveryRequest. If you call the submit() with the callback, the delivery process will start in the background and the submit() method will immediately return the control. Sample code follows:

```
// create delivery manager instance
    DeliveryManager dm = new DeliveryManager();

    // create a delivery request
    DeliveryRequest req =
dm.createRequest(DeliveryManager.TYPE_IPP_PRINTER);
          :
          :
    // submit request with the callback logic
    req.submit(new MyListener());
          :
          :
```

Document Filter Support

The Delivery API supports the document filter functionality for all the supported protocols. This functionality allows you to call the native OS command to transform the document before each delivery request. To specify the filter, pass the native OS command string with the two placeholders for the input and output filename: `{infile}` and `{outfile}`. You can set your filter in your delivery request as a delivery property. Following are two samples:

```
// The easiest filter, just copy the file :)
req.addProperty(DeliveryPropertyDefinitions.FILTER, "cp {infile}
{outfile}");


// Call "pdftops" utility to transform the PDF document into Postscript
format
 req.addProperty(DeliveryPropertyDefinitions.FILTER, "pdftops {infile}
{outfile}");
```

Alternatively, you can also specify the filter for each server in the configuration file (see Configuration File Support, page 12-45). In this case, the server will always use this filter for the requests to this server:

```
:
         :

<server name="printer1" type="ipp_printer" default="true">
<uri>ipp://myserver:80/printers/MyPrinter1/.printer</uri>
<filter>pdftops {infile} {outfile}</filter>
</server>
         :
         :
```

This is useful especially if you are trying to call IPP printers directly or IPP printers on Microsoft Internet Information Service (IIS) because those printers usually do not accept PDF documents, but only limited document formats. With this functionality, you can call any of the native OS commands to transform the document to the format that the target printer can understand. For example, if you need to call the HP LaserJet printer setup on the Microsoft IIS from Linux, you can set Ghostscript as a filter to transform the PDF document into the format that the HP LaserJet can understand.

```
// specify filter
     req.addProperty(DeliveryPropertyDefinitions.FILTER,
     "gs -q -dNOPAUSE -dBATCH -sDEVICE=laserjet -sOutputFile={outfile}
      {infile}");
```

Note that to use this functionality you must set the buffering mode must be enabled and a temporary directory must be specified. See Configuration File Support, page 12-45 for information on setting these properties.

Date Expression Support

Three properties support date expressions. Use the date expression if you want to name a file by the date, and have the date automatically set at runtime.

The properties that support date expressions are:

- SMTP_CONTENT_FILENAME
- FTP_REMOTE_FILENAME
- WEBDAV_REMOTE_FILENAME

The supported date expressions are:

- %y : 4 digit year (ex, 1972, 2005)
- %m : 2 digit month (00 - 12)
- %d : 2 digit date (00 - 31)
- %H : 24h based 2 digit hour (00 - 24)
- %M : 2 digit minute (00 - 59)
- %S : 2 digit sec (00 - 59)
- %l : 3 digit millisec (000 - 999)

For example, if you specify `my_file_%y%m%d.txt` for the filename, the actual filename will would be `my_file_20051108.txt` for November 8, 2005. All undefined expressions will be translated into 0 length string, for example, if you specify `my_file_%a%b%c.txt`, it would generate `my_file_.txt`. You can escape the '%' character by passing '%%'.

Internationalization Support

The Delivery Server API supports following internationalization features for the listed delivery channels:

SMTP

- Specify character encoding for the main document with SMTP_CONTENT_TYPE.
- Specify character encoding for the attachments by passing content type when you call addAttachment() method.
- Specify the character encoding for email To/From/CC/Subject with SMTP_CHARACTER_ENCODING property. The default value is "UTF-8".

IPP

- Specify character encoding for the IPP attributes by using IPP_ATTRIBUTE_CHARSET property. The default value is "UTF-8".
- Specify IPP_URL_CHARACTER_ENCODING property for encoding non-ASCII letters in a URL.

WebDAV

- Specify WEBDAV_URL_CHARACTER_ENCODING property for encoding non-ASCII letters in a URL.

FTP

- The FTP delivery channel automatically detects the internationalization support in the target FTP server. You can specify a non-ASCII directory name and file name only if the FTP server supports internationalization (see RFC 2640 for more detail). In that case, the UTF-8 encoding will be used automatically. If the server does not support internationalization and you specify a non-ASCII value, an exception will be thrown during the delivery process.

HTTP

- You can specify WEBDAV_URL_CHARACTER_ENCODING property for encoding non-ASCII letters in a URL.

Monitoring Delivery Status

The delivery system allows you to check the latest delivery status of your request by calling the `getStatus()` method. You can check the status of the request anytime, but currently you must retain the delivery request object. Status definitions are defined in the DeliveryRequest interface.

Monitoring delivery status is not available for the SMTP and HTTP delivery channels.

Example

```
// create delivery manager instance
     DeliveryManager dm = new DeliveryManager();

     // create a delivery request
     DeliveryRequest req =
dm.createRequest(DeliveryManager.TYPE_IPP_PRINTER);
           :
           :
     // submit request
     req.submit();
           :
           :

     // get request status
     int status = req.getStatus();
     if (status == DeliveryRequest.STATUS_SUCCESSFUL)
     {
        System.out.println("Request has been delivered successfully.");
     }
           :
           :
     // get request status again...
     status = req.getStatus();
           :
           :
```

Setting Global Properties

You can define the global properties to the DeliveryManager so that all the delivery requests inherit the global properties automatically.

The following global properties are supported:

Property	Description
BUFFERING_MODE	Valid values are "true" (default) and "false". See Direct and Buffering Modes, page 12-33 for more information.
TEMP_DIR	Define the location of the temporary directory.
CA_CERT_FILE	Define the location of the CA Certificate file generated by Oracle Wallet Manager. This is used for SSL connection with the Oracle SSL library. If not specified, the default CA Certificates are used.

Example

```
// create delivery manager instance
     DeliveryManager dm = new DeliveryManager();

     // set global properties
     dm.addProperty(DeliveryPropertyDefinitions.TEMP_DIR, "/tmp");
     dm.addProperty(DeliveryPropertyDefinitions.BUFFERING_MODE, "true");

     // create delivery requests
     DeliveryRequest req1 =
dm.createRequest(DeliveryManager.TYPE_IPP_PRINTER);
     DeliveryRequest req2 =
dm.createRequest(DeliveryManager.TYPE_IPP_FAX);
     DeliveryRequest req3 =
dm.createRequest(DeliveryManager.TYPE_SMTP_EMAIL);
           :
           :
```

Adding a Custom Delivery Channel

You can add custom delivery channels to the system by following the steps below:

1. Define the delivery properties
2. Implement the DeliveryRequest interface
3. Implement the DeliveryRequestHandler interface
4. Implement the DeliveryRequestFactory interface
5. Register your custom DeliveryRequestFactory to the DeliveryManager

The following sections detail how to create a custom delivery channel by creating a sample called "File delivery channel" that delivers documents to the local file system.

Define Delivery Properties

The first step to adding a custom delivery channel is to define the properties. These will vary depending on what you want your channel to do. You can define constants for your properties. Our example, a file delivery channel requires only one property, which is the destination.

Sample code is:

Example

```
package oracle.apps.xdo.delivery.file;

public interface FilePropertyDefinitions
  {
     /** Destination property definition.  */
     public static final String FILE_DESTINATION =
"FILE_DESTINATION:String";

   }
```

The value of each constant can be anything, as long as it is a String. It is recommend that you define the value in `[property name]:[property value type]` format so that the delivery system automatically validates the property value at runtime. In the example, the `FILE_DESTINATION` property is defined to have a String value.

Implement DeliveryRequest Interface

DeliveryRequest represents a delivery request that includes document information and delivery metadata, such as destination and other properties. To implement oracle.apps.xdo.delvery.DeliveryRequest you can extend the class oracle.apps.xdo.delivery.AbstractDeliveryRequest.

For example, to create a custom delivery channel to deliver documents to the local file system, the DeliveryRequest implementation will be as follows:

```
package oracle.apps.xdo.delivery.file;
import oracle.apps.xdo.delivery.AbstractDeliveryRequest;

public class FileDeliveryRequest extends AbstractDeliveryRequest
implements FilePropertyDefinitions
{
  private static final String[] MANDATORY_PROPS = {FILE_DESTINATION};

  /**
   * Returns mandatory property names
   */
  public String[] getMandatoryProperties()
  {
    return MANDATORY_PROPS;
  }
  /**
   * Returns optional property names
   */
  public String[] getOptionalProperties()
  {
    return null;
  }
}
```

Implement DeliveryRequestHandler Interface

DeliveryRequestHandler includes the logic for handling the delivery requests. A sample implementation of oracle.apps.xdo.delivery.DeliveryRequestHandler for the file delivery channel is as follows:

Example

```
package oracle.apps.xdo.delivery.file;
import java.io.BufferedOutputStream;
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.OutputStream;

import oracle.apps.xdo.delivery.DeliveryException;
import oracle.apps.xdo.delivery.DeliveryRequest;
import oracle.apps.xdo.delivery.DeliveryRequestHandler;
import oracle.apps.xdo.delivery.DeliveryStatusDefinitions;

public class FileDeliveryRequestHandler implements
DeliveryRequestHandler
{

  private FileDeliveryRequest mRequest;
  private boolean mIsOpen = false;
  private OutputStream mOut;

  /**
   * default constructor.
   */
  public FileDeliveryRequestHandler()
  {
  }

  /**
   * sets the request.
   */
  public void setRequest(DeliveryRequest pRequest)
  {
    mRequest = (FileDeliveryRequest) pRequest;
  }

  /**
   * returns the request.
   */
  public DeliveryRequest getRequest()
  {
    return mRequest;
  }

  /**
   * opens the output stream to the destination.
   */
  public OutputStream openRequest() throws DeliveryException
  {
    try
    {
      String filename =
        (String)
mRequest.getProperty(FileDeliveryRequest.FILE_DESTINATION);
      mOut = new BufferedOutputStream(new FileOutputStream(filename));

      mIsOpen = true;
      // set request status to open
      mRequest.setStatus(DeliveryStatusDefinitions.STATUS_OPEN);
      return mOut;
```

```
}
    catch (IOException e)
    {
      closeRequest();
      throw new DeliveryException(e);
    }

  }

  /**
   * flushes and closes the output stream to submit the request.
   */
  public void submitRequest() throws DeliveryException
  {
    try
    {
      // flush and close
      mOut.flush();
      mOut.close();
      // set request status
      mRequest.setStatus(DeliveryStatusDefinitions.STATUS_SUCCESSFUL);
      mIsOpen = false;
    }
    catch (IOException e)
    {
      closeRequest();
      throw new DeliveryException(e);
    }
  }

  /**
   * checks the delivery status.
   */
  public void updateRequestStatus() throws DeliveryException
  {

    // check if the file is successfully delivered
    String filename =
      (String)
mRequest.getProperty(FileDeliveryRequest.FILE_DESTINATION);
    File f = new File(filename);

    // set request status
    if (f.exists())
      mRequest.setStatus(DeliveryStatusDefinitions.STATUS_SUCCESSFUL);
    else

mRequest.setStatus(DeliveryStatusDefinitions.STATUS_FAILED_IO_ERROR);

  }
  /**
   * returns the request status.
   */
  public boolean isRequestOpen()
  {
    return mIsOpen;
  }

  /**
   * closes the request, frees all resources.
   */
```

```
public void closeRequest()
  {
    mIsOpen = false;
    try
    {
      if (mOut != null)
      {
        mOut.flush();
        mOut.close();
      }
    }
    catch (IOException e)
    {
    }
    finally
    {
      mOut = null;
    }
  }

}
```

Implement DeliveryRequestFactory Interface

Implement the DeliveryRequestFactory interface to register your custom delivery channel to the delivery system.

A sample implementation of oracle.apps.xdo.delivery.DeliveryRequestFactory is as follows:

Example

```
package oracle.apps.xdo.delivery.file;

import oracle.apps.xdo.delivery.DeliveryRequest;
import oracle.apps.xdo.delivery.DeliveryRequestFactory;
import oracle.apps.xdo.delivery.DeliveryRequestHandler;

public class FileDeliveryRequestFactory
implements DeliveryRequestFactory
{
  /**
   * default constructor.
   */
  public FileDeliveryRequestFactory()
  {
  }
  /**
   * returns delivery request.
   */
  public DeliveryRequest createRequest()
  {
    return new FileDeliveryRequest();
  }
  /**
   * returns delivery request handler.
   */
  public DeliveryRequestHandler createRequestHandler()
  {
    return new FileDeliveryRequestHandler();
  }
  /**
   * returns this
   */
  public DeliveryRequestFactory getFactory()
  {
    return this;
  }
}
```

Register your custom DeliveryRequestFactory to DeliveryManager

The final step is to register your custom delivery channel to the delivery system. You can register your delivery channel in two ways:

- Static method

  Use this method to register your delivery channel to the whole delivery system by specifying it in the configuration file. See Configuration File Support, page 12-45 for more information.

- Dynamic method

  Register the delivery channel to the Java VM instance by calling the Register API programmatically.

  Sample code to register the file delivery channel using the dynamic method and call the file delivery channel is as follows:

Example

```
package oracle.apps.xdo.delivery.file;

import oracle.apps.xdo.delivery.DeliveryManager;
import oracle.apps.xdo.delivery.DeliveryRequest;

public class FileDeliverySample
{
  public static void main(String[] args) throws Exception
  {
    // register the file delivery channel
    DeliveryManager.addRequestFactory("file",
"oracle.apps.xdo.delivery.file.FileDeliveryRequestFactory");

    // create delivery manager instance
    DeliveryManager dm = new DeliveryManager();
    // create a delivery request
    DeliveryRequest req = dm.createRequest("file");

    // set the destination
    req.addProperty(
      FileDeliveryRequest.FILE_DESTINATION,
      "d:/Temp/testDocument_delivered.pdf");
    // set the document to deliver
    req.setDocument("D:/Temp/testDocument.pdf");

    // submit the request
    req.submit();
    // close the request
    req.close();
  }
}
```

Configuration File Support

The delivery systems supports a configuration file to set default servers, default properties, and custom delivery channels. The location of the configuration file is

`{XDO_TOP}/resource/xdodelivery.cfg`

where `{XDO_TOP}` is a Java system property that points to the physical directory.

This system property can be set in two ways:

- Pass `-DXDO_TOP=/path/to/xdotop` to the Java startup parameter
- Use a Java API in your code, such as `java.lang.System.getProperties().put("XDO_TOP", "/path/to/xdotop")`

The system property must be defined before constructing a DeliveryManager object.

Following is a sample configuration file:

Example

```
<?xml version='1.0' encoding='UTF-8'?>
 <config xmlns="http://xmlns.oracle.com/oxp/delivery/config">
   <! -  ========================================================  - >
   <! -     servers section                                        - >
   <! -     List your pre-defined servers here.                    - >

   <! -  ========================================================  - >
   <servers>
     <server name="myprinter1" type="ipp_printer" default="true">
       <uri>ipp://myprinter1.oracle.com:631/printers/myprinter1</uri>

     </server>
     <server name="myprinter2" type="ipp_printer" >
       <host>myprinter2.oracle.com</host>
       <port>631</port>

       <uri>ipp://myprinter2.oracle.com:631/printers/myprinter2</uri>
       <authType>basic</authType>
       <username>xdo</username>
       <password>xdo</password>

     </server>
     <server name="myfax1" type="ipp_fax" default="true" >
       <host>myfax1.oracle.com</host>

       <port>631</port>
       <uri>ipp://myfax1.oracle.com:631/printers/myfax1</uri>
     </server>
     <server name="mysmtp1" type="smtp_email" default="true">

       <host>myprinter1.oracle.com</host>
       <port>25</port>
     </server>
     <server name="mysmtp2" type="smtp_email" >

       <host>mysmtp12.oracle.com</host>
       <port>25</port>
       <username>xdo</username>
       <password>xdo</password>

     </server>
   </servers>
   <! -  ========================================================  - >
   <! -     properties section                                     - >
   <! -     List the system properties here.                       - >
   <! -  ========================================================  - >
   <properties>

     <property name="ds-temp-dir">/tmp</property>
     <property name="ds-buffering">true</property>
   </properties>
   <! -  ========================================================  - >
   <! -      channels section                                      - >

   <! -      List the custom delivery channels here.               - >
   <! -  ========================================================  - >
   <channels>
     <channel
name="file">oracle.apps.xdo.delivery.file.FileDeliveryRequestFactory</ch
annel>
```

```
</channels>

 </config>
```

Defining Multiple Servers for a Delivery Channel

You can define multiple server entries for each delivery channel. For example, the preceding sample configuration file has two server entries for the "ipp_printer" delivery channel ("myprinter1" and "myprinter2").

Load a server entry for a delivery request by calling DeliveryRequest.setServer() method. Following is an example:

Example

```
// create delivery manager instance
     DeliveryManager dm = new DeliveryManager();
     // create a delivery request
     DeliveryRequest req =
dm.createRequest(DeliveryManager.TYPE_IPP_PRINTER);

     // load myprinter1 setting
     req.setServer("myprinter1");
```

Specifying a Default Server for a Delivery Channel

To define a default server for a delivery channel, specify default="true". In the configuration file example above, "myprinter1" is defined as the default sever for the "ipp_printer" delivery channel. If a user does not specify the server properties for "ipp_printer" delivery, the server properties under the default server will be used.

Supported Configuration File Properties and Elements

The following properties are supported in the `<properties>` section:

- `ds-temp-dir`: temporary directory location.
- `ds-buffering`: specify true or false for buffering mode.
- `ds-ca-cert-file`: specify the SSL certification file location.

The following elements are supported for `<server type="ipp_printer">`and `<server type="ipp_fax">`

- <host>
- <port>
- <printerName>
- <uri>
- <username>

- <password>
- <authType>
- <encType>
- <proxyHost>
- <proxyPort>
- <proxyUsername>
- <proxyPassword>
- <proxyAuthType>
- <filter>

The following elements are supported for `<server type="smtp_email">`

- <host>
- <port>
- <uri>
- <username>
- <password>
- <authType>
- <filter>

The following elements are supported for `<server type="webdav">`

- <host>
- <port>
- <uri>
- <username>
- <password>
- <authType>
- <encType>

- <proxyHost>
- <proxyPort>
- <proxyUsername>
- <proxyPassword>
- <proxyAuthType>
- <filter>

The following elements are supported for `<server type="ftp">` and `<server type="sftp">`

- <host>
- <port>
- <uri>
- <username>
- <password>
- <filter>

The following elements are supported for `<server type="external">`

- <command>
- <filter>

13

Setting Up an After Report Trigger

This chapter covers the following topics:

- Overview
- Setting Up the After Report Trigger

Overview

BI Publisher enables you to set up an HTTP notification that will execute after report generation as an after report trigger. This enables you to integrate BI Publisher with other Oracle and third-party applications such as a BPEL process, Content Management applications, or other workflow applications.

Limitations

Note that immediately upon the generation of the report in BI Publisher, the notification will execute. There is currently no ability to call back or introduce a listener or process between the report generation and the HTTP notification to your servlet.

Process Overview for Adding the After Report Trigger to a Report

The following tasks are required to complete the setup of an after report trigger for your report:

1. Create your servlet or third-party application, as described in this chapter.
2. Register your servlet URL as an HTTP delivery server in the BI Publisher Admin page. See Set Up an HTTP Server, page 7-4.
3. Create a schedule for the report, choosing HTTP Notification. See Scheduling a Report, *Oracle Business Intelligence Publisher Report Designer's Guide*.

Setting Up the After Report Trigger

When the report generation has completed BI Publisher will call the HTTP notification as a postprocess and submit the URL (that you registered as an HTTP server) with the following additional parameters:

- jobid
- report_url
- status

  Values for status are "S" for success and "F" for failure.

Your remote application can then access these parameters using BI Publisher's APIs and Web services to access the job details, including report output and XML data as shown in the following code sample:

```
String id=    request.getParameter("jobid");
      String report_url = request.getParameter("report_url");
      String status = request.getParameter("status");

      try
      {
       Scheduler sch =new SchedulerImpl();
       JobHistoryInfo[] jobs= sch.getJobHistoryInfo(id);
       for (int i = 0; i<jobs.length; i++){
       JobHistoryInfo outinfo = jobs[i];
       FileOutputStream fos = new FileOutputStream(targetDir+id+".pdf");
       byte[] buf = new byte[256];
       int read = 0;
       InputStream in  = outinfo.getDocumentOutput();

       while ((read =in.read(buf)) > 0) {
                      fos.write(buf, 0, read);
     }
                      in.close();
                      fos.close();
       }
      } catch (Exception e) {
         Logger.log(e);
      }
```

Sample Program

Following is a sample HTTP servlet that is called as an HTTP Notification. In this example, the servlet is deployed on the same server as the BI Publisher application. If your servlet is deployed on a remote server, use the BI Publisher Web service APIs to access the report details. For more information about the BI Publisher Web service APIs, see Using the BI Publisher Web Services, page 10-1.

In this sample, the servlet uses the information provided by the HTTP request as input to the BI Publisher Web services to retrieve the report output. This could then be used

to insert in an approval workflow.

```
ackage oracle.xdo.service.scheduling;

import java.io.FileOutputStream;


import java.io.IOException;
import java.io.InputStream;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import oracle.xdo.common.log.Logger;
import oracle.xdo.server.JobHistoryInfo;
import oracle.xdo.server.Scheduler;
import oracle.xdo.server.impl.SchedulerImpl;

public class HttpNotificationTest extends HttpServlet
{
 public String targetDir =
"c://oc4j1013/j2ee/home/applications/xmlpserver/xmlpserver/output/";
  public void doGet(HttpServletRequest request,
                    HttpServletResponse response) throws
ServletException, IOException
  {
    doPost(request, response);
  }
 public void doPost(HttpServletRequest request,
                     HttpServletResponse response) throws
ServletException, IOException
  {


      String id=    request.getParameter("jobid");
      String report_url = request.getParameter("report_url");
      String status = request.getParameter("status");

      try
      {
       Scheduler sch =new SchedulerImpl();
       JobHistoryInfo[] jobs= sch.getJobHistoryInfo(id);
       for (int i = 0; i<jobs.length; i++){
       JobHistoryInfo outinfo = jobs[i];
       FileOutputStream fos = new
FileOutputStream(targetDir+id+"."+getFileExtension(outinfo.getDocumentDa
taContentType()));
       byte[] buf = new byte[256];
       int read = 0;
       InputStream in  = outinfo.getDocumentOutput();

       while ((read =in.read(buf)) > 0) {
                       fos.write(buf, 0, read);
     }
                       in.close();
                       fos.close();
       }
      } catch (Exception e) {
         Logger.log(e);
      }
      }
```

```
public static String getFileExtension(String contentType)
  {
   String ext="pdf";

   if (contentType == "application/pdf")
   ext="pdf" ;

   else if (contentType == "text/html; charset=UTF-8")
   ext="html";
   return ext;
  }
```

A

Configuration File Reference

BI Publisher Configuration Files

This chapter contains reference information about the following BI Publisher configuration files:

- Runtime Configuration Properties File
- Server Configuration File

The properties in the Runtime Configuration file are set through the Runtime Configuration Properties and Font Mappings pages (see Setting Runtime Properties, page 8-1). The properties in the Server Configuration file are set through the following Admin pages:

- System Maintenance Server Configuration
- System Maintenance Scheduler Configuration
- Security Center Security Configuration

Setting Properties in the Runtime Configuration File

As of the BI Publisher Enterprise version 10.1.3.2, the runtime properties are set through the Runtime Configuration Properties page and the Font Mappings page in the Admin interface.

If you do not use the Admin interface to set the properties, BI Publisher will fall back to the properties set in this file. Therefore if you are upgrading from XML Publisher Enterprise 5.6.2, you can use the settings in your existing xdo.cfg file, although it is recommended that you migrate your settings in this file to the Admin interface.

It is important to note that the Admin interface does not update this file. Any settings in the Admin interface will take precedence over the settings in the xdo.cfg file.

File Name and Location

The configuration file is named `xdo.cfg`.

The file is located under the `<JRE_TOP>/jre/lib`, for example: `jdk/jre/lib`.

Namespace

The namespace for this configuration file is:

`http://xmlns.oracle.com/oxp/config/`

Configuration File Example

Following is a sample configuration file:

```
<config version="1.0.0"
  xmlns="http://xmlns.oracle.com/oxp/config/"><!-- Properties -->
<properties>
      <!-- System level properties -->
      <property name="system-temp-dir">/tmp</property>

      <!--  PDF compression -->
      <property name="pdf-compression">true</property>

     <!--  PDF Security -->
     <property name="pdf-security">true</property>
     <property name="pdf-open-password">user</property>
     <property name="pdf-permissions-password">owner</property>
     <property name="pdf-no-printing">true</property>
     <property name="pdf-no-changing-the-document">true</property>
    </properties>

   <!-- Font setting -->
   <fonts>
      <!-- Font setting (for FO to PDF etc...) -->
      <font family="Arial" style="normal" weight="normal">
         <truetype path="/fonts/Arial.ttf" />
      </font>
      <font family="Default" style="normal" weight="normal">
         <truetype path="/fonts/ALBANWTJ.ttf" />
      </font>

      <!--Font substitute setting (for PDFForm filling etc...) -->
      <font-substitute name="MSGothic">
         <truetype path="/fonts/msgothic.ttc" ttcno="0" />
      </font-substitute>
   </fonts>
</config>
```

How to Read the Element Specifications

The following is an example of an element specification:

```
<Element Name Attribute1="value"
              Attribute2="value"
      AttributeN="value"
   <Subelement Name1/>[occurrence-spec]
   <Subelement Name2>...</Subelement Name2>
   <Subelement NameN>...</Subelement NameN>
</Element Name>
```

The `[occurrence-spec]` describes the cardinality of the element, and corresponds to the following set of patterns:

- [0..1] - indicates the element is optional, and may occur only once.
- [0..n] - indicates the element is optional, and may occur multiple times.

Structure

The `<config>` element is the root element. It has the following structure:

```
<config version="cdata" xmlns="http://xmlns.oracle.com/oxp/config/">
   <fonts> ... </fonts> [0..n]
   <properties> ... </properties> [0..n]
</config>
```

Attributes

version — The version number of the configuration file format. Specify 1.0.0.

xmlns — The namespace for BI Publisher's configuration file. Must be `http://xmlns.oracle.com/oxp/config/`

Description

The root element of the configuration file. The configuration file consists of two parts:

- Properties (`<properties>` elements)
- Font definitions (`<fonts>` elements)

The `<fonts>` and `<properties>` elements can appear multiple times. If conflicting definitions are set up, the last occurrence prevails.

Properties

This section describes the `<properties>` element and the `<property>` element.

The <properties> element

The properties element is structured as follows:

```
<properties locales="cdata">
   <property>...
   </property> [0..n]
</properties>
```

Description

The <properties> element defines a set of properties. You can specify the locales attribute to define locale-specific properties. Following is an example:

Example

```
<!-- Properties for all locales -->
<properties>...Property definitions here...
</properties>

<!--Korean specific properties-->
<properties locales="ko-KR">
 ...Korean-specific property definitions here...
</properties>
```

The <property> element

The <property> element has the following structure:

```
<property name="cdata"> ...pcdata...
</property>
```

Attributes

name	Specify the property name.

Description

Property is a name-value pair. Specify the internal property name (key) to the name attribute and the value to the element value. See List of Properties, page A-12 for the list of the internal property names.

Example

```
<properties>
  <property name="system-temp-dir">d:\tmp</property>
  <property name="system-cache-page-size">50</property>
  <property name="pdf-replace-smart-quotes">false</property>
</properties>
```

Font Definitions

Font definitions include the following elements:

- <fonts>
- <font>
- <font-substitute>
- <truetype>

- `<type1>`

For the list of Truetype and Type1 fonts, see Predefined Fonts, page A-7.

<fonts> element

The `<fonts>` element is structured as follows:

```
<fonts locales="cdata">
   <font> ... </font> [0..n]
   <font-substitute> ... </font-substitute> [0..n]
</fonts>
```

Attributes

locales — Specify the locales for this font definition. This attribute is optional.

Description

The `<fonts>` element defines a set of fonts. Specify the locales attribute to define locale-specific fonts.

Example

```
<!-- Font definitions for all locales -->
<fonts>
  ..Font definitions here...
</fonts>

<!-- Korean-specific font definitions -->
<fonts locales="ko-KR">
... Korean Font definitions here...
</fonts>
```

<font> element

Following is the structure of the `<font>` element:

```
<font family="cdata" style="normalitalic"
weight="normalbold">
   <truetype>...</truetype>
or <type1> ... <type1>
</font>
```

Attributes

family — Specify any family name for the font. If you specify "Default" for this attribute, you can define a default fallback font. The **family** attribute is case-insensitive.

style — Specify "normal" or "italic" for the font style.

weight — Specify "normal" or "bold" for the font weight.

Description

Defines a BI Publisher font. This element is primarily used to define fonts for FO-to-PDF processing (RTF to PDF). The PDF Form Processor (used for PDF templates) does not refer to this element.

Example

```
<!-- Define "Arial" font -->
<font family="Arial" style="normal" weight="normal">
  <truetype path="/fonts/Arial.ttf"/>
</font>
```

<font-substitute> element

Following is the structure of the font-substitute element:

```
<font-substitute name="cdata">
   <truetype>...</truetype>
or <type1>...</type1>
</font-substitute>
```

Attributes

name	Specify the name of the font to be substituted.

Description

Defines a font substitution. This element is used to define fonts for the PDF Form Processor.

Example

```
<font-substitute name="MSGothic">
   <truetype path="/fonts/msgothic.ttc" ttccno=0"/>
</font-substitute>
```

<type1> element>

The form of the <type1> element is as follows:

```
<type1 name="cdata"/>
```

Attributes

name	Specify one of the Adobe standard Latin1 fonts, such as "Courier".

Description

<type1> element defines an Adobe Type1 font.

Example

```
<!--Define "Helvetica" font as "Serif" -->
<font family="serif" style="normal" weight="normal">
  <type1 name="Helvetica"/>
</font>
```

Predefined Fonts

BI Publisher provides a set of Type1 fonts and a set of TrueType fonts. You can select any of these fonts as a target font with no additional setup required.

The Type1 fonts are listed in the following table:

Type 1 Fonts

Number	Font Family	Style	Weight	Font Name
1	serif	normal	normal	Time-Roman
1	serif	normal	bold	Times-Bold
1	serif	italic	normal	Times-Italic
1	serif	italic	bold	Times-BoldItalic
2	sans-serif	normal	normal	Helvetica
2	sans-serif	normal	bold	Helvetica-Bold
2	sans-serif	italic	normal	Helvetica-Oblique
2	sans-serif	italic	bold	Helvetica-BoldObliq ue
3	monospace	normal	normal	Courier
3	monospace	normal	bold	Courier-Bold
3	monospace	italic	normal	Courier-Oblique
3	monospace	italic	bold	Courier-BoldOblique
4	Courier	normal	normal	Courier
4	Courier	normal	bold	Courier-Bold
4	Courier	italic	normal	Courier-Oblique

Number	Font Family	Style	Weight	Font Name
4	Courier	italic	bold	Courier-BoldOblique
5	Helvetica	normal	normal	Helvetica
5	Helvetica	normal	bold	Helvetica-Bold
5	Helvetica	italic	normal	Helvetica-Oblique
5	Helvetica	italic	bold	Helvetica-BoldObliq ue
6	Times	normal	normal	Times
6	Times	normal	bold	Times-Bold
6	Times	italic	normal	Times-Italic
6	Times	italic	bold	Times-BoldItalic
7	Symbol	normal	normal	Symbol
8	ZapfDingbats	normal	normal	ZapfDingbats

The TrueType fonts are listed in the following table. All TrueType fonts will be subsetted and embedded into PDF.

Number	Font Family Name	Style	Weight	Actual Font	Actual Font Type
1	Albany WT	normal	normal	ALBANYWT.ttf	TrueType (Latin1 only)
2	Albany WT J	normal	normal	ALBANWTJ.ttf	TrueType (Japanese flavor)
3	Albany WT K	normal	normal	ALBANWTK.ttf	TrueType (Korean flavor)

Number	Font Family Name	Style	Weight	Actual Font	Actual Font Type
4	Albany WT SC	normal	normal	ALBANWTS.ttf	TrueType (Simplified Chinese flavor)
5	Albany WT TC	normal	normal	ALBANWTT.ttf	TrueType (Traditional Chinese flavor)
6	Andale Duospace WT	normal	normal	ADUO.ttf	TrueType (Latin1 only, Fixed width)
6	Andale Duospace WT	bold	bold	ADUOB.ttf	TrueType (Latin1 only, Fixed width)
7	Andale Duospace WT J	normal	normal	ADUOJ.ttf	TrueType (Japanese flavor, Fixed width)
7	Andale Duospace WT J	bold	bold	ADUOJB.ttf	TrueType (Japanese flavor, Fixed width)
8	Andale Duospace WT K	normal	normal	ADUOK.ttf	TrueType (Korean flavor, Fixed width)
8	Andale Duospace WT K	bold	bold	ADUOKB.ttf	TrueType (Korean flavor, Fixed width)
9	Andale Duospace WT SC	normal	normal	ADUOSC.ttf	TrueType (Simplified Chinese flavor, Fixed width)
9	Andale Duospace WT SC	bold	bold	ADUOSCB.ttf	TrueType (Simplified Chinese flavor, Fixed width)

Number	Font Family Name	Style	Weight	Actual Font	Actual Font Type
10	Andale Duospace WT TC	normal	normal	ADUOTC.ttf	TrueType (Traditional Chinese flavor, Fixed width)
10	Andale Duospace WT TC	bold	bold	ADUOTCB.ttf	TrueType (Traditional Chinese flavor, Fixed width)

The Server Configuration Files

The server configuration properties are set through the following pages under the Admin tab:

- System Maintenance Server Configuration
- System Maintenance Scheduler Configuration
- Security Center Security Configuration

The Repository location defined under System Maintenance Server Configuration is stored in `WEB-INF/xmlp-server-config.xml`.

All other server configuration properties are stored in a second instance of `xmlp-server-config.xml`. This file is located in `Admin/Configuration/xmlp-server-config.xml` in the repository.

The Admin user interface pages write the settings to these two files. Therefore the files can be manually updated. However, this is not recommended because the user interface ensures the validity of related property settings.

> **Important:** Whenever you change any of the properties in this configuration file, you must restart the server in order for the changes to take effect.

Configuration File Structure

The file consists of the following sections: XML header, resource, scheduler, and general properties.

XML Header

The header consists of the XML declaration and the root element with the namespace attribute:

```
<?xml version="1.0" encoding="UTF-8"?>
<xmlpConfig xmlns="http://xmlns.oracle.com/oxp/xmlp">
```

Resource Section

> **Note:** The repository location is now set using the Server Configuration page. See Setting Server Configuration Options, page 6-3.

The resource section defines the location of your repository. The resource section is mandatory. The resource elements are enclosed within `<resource> </resource>` tags.

Within the `<resource>` tags you must define either the `<file>` element or the `<xdb>` element to specify the path to your reports repository.

The <file> Element

If your reports repository exists on your file system, declare the absolute file path using the `file` element with its `path` attribute as follows:

```
<file path=""/>
```

Example: `<file path="d:/reports"/>`

The <xdb> element

If your reports repository is set up on your database, declare the absolute path using the `xdb` element with its `path` attribute as follows:

```
<xdb path="">
```

Example: `<xdb path="/public/Reports">`

The xdb element requires the `<connection>` element. Within the `<connection> </connection>` tags, define the `<connectionType>`. Valid values for `<connectionType>` are "jdbc" or "jndi".

jdbc connectionType

If the `<connectionType>` is jdbc, the following elements are required:

```
<url>
<username>
<password>
<driver>
```

Example:

```
<resource>
 <xdb path="/public/Reports">
   <connection>
     <connectionType>jdbc</connectionType>
     <url>jdbc:oracle:thin:@rpts.mycompany.com:1525:ora10g</url>
     <username>scott</username>
     <password>tiger</password>
     <driver>oracle.jdbc.driver.oracleDriver</driver>
   </connection>
  </xdb>
</resource>
```

jndi connectionType

If the connection type is "jndi", the following element is required:

```
<jndiName>
```

Example:

```
<resource>
  <xdb path="/public/Reports">
   <connection>
     <connectionType>jndi</connectionType>
     <jndiName>jdbc/pool/mydb</jndiName>
   </connection>
  </xdb>
</resource>
```

General Properties

The following table lists the general properties that can be specified in the configuration file. Specify the properties according to the following syntax:

```
<property name = "PROPERTY_NAME" value="value"/>
```

Example:

```
<property name = "CACHE_EXPIRATION" value="120"/>
```

Use the Admin Server Configuration page to set these properties (see Setting Server Configuration Options, page 6-3), with the exception of Guest Folder access. Guest Folder access is now set on the Security Configuration page (see Allowing Guest Access, page 2-2).

The properties listed here are not required. If not specified, the default value will be used. The following table lists the name, valid values, default value, and description of each property.

Property Name	Values	Description
CACHE_EXPIRATION	Default: 30	Enter the expiration period for the dataset cache in minutes. For reports that execute a SQL query, you have the option of caching the dataset returned by the query. The returned dataset will remain in cache for the period specified by this property. For more information about setting this option, see Create a New Report, *Oracle Business Intelligence Publisher Report Designer's Guide*.
CACHE_SIZE_LIMIT	Default: 1000	Sets the maximum number of datasets that will be maintained in the cache at a given time.
CACHED_REPORT_LIMIT	Default: 50	Specifies the number of reports that can be cached in memory at any given time.
OUTPUT_FORMAT	html, pdf, rtf, excel, xml	The output types specified in this property will be displayed to the user by default for every report (PDF templates will still only allow PDF output). Enter each output type separated by a comma. Valid values are: html, pdf, rtf, excel, xml. This value is overridden by the Output Format types selected in the report definition. See Create a New Report, *Oracle Business Intelligence Publisher Report Designer's Guide*.
DEBUG_LEVEL	`exception` (Default), `debug`	Controls the amount of debug information generated by the system. If set to `exception`, only error information is generated. If set to `debug`, all system output is generated.

Property Name	Values	Description
GUEST_FOLDER	true (Default), false	Enables a "guest" folder for your installation. A guest folder is a public folder accessible to anyone who can view the login URL. No credentials are required to view the reports in the guest folder.
GUEST_FOLDER_NAME	Default: Guest	Sets the name of the guest folder.

The following properties must be specified if you are using an LDAP server with BI Publisher Enterprise. Set these properties from the Admin user interface. For more information about LDAP integration, see Integrating with LDAP, page 2-7.

Property Name	Values	Description
LDAP_PROVIDER_URL	Example: `ldap://myserver.mycompany.com:3060/`	Enter the URL for the LDAP server.
LDAP_PROVIDER_ADMIN_USERNAME	Example: `Admin`	Enter the administrator user name for the LDAP server.
LDAP_PROVIDER_ADMIN_PASSWORD	Example: `welcome`	Enter the administrator password for the username entered.
LDAP_PROVIDER_USER_DN	Example: `cn=xdo,dc=myserver,dc=com`	The LDAP distinguished name user suffix that distinguishes the group of users to have access to BI Publisher.

Property Name	Values	Description
LDAP_PROVIDER_FACTORY	Example: `com.sun.jndi.ldap.ctl.LdapCtxFactory`	The value of this property is the fully qualified class name of the factory class which creates the initial context for the LDAP service provider. It is used to select a particular LDAP service provider; it is not used by the provider itself. This property need not be set when the name argument to initial context methods is a URL.
LDAP_PROVIDER_GROUP_SEARCH	Example: `(&(objectclass=groupofuniquenames)(cn=*))`	The search criteria to locate the qualified groups. This will be based on your LDAP server schema.
LDAP_PROVIDER_GROUP_SEARCH_ROOT	Example: `cn=OracleDefaultDomain,cn=OracleDBSecurity,cn=Products,cn=OracleContext,dc=mypc11,dc=com`	Indicates where in the tree structure to apply the group search criteria.
LDAP_PROVIDER_GROUP_ATTR_NAME	Example: `cn`	Indicates which attribute contains the Group name.
LDAP_PROVIDER_GROUP_ATTR_MEMBER	Example: `uniquemember`	Indicates which attribute contains the member names of the Group.
LDAP_PROVIDER_GROUP_ATTR_DESCRIPTION	Example:`description`	Indicates which attribute contains the description of the Group.

The Oracle Single Sign-On properties are listed in the following table. These properties are now set through the Admin interface. For more information about setting up Single Sign-On, see Setting Up Oracle Single Sign-On, page 2-17.

Property Name	Values	Description
SINGLE_SIGN_OFF_URL	Example: `http://server1.mycompany.com:7777/pls/orasso/orasso.wwsso_app_admin.ls_logout`	Enter the Single Sign-Off URL retrieved from the SSO Partner Application Login page.

Sample Configuration Files

Following is a sample `WEB-INF/xmlp-server-config.xml` file containing repository information:

```
<?xml version="1.0" encoding="UTF-8"?>
<xmlpConfig xmlns="http://xmlns.oracle.com/oxp/xmlp">
 <resource>
   <file path="d:/reports"/>
<! -     <xdb path="/public/Reports"> - >
<! -       <connection> - >
<! -         <connectionType>jndi</connectionType> - >
<! -         <jndiName>jdbc/pool/mydb</jndiName> - >
<! -       </connection> - >
<! -     </xdb> - >
 </resource>
</xmlpConfig>
```

Following is a sample `Admin/Configuration/xmlp-server-config.xml` file containing the BI Publisher server general and LDAP properties:

```
<?xml version="1.0" encoding="UTF-8"?>
<xmlpConfig xmlns="http://xmlns.oracle.com/oxp/xmlp">

<property name="CACHE_EXPIRATION" value="120"/>
<property name="CACHE_SIZE_LIMIT" value="1000"/>
<property name="OUTPUT_FORMAT" value="html, pdf, rtf, excel, xml"/>
<property name="DEBUG_LEVEL" value="debug"/>
<property name="CACHED_REPORT_LIMIT" value="10"/>
<property name="LDAP_PROVIDER_URL"
          value="ldap://myldapserver.com:3060/"/>
<property name="LDAP_PROVIDER_ADMIN_USERNAME" value="orcladmin"/>
<property name="LDAP_PROVIDER_ADMIN_PASSWORD" value="welcome1"/>
<property name="LDAP_PROVIDER_USER_DN"
          value="cn=xdo,dc=myserver,dc=com"/>
</xmlpConfig>
```

Index