Oracle® Business Intelligence Publisher

New Features Guide

Release 10.1.3.4.2

E14667-02

July 2011

ORACLE®

Oracle Business Intelligence Publisher New Features Guide, Release 10.1.3.4.2

E14667-02

Contents

Preface

This guide describes the new features in Releases 10.1.3.4.1 and 10.1.3.4.2 of Oracle Business Intelligence Publisher (Oracle BI Publisher).

This preface contains the following sections:

Audience

This document is intended for:

- Oracle Business Intelligence Publisher administrators, report authors, and end users

Documentation Accessibility

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

Access to Oracle Support

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

Related Documents

More documentation about Oracle BI Publisher is available as follows:

- Within the product, as online help.
- On OTN, at the Oracle BI Publisher Documentation page:

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

For detailed information about new features and functionality in Oracle Business Intelligence Publisher Release 10.1.3.4 and earlier, refer to *Oracle Business Intelligence New Features Guide*.

You can also find additional information at the following locations

- The Oracle Business Intelligence Publisher Product web site:

  `http://www.oracle.com/technology/products/xml-publisher/index.html`

- The Oracle BI Enterprise Edition Product Web site:

  `http://www.oracle.com/technology/products/bi/enterprise-edition.html`

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

1

New Features in BI Publisher Release 10.1.3.4.1

This chapter describes features of BI Publisher that were introduced in Release 10.1.3.4.1. This chapter contains the following sections:

1.1 Support for Oracle WebLogic Server 10.3

The 10.1.3.4.1 release of BI Publisher introduces support for Oracle WebLogic Server 10.3. Follow the manual deployment instructions below to deploy BI Publisher to WebLogic Server 10.3. You will also need the *Oracle Business Intelligence Publisher Installation Guide* to complete the configurations required for the deployment.

1.1.1 Deploying BI Publisher to Oracle WebLogic Server 10.3

Note: Do not deploy the xmlpserver.war or xmlpserver.ear file on the WebLogic Server by uploading it from the WebLogic console, because the console deploys the application (or Web module) in an archived file format. This is problematic to a BI Publisher configuration, because you must update WEB-INF/xmlp-server-config.xml manually after the deployment. To work around this issue, use an "exploded archive" directory, as described in this procedure.

1. Create a new WebLogic Server domain using a JDK such as Sun JDK 1.5.x or higher.
2. Create an exploded archive directory from the xmlpserver.ear or xmlpserver.war file using the new domain, by following these steps:
   a. Copy the xmlpserver.ear or xmlpserver.war file in the \xmlpserver\web directory to your destination directory, as shown in the following example.

      ```
      -mkdir c:mydestination\xmlpserver\
      -cp c:\xmlpserver\web\xmlpserver.war to
      c:mydestination\xmlpserver\
      ```

   b. Manually unpack the xmlpserver.ear or xmlpserver.war file in your destination folder using a jar command, as shown in the following example.

      ```
      cd c:mydestination\xmlpserver\
      jar -xvf xmlpserver.war
      ```

      This is the directory where the application will be deployed.
3. Set up the BI Publisher repository.

   To set up the BI Publisher repository, copy the XMLP repository to a location on your server, then edit the configuration file to point to the saved location, as follows:
   a. Copy the XMLP repository directory from \manual on your installation media to your server.
   b. Open the xmlp-server-config.xml file located in the `c:mydestination\xmlpserver\WEB-INF\`

      directory where you unpacked the ear or war file.
   c. Replace `${oracle.home}/xdo/repository` with the file path to the location where you copied the XMLP repository directory on your server. For example, if you copied the repository directory to `/home/repository/XMLP`, update the xmlp-server-config.xml file as follows:

      ```
      <?xml version="1.0" encoding="UTF-8"?>
      <xmlpConfig xmlns="http://xmlns.oracle.com/oxp/xmlp">
      <resource>
      <file path="/home/repository/XMLP"/>
      </resource>
      </xmlpConfig>
      ```

   d. Save xmlp-server-config.xml after updating.
4. In a browser, open the WebLogic Administration console (for example: `http://hostname:7001/console`).
5. If you have not already done so, in the **Change Center** of the Administration Console, click **Lock & Edit**.
6. In the left pane of the Administration Console, click **Deployments**.
7. In the right pane, click **Install**.
8. Select `c:mydestination\xmlpserver\xmlpserver` and click **Next**.
9. Select "Install this deployment as an application" and click **Next.**
10. Select "I will make the deployment accessible from the following location" from Source accessibility.
11. Click **Finish**.

12. To activate these changes, in the **Change Center** of the Administration Console click **Activate Changes**.
13. Install the BI Publisher fonts as described in the section "Configuring the BI Publisher Repository and Installing Fonts," in the *Oracle Business Intelligence Publisher Installation Guide*.

1.1.2 Required Settings for the WebLogic Start Script

There are two updates you must make to the WebLogic start script (`startWebLogic.sh`) for the domain that contains the BI Publisher server. The start script is located as follows:

```
/weblogic_home/user_projects/domains/base_
domain/bin/startWebLogic.sh
```

- WebLogic Server 10.3 includes two implementations of SAAJ (SOAP with Attachments API for Java). In order for BI Publisher to work properly, you must ensure that your WebLogic Server start script points to the SOAP Message Factory in the SAAJ 1.3 implementation. The SAAJ 1.3 implementation is found in package weblogic.xml.saaj.

  To ensure that your deployment is using the correct version, include the following setting in startWebLogic.sh :

  ```
  Djavax.xml.soap.MessageFactory=weblogic.xml.saaj.MessageFactoryImpl
  ```

- The following setting is required to enable BI Publisher to find the TopLink JAR files to create the Scheduler tables:

  ```
  -Dtoplink.xml.platform=oracle.toplink.platform.xml.jaxp.JAXPPlatform
  ```

1.2 Extended Support for Single Sign-On Providers

In previous versions of BI Publisher only Oracle Single Sign-On was supported.

Earlier versions of BI Publisher also required that if an administrator configured BI Publisher with Oracle Single Sign-On that BI Publisher must also use Oracle Internet Directory as the security model. This restriction is removed in 10.1.3.4.1 of BI Publisher. Customers may now use any of the supported security models when implementing Single Sign-On. Please note that it is the customer's responsibility to ensure that the user community of the security model matches the user community from the Single Sign-On server.

To configure BI Publisher to use Single Sign-On you must enter your Single Sign-On server information in BI Publisher's Security Configuration page under the Admin tab. You will need to provide details on how the Single Sign-On server passes the user name and the user locale to BI Publisher as well as the parameter names for both user name and user locale. For the certified Single Sign-On servers, the configuration page will provide default values for these configuration settings.

1.2.1 Configuring Single Sign-On for BI Publisher

1. Create a BI Publisher Local Superuser.

   Before performing any security updates, set up a BI Publisher Local Superuser to ensure access to BI Publisher regardless of your selected security configuration.

See "Defining a Local Superuser" in the BI Publisher Help or in the *Oracle Business Intelligence Publisher Administrator's and Developer's Guide* for more information.

2. Configure the Single Sign-On server to include BI Publisher as a partner or protected application. See your Single Sign-On server documentation for details.
3. BI Publisher requires that certain locations are unprotected by your Single Sign-On application to enable communication between the BI Publisher server and external components. To enable the following services, you must unprotect the specified location. Follow the guidelines in your Single Sign-On server documentation for unprotecting specific locations.

Location to Unprotect	Needed For:
/xmlpserver/services/	Web service communication between BI Publisher and its client components: Template Builder and Excel Analyzer Web service communication between BI Publisher and BI Presentation Services
/xmlpserver/report_service/	Also needed for Web service communication between BI Publisher and its client components: Template Builder and Excel Analyzer
/xmlpserver/ReportTemplateService.xls/	Web service communication between BI Publisher and Excel Analyzer
/xmlpserver/Guest/	Guest folder access to users not signed on through SSO

4. Log in to BI Publisher with Administrator privileges. Navigate to the **Admin tab**, then select **Security Configuration**.
5. In the Single Sign-On region of the page, select **Use Single Sign-On**.
6. In the details of the Single Sign-On region enter the following:
   - Select from the certified types of certified Single Sign-On servers:

Oracle Single Sign On
Oracle Access Manager
Siteminder

   - Optionally enter the **Single Sign-Off URL**.
   - Based on the type of server you selected, the following fields will be automatically populated with default values:
     - **How to get Username**
     - **User Name Parameter**
     - **How to get User Locale**
     - **User Locale Parameter**
7. If your Single Sign-On server is configured to use different parameters or methods, you can delete the default values and enter the appropriate values for your system.
8. Restart BI Publisher.

1.3 Automatic Refresh of LDAP Cache

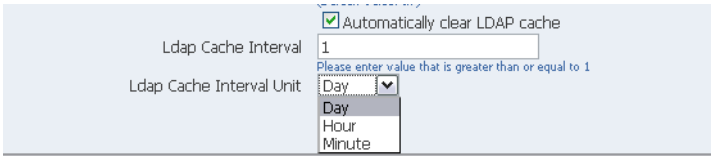
The LDAP Security Model configuration has been enhanced to enable you to schedule the automatic refresh of the LDAP cache. Previously in order to refresh the LDAP

cache, the administrator had to navigate to the Refresh Metadata function and click the Synchronize button. With this enhancement, the LDAP cache can be automatically refreshed per a designated interval.

To activate the LDAP cache refresh:

1. Navigate to the Security Configuration page By clicking the **Admin** tab, then choosing **Security Configuration**.
2. Scroll down to your LDAP settings.

   The three new fields added to support this new feature are shown in the following figure:



3. Select **Automatically clear LDAP cache**.
4. Enter an integer for **Ldap Cache Interval**. For example, to clear the LDAP cache once a day, enter 1.
5. Select the appropriate **Ldap Cache Interval Unit**: Day, Hour, or Minute.
6. Restart BI Publisher.

1.4 Support for Siebel CRM Security

If you are using BI Publisher integrated with Siebel Customer Relationship Management (CRM) you can now use the Siebel security model for BI Publisher.

To configure BI Publisher to integrate with Siebel security, perform the following three general tasks (described in more detail below):

1. Set up BI Publisher roles as Siebel CRM responsibilities.
2. Configure BI Publisher to user Siebel Security.
3. Authorize Siebel responsibilities to access data sources and folders in BI Publisher.

1.4.1 Setting Up BI Publisher Roles as Siebel CRM Responsibilities

1. Using Siebel Administrator credentials, navigate to Administration - Application, and then Responsibilities.
2. In the Responsibilities list, add a new record for each of the BI Publisher functional roles:
   - XMLP_ADMIN - this is the administrator role for the BI Publisher.
   - XMLP_DEVELOPER - enables users to build reports in the system.
   - XMLP_SCHEDULER - enables users to schedule reports.
   - XMLP_ANALYZER_EXCEL - enables users to use the Excel analysis feature.
   - XMLP_ANALYZER_ONLINE - enables users to use the online analysis feature.

- XMLP_TEMPLATE_BUILDER - enables users to connect to the BI Publisher server from the Template Builder for Microsoft Word and to upload and download templates directly from Microsoft Word.

3. Assign these responsibilities to the appropriate users. You may also want to create additional reporting responsibilities that you can utilize when setting up your report privileges in the BI Publisher application.

   Ensure to assign the XMLP_ADMIN role to a user with administration privileges.

1.4.2 Configure BI Publisher to Use Siebel Security

1. In the BI Publisher application, log in with Administrator privileges. From the **Admin** tab select **Security Configuration.**
2. Define a Local Superuser in BI Publisher:
   - Under **Local Superuser**, select the box and enter the credentials for the Superuser.

   The Local Superuser credentials will enable you to access BI Publisher in case of issues with the configured security model.
3. In the **Security Model** section of the page, select **Siebel Security** from the list.
4. Provide the following connection information:
   - Siebel Web Service Endpoint
   - Administrator Username
   - Administrator Password
5. Restart BI Publisher for the security changes to take effect.

1.4.3 Authorize Siebel Responsibilities to Access BI Publisher Data Sources and Folders

1. After you restart BI Publisher, log in to BI Publisher using the Siebel user credentials to which you assigned the XMLP_ADMIN role.
2. From the **Admin** tab, under **Security Center**, select **Roles and Permissions**. This will invoke the **Security Center** page. Here you can see the Siebel CRM responsibilities to which you assigned BI Publisher roles.
3. To add report folder access for a role, select the **Add Folders** icon. From the **Add Folders** page, use the shuttle buttons to select the appropriate folders for the role.
4. To add data source access for a role, select the **Add Data Sources** icon. From the **Add Data Sources** page, use the shuttle buttons to select the appropriate data sources for the role.

For more information about BI Publisher's roles and permissions, see the topic "Understanding BI Publisher's Users and Roles" in the *Oracle Business Intelligence Publisher Administrator's and Developer's Guide*.

1.5 Support for Secure File Transfer Protocol (SFTP) for Burst Reports

BI Publisher now supports Secure FTP (SFTP) as a delivery channel for burst reports. To enable SFTP for your burst report, follow the guidelines for defining the delivery data set for FTP, then specify "true" as the value for parameter 6.

The parameter mappings for the delivery data set are shown in the following table:

Channel	Parameter 1	Parameter 2	Parameter 3	Parameter 4	Parameter 5	Parameter 6
FTP	Server Name	User Name	Password	Remote Directory	Remote File Name	Secure Value: true

For the complete description of how to enable a report for bursting, see the topic: "Enabling Bursting" in the *Oracle Business Intelligence Publisher Report Designer's Guide*.

1.6 Data Model Enhancements

The following enhancements for Web service and SQL Query data set types are included in this release:

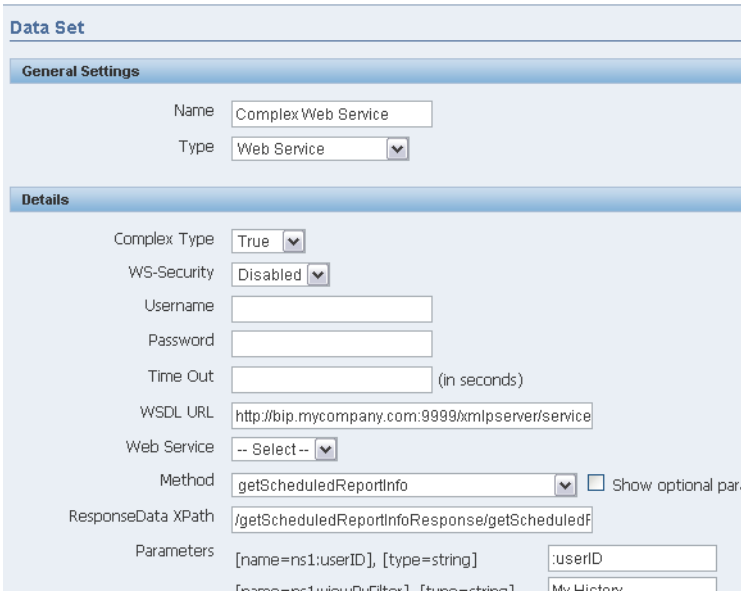
- Section 1.6.1, "Web Service Data Set Type Enhancement: Specify Path to Data in SOAP Response"
- Section 1.6.2, "SQL Query Data Set New Property: Use Default Schema Only"

1.6.1 Web Service Data Set Type Enhancement: Specify Path to Data in SOAP Response

When using a Web service as a data source, the start of the XML data that you wish to use for your report can often be deeply embedded in the response XML generated by the Web service request. This can make it difficult to work with the data in the BI Publisher template design tools.

In this release, a new field has been added to the Web Service data set type definition: **ResponseData XPath**. Use this field to specify the path to the data that you wish to use in your BI Publisher report.

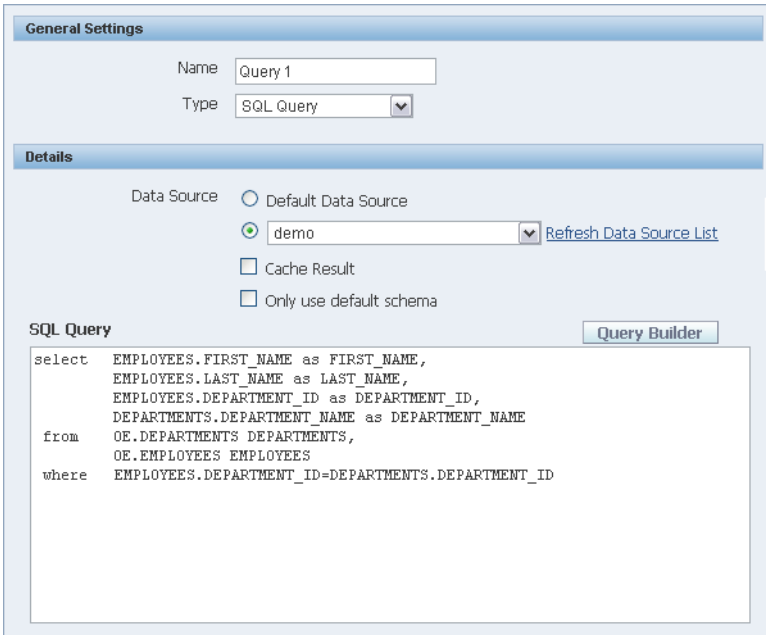
The new field is shown in the following figure of the Web Service data set definition page:



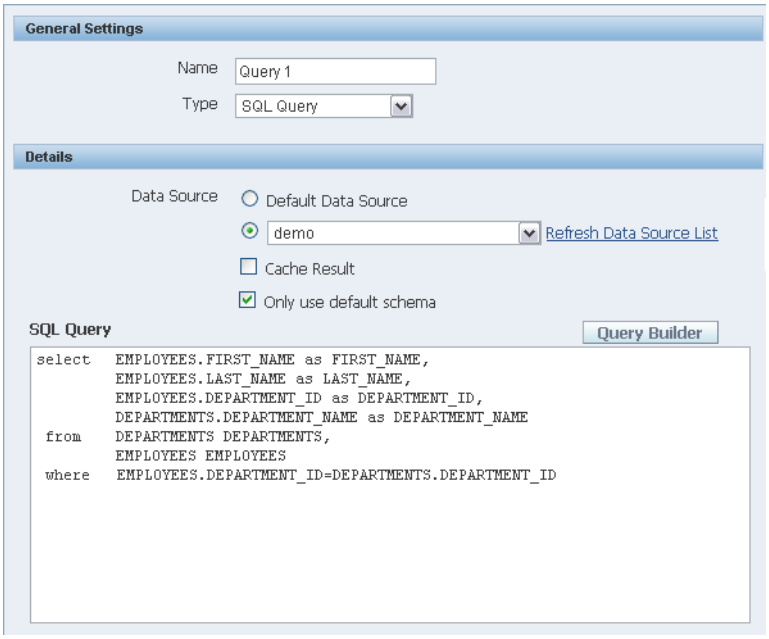
1.6.2 SQL Query Data Set New Property: Use Default Schema Only

When building a SQL query data set type using BI Publisher's Query Builder, by default the schema name is prepended to the table name from which you are selecting data. In this release, a new option has been added to the SQL query data set type definition: **Only use default schema**. When you enable this option, the query builder constructs the query without the schema name prepended to the table name.

For example, the following figure shows a sample query with the new property not enabled. Note that the table names are specified with the schema name prepended: OE.DEPARTMENTS and OE.EMPLOYEES



The next figure shows the same query with the new property enabled. Note that the table names are now specified as DEPARTMENTS and EMPLOYEES.



Enable this property when your development environment schema may be different from your production environment schema. Enabling this property makes your query portable against different schema.

Important Considerations

Because the query no longer includes the schema identifier, you cannot use this property if your query requires table joins across different schema.

Because the Query Builder builds the query based on the setting of this option, ensure to enable this property before you build the query with the Query Builder.

1.7 Support for JNDI for Scheduler Connections

In environments with many simultaneous scheduled jobs or requests to display completed reports from job history, the JDBC connection to the scheduler database may get overwhelmed causing performance to be sluggish. To enable better performance the scheduler database now supports using a JNDI connection pool defined on the application server.

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 the Scheduler Configuration page. For information on setting up a connection pool in OC4J, see the chapter "Data Sources" in the *Oracle Containers for J2EE Services Guide 10g*.

To connect the Scheduler to your database via a JNDI connection pool

Perform the following:

1. Define a JNDI data source in your application server. Select JDBC connection pool and set an appropriate number of connections for the pool size. The optimum number of connections will depend on the volume of report scheduling required at your site.
2. From the BI Publisher **Admin** tab, navigate to the **Scheduler Configuration** page.
3. For the **Database Connection Type**, select jndi.
4. Enter the **JNDI Name** that you defined on your application server.
5. Test the connection. If it is successful, select **Install Schema** to install the Scheduler schema to your database.
6. Restart BI Publisher.

1.8 Support for Expressions to Calculate Date Parameters

Previously when you created a schedule for a report that included date parameters in the data model, each time the report ran under the schedule, the original entries for the data parameters could not be updated. Now you can create a schedule for a report with date parameters and use a function in the data parameter field to programmatically calculate the dates each time the report runs under that schedule.

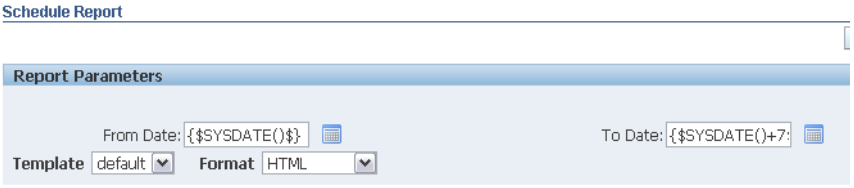
Enter one of the following functions using the syntax shown to calculate the appropriate date at the scheduled runtime for the report:

- `{$SYSDATE()$}` - current date (the system date of the server on which BI Publisher is running)
- `{$FIRST_DAY_OF_MONTH()$}` - first day of the current month
- `{$LAST_DAY_OF_MONTH()$}` - last day of the current month
- `{$FIRST_DAY_OF_YEAR)$}` - first day of the current year
- `{$LAST_DAY_OF_YEAR)$}` - last day of the current year

Note that you can also enter expressions using the plus sign "+" and minus sign "-" to add or subtract days as follows:

- `{$SYSDATE()+1$}`
- `{$SYSDATE()-7$}`

The following figure shows a sample of the Scheduler page with the date expressions entered for the parameter values:



The date function calls in the parameter values are not evaluated until the report is executed by the Scheduler.

The date functions can also be set up as default parameter values in the data model. In this case, every time a user views the report from the report viewer, the date parameters will be calculated according to the expression supplied for the default value.

1.9 RTF Template Enhancements

This release includes the following enhancements for RTF template development:

- Section 1.9.1, "Get List of BI Publisher Configuration Settings"
- Section 1.9.2, "Enable Debug Mode for an RTF Template"
- Section 1.9.3, "Number to Word Conversion in Report Output"

1.9.1 Get List of BI Publisher Configuration Settings

This release introduces a new function that can be inserted into an RTF template to return the list of configuration properties and values.

Insert the following command in an RTF template to return the list of configuration settings:

```
<?xdoxslt:getXDOProperties($_XDOCTX)?>
```

Note that if the report has been configured to override the system settings, the report-specific settings will be returned.

1.9.2 Enable Debug Mode for an RTF Template

This release introduces a new command for RTF templates to enable debug mode for the specific template that contains the command. Setting this property during testing of a template will enable you to receive targeted error information about a specific template without impacting system debug settings.

The new command is `<?xdo-debug-level: debug_level?>` where the valid values are:

```
<?xdo-debug-level:"STATEMENT"?>
<?xdo-debug-level:"ERROR"?>
<?xdo-debug-level:"OFF"?>
```

Note that *debug_level* is not case sensitive, `<?xdo-debug-level:"STATEMENT"?>` and `<?xdo-debug-level:"statement"?>` are both valid.

The debug log directory will be generated in one of the following locations:

- <system-temp-dir>/xdodebug
- <java.io.tmpdir>/xdodebug (if system-temp-dir is not defined

---

Note: This command can also be used in an XSL template or FO file by adding the following comment:

```
<!-- xdo-debug-level="STATEMENT" -->
```

The comment must reside within the first 512 bytes in the file.

1.9.3 Number to Word Conversion in Report Output

This release introduces a function that enables the conversion of numbers to words for RTF template output. This is a common requirement for check printing.

The new function is "to_check_number". The syntax of this function is

```
<?xdofx:to_check_number(amount, precisionOrCurrency, caseType, decimalStyle)?>
```

The following table describes the function attributes:

Attribute	Description	Available Value
amount	The number to be transformed.	Any number
precisionOrCurrency	For this attribute you can specify either the precision, which is the number of digits after the decimal point; or the currency code, which will govern the number of digits after the decimal point. The currency code does not generate a currency symbol in the output.	An integer, such as 2; or a currency code, such as 'USD'.
caseType	The case type of the output.	Valid values are: 'CASE_UPPER', 'CASE_LOWER', 'CASE_INIT_CAP'
decimalStyle	Output type of the decimal fraction area.	Valid values are: 'DECIMAL_STYLE_FRACTION1', 'DECIMAL_STYLE_FRACTION2', 'DECIMAL_STYLE_WORD'

The following examples display the function as entered in an RTF template and the returned output:

RTF Template Entry: <?xdofx:to_check_number(12345.67, 2)?>

Returned Output: Twelve thousand three hundred forty-five and 67/100

RTF Template Entry: <?xdofx:to_check_number(12345.67, 'USD')?>

Returned Output: Twelve thousand three hundred forty-five and 67/100

RTF Template Entry: <?xdofx:to_check_number(12345, 'JPY', 'CASE_UPPER')?>

Returned Output: TWELVE THOUSAND THREE HUNDRED FORTY-FIVE

RTF Template Entry: <?xdofx:to_check_number(12345.67, 'EUR', 'CASE_LOWER', 'DECIMAL_STYLE_WORDS')?>

Returned Output: twelve thousand three hundred forty-five and sixty-seven

1.10 Remove Logos and Links from the BI Publisher Header

If you are integrating BI Publisher into a custom application, you may want to remove the BI Publisher header to integrate more seamlessly into your enterprise applications.

In this release, BI Publisher introduces a configuration property to enable you to remove the header information from the BI Publisher application.

To remove the BI Publisher header region:

1. Navigate to the configuration file located at: *<BI Publisher repository>*/Admin/Configuration/xmlp-server-config.xml
2. Add the following property and value to the file:

```
<property name="DISPLAY_HEADER_LOGO_MENU" value="false"/>
```

3. Restart BI Publisher.

1.11 Disable Access to Guest Page

Currently, if you have activated Guest user access, then any user can access the BI Publisher Guest report page. If your company does not wish to expose the BI Publisher Guest report page to all users, you can now restrict access to the guest page to only authorized users.

This feature enables you to restrict access by defining a mandatory role that users must have to access the Guest page.

1. Navigate to the *<BI Publisher repository>*/Admin/Configuration/xmlp-server-config.xml file.
2. Add the following property to the file:

   ```
   <property name="MANDATORY_USER_ROLE" value="ROLE_NAME"/>
   ```

   where ROLE_NAME is the name of the role that is required for a user to access the guest page.
3. Restart BI Publisher server.

Any user without this new role who attempts to access the BI Publisher Guest log in page will receive the HTTP 403 Forbidden Error.

> **Note:** Once an unauthorized user has attempted to log in, the browser must be closed before an authorized user can log in.

1.12 Updates to the BI Publisher Web Service

In release 10.1.3.4.1, BI Publisher introduces a new version of PublicReportService, called PublicReportService_v11. This Web service includes all the functionality available in PublicReportService, plus introduces additional features.

If you have implemented PublicReportService and do not wish to make any changes to your existing implementation, you can continue to use PublicReportService when you uptake BI Publisher 10.1.3.4.1.

If this is the first time you are using the BI Publisher Web service, use PublicReportService_v11 to get the latest functionality.

This section describes the changes between PublicReportService (as documented in release 10.1.3.4) and PublicReportService_v11. For the full description of PublicReportService, see the *Oracle Business Intelligence Publisher Administrator's and Developer's Guide*.

1.12.1 Summary of Changes

This topic contains the following sections to describe the changes between PublicReportService and PublicReportService_v11:

1.12.2 New Complex Types

The following complex types are new in PublicReportService_v11:

- BIPDataSource
- JDBCDataSource
- FileDataSource
- DeliveryServiceDefinition
- TemplateFormatLabelValue
- ArrayOfTemplateFormatLabelValue
- TemplateFormatsLabelValues
- ArrayOfTemplateFormatsLabelValues

1.12.2.1 BIPDataSource

The new complex type BIPDataSource has been added to enable you to dynamically specify a data source when using the runReport operation. The definition is as follows:

```
  <complexType name="BIPDataSource">
        <sequence>
          <element name="JDBCDataSource" nillable="true"
type="impl:JDBCDataSource"/>
          <element name="fileDataSource" nillable="true"
type="impl:FileDataSource"/>
        </sequence>
      </complexType>
```

BIPDataSource is used by the ReportRequest structure.

The following table describes the elements:

Parameter	Type	Description
JDBCDataSource	JDBCDataSource	Contains the elements to describe a JDBC data source. See Section 1.12.2.2, "JDBCDataSource."
fileDataSource	FileDataSource	Contains the elements to describe a file data source. See Section 1.12.2.3, "FileDataSource."

1.12.2.2 JDBCDataSource

The new complex type JDBCDataSource has been added to enable you to specify a JDBC data source. The elements that make up the JDBCDataSource structure are as follows:

```
<complexType name="JDBCDataSource">
        <sequence>
          <element name="JDBCDriverClass" nillable="true" type="xsd:string"/>
          <element name="JDBCDriverType" nillable="true" type="xsd:string"/>
          <element name="JDBCPassword" nillable="true" type="xsd:string"/>
          <element name="JDBCURL" nillable="true" type="xsd:string"/>
          <element name="JDBCUserName" nillable="true" type="xsd:string"/>
          <element name="dataSourceName" nillable="true" type="xsd:string"/>
        </sequence>
      </complexType>
```

This structure is used by the BIPDataSource structure.

The elements are described in the following table:

Parameter	Type	Description
JDBCDriverClass	string	The JDBC driver class for the data source, for example: "oracle.jdbc.OracleDriver"
JDBCDriverType	string	The driver type as defined in the BI Publisher data source definition page, for example: "Oracle 9i/10g/11g".
JDBCPassword	string	The password for the data source as defined in the BI Publisher data source definition page.
JDBCURL	string	The connection string for the data source, for example: "jdbc:oracle:thin:@mydatabase.mycompany.com:1521:orcl".
JDBCUserName	string	The user name for the data source as defined in the BI Publisher data source definition page.
dataSourceName	string	The Data Source Name assigned to the data source in the BI Publisher data source definition page, for example: "Oracle".

1.12.2.3 FileDataSource

The new complex type FileDataSource has been added to specify a file data source for your report. You can specify a direct path to a location on your server, or indicate that the file is in the temporary directory. The elements that make up FileDataSource are as follows:

```
<complexType name="FileDataSource">
        <sequence>
          <element name="dynamicDataSourcePath" nillable="true"
type="xsd:string"/>
          <element name="temporaryDataSource" type="xsd:boolean"/>
        </sequence>
      </complexType>
```

This structure is used by the BIPDataSource structure.

The elements are described in the following table:

Parameter	Type	Description
dynamicDataSourcePath	string	If you want to specify a path to a data source that resides on an available server, specify the full path to the data source and set temporaryDataSource to "false". For example: "D:\BI\OracleBI\xmlp\XMLP\DemoFiles\Balance.xml")" If the file is located in the system temporary directory, set temporaryDataSource to true, and specify the file name here. For example: "Balance.xml".
temporaryDataSource	boolean	Set to true when the file data source is in the system temporary directory. Set to false when dynamicDataSourcePath specifies the full path.

1.12.2.4 DeliveryServiceDefinition

The new complex type DeliveryServiceDefinition is returned in the response for the getDeliveryServiceDefinition operation. This operation enables you to obtain information about the delivery servers set up for BI Publisher. The definition is as follows:

```
<complexType name="DeliveryServiceDefinition">
        <sequence>
          <element name="EMailServerNames" nillable="true"
type="tns1:ArrayOfString"/>
          <element name="FTPServerNames" nillable="true"
type="tns1:ArrayOfString"/>
```

```
          <element name="HTTPServerNames" nillable="true"
type="tns1:ArrayOfString"/>
          <element name="SFTPServerNames" nillable="true"
type="tns1:ArrayOfString"/>
          <element name="defaultServerNames" nillable="true"
type="tns1:ArrayOfString"/>
          <element name="faxServerNames" nillable="true"
type="tns1:ArrayOfString"/>
          <element name="printerNames" nillable="true" type="tns1:ArrayOfString"/>
          <element name="webDAVServerNames" nillable="true"
type="tns1:ArrayOfString"/>
        </sequence>
      </complexType>
```

The following table describes the elements:

Parameter	Type	Description
EMailServerNames	ArrayOfString	The list of e-mail server names returned in the ArrayOfString structure.
FTPServerNames	ArrayOfString	The list of FTP server names returned in the ArrayOfString structure.
HTTPServerNames	ArrayOfString	The list of HTTP server names returned in the ArrayOfString structure.
SFTPServerNames	ArrayOfString	The list of SFTP server names returned in the ArrayOfString structure.
defaultServerNames	ArrayOfString	The list of the default server names for each type defined returned in the ArrayOfString structure.
faxServerNames	ArrayOfString	The list of fax server names returned in the ArrayOfString structure.
printerNames	ArrayOfString	The list of printer names returned in the ArrayOfString structure.
webDAVServerNames	ArrayOfString	The list of WebDAV server names returned in the ArrayOfString structure.

1.12.2.5 TemplateFormatLabelValue

The new complex type TemplateFormatLabelValue has been added to specify the template format labels and values for a report. TemplateFormatLabelValue is included in the "ArrayOfTemplateFormatLabelValue" structure. The elements that make up TemplateFormatLabelValue are as follows:

```
<complexType name="TemplateFormatLabelValue">
        <sequence>
          <element name="templateFormatLabel" nillable="true" type="xsd:string"/>
          <element name="templateFormatValue" nillable="true" type="xsd:string"/>
        </sequence>
      </complexType>
```

The elements are described in the following table:

Parameter	Type	Description
templateFormatLabel	string	The label that displays for a template format. Examples include: "HTML" "PDF" "Excel"

Parameter	Type	Description
templateFormatValue	string	The template format value that corresponds to the label. Examples include: "html" "pdf" "excel"

1.12.2.6 ArrayOfTemplateFormatLabelValue

The new complex type ArrayOfTemplateFormatLabelValue is an array structure to include the TemplateFormatLabelValue label-value pairs. ArrayOfTemplateFormatLabelValue is included in the TemplateFormatsLabelValues structure. The elements that make up ArrayOfTemplateFormatLabelValue are as follows:

```
<complexType name="ArrayOfTemplateFormatLabelValue">
        <sequence>
          <element name="item" type="tns1:TemplateFormatLabelValue" minOccurs="0"
maxOccurs="unbounded"/>
        </sequence>
      </complexType>
```

The elements are described in the following table:

Parameter	Type	Description
item	TemplateFormatLabelValue	Contains the TemplateFormatLabelValue label-value pairs. See Section 1.12.2.5, "TemplateFormatLabelValue"

1.12.2.7 TemplateFormatsLabelValues

TemplateFormatsLabelValues is a new complex type that provides detailed information about template formats stored in the BI Publisher repository. TemplateFormatsLabelValues is included in the ArrayOfTemplateFormatsLabelValues structure.

The elements that make up TemplateFormatsLabelValues are as follows:

```
<complexType name="TemplateFormatsLabelValues">
        <sequence>
          <element name="listOfTemplateFormatLabelValue" nillable="true"
type="tns1:ArrayOfTemplateFormatLabelValue"/>
          <element name="templateAvailableLocales" nillable="true"
type="tns1:ArrayOfString"/>
          <element name="templateID" nillable="true" type="xsd:string"/>
          <element name="templateType" nillable="true" type="xsd:string"/>
          <element name="templateURL" nillable="true" type="xsd:string"/>
        </sequence>
      </complexType>
```

The elements are described in the following table:

Parameter	Type	Description
listOfTemplateFormatLabelValue	ArrayOfTemplateFormatLabelValue	Contains the TemplateFormatLabelValue label-value pairs. See Section 1.12.2.6, "ArrayOfTemplateFormatLabelValue"
templateAvailableLocales	ArrayOfString	The available locale options defined for a template passed in the ArrayOfString object.
templateID	string	The name assigned to the template in BI Publisher, for example: "Employee Listing".

Parameter	Type	Description
templateType	string	The type of template, for example: "rtf" or "pdf".
templateURL	string	The template file name in the BI Publisher repository, for example: "Employee Listing.rtf".

1.12.2.8 ArrayOfTemplateFormatsLabelValues

The new complex type ArrayOfTemplateFormatsLabelValues is a structure to include an array of the TemplateFormatsLabelValues structure. ArrayOfTemplateFormatsLabelValues is included in the ReportDefinition structure to contain the specific fields to describe the available template formats. The elements that make up ArrayOfTemplateFormatsLabelValues are as follows:

```
<complexType name="ArrayOfTemplateFormatsLabelValues">
        <sequence>
          <element name="item" type="impl:TemplateFormatsLabelValues"
minOccurs="0" maxOccurs="unbounded"/>
        </sequence>
      </complexType>
```

The elements are described in the following table:

Parameter	Type	Description
item	TemplateFormatsLabelValues	Contains the fields that describe the available template formats. See Section 1.12.2.7, "TemplateFormatsLabelValues"

1.12.3 Changed Complex Types

The following complex types have changed in PublicReportService_v11:

- DeliveryRequest
- EmailDeliveryOption
- ParamNameValue
- PrintDeliveryOption
- ReportRequest
- ReportResponse

1.12.3.1 DeliveryRequest

The complex type DeliveryRequest provides the options to deliver a report via the specified delivery methods. Three new elements are added for PublicReportService_v11. The complete complex type definition is as follows:

```
<complexType name="DeliveryRequest">
   <sequence>
     <element name="contentType" nillable="true" type="xsd:string"/>
     <element name="documentData" nillable="true" type="xsd:base64Binary"/>
     <element name="dynamicDataSource" nillable="true" type="impl:BIPDataSource"/>
     <element name="emailOption" nillable="true" type="impl:EMailDeliveryOption"/>
     <element name="faxOption" nillable="true" type="tns1:FaxDeliveryOption"/>
     <element name="ftpOption" nillable="true" type="tns1:FTPDeliveryOption"/>
     <element name="localOption" nillable="true" type="tns1:LocalDeliveryOption"/>
     <element name="printOption" nillable="true" type="impl:PrintDeliveryOption"/>
     <element name="webDAVOption" nillable="true"
type="tns1:WebDAVDeliveryOption"/>
  </sequence>
```

```
</complexType>
```

The three new elements added are shown in the following table:

Parameter	Type	Description
contentType	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-powerpoint" "application/vnd.ms-excel" "application/msword" "application/x-shockwave-flash" "text/xml" "message/rfc822"
documentData	base64Binary	The output document.
dynamicDataSource	BIPDataSource	See Section 1.12.2.1, "BIPDataSource"

1.12.3.2 EmailDeliveryOption

The complex type EmailDeliveryOption added one new element in PublicReportService_v11. The complete complex type definition is as follows:

```
<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="true" type="xsd:string"/>
          <element name="emailReplyTo" nillable="true" type="xsd:string"/>
          <element name="emailServerName" nillable="true" type="xsd:string"/>
          <element name="emailSubject" nillable="true" type="xsd:string"/>
          <element name="emailTo" nillable="true" type="xsd:string"/>
        </sequence>
</complexType>
```

The new element is described in the following table:

Parameter	Type	Description
emailServerName	string	The e-mail server name, for example: "Oracle Mail".

1.12.3.3 ParamNameValue

Sixteen new elements were added to the ParamNameValue complex type in PublicReportService_v11. This structure describes parameters defined for a reports and templates in BI Publisher. The complete definition is as follows:

```
<complexType name="ParamNameValue">
        <sequence>
          <element name="UIType" nillable="true" type="xsd:string"/>
          <element name="dataType" nillable="true" type="xsd:string"/>
          <element name="dateFormatString" nillable="true" type="xsd:string"/>
          <element name="dateFrom" nillable="true" type="xsd:string"/>
          <element name="dateTo" nillable="true" type="xsd:string"/>
          <element name="defaultValue" nillable="true" type="xsd:string"/>
          <element name="fieldSize" nillable="true" type="xsd:string"/>
          <element name="label" nillable="true" type="xsd:string"/>
          <element name="lovLabels" nillable="true" type="tns1:ArrayOfString"/>
          <element name="multiValuesAllowed" type="xsd:boolean"/>
          <element name="name" nillable="true" type="xsd:string"/>
```

```
        <element name="refreshParamOnChange" type="xsd:boolean"/>
        <element name="selectAll" type="xsd:boolean"/>
        <element name="templateParam" type="xsd:boolean"/>
        <element name="useNullForAll" type="xsd:boolean"/>
        <element name="values" nillable="true" type="tns1:ArrayOfString"/>
      </sequence>
    </complexType>
```

The ParamNameValue structure is used in the ArrayOfParamNameValue, which is included in the ReportRequest, ReportDefinition, and JobInfo structures. ParamNameValue is also returned by the getTemplateParameters and getReportParameters operations.

The new elements are described in the following table:

Parameter	Type	Description
UIType	string	The type of parameter as defined in the BI Publisher data model user interface. Valid values are: "Menu" "Text" "Date" "Hidden" "Search"
dataType	string	The data type of the parameter. Valid values are: "String" "Integer" "Boolean" "Date" "Float"
dateFormatString	string	If the parameter type is "Date", this specifies the Date Format String. The date format string must be a Java date format, for example: "MM-dd-YYYY".
dateFrom	string	If the parameter type is "Date", this specifies the begin value of the date.
dateTo	string	If the parameter type is "Date", this specifies the end value of the date.
defaultValue	string	Specifies the default value of the parameter.
fieldSize	string	For parameter types "Text" and "Date" specifies the text field size for the parameter.
label	string	For all parameter types except "Hidden", specifies the display label for the parameter.
lovLabels	ArrayOfString	If the parameter type is "Menu", specifies the values displayed in the list of values to the user.
multiValuesAllowed	boolean	True indicates that multiple values are allowed for the parameter.
name	string	The parameter Identifier as defined in the BI Publisher parameter definition page.
refreshParamOnChange	boolean	For parameter types "Text" and "Menu", a true value for this parameter indicates that other defined parameters should be refreshed when a selection is made for this parameter.
selectAll	boolean	For parameter type Menu: True indicates that all values can be selected for the LOV.

Parameter	Type	Description
templateParam	boolean	True indicates the parameter is defined in the RTF template.
useNullForAll	boolean	For parameter type Menu: True indicates that a null will be passed if all values are selected for the parameter.
values	ArrayOfString	The values for the parameter.

1.12.3.4 PrintDeliveryOption

One new element was added to the PrintDeliveryOption complex type in PublicReportService_v11. The complete definition is as follows:

```
<complexType name="PrintDeliveryOption">
        <sequence>
          <element name="printNumberOfCopy" nillable="true" type="xsd:string"/>
          <element name="printOrientation" 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="true" type="xsd:string"/>
        </sequence>
      </complexType>
```

This structure is used by the DeliveryRequest structure.

The new element is described in the following table:

Parameter	Type	Description
printOrientation	string	Valid values are: "Portrait" and "Landscape".

1.12.3.5 ReportRequest

Five new elements are added to the ReportRequest complex type in PublicReportService_v11. The complete definition is as follows:

```
 <complexType name="ReportRequest">
        <sequence>
          <element name="attributeCalendar" nillable="true" type="xsd:string"/>
          <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="attributeTimezone" nillable="true" type="xsd:string"/>
          <element name="byPassCache" type="xsd:boolean"/>
          <element name="dynamicDataSource" nillable="true"
type="impl:BIPDataSource"/>
          <element name="flattenXML" type="xsd:boolean"/>
          <element name="parameterNameValues" nillable="true"
type="impl:ArrayOfParamNameValue"/>
          <element name="reportAbsolutePath" nillable="true" type="xsd:string"/>
          <element name="reportData" nillable="true" type="xsd:base64Binary"/>
          <element name="reportOutputPath" nillable="true" type="xsd:string"/>
          <element name="reportRawData" nillable="true" type="xsd:string"/>
          <element name="sizeOfDataChunkDownload" type="xsd:int"/>
        </sequence>
      </complexType>
```

The new elements are described in the following table:

Parameter	Type	Description
attributeCalendar	string	The formatting calendar to use for the report request. Valid values are: "Gregorian", "Arabic Hijrah", "English Hijrah", "Japanese Imperial", "Thai Buddha", and "ROC Official".
attributeTimeZone	string	Specifies the time zone to use for the request, using a supported Java time zone ID. For example: "America/Los_ Angeles".
byPassCache	boolean	True indicates to bypass document cache.
dynamicDataSource	BIPDataSource	See Section 1.12.2.1, "BIPDataSource"
reportOutputPath	string	Specifies the output path for the generated report.
reportRawData	string	If raw XML data is used for the report, this element contains the XML data.

1.12.4 New Operations

The following operations are new in PublicReportService_v11:

- getTemplateParameters
- getDeliveryServiceDefinition
- deliveryService

Also new are the corresponding InSession operations for the new operations listed above:

- getTemplateParametersInSession
- getDeliveryServiceDefinitionInSession
- deliveryServiceInSession

For information on using the InSession operations, see Section 1.12.6, "Using the InSession Operations."

1.12.4.1 getTemplateParameters

Use getTemplateParameters to retrieve information about parameters for a template.

1.12.4.1.1 Input Message: getTemplateParametersRequest The following table displays the parameters required for getTemplateParameters:

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

1.12.4.1.2 Return Message: getTemplateParametersResponse The response message returns the parameter information in the ParamNameValue structure:

Parameter	Type	maxOccurs	Description
getTemplateParametersReturn	ParamNameValue	unbounded	See Section 1.12.3.3, "ParamNameValue."

1.12.4.2 getDeliveryServiceDefinition

Use getDeliveryServiceDefinition to retrieve information about the BI Publisher delivery servers.

1.12.4.2.1 Input Message: getDeliveryServiceDefinitionRequest The following table describes the parameters required for getDeliveryServiceDefinition:

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

1.12.4.2.2 Return Message: getDeliveryServiceDefinitionResponse The response message returns information about the delivery servers in the DeliveryServiceDefinition structure:

Parameter	Type	Description
getDeliveryServiceDefinitionReturn	DeliveryServiceDefinition	See Section 1.12.2.4, "DeliveryServiceDefinition"

1.12.4.3 deliveryService

Use deliveryService to execute a report delivery request.

1.12.4.3.1 Input Message: deliveryServiceRequest The following table describes the parameters required for deliveryService:

Parameter	Type	Nillable	Description
deliveryRequest	DeliveryRequest	no	Includes parameters required to execute a report delivery via a supported delivery channel. See Section 1.12.3.1, "DeliveryRequest."
userID	string	no	The BI Publisher user name.
password	string	no	The password for the entered user name.

1.12.4.3.2 Return Message: deliveryServiceResponse The following table describes the parameter for deliveryServiceResponse:

Parameter	Type	Description
deliveryServiceReturn	string	The return response.

1.12.5 Changed Operations

The following operations are impacted by changes in the noted complex type:

PublicReportService Operation	Includes Complex Type:
getReportParameters	ReportRequest
runReport	ReportRequest
scheduleReport	ScheduleRequest
updateReportDefinition	ReportDefinition

1.12.6 Using the InSession Operations

For most operations in PublicReportService and PublicReportService_v11, an "InSession" version of the operation is also available. These operations enable you to pass a BI Publisher token, rather than the username and password and otherwise take the identical parameters. Use the `login` operation to obtain a token to pass to the InSession operations. Use the `logout` operation to terminate the session.

1.12.6.1 login

Use the login operation to establish a BI Publisher session and return the session token for you to pass to the InSession operations.

1.12.6.1.1 Input Message: loginRequest The following table describes the parameters defined for login:

Parameter	Type	Description
userID	string	The BI Publisher user name.
password	string	The password for the entered user name.

1.12.6.1.2 Return Message: loginResponse The following table describes the parameter for loginResponse:

Parameter	Type	Description
loginReturn	string	The BI Publisher session token. Use this token with the InSession operations.

1.12.6.2 logout

Use the logout operation to terminate a BI Publisher session.

1.12.6.2.1 Input Message: logoutRequest The following table describes the parameters defined for logout:

Parameter	Type	Description
bipSessionToken	string	The token established for your BI Publisher session.

1.12.6.2.2 Return Message: logoutResponse The following table describes the parameter for logoutResponse:

Parameter	Type	Description
logoutReturn	boolean	True indicates that the BI Publisher session was successfully terminated.

2

New Features in BI Publisher Release 10.1.3.4.2

This chapter describes features of BI Publisher that were introduced in Release 10.1.3.4.2. This chapter contains the following sections:

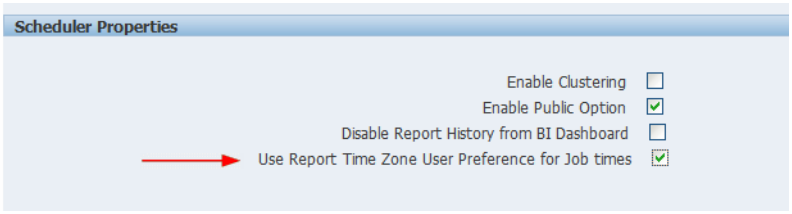
2.1 Display Scheduler Job Times in the User's Preferred Time Zone

Previously when a user viewed Scheduled Report History, the Start Processing Time and End Processing Time values displayed in the server time zone. A new property in the **Scheduler Configuration** page, "Use Report Time Zone User Preference for Job times" enables the Administrator to configure the system so that processing times will display in the time zone preferred by each user, as defined in the user's Preference setting for Report Time Zone.

To enable this property:

1. Navigate to the **Scheduler Configuration** page: From the **Admin** tab, under **System Maintenance**, click **Scheduler Configuration**.
2. In the Scheduler Properties region, select checkbox for the **Use Report Time Zone User Preference for Job times** property. The new property is shown in Figure 2–1.

Figure 2–1 Detail of Scheduler Properties

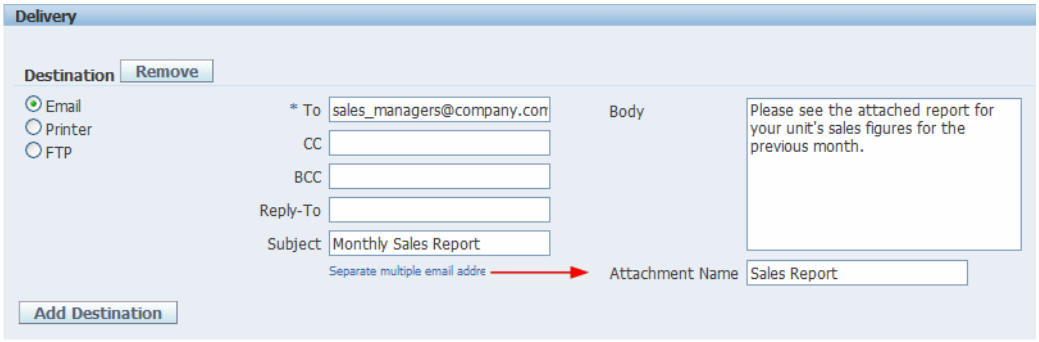


Note that the property is disabled by default.

2.2 Attachment Name Field for E-mail Deliveries

A new **Attachment Name** field has been added to the Email delivery specification options in the **Schedule Report** and **Send** pages to enable you to define the file name of the report when it is attached to the email. The new field is shown in Figure 2–2:

Figure 2–2 Attachment Name Field in Email Delivery Options



You can enter a static name for the attachment, such as Sales Report (the .pdf extension will be added); or, you can also define a dynamic name using date expressions as follows:

Table 2–1 Supported Date Expressions

Expression	Description
%y	Displays the year in four digits: Example: 2011
%m	Displays the month in two digits: 01-12 (where 01 = January)
%d	Displays the date in two digits: 01-31
%H	Displays the hour in two digits based on 24-hour day: 00-24
%M	Displays the minute in two digits: 00 - 59
%S	Displays the number of seconds in two digits: 00 - 59
%l	Displays milliseconds in three digits: 000 - 999

Examples:

To create a file name that appends the day, month, and year, such as:

Sales Report 07_11_2010.pdf

Enter the following in the **Attachment Name** field:

Sales Report %d_%m_%y.pdf

To create a file name that prefixes the day, month, and year and appends the hour and minute, such as:

07_11_2011 Sales Report 08_55.pdf

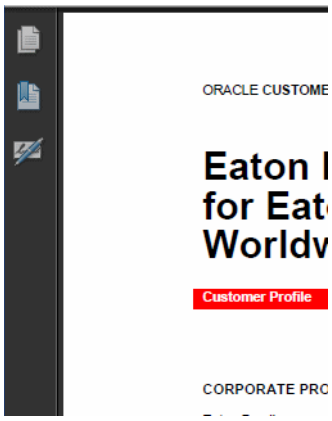
Enter the following:

%d_%m_%y Sales Report %H_%M.pdf

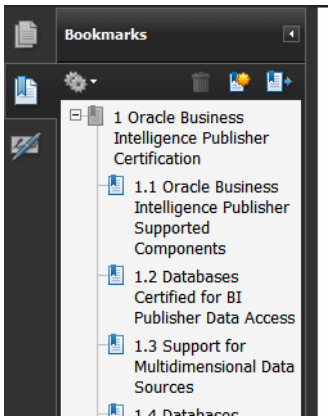
2.3 Control Initial Display of PDF Navigation Panel

A new property added to the Runtime Configuration properties page enables you to control the navigation panel view that is presented when a user first opens a PDF report. The view can be set to one of the following:

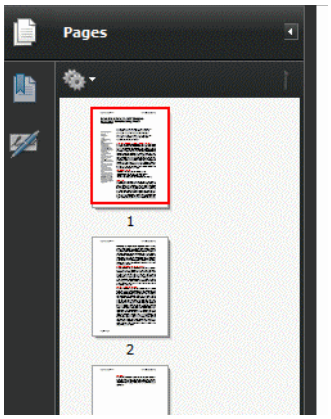
- Panels Collapsed - displays the PDF document with the navigation panel collapsed, as shown in the following figure:



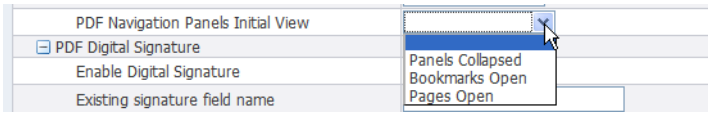
- Bookmarks Open (default) - displays the bookmark links for easy navigation as shown in the following figure:



- Pages Open - displays each page of the PDF as a thumbnail that you can click to navigate to, as shown in the following figure:



The new property is located in the PDF Output group of properties and is shown in the following figure:



2.4 Support for Safe Divide Function

Previously when performing a divide function you had to check the value of the numeric values for a zero when doing a divide operation to avoid the "not a number" (NaN) error.

The safe divide function enables you to specify a value to be returned if the result of the divide function is not a number.

The syntax of the safe divide function is:

<?xdoxslt:sdiv(*num1*,*num2*, string)?>

where

num1 is the dividend

num2 is the divisor

string is the value to return when the result is not a number.

For example:

<?xdoxslt:sdiv(10,0, '0')?> would yield 0

<?xdoxslt:sdiv(10,0, 'None')?> would yield None

2.5 Support for Hyperlinking Across Templates

If you are utilizing the PDF Bookbinder API to merge documents at runtime, you can use this new feature to create a hyperlink across the RTF or PDF templates that you are merging. The syntax described here enables you to declare a link ID in the linking template that you match in the target template so that at runtime when the documents are merged the intradocument hyperlink is created.

To create the hyperlink:

1. In the linking template (Template A), enter the following:

   ```
   <?pdf-named-destination:true?>
   <?link-source:'linkid';'link_displaytext';true>
   ```
   where

   `'linkid'` is a unique identifier of the target

   `'link_display_text'` is the hyperlink text

   For example:

   ```
   <?link-source:'MyTarget';'This is a link';true>
   ```
2. In the target template (Template B), enter:

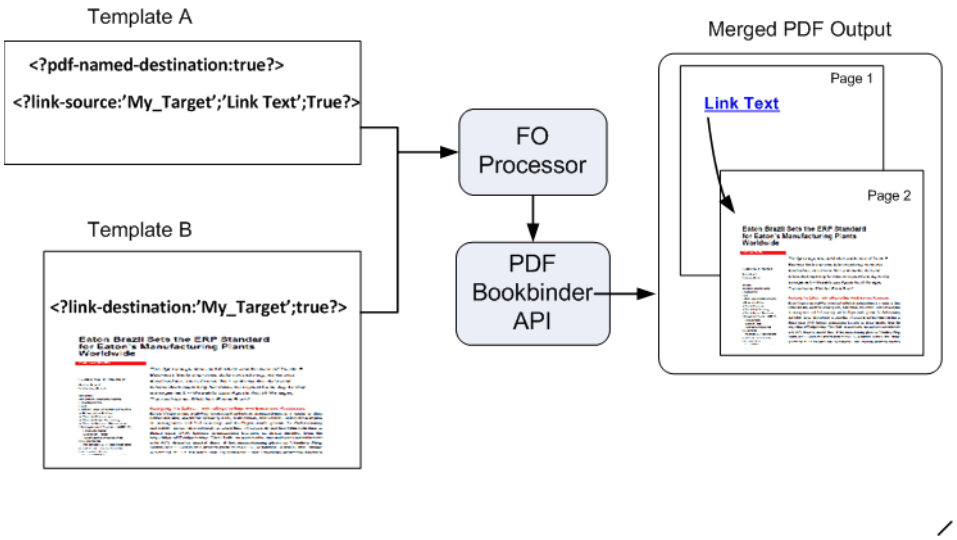
   ```
   <?link-destination:'linkid';true?>
   ```
   where `'linkid'` matches the link-source `'linkid'` in Template A.

   For example:

   ```
   <?link-destination:'MyTarget';true?>
   ```

At runtime, when the PDF Bookbinder API is called, the hyperlink will be created in the merged PDF document. Figure 2–3 illustrates the process:

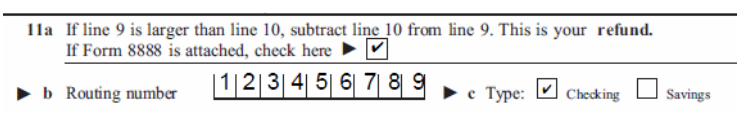
Figure 2–3 Creating a Hyperlink Between Merged Documents



2.6 Support for "Comb of Characters" Option for Form Fields in PDF Templates

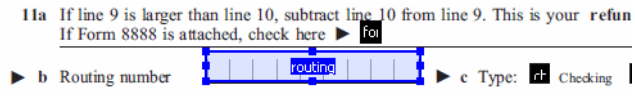
The comb of characters option for a PDF form field in Adobe Acrobat spreads the text evenly across the width of the text field. Use this option when the form field requires the characters to be entered in specific positions, as the Routing number field shown in Figure 2–4:

Figure 2–4 Example of PDF Form Field Requiring Characters in Specific Positions



To use this feature in a PDF template, perform the following:

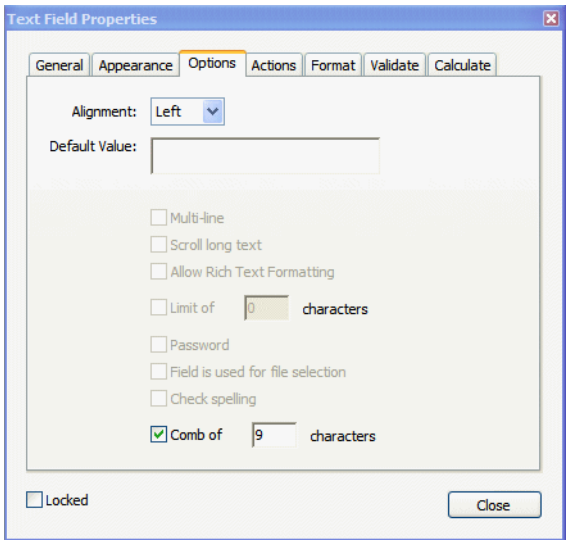
1. In Adobe Acrobat Professional, add the form field as a text field. An example is shown in the following figure:



2. Open the **Text Field Properties** dialog and click the **Options** tab. Clear all check boxes and select the **Comb of characters** check box.

> **Note:** The **Comb of characters** option is only enabled when all other options are cleared.

3. Enter the number of characters in the text field. For the routing number example, a value of 9 is entered as shown in the following figure:



   If your data may not contain the number of characters specified each time, you can set the **Alignment** option to specify whether the value will be aligned to the right, left, or center within the field.

When you run the report, the characters comprising the value for the routing field will be spread across the text field as shown in Figure 2–5:

Figure 2–5 Example with Data

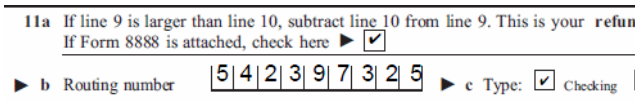
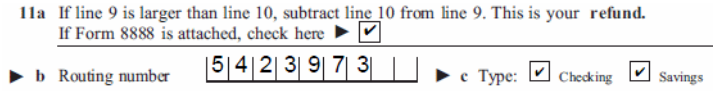


Figure 2–6 shows how the data will display in the field when the data for the routing field does not contain the full nine characters and the **Alignment** option is set to Left:

Figure 2–6 Example of Left-Aligned Data

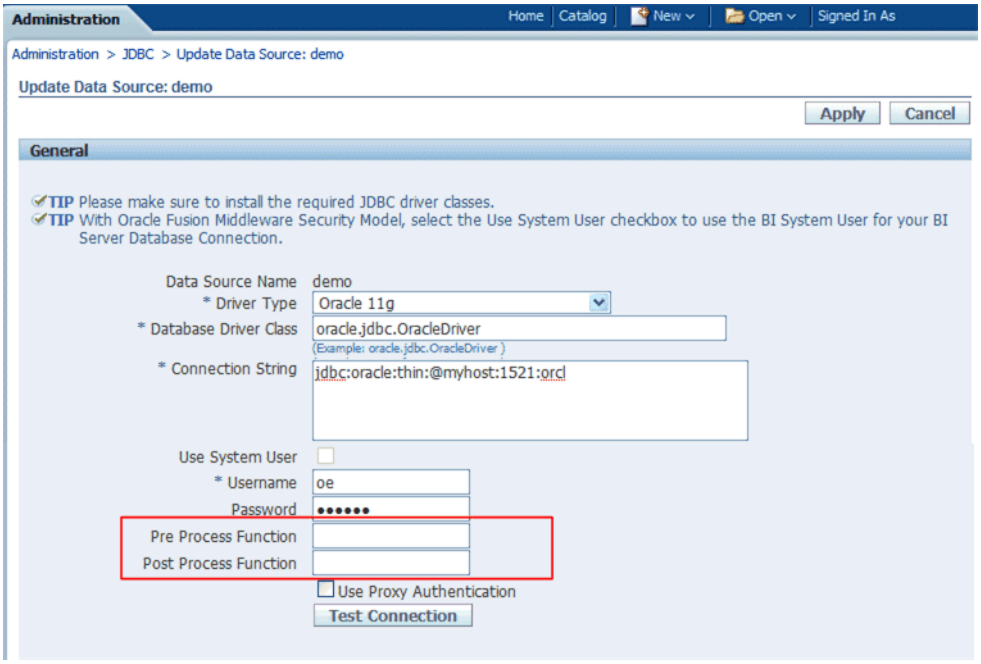


For guidelines for creating PDF templates see "Creating a PDF Template" in the *Oracle Business Intelligence Publisher Report Designer's Guide*.

2.7 Support for JDBC Triggers

Two new fields enable you to define PL/SQL functions for BI Publisher to execute when a connection to a JDBC data source is created (preprocess function) or closed (postprocess function). The function must return a boolean value. This feature is supported for Oracle databases only. Figure 2–7 shows the new fields.

Figure 2–7 JDBC Data Source Definition Page Showing New Trigger Fields



These two fields enable the administrator to set a user's context attributes before a connection is made to a database and then to dismiss the attributes after the connection is broken by the extraction engine.

The system variable :xdo_user_name can be used as a bind variable to pass the login username to the PL/SQL function calls. Setting the login user context in this way enables you to secure data at the data source level (rather than at the SQL query level).

For example, assume you have defined the following sample function:

```
FUNCTION set_per_process_username (username_in IN VARCHAR2)
   RETURN BOOLEAN IS
     BEGIN
     SETUSERCONTEXT(username_in);
     return TRUE;
   END set_per_process_username
```

To call this function every time a connection is made to the database, enter the following in the **Pre Process Function** field: set_per_process_username(:xdo_user_name)

Another sample usage might be to insert a row to the LOGTAB table every time a user connects or disconnects:

```
CREATE OR REPLACE FUNCTION BIP_LOG (user_name_in IN VARCHAR2, smode IN VARCHAR2)
RETURN BOOLEAN AS
   BEGIN
   INSERT INTO LOGTAB VALUES(user_name_in, sysdate,smode);
   RETURN true;
   END BIP_LOG;
```

In the **Pre Process Function** field enter: BIP_LOG(:xdo_user_name)

As a new connection is made to the database, it is logged in the LOGTAB table. The SMODE value specifies the activity as an entry or an exit. Calling this function as a **Post Process Function** as well returns results such as those shown in Figure 2–8.

Figure 2–8 LOGTAB table

NAME	UPDATE_DATE	S_FLAG
oracle	14-MAY-10 09.51.34.000000000	AMStart
oracle	14-MAY-10 10.23.57.000000000	AMFinish
administrator	14-MAY-10 09.51.38.000000000	AMStart
administrator	14-MAY-10 09.51.38.000000000	AMFinish
oracle	14-MAY-10 09.51.42.000000000	AMStart
oracle	14-MAY-10 09.51.42.000000000	AMFinish

2.8 Integrating with Microsoft Active Directory

Microsoft Active Directory supports the LDAP interface and therefore can be configured with BI Publisher using LDAP Security.

2.8.1 Configuring Active Directory for BI Publisher

To configure active directory:

1. Add users who must access BI Publisher.

   Add the users under "Users" or any other organization unit in the Domain Root.

2. Add the BI Publisher system groups. The Scope of the groups must be Domain Local.

   Table 2–2 describes the BI Publisher system groups that must be added.

Table 2–2 BI Publisher System Groups

BI Publisher System Group	Description
XMLP_ADMIN	The administrator role for the BI Publisher server. You must assign the Administrator account used to access your LDAP server the XMLP_ADMIN group.
XMLP_DEVELOPER	Allows users to create and edit reports and data models.
XMLP_SCHEDULER	Allows users to schedule reports.
XMLP_ANALYZER_EXCEL	Allows users to use the Excel Analyzer feature.

Table 2–2 (Cont.) BI Publisher System Groups

BI Publisher System Group	Description
XMLP_ANALYZER_ONLINE	Allows users to use the online analysis feature (online analyzer).
XMLP_TEMPLATE_BUILDER	Allows users to connect to the BI Publisher server from the Template Builder for Word and to upload and download templates.

3. Grant BI Publisher system groups to global groups or users.

   You can grant BI Publisher system groups directly to users or through global groups.

Example 1: Grant Users the BI Publisher Administrator Role

1. Under the **Active Directory User and Computers**, open the XMLP_ADMIN group and click the **Members** tab.
2. Click **Add** to add users who need access to BI Publisher Administrator privileges.

Example 2: Grant Users Access to Scheduling Reports

A global group called "HR Manager" is defined under "Users".

All users in this group will need to schedule reports.

To achieve this, add "HR Manager" as a Member of the XMLP_SCHEDULER group.

2.8.2 Configuring BI Publisher to Use Active Directory

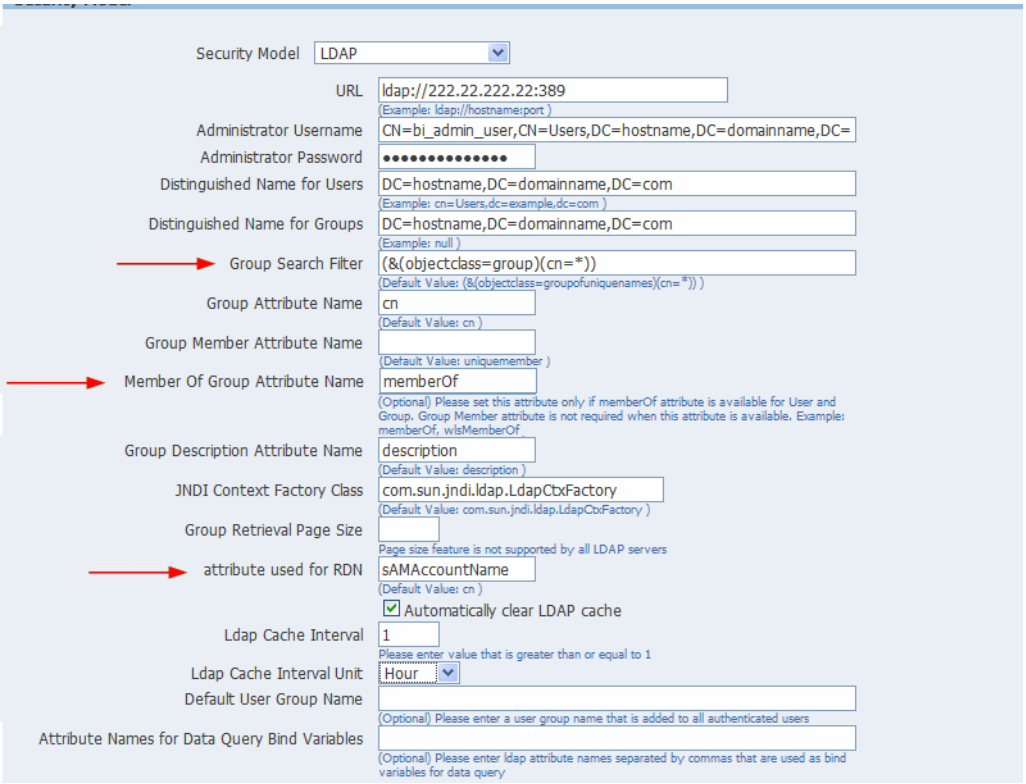
To configure BI Publisher:

1. On the **Admin** tab, click **Security Configuration**.
2. Set up a Local Superuser if one has not been configured. This is highly recommended. If the security configuration fails you will still be able to log in to BI Publisher using the Superuser credentials.
3. In the **Security Model** region of the page, select LDAP from the **Security Model** list.
4. Enter the details for the Active Directory server, as described in the section "Integrating with LDAP" in the *Oracle Business Intelligence Publisher Administrator's and Developer's Guide*, noting the following specific information for Active Directory:
   - Set **Group Search Filter** objectclass to "group"
   - Set **Member of Group Attribute Name** to "memberOf" (**Group Member Attribute Name** can be left blank).
   - Set **Attribute used for RDN** to "sAMAccountName".
   - If your are using LDAP over SSL note the following:
     - the protocol is "ldaps"
     - the default port is 636

     An example URL would be: ldaps://example.com:636/

   Figure 2–9 shows an example configuration highlighting the recommendations stated above.

Figure 2–9 Example Active Directory Configuration



5. Restart the BI Publisher application.

If you are configuring BI Publisher to use LDAP over SSL, then you must also configure the Java keystore to add the server certificate to the JVM.

2.8.3 Logging In to BI Publisher Using the Active Directory Credentials

The User login name defined in **Active Directory Users and Computers** >**User Properties** >**Account** is used for the BI Publisher login name. Add the Domain to the user name to log in to BI Publisher. For example: "scott_tiger@domainname.com".

Note the following:

- The **Attribute used for RDN** can be sAMAccountName instead of userPrincipalName.
- You must use sAMAccountName for the **Attribute used for RDN** when the "User logon name (pre-Windows 2000)" is required to use for the BI Publisher login username.
- User names must be unique across all organization units.

2.8.4 Assign Data Access and Folder Permissions to Roles

To assign data access and folder permissions to roles:

1. Log in to BI Publisher as a user assigned the XMLP_ADMIN role Active Directory.
2. On the **Admin** tab click **Roles and Permissions**.

   You see the roles that you created in Active Directory to which you assigned the XMLP_ roles. Note the following:

- The XMLP_ roles are not shown because these are controlled through the Active Directory interface.
- The Users tab is no longer available under the Security Center because users are now managed through Active Directory.
- Roles are not updatable in the BI Publisher interface, except for adding folders and data sources.

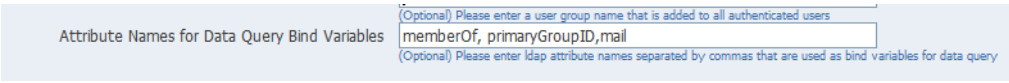
3. Click **Add Data Sources** to add BI Publisher data sources to the role. A role must be assigned access to a data source to run reports from that data source or to build data models from the data source. For more information see the topic "Assign Folders and Data Sources to Roles" in the *Oracle Business Intelligence Publisher Administrator's and Developer's Guide*.
4. Click **Add Folders** to add access to specific folders to the role. A role must be assigned access to a folder to run or edit the reports it contains. See the topic "Assign Folders and Data Sources to Roles" in the *Oracle Business Intelligence Publisher Administrator's and Developer's Guide*.

2.9 Using LDAP Attribute Values as Bind Variables in Data Queries

You can now use LDAP (and Microsoft Active Directory) attribute values as bind variables in data queries. To use this feature, specify the LDAP attribute values you want to use as bind variables, then use the syntax defined in this section in your data query.

The LDAP configuration page contains a new field **Attribute Names for Data Query Bind Variables** and is shown in Figure 2–10.

Figure 2–10 LDAP Configuration Page Detail



Enter the LDAP attribute names that you wish to use as bind variables; enter multiple attributes separated by a comma, for example: memberOf, primaryGroupID,mail

To use these attribute values as bind variables in a data query:

Reference the attribute names that you enter in the **Attribute Names for Data Query Bind Variables** field as follows:

```
xdo_<attribute name>
```

Assume that you have entered the sample attributes: memberOf, primaryGroupID, mail. These can then be used in a query as the following bind variables:

```
xdo_memberof
xdo_primaryGroupID
xdo_mail
```

Note that the case of the attribute is ignored; however, the "xdo_" prefix must be lowercase.

Use these in a data model as follows:

```
SELECT
:xdo_user_name AS USER_NAME ,
:xdo_user_roles AS USER_ROLES,
```

```
:xdo_user_ui_oracle_lang AS USER_UI_LANG,
:xdo_user_report_oracle_lang AS USER_REPORT_LANG,
:xdo_user_ui_locale AS USER_UI_LOCALE,
:xdo_user_report_locale AS USER_REPORT_LOCALE,
:xdo_memberof as MEMBER_OF,
:xdo_primaryGroupID as PRIMARY_GROUP_ID,
:xdo_mail as MAIL
FROM DUAL
```

The LDAP bind variables will return the values stored in the LDAP directory for the user that is logged in.

2.10 Support for IBM WebSphere 7.x

Release 10.1.3.4.2 adds support for IBM WebSphere 7.x as an application server. To deploy BI Publisher to IBM WebSphere 7.x, follow the instructions in the *Oracle Business Intelligence Publisher Installation Guide* for IBM WebSphere 6.1 deployment, adding the following step:

1. Navigate to the xmlp-server-config.xml file located at:

   <bip_repository>/Admin/Configuration/xmlp-server-config.xml

2. Add the USE_HTTPONLY_SESSIONID property to the configuration file, setting it to "false" as shown:

   ```
   <xmlpConfig xmlns="http://xmlns.oracle.com/oxp/xmlp">
   ...
   <property name="USE_HTTPONLY_SESSIONID" value="false"/>
   ...
   </xmlpConfig>
   ```

3. Restart the BI Publisher server.

Oracle recommends configuring IBM WebSphere to use HTTPOnly JSESSIONID cookie. Please refer to IBM WebSphere documentations for details.

Important: See also the *Oracle Business Intelligence Publisher Release Notes* for the issue: "WebSphere 6.1: Class Loader Configuration Is Required When "HTTP 500 Internal Server Error" Prevents User Login."

3

Creating Excel Templates in Release 10.1.3.4.2

This chapter describes how to create Excel templates. It covers the following topics:

3.1 Overview

An Excel template is a report layout designed in Microsoft Excel for retrieving and formatting enterprise reporting data in Excel. Excel templates provide a set of special features for mapping data to worksheets and for performing additional processing to control how the data is output to Excel workbooks.

3.1.1 Features of Excel Templates

With Excel templates you can:

- Define the structure for the data in Excel output
- Split hierarchical data across multiple sheets and dynamically name the sheets
- Create sheets of data that have master-detail relationships
- Use native XSL functions in the data to manipulate it prior to rendering
- Use native Excel functionality

3.1.2 Limitations of Excel Templates

The following are limitations of Excel templates:

- For reports that split the data into multiple sheets, images are not supported. If the template sheet includes images, when the data is split into multiple sheets, the images are displayed only on the first sheet.

- BI Publisher does not provide a tool to facilitate the markup of the template with BI Publisher tags; all tags must be manually coded. Some features require the use of XSL and XSL Transformation (XSLT) specifications

3.1.3 Prerequisites

Following are prerequisites for designing Excel templates:

- Microsoft Excel 2003 or later. The template file must be saved as Excel 97-2003 Workbook binary format (*.xls).
- To use some of the advanced features, the report designer must have experience with XSL and XSLT.
- The report data model has been created.

3.1.4 Supported Output

Excel templates generate Excel binary (.xls) output only.

3.1.5 Desktop Tools

You can use the Template Viewer to preview Excel templates from your desktop.

The Template Viewer is installed automatically when you install the Template Builder for Word. For information on obtaining the utility, see the topic "Installing Oracle BI Publisher Desktop Tools" in the *Oracle Business Intelligence Publisher Installation Guide*.

3.2 Concepts

Similar to RTF template design, Excel template design follows the paradigm of mapping fields from the XML data to positions in the Excel worksheet. Excel templates make use of features of Excel in conjunction with special BI Publisher syntax to achieve this mapping. In addition to direct mapping of data elements, Excel templates also utilize a special sheet (the XDO_METADATA sheet) to specify and map more complex formatting instructions.

3.2.1 Identifying Data Field Placeholders and Groups

Excel templates use named cells and groups of cells to enable BI Publisher to insert data elements. Cells are named using BI Publisher syntax to establish the mapping back to the XML data. The cell names are also used to establish a mapping within the template between the named cell and calculations and formatting instructions that are defined on the XDO_METADATA sheet.

The template content and layout must correspond to the content and hierarchy of the XML data file used as input to the report. Each group of repeating elements in the template must correspond to a parent-child relationship in the XML file. If the data is not structured to match the desired layout in Excel it is possible to regroup the data using XSLT preprocessing or the grouping functions. However, for the best performance and least complexity it is recommended that the data model be designed with the report layout in mind.

Note: See Section 3.6, "Preprocessing the Data Using an XSL Transformation (XSLT) File" and Section 3.5.3, "Grouping Functions" for more information about these options.

3.2.2 Use of Excel Defined Names

The Excel defined names feature is used to identify data fields and repeating elements. A defined name in Excel is a name that represents a cell, range of cells, formula, or constant value.

> **Tip:** To learn more about defined names and their usage in Microsoft Excel 2007, see the Microsoft help topic: "Define and use names in formulas."

The defined names used in the Excel template must use the syntax described in this chapter, as well as follow the Microsoft guidelines described in the Microsoft Excel help document. Note that BI Publisher defined names are within the scope of the template sheet.

3.2.3 About the XDO_ Defined Names

The BI Publisher defined names are Excel defined names identified by the prefix "XDO_". Marking up the placeholders in the template files creates the connection between the position of the placeholders in the template and the XML data elements, and also maintains the ability to dynamically grow data ranges in the output reports, so that these data ranges can be referenced by other formula calculations, charts, and macros.

3.2.4 Using Native Excel Functions

You can use the XDO_ defined names in Excel native formulas as long as the defined names are used in a simple table. When a report is generated, BI Publisher automatically adjusts the region ranges for those named regions so that the formulas calculate correctly.

However, if you create nested groups in the template, then the cells generated in the final report within the grouping can no longer be properly associated to the correct name. In this case, the use of XDO_ defined names with native Excel functions cannot be supported.

3.2.5 About the XDO_METADATA Sheet

Each Excel template requires a sheet within the template workbook called "XDO_METADATA". Use this sheet to identify the template to BI Publisher as an Excel template. This sheet is also used to specify calculations and processing instructions to perform on fields or groups in the template. BI Publisher provides a set of functions to provide specific report features. Other formatting and calculations can be expressed in XSLT.

It is recommended that you hide the XDO_METADATA sheet before you upload the completed template to the BI Publisher repository. This specification prevents report consumers from seeing it in the final report output.

Note: For more information see Section 3.3.4.1, "Format of the XDO_METADATA Sheet" and Section 3.5, "Defining BI Publisher Functions."

3.3 Building a Simple Template

This section demonstrates the concepts of Excel templates by walking through the steps to create a simple Excel template and testing it with the Template Viewer. This procedure follows these steps:

- Step 1: Obtain Sample XML Data from the Data Model
- Step 2: Design the Layout in Excel
- Step 3: Assign the BI Publisher Defined Names
- Step 4: Prepare the XDO_METADATA Sheet
- Step 5: Test the Template

3.3.1 Step 1: Obtain Sample XML Data from the Data Model

You need sample data in order to know the element names and the hierarchical relationships to properly mark up the template.

To save data from the report viewer:

1. From the Report Editor or from the Reports page, select **View**.
2. If no layouts are defined for your report, then the output type will default to xml, otherwise, choose **data** for the output type.
3. Select **Export**. Save the results as an XML file to a local directory

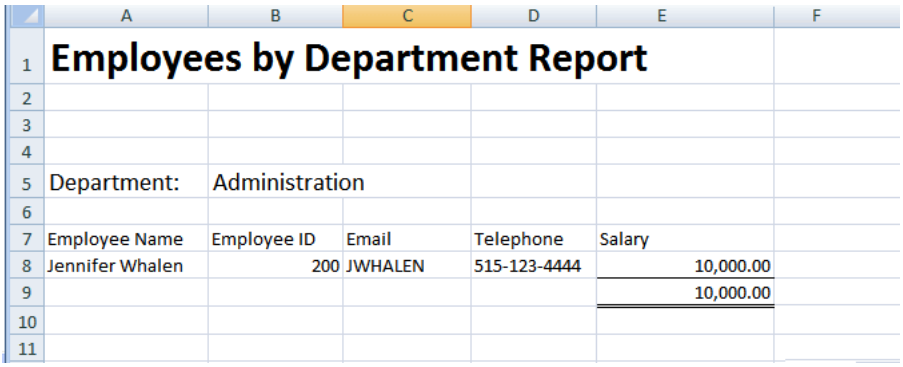
The sample data for this example is a list of employees by department. Note that employees are grouped and listed under the department.

```
<?xml version="1.0" encoding="UTF-8"?>
<DATA>
 <DEPT>
   <DEPARTMENT_ID>20</DEPARTMENT_ID>
   <DEPARTMENT_NAME>Marketing</DEPARTMENT_NAME>
     <EMPS>
       <EMPLOYEE_ID>201</EMPLOYEE_ID>
       <EMP_NAME>Michael Hartstein</EMP_NAME>
       <EMAIL>MHARTSTE</EMAIL>
       <PHONE_NUMBER>515.123.5555</PHONE_NUMBER>
       <HIRE_DATE>1996-02-17T00:00:00.000+00:00</HIRE_DATE>
       <SALARY>13000</SALARY>
     </EMPS>
     <EMPS>
       <EMPLOYEE_ID>202</EMPLOYEE_ID>
       <EMP_NAME>Pat Fay</EMP_NAME>
       <EMAIL>PFAY</EMAIL>
       <PHONE_NUMBER>603.123.6666</PHONE_NUMBER>
       <HIRE_DATE>1997-08-17T00:00:00.000+00:00</HIRE_DATE>
       <SALARY>6000</SALARY>
     </EMPS>
  </DEPT>
<DEPT>
...
...
</DEPT>
</DATA>
```

3.3.2 Step 2: Design the Layout in Excel

In Excel, determine how you want to render the data and create a sample design, as shown in Figure 3–1.

Figure 3–1 A Sample Design



	A	B	C	D	E	F
1	**Employees by Department Report**					
2						
3						
4						
5	Department:	Administration				
6						
7	Employee Name	Employee ID	Email	Telephone	Salary	
8	Jennifer Whalen	200	JWHALEN	515-123-4444	10,000.00	
9					10,000.00	
10						
11						

The design shows a department name and a row for each employee within the department. You can apply Excel formatting to the design, such as font style, shading, and alignment. Note that this layout includes a total field. The value for this field is not available in the data and requires a calculation.

3.3.3 Step 3: Assign the BI Publisher Defined Names

To code this design as a template, mark up the cells with the XDO_ defined names to map them to data elements. The cells must be named according to the following format:

- Data elements: XDO_?*element_name*?

  where

  XDO_ is the required prefix and

  ?*element_name*? is either:

  - the XML tag name from the data delimited by "?"
  - a unique name that you use to map a derived value to the cell

  For example: XDO_?EMPLOYEE_ID?

- Data groups: XDO_GROUP_?*group_name*?

  where

  XDO_GROUP_ is the required prefix and ?*group_name*? is either

  - the XML tag name for the parent element in the XML data delimited by "?".
  - a unique name that you use to define a derived grouping logic

  For example: XDO_GROUP_?DEPT?

  Note that the question mark delimiter, the *group_name*, and the *element_name* are case sensitive.

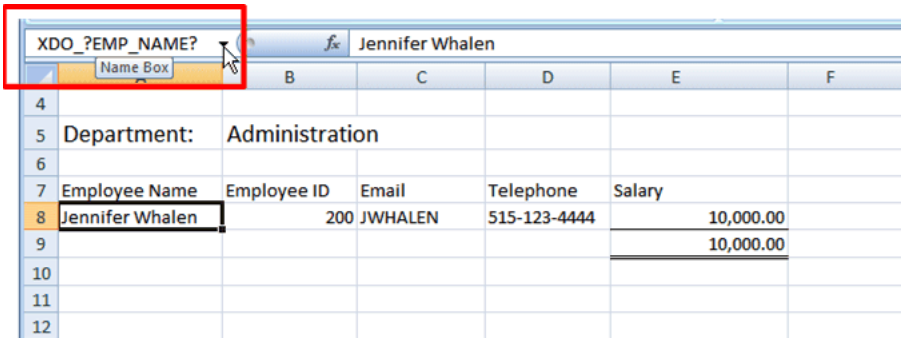
3.3.3.1 Applying a Defined Name to a Cell

To apply a defined name to a cell:

1. Click the cell in the Excel worksheet.
2. Click the Name box at the left end of the formula bar. The default name is displayed in the Name box. By default, all cells are named according to position, for example: A8.
3. In the Name box, enter the name using the XDO_ prefix and the tag name from the data. For example: XDO_?EMP_NAME?
4. Press Enter.

   Figure 3–2 shows the defined name for the Employee Name field entered in the Name box.

Figure 3–2 Defined Name for the Employee Name Field Entered in the Name Box



5. Repeat for each of the following data fields: DEPARTMENT_NAME, EMPLOYEE_ID, EMAIL, PHONE_NUMBER, and SALARY.

   Tip: If you navigate out of the Name box without pressing Enter, then the name that you entered is not maintained.

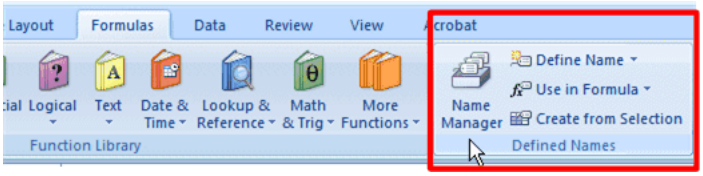
   You cannot edit the Name box while you are editing the cell contents.

   The name cannot be more than 255 characters in length.

6. For the total salary field, a calculation is mapped to that cell. For now, name that cell XDO_?TOTAL_SALARY?. The calculation is added later.

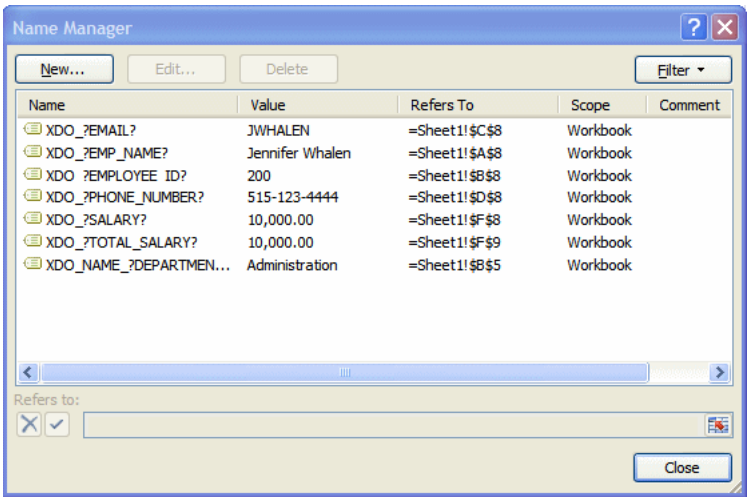
After you have entered all the fields, you can review the names and make any corrections or edits using the Name Manager feature of Excel. Access the Name Manager from the Formulas tab in Excel, as shown in Figure 3–3.

Figure 3–3 The Name Manager



The Name Manager dialog is shown in Figure 3–4 after completing all cell entries for this example:

Figure 3–4 The Name Manager Dialog



You can review all your entries and update any errors through this dialog.

3.3.3.2 Understanding Groups

A group is a set of data that repeats for each occurrence of a particular element. In the sample template design, there are two groups:

- For each occurrence of the <EMPS> element, the employee's data (name, e-mail, telephone, salary) is displayed in the worksheet.
- For each occurrence of the <DEPT> element, the department name and the list of employees belonging to that department are displayed.

In other words, the employees are "grouped" by department and each employee's data is "grouped" by the employee element. To achieve this in the final report, add grouping tags around the cells that are to repeat for each grouping element.

Note that the data must be structured according to the groups that you want to create in the template. The structure of the data for this example

```
<DATA>
   <DEPT>
      <EMPS>
```

establishes the grouping desired for the report.

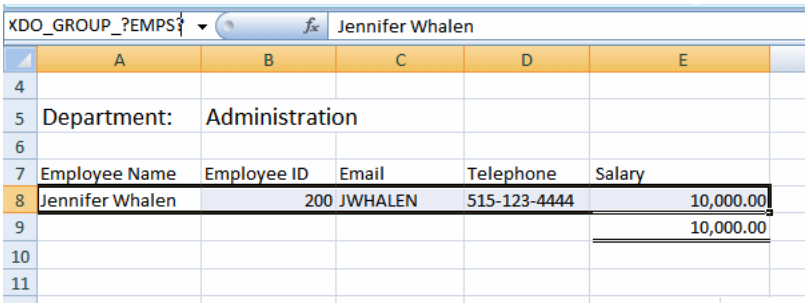
3.3.3.3 Creating Groups in the Template

To create groups in the template:

1. Highlight the cells that make up the group. In this example the cells are A8 - E8.
2. Click the Name box at the left end of the formula bar and enter the name using the XDO_GROUP_ prefix and the tag name for the group from the data. For example: XDO_GROUP_?EMPS?
3. Press Enter.

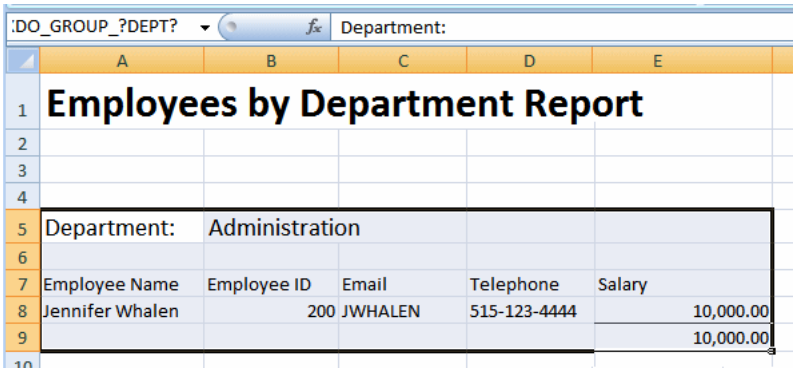
Figure 3–5 shows the XDO_GROUP_ defined name entered for the Employees group. Note that just the row of employee data is highlighted. Do not highlight the headers. Note also that the total cell XDO_?TOTAL_SALARY? is not highlighted.

Figure 3–5 The XDO_GROUP_ Defined Name Entered for the Employees Group



To define the department group, include the department name cell and all the employee fields beneath it (A5-E9) as shown in Figure 3–6.

Figure 3–6 The Department Name Cell and All Employee Fields



Enter the name for this group as: XDO_GROUP_?DEPT? to match the group in the data. Note that the XDO_?TOTAL_SALARY? cell is included in the department group to ensure it repeats at the department level.

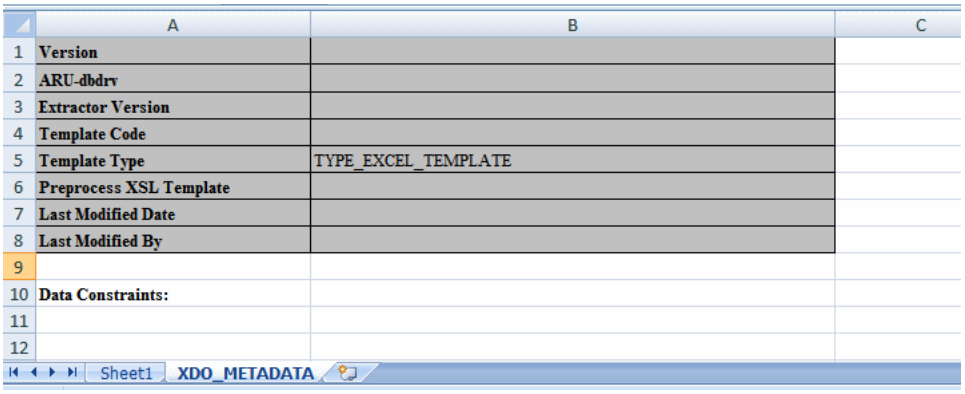
3.3.4 Step 4: Prepare the XDO_METADATA Sheet

BI Publisher requires the presence of a sheet called "XDO_METADATA" to process the template. This sheet must follow the specifications defined here.

3.3.4.1 Format of the XDO_METADATA Sheet

The XDO_METADATA sheet must have the format shown in Figure 3–7.

Figure 3–7 Format of the XDO_METADATA Sheet



The format consists of two sections: the header section and the data constraints section. Both sections are required. In the header section, all the entries in column A must be listed, but a value is required for only one: Template Type, as shown. The Data Constraints section does not require any content, but also must be present as shown.

This procedure describes how to set up the sheet for this sample Excel template to run. For the detailed description of the functionality provided by the XDO_METADATA sheet see Section 3.5, "Defining BI Publisher Functions".

3.3.4.2 Creating the XDO_METADATA Sheet

1. Create a new sheet in the Excel workbook and name it "XDO_METADATA".
2. Create the header section by entering the following variable names in column A, one per row, starting with row 1:
   - Version
   - ARU-dbdrv
   - Extractor Version
   - Template Code
   - Template Type
   - Preprocess XSLT File
   - Last Modified Date
   - Last Modified By
3. Skip a row and enter "Data Constraints" in column A of row 10.
4. In the header region, for the variable "Template Type" enter the value: TYPE_EXCEL_TEMPLATE

3.3.4.3 Adding the Calculation for the XDO_?TOTAL_SALARY? Field

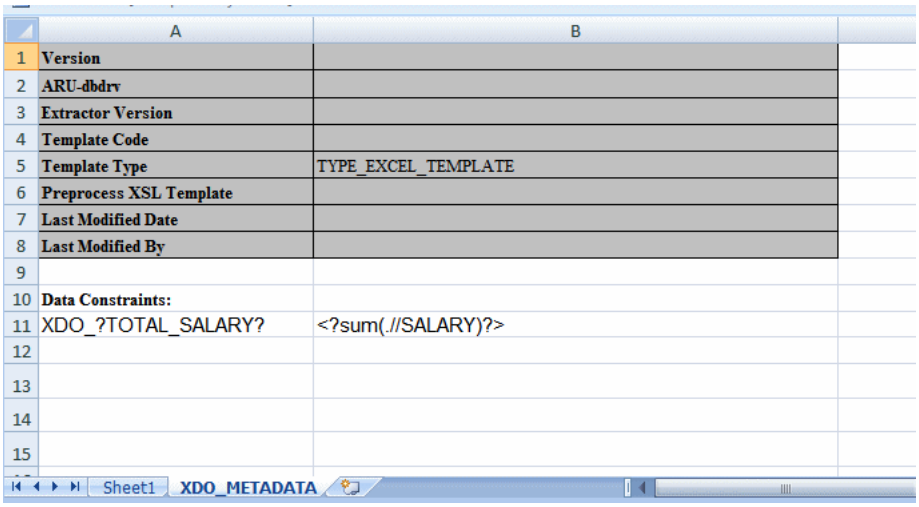
Earlier in this procedure, you assigned the defined name XDO_?TOTAL_SALARY? to the cell that is to display the total salaries listed in the SALARY column. In this step, you add the calculation to the Data Constraints section of the XDO_METADATA sheet and map the calculation to the XDO_?TOTAL_SALARY? field.

To add the calculation to the field:

1. In the Data Constraints section, in Column A, enter the defined name of the cell: XDO_?TOTAL_SALARY?
2. In Column B enter the calculation as an XPATH function. To calculate the sum of the SALARY element for all employees in the group, enter the following: <?sum(.//SALARY)?>

The completed XDO_METADATA sheet is shown in Figure 3–8.

Figure 3–8 A Completed XDO_METADATA Sheet



	A	B
1	**Version**	
2	**ARU-dbdrv**	
3	**Extractor Version**	
4	**Template Code**	
5	**Template Type**	TYPE_EXCEL_TEMPLATE
6	**Preprocess XSL Template**	
7	**Last Modified Date**	
8	**Last Modified By**	
9		
10	**Data Constraints:**	
11	XDO_?TOTAL_SALARY?	<?sum(.//SALARY)?>
12		
13		
14		
15		

Sheet1 XDO_METADATA

3.3.5 Step 5: Test the Template

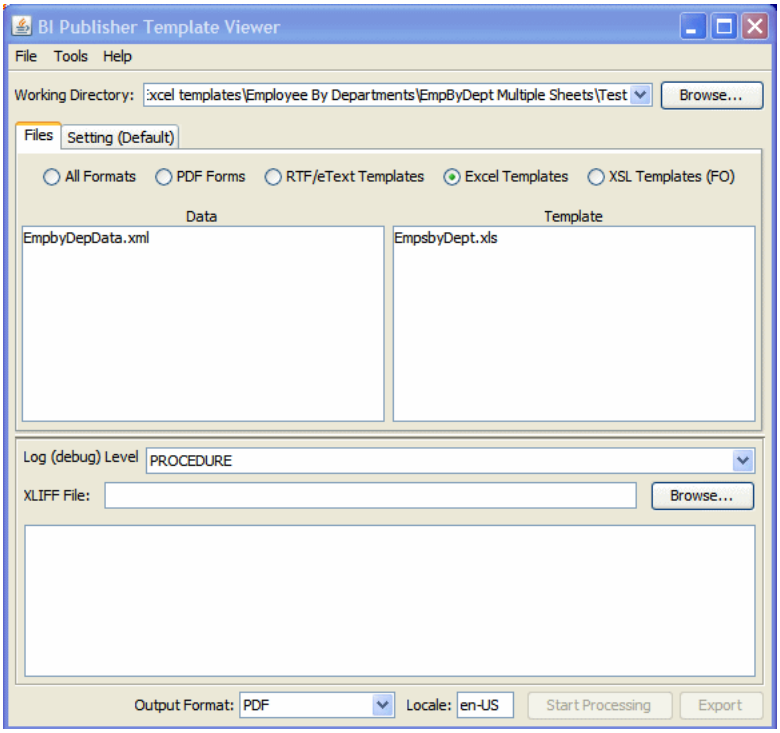
The Template Viewer is installed when you install the Template Builder for Word; see Section 3.1.5, "Desktop Tools" for more information.

To preview with the Template Viewer:

1. Open the Template Viewer:

   From the Windows desktop, click **Start**, then **Programs**, then **Oracle BI Publisher Desktop**, then **Template Viewer**.
2. Click **Browse** to locate the folder that contains the sample data file and template file. The data file and template file must reside in the same folder.
3. Select **Excel Templates**. The **Data** and **Template** regions display all XML files and all XLS files present in the directory, as shown in Figure 3–31.

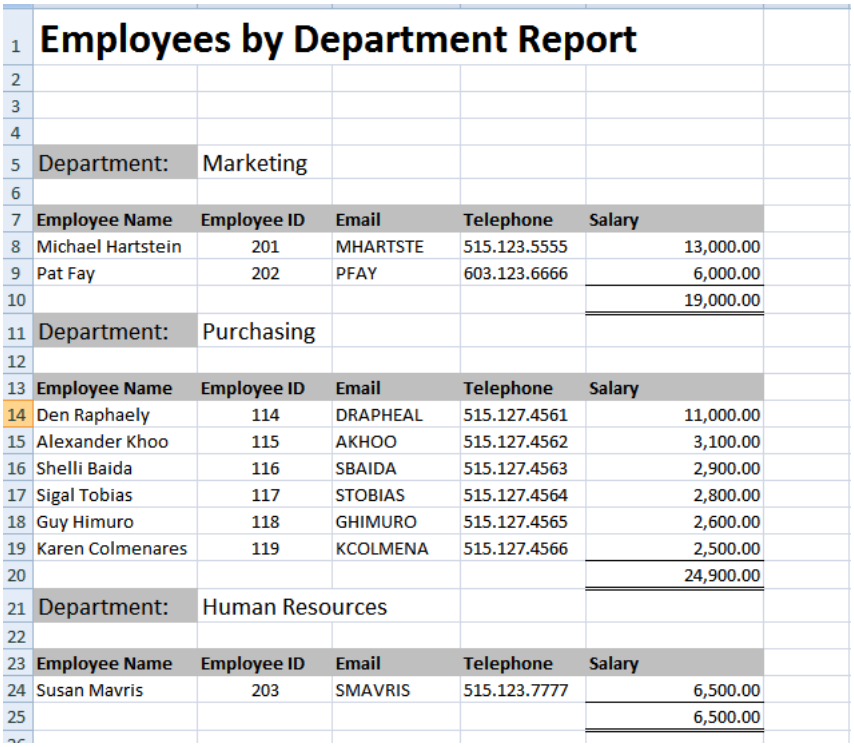
Figure 3–9 The Data and Template Regions Showing All .xml and .xls Files



4. Click the appropriate data and template files to select them.
5. From the **Output Format** list, select Excel.
6. Click **Start Processing**.

   The Template Viewer merges the sample data with the template and the output document is opened in a new workbook. Figure 3–10 shows the preview of the template with the sample data.

Figure 3–10 A Preview of a Template with Sample Data



1	**Employees by Department Report**				
2					
3					
4					
5	Department:	Marketing			
6					
7	**Employee Name**	**Employee ID**	**Email**	**Telephone**	**Salary**
8	Michael Hartstein	201	MHARTSTE	515.123.5555	13,000.00
9	Pat Fay	202	PFAY	603.123.6666	6,000.00
10					19,000.00
11	Department:	Purchasing			
12					
13	**Employee Name**	**Employee ID**	**Email**	**Telephone**	**Salary**
14	Den Raphaely	114	DRAPHEAL	515.127.4561	11,000.00
15	Alexander Khoo	115	AKHOO	515.127.4562	3,100.00
16	Shelli Baida	116	SBAIDA	515.127.4563	2,900.00
17	Sigal Tobias	117	STOBIAS	515.127.4564	2,800.00
18	Guy Himuro	118	GHIMURO	515.127.4565	2,600.00
19	Karen Colmenares	119	KCOLMENA	515.127.4566	2,500.00
20					24,900.00
21	Department:	Human Resources			
22					
23	**Employee Name**	**Employee ID**	**Email**	**Telephone**	**Salary**
24	Susan Mavris	203	SMAVRIS	515.123.7777	6,500.00
25					6,500.00

If the template preview is not generating the results expected, then you can use the Template Viewer to enable trace settings to view debug messages, see Section 3.7, "Debugging a Template Using the Template Viewer."

3.4 Formatting Dates

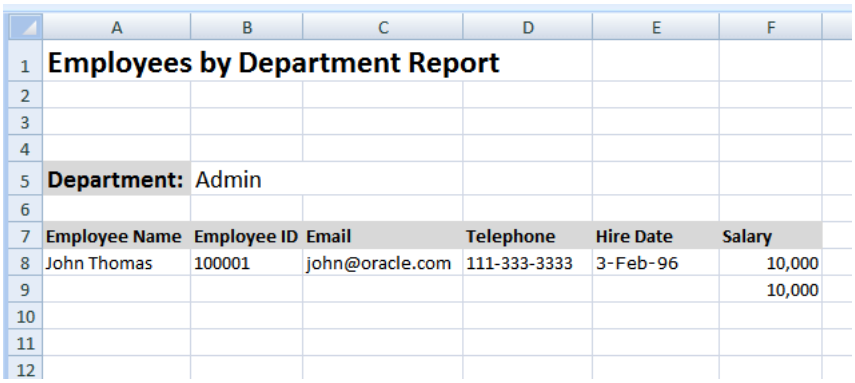
Excel cannot recognize canonical date format. If the date format in the XML data is in canonical format, that is, YYYY-MM-DDThh:mm:ss+HH:MM, you must apply a function to display it properly.

One option to display a date is to use the Excel REPLACE and SUBSTITUTE functions. This option retains the full date and timestamp. If you only require the date portion in the data (YYY-MM-DD), then another option is to use the DATEVALUE function. The following example shows how to use both options.

Example: Formatting a Canonical Date in Excel

Using the Employee by Department template and data from the first example, assume you want to add the HIRE_DATE element to the layout to and display the date as shown in Column E of Figure 3–11.

Figure 3–11 The Employee by Department Template Showing the Hire Date



	A	B	C	D	E	F
1	**Employees by Department Report**					
2						
3						
4						
5	**Department:**	Admin				
6						
7	**Employee Name**	**Employee ID**	**Email**	**Telephone**	**Hire Date**	**Salary**
8	John Thomas	100001	john@oracle.com	111-333-3333	3-Feb-96	10,000
9						10,000
10						
11						
12						

To format the date:

1. Copy and paste a sample value for HIRE_DATE from the XML data into the cell that is to display the HIRE_DATE field. For example:

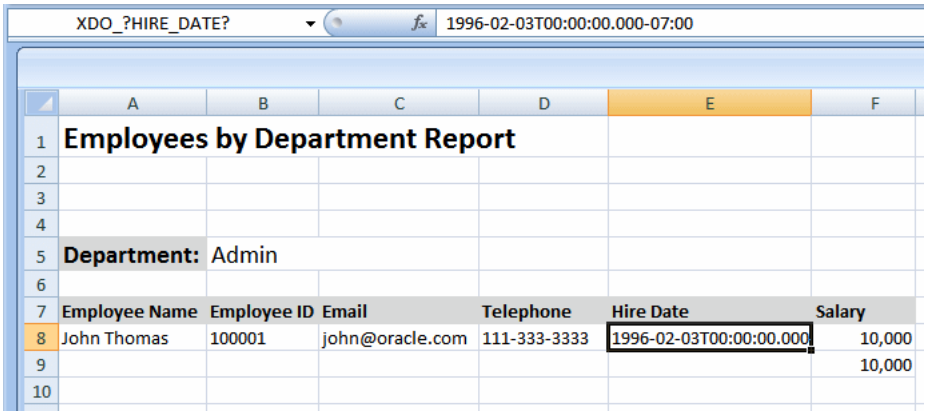
   Copy and paste

   1996-02-03T00:00:00.000-07:00

   into the E8 cell.

2. Assign the cell the defined name XDO_?HIRE_DATE? to map it to the HIRE_DATE element in the data, as shown in Figure 3–12.

Figure 3–12 Assigning the Defined Name XDO_?HIRE_DATE?



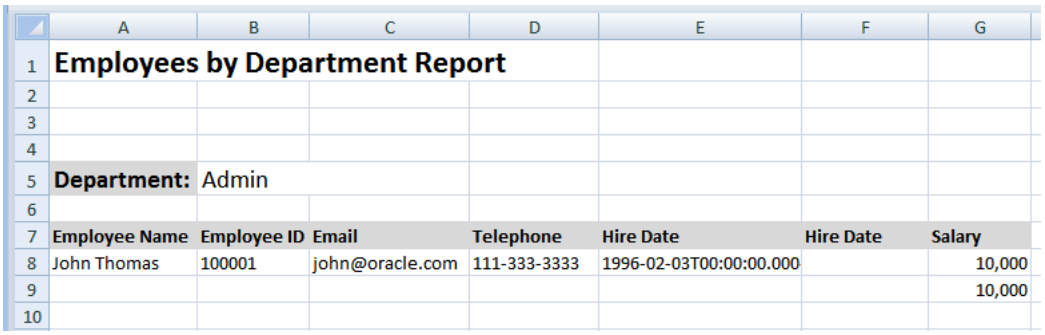
XDO_?HIRE_DATE? | 1996-02-03T00:00:00.000-07:00

	A	B	C	D	E	F
1	**Employees by Department Report**					
2						
3						
4						
5	**Department:**	Admin				
6						
7	**Employee Name**	**Employee ID**	**Email**	**Telephone**	**Hire Date**	**Salary**
8	John Thomas	100001	john@oracle.com	111-333-3333	1996-02-03T00:00:00.000	10,000
9						10,000
10						

   If you do nothing else, the HIRE_DATE value is displayed as shown. To format the date as "3-Feb-96", you must apply a function to that field and display the results in a new field.

3. Insert a new Hire Date column. This is now column F, as shown in Figure 3–13.

Figure 3–13 The New Hire Date Column in Column F



	A	B	C	D	E	F	G
1	**Employees by Department Report**						
2							
3							
4							
5	**Department:**	Admin					
6							
7	**Employee Name**	**Employee ID**	**Email**	**Telephone**	**Hire Date**	**Hire Date**	**Salary**
8	John Thomas	100001	john@oracle.com	111-333-3333	1996-02-03T00:00:00.000		10,000
9							10,000
10							

4. In the new Hire Date cell (F8), enter one of the following Excel functions:

   - To retain the full date and timestamp, enter:

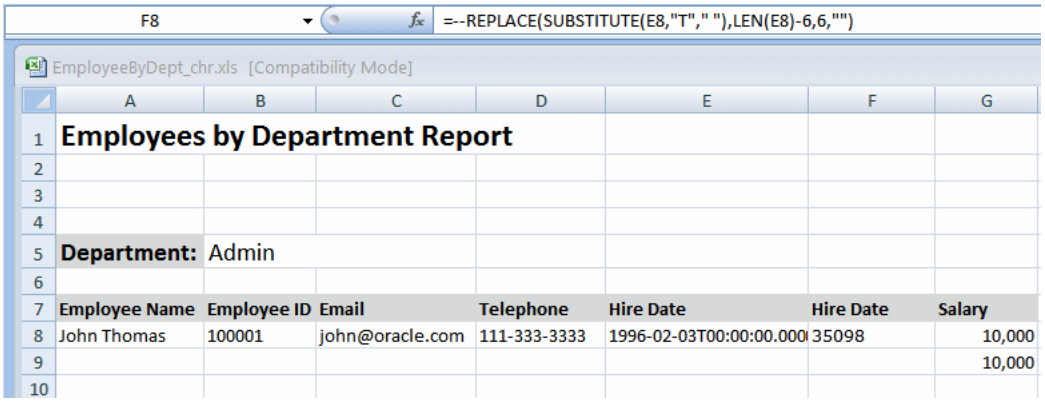
     ```
     =--REPLACE(SUBSTITUTE(E8,"T"," "),LEN(E8)-6,6,"")
     ```

   - To retain only the date portion (YYY-MM-DD), enter:

     ```
     DATEVALUE(LEFT(E8,10))
     ```

   After you enter the function, it populates the F8 cell as shown in Figure 3–14.

Figure 3–14 Hire Date Cell (F8) Populated



F8 =--REPLACE(SUBSTITUTE(E8,"T"," "),LEN(E8)-6,6,"")

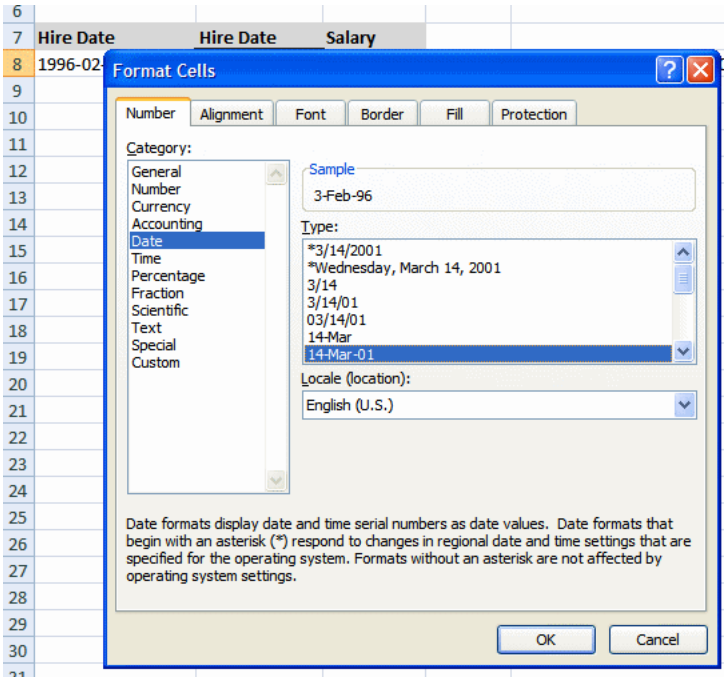
EmployeeByDept_chr.xls [Compatibility Mode]

	A	B	C	D	E	F	G
1	**Employees by Department Report**						
2							
3							
4							
5	**Department:**	Admin					
6							
7	**Employee Name**	**Employee ID**	**Email**	**Telephone**	**Hire Date**	**Hire Date**	**Salary**
8	John Thomas	100001	john@oracle.com	111-333-3333	1996-02-03T00:00:00.000	35098	10,000
9							10,000
10							

5. Apply formatting to the cell.

   Right-click the F8 cell. From the menu, select Format Cells. In the Format Cells dialog, select Date and the desired format, as shown in Figure 3–15.

Figure 3–15 Applying the Format for the Date in the Format Cells Dialog

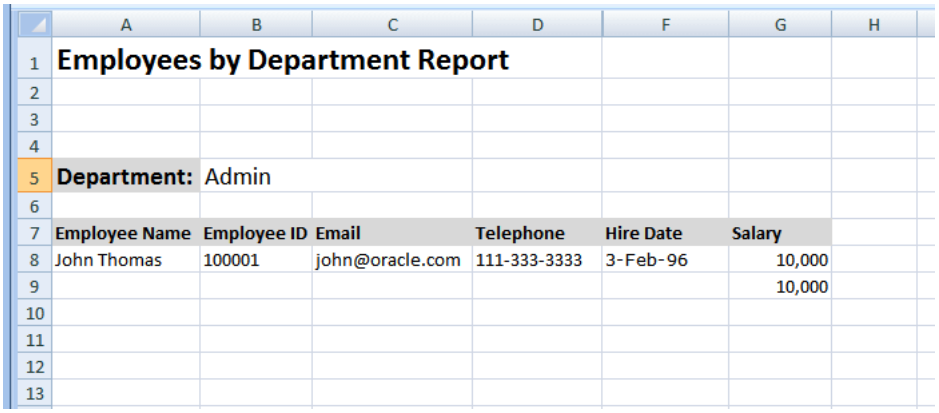


The sample data in the F8 cell now displays as 3-Feb-96.

6. Hide the E column, so that report consumers do not see the canonical date that is converted.

   Figure 3–16 shows column E hidden.

Figure 3–16 Column E Hidden



3.5 Defining BI Publisher Functions

BI Publisher provides a set of functions to achieve additional reporting functionality. You define these functions in the Data Constraints region of the XDO_METADATA sheet.

The functions make use of Columns A, B, and C in the XDO_METADATA sheet as follows:

Use Column A to declare the function or to specify the defined name of the object to which to map the results of a calculation or XSL evaluation.

Use Column B to enter the special XDO-XSL syntax to describe how to control the data constraints for the XDO function, or the XSL syntax that describes the special constraint to apply to the XDO_ named elements.

Use Column C to specify additional instructions for a few functions.

The functions are described in the following three sections:

- Section 3.5.1, "Reporting Functions"
- Section 3.5.2, "Formatting Functions That Rely on Specific Data Attribute Values"
- Section 3.5.3, "Grouping Functions"

3.5.1 Reporting Functions

Table 3–1 lists functions that you can add to a template using the commands shown and a combination of BI Publisher syntax and XSL. A summary list of the commands is shown in Table 3–1. See the corresponding section for details on usage.

Table 3–1 Reporting Functions

Function	Commands
Section 3.5.1.1, "Splitting the Report into Multiple Sheets"	XDO_SHEET_? with XDO_SHEET_NAME_?
Section 3.5.1.2, "Declaring and Passing Parameters"	XDO_PARAM_?*n*?
Section 3.5.1.3, "Defining a Link"	XDO_LINK_?*link object name*?
Section 3.5.1.4, "Importing and Calling a Subtemplate"	XDO_SUBTEMPLATE_?*n*?
Section 3.5.1.5, "Referencing Java Extension Libraries"	XDO_EXT_?*n*?

3.5.1.1 Splitting the Report into Multiple Sheets

Note: Images are not supported across multiple sheets. If the template sheet includes images, when the data is split into multiple sheets, the images are displayed only on the first sheet.

Use the set of commands to define the logic to split the report data into multiple sheets, as described in the following list:

- Use XDO_SHEET_? to define the logic by which to split the data onto a new sheet.
- Use XDO_SHEET_NAME_? to specify the naming convention for each sheet.

Table 3–2 describes the column entries.

Table 3–2 Column Entries

Column A Entry	Column B Entry	Column C Entry
XDO_SHEET_?	<?*xsl_evaluation to split the data*?> Example: <?.//DEPT?>	n/a

Table 3–2 (Cont.) Column Entries

Column A Entry	Column B Entry	Column C Entry
XDO_SHEET_ NAME_?	<?*xsl_expression to name the sheet*?> Example: <?concat(.//DEPARTMENT_ NAME,'-',count(.//EMP_NAME))?>	(Optional) <?*original sheet name*?> Example: <?Sheet3?>

XDO_SHEET_? must refer to an existing high-level node in the XML data. The example <?.//DEPT?> creates a new sheet for each occurrence of <DEPT> in the data.

If the data is flat, then you cannot use this command unless you first preprocess the data to create the desired hierarchy. To preprocess the data, define the transformation in an XSLT file, then specify this file in the Preprocess XSLT File field of the header section of the XDO _METADATA sheet. For more information, see Section 3.6, "Preprocessing the Data Using an XSL Transformation (XSLT) File."

Use XDO_SHEET_NAME_? to define the name to apply to the sheets. In Column B enter the XSL expression to derive the new sheet name. The expression can reference a value for an element or attribute in the XML data, or you can use the string operation on those elements to define the final sheet name. This example:

```
<?concat(.//DEPARTMENT_NAME,'-',count(.//EMP_NAME))?>
```

names each sheet using the value of DEPARTMENT_NAME concatenated with "-" and the count of employees in the DEPT group.

The original sheet name entry in Column C tells BI Publisher on which sheet to begin the specified sheet naming. If this parameter is not entered, BI Publisher applies the naming to the first sheet in the workbook that contains XDO_ names. You must enter this parameter if, for example, you have a report that contains summary data in the first two worksheets and the burst data should begin on Sheet3. In this case, you enter <?SHEET3?> in Column C.

Example: Splitting the data into multiple sheets

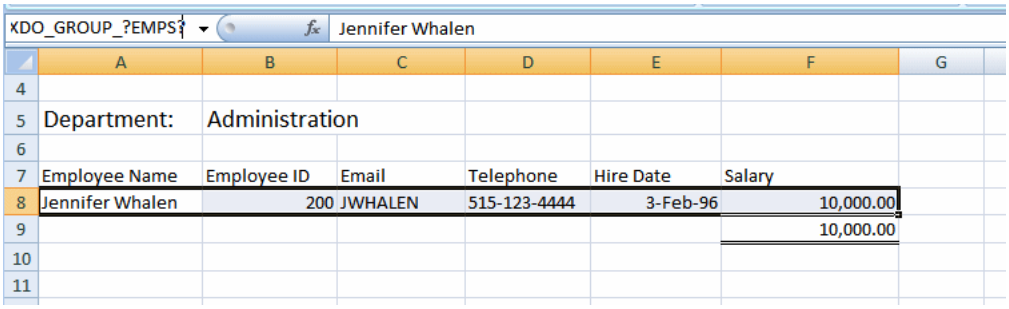
Using the employee data shown in the previous example. This example:

- Creates a new worksheet for each department
- Names each worksheet the name of the department with the number of employees in the department, for example: Sales-21.

To split the data into sheets:

1. Enter the defined names for each cell of employee data and create the group for the repeating employee data, as shown in Figure 3–17.

Figure 3–17 Defining Employee Data and the Group for Repeating Employee Data



Note: Do not create the grouping around the department because the data is split by department.

2. Enter the values that are described in Table 3–3 in the Data Constraints section of the XDO_METADATA sheet.

Table 3–3 Data Constraints Values

Column A Entry	Column B Entry
XDO_SHEET_?	<?.//DEPT?>
XDO_SHEET_NAME_?	<?concat(.//DEPARTMENT_NAME,'-',count(.//EMP_NAME))?>

The entries are shown in Figure 3–18.

Figure 3–18 Entries for Data Constraints

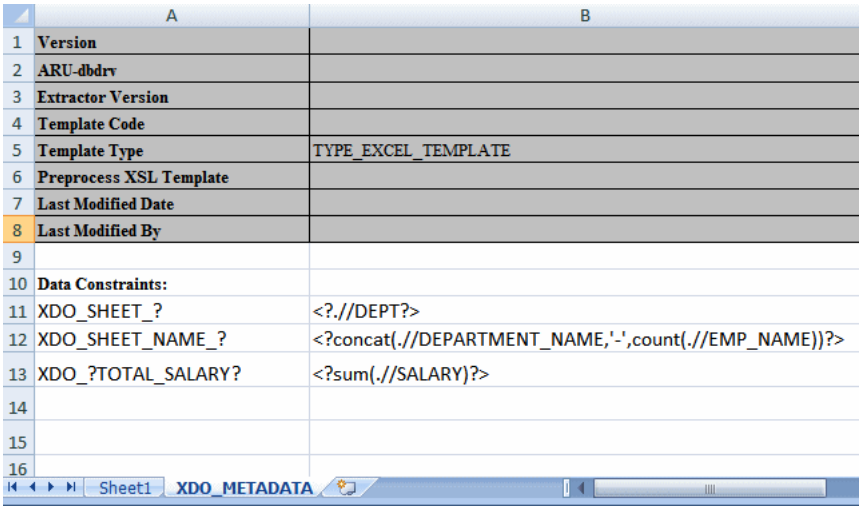
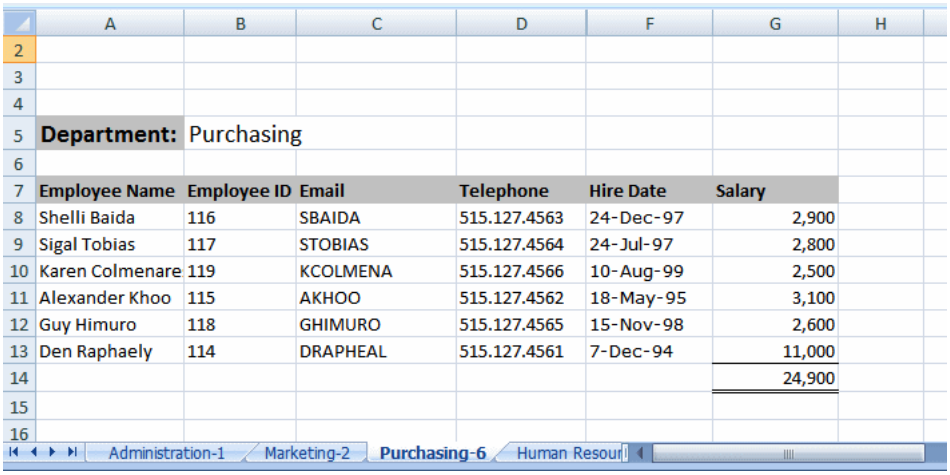


Figure 3–19 shows the generated report. Each department data now displays on its own sheet, which shows the naming convention specified.

Figure 3–19 Example of a Generated Report



3.5.1.2 Declaring and Passing Parameters

To define a parameter, use the XDO_PARAM_?*n*? function to declare the parameter, then use the $*parameter_name* syntax to pass a value to the parameter. A parameter must be defined in the data model.

To declare the parameter, use the command that is described in Table 3–4.

Table 3–4 Command for Declaring Parameters

Column A Entry	Column B Entry
XDO_PARAM_?*n*? where *n* is unique identifier for the parameter	<?param@begin:*parameter_name*;*parameter_value*?> where *parameter_name* is the name of the parameter from the data model and *parameter_value* is the optional default value. For example: <?param@begin:Country;US?>

To use the value of the parameter directly in a cell, refer to the parameter as $*parameter_name* in the definition for the XDO_ defined name, as described in Table 3–5.

Table 3–5 Using a Parameter Directly

Column A Entry	Column B Entry
XDO_PARAM_?parameter_name? For example: XDO_PARAM_?Country?	<?$*parameter_name*?>. For example: <?$Country?>

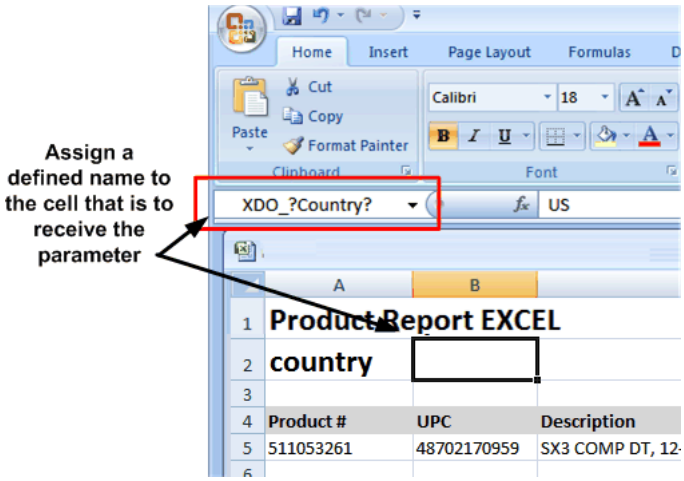
You can also refer to the parameter in other logic or calculations in the XDO_METADATA sheet using $parameter_name.

Example: Defining and Passing a Parameter

To declare and reference a parameter named Country:

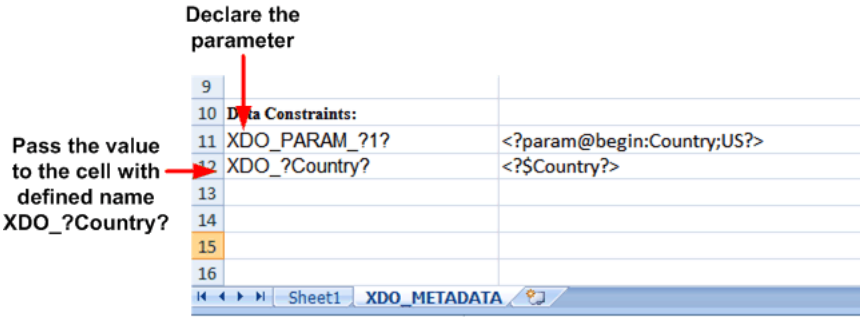
1. In the template sheet, mark the cell with a defined name. In the figure below, the cell has been marked with the defined name XDO_?Country?

Figure 3–20 Cell Marked with the Defined Name XDO_?Country?



2. In the hidden sheet assign that cell the parameter value, as shown in Figure 3–21.

Figure 3–21 Assigning a Parameter Value to the XDO_?Country?Cell



3.5.1.3 Defining a Link

Use the XDO_LINK_? command to define a hyperlink for any data cell, as described in Table 3–6.

Example: Defining a Link

Table 3–6 Defining a Link

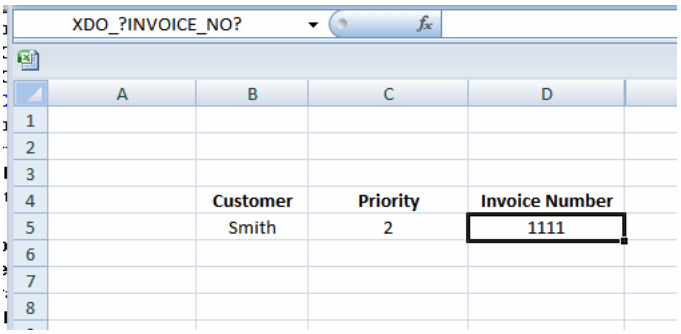
Column A Entry	Column B Entry
XDO_LINK_?*cell object name*?	*<xsl statement to build the dynamic URL>*
For example:	For example:
XDO_LINK_?INVOICE_NO?	<xsl:value-of select="concat('https://server.company.com/documents/invoice_print.show?c_rptno=',./INVOICE_NO)"/>

Assume your company generates customer invoices. The invoices are stored in a central location accessible by a Web server and can be identified by the invoice number (INVOICE_NO).

To generate a report that creates a dynamic link to each invoice:

1. In the template sheet, assign the cell that is to display the INVOICE_NO the XDO defined name: XDO_?INVOICE_NO?, as shown in Figure 3–22.

Figure 3–22 Assigning the Defined Name XDO_?INVOICE_NO?



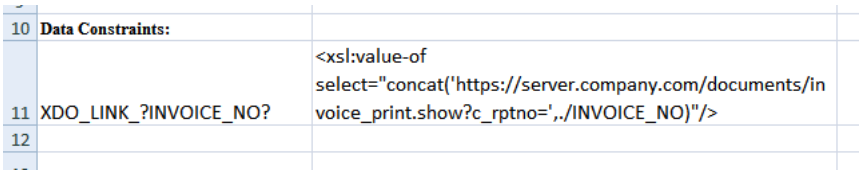
2. In the XDO_METADATA sheet, enter the appropriate values, as described in Table 3–7:

Table 3–7 Generating a Report with a Link

Column A Entry	Column B Entry
XDO_LINK_?INVOICE_NO?	<xsl:value-of select="concat('https://server.company.com/documents/invoice_print.show?c_rptno=',./INVOICE_NO)"/>

The entries in Excel are shown in Figure 3–23.

Figure 3–23 XDO_LINK?_?INVOICE_NO? Entries in Excel



The report output is displayed as shown in Figure 3–24. The logic that is defined in the XDO_METADATA sheet is applied to create a hyperlink for each INVOICE_NO entry.

Figure 3–24 Example of the Report Output

Customer	Priority	Invoice Number
Aaron	3	5952174
Abner	3	8280605
Acheson	2	7604618
Addison	3	9645172
Aeschelle	3	9616238
Agers	2	9685385
Ahab	3	9644915
Aiden	3	9663648
Ajackal	3	9685016
Akkers	3	9706403

3.5.1.4 Importing and Calling a Subtemplate

Use these commands to declare XSL subtemplates that you can then call and reference in any of the XDO_ commands.

> **Note:** The Template Viewer does not support preview for templates that import subtemplates.

To import the subtemplate, enter the command shown in Table 3–8.

Table 3–8 Importing a Subtemplate

Column A Entry	Column B Entry
XDO_SUBTEMPLATE_?*n*? where *n* is a unique identifier. For example: XDO_SUBTEMPLATE_?1?	<xsl:import href="xdoxsl:///*path to subtemplate*"/> For example: <xsl:import href="http://server.company.com:8080/subtemplates/PaymentsSummary-SubTemplate.xsl"/>

To call the subtemplate, declare the cell name for which the results should be returned in Column A, then enter the call-template syntax with any other XSL processing to be performed. The commands are shown in Table 3–9.

Table 3–9 Calling a Subtemplate

Column A Entry	Column B Entry
XDO_?*cell object name*?	<xsl:call-template name="*template_name*"> </xsl:call-template>

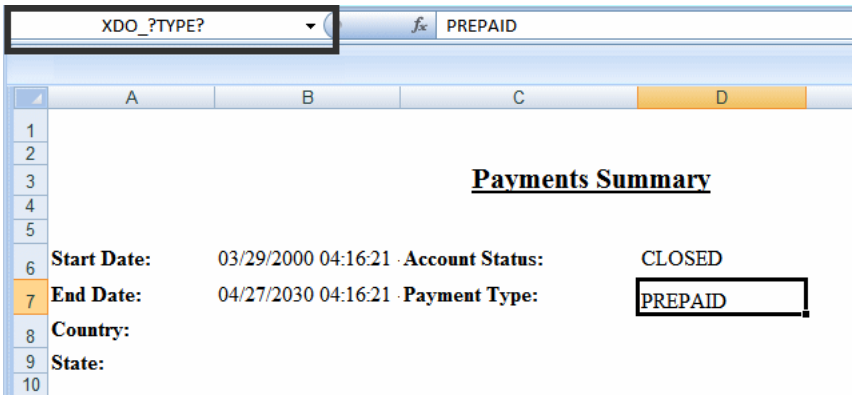
Example: Importing and Calling a Subtemplate

Assume you have the following subtemplate to evaluate the value of a parameter named pPayType and based on the value, will return a string that indicates the payment type:

```
<?xml version="1.0" encoding="utf-8"?>
<xsl:stylesheet version="2.0"
 xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="/">
    </xsl:template>
      <xsl:template name="BRM_PAY_TYPES">
         <xsl:param name="pPayType" select="string('ALL')"/>
         <xsl:choose>
            <xsl:when test="$pPayType = '0'">UNDEFINED</xsl:when>
            <xsl:when test="$pPayType=string('10000')">PREPAID</xsl:when>
            <xsl:when test="$pPayType=string('10001')">INVOICE</xsl:when>
            <xsl:when test="$pPayType=string('10003')">CREDIT CARD</xsl:when>
            <xsl:when test="$pPayType=string('10005')">DIRECT DEBIT</xsl:when>
            <xsl:when test="$pPayType=string('10011')">CASH</xsl:when>
            <xsl:when test="$pPayType=string('10012')">CHECK</xsl:when>
            <xsl:when test="$pPayType=string('ALL')">ALL</xsl:when>
            </xsl:choose>
        </xsl:template>
</xsl:stylesheet>
```

In the Excel template, you have defined a field with the XDO Defined Name XDO_?TYPE?, which is populated based on the string returned from code performed in the subtemplate, as shown in Figure 3–25.

Figure 3–25 Populated XDO_?TYPE? Field



Enter the commands shown in Table 3–10 in the Data Constraints region.

Table 3–10 Commands for Data Constraints Region

Column A Entry	Column B Entry
XDO_SUBTEMPLATE_?1?	<xsl:import href="http://server.company.com:8080/subtemplates/PaymentsSummary-SubTemplate.xsl"/>
XDO_?TYPE?	<xsl:call-template name="BRM_PAY_TYPES"> <xsl:with-param name="pPayType" select="string('10000')"/> </xsl:call-template>

The XDO_SUBTEMPLATE_?1? function imports the subtemplate from the specified location.

The XDO_?TYPE? cell entry maps the results of the subtemplate processing entered in Column B.

3.5.1.5 Referencing Java Extension Libraries

You can include the reference to a Java extension library in the template and then call methods from this library to perform processing in the template. Use the command shown in Table 3–11 to reference the Java extension libraries.

Table 3–11 Referencing Java Extension Libraries

Column A Entry	Column B Entry
XDO_EXT_?*n*? where *n* is a unique identifier. Example: XDO_EXT?1?	<?namespace:xmlns:bipext="*extension library*"?> Example: <?namespace:xmlns:bipext="http://www.company.com/XSL/Transform/java/company.com.xmlpublisher.reports.BIPExtension"?>

You can have multiple extension libraries defined in a single template file.

Example: Calling a Java Extension Library

Assume the extension library includes the following two methods that you want to call in the template:

- bipext:infTimeToStr()
- bipext:infStrToTimet()

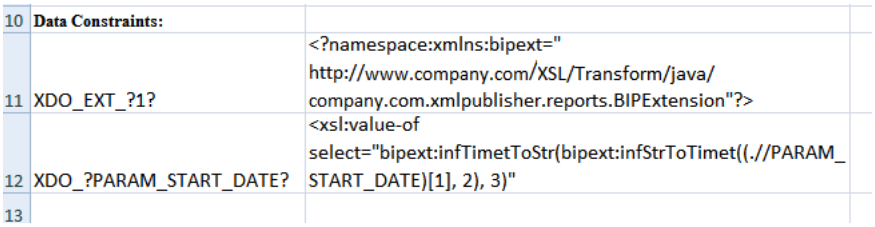
After you have declared the library as shown above, specify the cell to which you want to apply the method by entering the XDO defined name in Column A and calling the function in Column B. Table 3–12 shows example commands.

Table 3–12 Example: Calling a Java Extension Library

Column A Entry	Column B Entry
XDO_?PARAM_START_DATE?	<xsl:value-of select="bipext:infTimeToStr(bipext:infStrToTimet((.//PARAM_START_DATE)[1],2),3)"

The entries in the XDO_METADATA sheet to declare and call the Java extension libraries are shown in Figure 3–26.

Figure 3–26 Entries in the XDO_METADATA Sheet to Declare and Call the Java Extension Libraries



10	**Data Constraints:**	
11	XDO_EXT_?1?	<?namespace:xmlns:bipext=" http://www.company.com/XSL/Transform/java/ company.com.xmlpublisher.reports.BIPExtension"?>
12	XDO_?PARAM_START_DATE?	<xsl:value-of select="bipext:infTimetToStr(bipext:infStrToTimet((.//PARAM_ START_DATE)[1], 2), 3)"
13		

3.5.2 Formatting Functions That Rely on Specific Data Attribute Values

The following commands require that specific formatting attributes be present in the XML data file. A summary list of the commands is shown in Table 3–13. See the corresponding section for details on usage.

Table 3–13 Commands for Specific Formatting Attributes

Function	Command
Section 3.5.2.1, "Defining Border and Underline Styles"	XDO_STYLE_*n*_?*cell object name*?
Section 3.5.2.2, "Skipping a Row"	XDO_SKIPROW_?*cell object name*?

3.5.2.1 Defining Border and Underline Styles

While you can define a consistent style in the template using Excel formatting, the XDO_STYLE command enables you to define a different style for any data cell dynamically based on the XML data.

With the XDO_STYLE command you specify the cell to which to apply the style, the logic to determine when to apply the style, and the style type to apply. The style value must be present in the XML data. Table 3–14 provides examples.

Table 3–14 Defining Border and Underline Styles

Column A Entry	Column B Entry	Column C Entry
XDO_STYLE_n_?*cell_object_name*? For example: XDO_STYLE_1_?TOTAL_ SALARY?	*<xsl evaluation that returns a supported value>* For example: <xsl:value-of select=".//TOTAL_ SALARY/@borderStyle"/>	Style type For example: BottomBorderStyle

BI Publisher supports the normal Excel style types and values as shown in Table 3–15.

Table 3–15 Excel Style Types and Values

Style Type	Supported Values (Must be in returned by evaluation in Column B)	Supported Types (Enter in Column C)
Normal	BORDER_NONE BORDER_THIN BORDER_MEDIUM BORDER_DASHED BORDER_DOTTED BORDER_THICK BORDER_DOUBLE BORDER_HAIR BORDER_MEDIUM_DASHED BORDER_DASH_DOT BORDER_MEDIUM_DASH_DOT BORDER_DASH_DOT_DOT BORDER_MEDIUM_DASH_DOT_DOT BORDER_SLANTED_DASH_DOT	BottomBorderStyle TopBorderStyle LeftBorderStyle RightBorderStyle DiagonalLineStyle

You can also set a color using one of the types shown in Table 3–16.

Table 3–16 Color Types

Style Type	Supported Value (Must be in returned by evaluation in Column B)	Supported Types (Enter in Column C)
Normal	When you set Color Style, give the value in RRBBGG hex format, for example: borderColor="0000FF"	BottomBorderColor TopBorderColor LeftBorderColor RightBorderColor DiagonalLineColor

BI Publisher also supports the underline type with the values shown in Table 3–17.

Table 3–17 Underline Types

Style Type	Supported Values (Must be in returned by evaluation in Column B)	Supported Type (Enter in Column C)
Underline	UNDERLINE_NONE UNDERLINE_SINGLE UNDERLINE_DOUBLE UNDERLINE_SINGLE_ACCOUNTING UNDERLINE_DOUBLE_ACCOUNTING	UnderlineStyle

You can have multiple underline styles defined for a single cell.

Example: Defining Styles

To apply a style in a template, the style value must be present in the data. In this example, a border style and an underline style are applied to the DEPT_TOTAL_SALARY field shown in the Excel template.

For this example, the following data is used. Note that the DEPT_TOTAL_SALARY element in the data has these attributes defined:

- borderStyle
- underLineStyle
- borderColor

The value of each of these attributes is used to apply the defined style based on logic defined in the template.

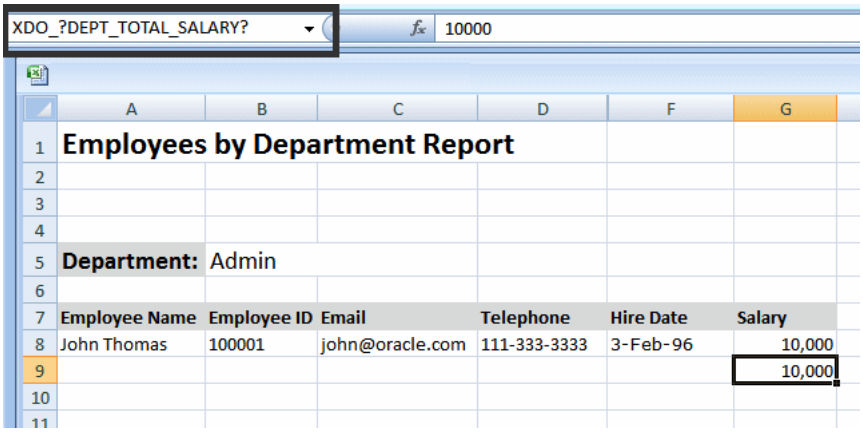
```
<?xml version="1.0" encoding="UTF-8"?>

<EMPLOYEES>
  <G_DEPT>
    <DEPARTMENT_ID>10</DEPARTMENT_ID>
    <DEPARTMENT_NAME>Administration</DEPARTMENT_NAME>
    <LIST_G_EMP>
      <G_EMP>
        <EMPLOYEE_ID>200</EMPLOYEE_ID>
        <EMP_NAME>Jennifer Whalen</EMP_NAME>
        <EMAIL>JWHALEN</EMAIL>
        <PHONE_NUMBER>515.123.4444</PHONE_NUMBER>
        <HIRE_DATE>1987-09-17T00:00:00.000-06:00</HIRE_DATE>
        <SALARY>4400</SALARY>
      </G_EMP>
    </LIST_G_EMP>

    <DEPT_TOTAL_SALARY borderStyle="BORDER_DOUBLE" underLineStyle="UNDERLINE_
DOUBLE_ACCOUNTING" borderColor="0000FF">4400</DEPT_TOTAL_SALARY>
  </G_DEPT>
  <G_DEPT>
    <DEPARTMENT_ID>20</DEPARTMENT_ID>
    <DEPARTMENT_NAME>Marketing</DEPARTMENT_NAME>
    <LIST_G_EMP>
      <G_EMP>
        <EMPLOYEE_ID>201</EMPLOYEE_ID>
        <EMP_NAME>Michael Hartstein</EMP_NAME>
        <EMAIL>MHARTSTE</EMAIL>
        <PHONE_NUMBER>515.123.5555</PHONE_NUMBER>
        <HIRE_DATE>1996-02-17T00:00:00.000-07:00</HIRE_DATE>
        <SALARY>13000</SALARY>
      </G_EMP>
      <G_EMP>
        <EMPLOYEE_ID>202</EMPLOYEE_ID>
        <EMP_NAME>Pat Fay</EMP_NAME>
        <EMAIL>PFAY</EMAIL>
        <PHONE_NUMBER>603.123.6666</PHONE_NUMBER>
        <HIRE_DATE>1997-08-17T00:00:00.000-06:00</HIRE_DATE>
        <SALARY>6000</SALARY>
        </G_EMP>
      </LIST_G_EMP>

      <DEPT_TOTAL_SALARY borderStyle="BORDER_DOUBLE" underLineStyle="UNDERLINE_
DOUBLE_ACCOUNTING" borderColor="0000FF">19000</DEPT_TOTAL_SALARY>
```

```
  </G_DEPT>

...

</EMPLOYEES>
```

To define a style:

1. In the Excel template, assign the defined name XDO_?DEPT_TOTAL_SALARY? to the field that is to display the DEPT_TOTAL_SALARY from the data, as shown in Figure 3–27.

Figure 3–27 Assigning the Defined Name XDO_?DEPT_TOTAL_SALARY?



2. In the XDO_METADATA sheet, enter the following:
   - To define the top border style, use the entries shown in Table 3–18.

Table 3–18 Top Border Style

Column A Entry	Column B Entry	Column C Entry
XDO_STYLE_1_?DEPT_TOTAL_SALARY?	<xsl:value-of select=".//DEPT_TOTAL_SALARY/@borderStyle"/>	TopBorderStyle

The entry in Column A maps this style command to the cell assigned the name XDO_?DEPT_TOTAL_SALARY?

The entry in Column B retrieves the style value from the attribute borderStyle of the DEPT_TOTAL_SALARY element. Note from the sample data that the value for borderStyle is "BORDER_DOUBLE".

The entry in Column C tells BI Publisher to apply a TopBorderStyle to the cell.

- To define the top border color, use the entries shown in Table 3–19.

Table 3–19 Top Border Color

Column A Entry	Column B Entry	Column C Entry
XDO_STYLE_2_?DEPT_TOTAL_SALARY?	<?.//DEPT_TOTAL_SALARY/@borderColor?>	TopBorderColor

The entry in Column A maps this style command to the cell assigned the name XDO_?DEPT_TOTAL_SALARY?

The entry in Column B retrieves the style value from the attribute borderColor of the DEPT_TOTAL_SALARY element. Note from the sample data that the value for borderColor is "0000FF" (blue).

The entry in Column C tells BI Publisher to apply a TopBorderColor to the cell.

- To define the underline style, use the entries shown in Table 3–20.

Table 3–20 Underline Style

Column A Entry	Column B Entry	Column C Entry
XDO_STYLE_3_?DEPT_TOTAL_SALARY?	<?.//DEPT_TOTAL_SALARY/@underLineStyle?>	UnderlineStyle

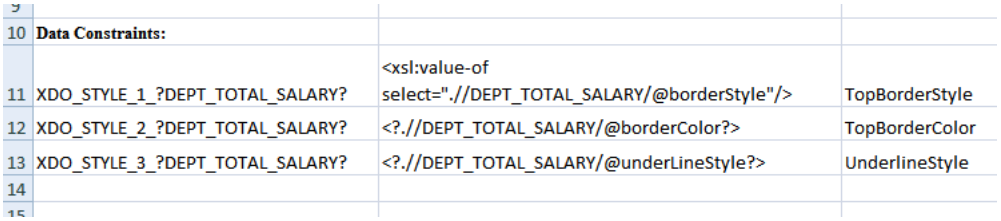
The entry in Column A maps this style command to the cell assigned the name XDO_?DEPT_TOTAL_SALARY?

The entry in Column B retrieves the style value from the attribute underLineStyle of the DEPT_TOTAL_SALARY element. Note from the sample data that the value for underLineStyle is "UNDERLINE_DOUBLE_ACCOUNTING".

The entry in Column C tells BI Publisher to apply the UnderLineStyle to the cell.

Figure 3–28 shows the three entries in the Data Constraints region.

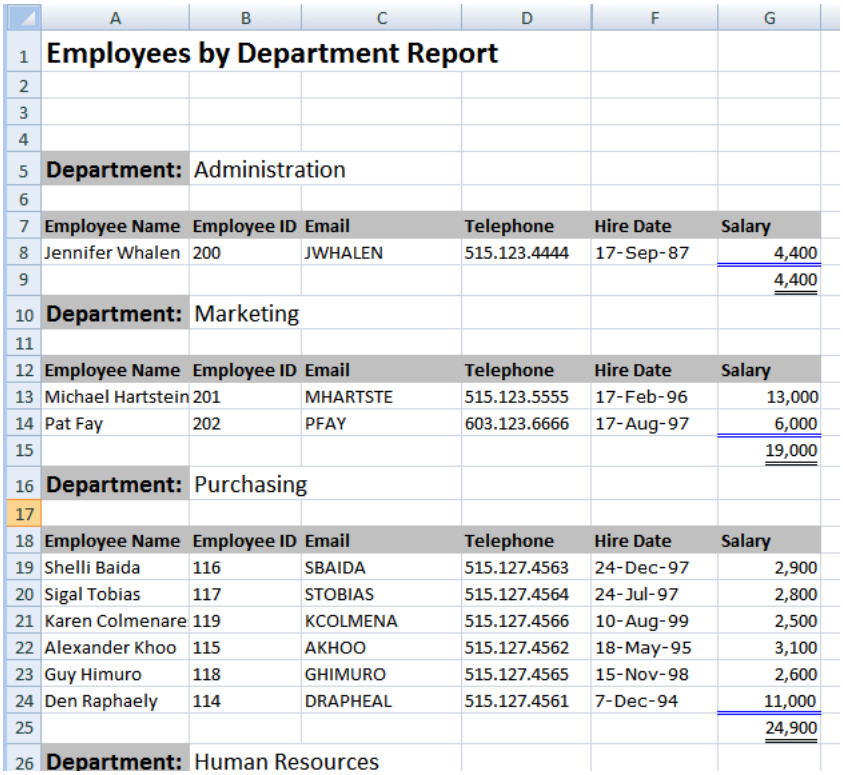
Figure 3–28 Entries for Data Constraints



10	**Data Constraints:**		
11	XDO_STYLE_1_?DEPT_TOTAL_SALARY?	<xsl:value-of select=".//DEPT_TOTAL_SALARY/@borderStyle"/>	TopBorderStyle
12	XDO_STYLE_2_?DEPT_TOTAL_SALARY?	<?.//DEPT_TOTAL_SALARY/@borderColor?>	TopBorderColor
13	XDO_STYLE_3_?DEPT_TOTAL_SALARY?	<?.//DEPT_TOTAL_SALARY/@underLineStyle?>	UnderlineStyle
14			

When you run the report, the style commands are applied to the XDO_?DEPT_TOTAL_SALARY? cell, as shown in Figure 3–29.

Figure 3–29 A Generated Report Showing Style Commands Applied to the XDO_?DEPT_TOTAL_SALARY? Cell



	A	B	C	D	F	G
1	**Employees by Department Report**					
2						
3						
4						
5	**Department:**	Administration				
6						
7	**Employee Name**	**Employee ID**	**Email**	**Telephone**	**Hire Date**	**Salary**
8	Jennifer Whalen	200	JWHALEN	515.123.4444	17-Sep-87	4,400
9						4,400
10	**Department:**	Marketing				
11						
12	**Employee Name**	**Employee ID**	**Email**	**Telephone**	**Hire Date**	**Salary**
13	Michael Hartstein	201	MHARTSTE	515.123.5555	17-Feb-96	13,000
14	Pat Fay	202	PFAY	603.123.6666	17-Aug-97	6,000
15						19,000
16	**Department:**	Purchasing				
17						
18	**Employee Name**	**Employee ID**	**Email**	**Telephone**	**Hire Date**	**Salary**
19	Shelli Baida	116	SBAIDA	515.127.4563	24-Dec-97	2,900
20	Sigal Tobias	117	STOBIAS	515.127.4564	24-Jul-97	2,800
21	Karen Colmenare	119	KCOLMENA	515.127.4566	10-Aug-99	2,500
22	Alexander Khoo	115	AKHOO	515.127.4562	18-May-95	3,100
23	Guy Himuro	118	GHIMURO	515.127.4565	15-Nov-98	2,600
24	Den Raphaely	114	DRAPHEAL	515.127.4561	7-Dec-94	11,000
25						24,900
26	**Department:**	Human Resources				

3.5.2.2 Skipping a Row

Use the XDO_SKIPROW command to suppress the display of a row of data in a table when the results of an evaluation defined in Column B return the case insensitive string "True". Example entries are shown in Table 3–21.

Table 3–21 Skipping a Row

Column A Entry	Column B Entry
XDO_SKIPROW_?*cell_object_name*? For example: XDO_SKIPROW_?EMPLOYEE_ID?	*<xsl evaluation that returns the string "True"/>* For example: <xsl:if test="string-length(./EMPLOYEE_ID/@MANAGER) != 0"> <xsl:value-of select="./EMPLOYEE_ID/@MANAGER"/> </xsl:if>

Example: Skipping a Row Based on Data Element Attribute

In this example, the Excel template suppresses the display of the row of employee data when the EMPLOYEE_ID element includes a "MANAGER" attribute with the value "True".

Assume data as shown below. Note that the EMPLOYEE_ID element for employee Michael Hartstein has the MANAGER attribute with the value "True". The other EMPLOYEE_ID elements in this set do not have the attribute.

```
<?xml version="1.0" encoding="UTF-8"?>

<EMPLOYEES>
  <G_DEPT>
```

```
    <DEPARTMENT_ID>20</DEPARTMENT_ID>
    <DEPARTMENT_NAME>Marketing</DEPARTMENT_NAME>
    <LIST_G_EMP>
      <G_EMP>
        <EMPLOYEE_ID MANAGER="TRUE">201</EMPLOYEE_ID>
        <EMP_NAME>Michael Hartstein</EMP_NAME>
        <EMAIL>MHARTSTE</EMAIL>
        <PHONE_NUMBER>515.123.5555</PHONE_NUMBER>
        <HIRE_DATE>1996-02-17T00:00:00.000-07:00</HIRE_DATE>
        <SALARY>13000</SALARY>
       </G_EMP>
       <G_EMP>
        <EMPLOYEE_ID>202</EMPLOYEE_ID>
        <EMP_NAME>Pat Fay</EMP_NAME>
        <EMAIL>PFAY</EMAIL>
        <PHONE_NUMBER>603.123.6666</PHONE_NUMBER>
        <HIRE_DATE>1997-08-17T00:00:00.000-06:00</HIRE_DATE>
        <SALARY>6000</SALARY>
       </G_EMP>
       <G_EMP>
        <EMPLOYEE_ID>652</EMPLOYEE_ID>
        <EMP_NAME>William Morgan</EMP_NAME>
        <EMAIL>WMORGAN</EMAIL>
        <PHONE_NUMBER>219.123.7776</PHONE_NUMBER>
        <HIRE_DATE>1994-10-17T00:00:00.000-06:00</HIRE_DATE>
        <SALARY>8000</SALARY>
       </G_EMP>
      </LIST_G_EMP>
  </G_DEPT>

...

</EMPLOYEES>
```

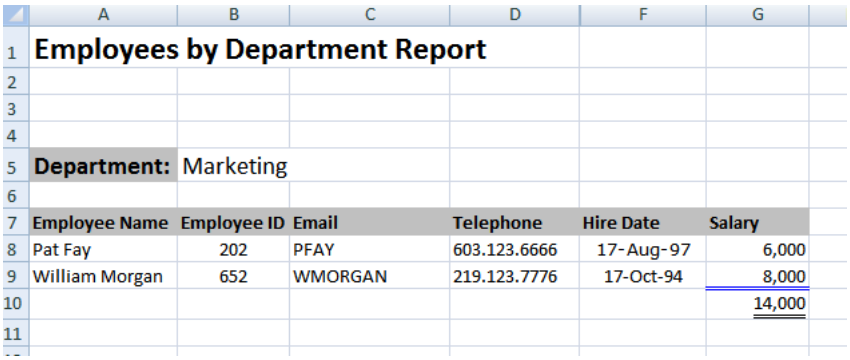
To suppress the display of the row of the employee data when the MANAGER attribute is set to "True", enter the entries shown in Table 3–22 in the Data Constraints section.

Table 3–22 Skipping a Row Based on Data Element Attribute

Column A Entry	Column B Entry
XDO_SKIPROW_?EMPLOYEE_ ID?	<xsl:if test="string-length(./EMPLOYEE_ ID/@MANAGER)!= 0"> <xsl:value-of select="./EMPLOYEE_ID/@MANAGER"/> </xsl:if>

The output from this template is shown in Figure 3–30. Note that the employee Michael Hartstein is not included in the report.

Figure 3–30 Output for a Skipped Row



	A	B	C	D	F	G
1	**Employees by Department Report**					
2						
3						
4						
5	**Department:**	Marketing				
6						
7	**Employee Name**	**Employee ID**	**Email**	**Telephone**	**Hire Date**	**Salary**
8	Pat Fay	202	PFAY	603.123.6666	17-Aug-97	6,000
9	William Morgan	652	WMORGAN	219.123.7776	17-Oct-94	8,000
10						14,000
11						

3.5.3 Grouping Functions

Use the functions shown in Table 3–23 to create groupings of data in the template.

Table 3–23 Creating Groupings of Data

Function	Command
Section 3.5.3.1, "Grouping the data"	XDO_GROUP_?*group element*?
Section 3.5.3.2, "Regrouping the Data"	XDO_REGROUP_?

3.5.3.1 Grouping the data

Use the XDO_GROUP command to group flat data when the layout requires a specific data grouping, for example, to split the data across multiple sheets. Example entries are shown in Table 3–24.

Table 3–24 Grouping the Data

Column A Entry	Column B Entry	Column C Entry
XDO_GROUP_?*group element*? For example: XDO_GROUP_?STATE_GROUP?	*<xsl beginning groupng logic/>* For example: <xsl:for-each-group select="current-group()" group-by="./STATE"> <xsl:for-each-group select="current-group()" group-by="./RESOURCE_NAME"> <xsl:for-each select="current-group()">	*<xsl ending groupng tags/>* For example: </xsl:for-each> </xsl:for-each-group> </xsl:for-each-group>

Define the XSL statements to be placed at the beginning and ending of the section of the group definition marked up by XDO_?*cell object name*?. You can mark multiple groups nested in the template, giving each the definition appropriate to the corresponding group.

3.5.3.2 Regrouping the Data

The XDO_REGROUP regroups the data by declaring the structure using the defined names. It does not require the XSLT logic. Entries are shown in Table 3–25.

Table 3–25 Regrouping the Data

Column A Entry	Column B Entry
XDO_REGROUP_?	XDO_REGROUP_ ?UniqueGroupID?levelName?groupByName?sortByName?sortByNam e?sortByName? where ■ UniqueGroupID is the ID of the group. It can be the same as the levelName or you can assign it unique name. ■ levelName is the XML level tag name in the XML data file or current-group() in the context of the XDO_ grouping structure. ■ groupByName is the field name that you want to use for the GroupBy operation for the current group. This name can be empty if the XDO_REGROUP_? command is used for the most inner group. ■ sortByName is the field name that you want to sort the group by. You can have multiple sortBy fields. If no sortByName is declared, then the data from the XML file is not sorted.

Table 3–26, Table 3–27, and Table 3–28 show an example of how to create three nested groupings.

Table 3–26 Creating Nested Groupings, Example 1

Column A Entry	Column B Entry
XDO_REGROUP_?	XDO_REGROUP_?PAYMENTSUMMARY_ Q1?PAYMENTSUMMARY_Q1?PAY_TYPE_NAME?

In the definition shown in Table 3–26, the most outer group is defined as PAYMENTSUMMARY_Q1, and it is grouped by PAY_TYPE_NAME

Table 3–27 Creating Nested Groupings, Example 2

Column A Entry	Column B Entry
XDO_REGROUP_?	XDO_REGROUP_?COUNTRYGRP?XDO_CURRGRP_?COUNTRY?

The definition shown in Table 3–27 creates a second outer group. The group is assigned the name COUNTRY_GRP and it is grouped by the element COUNTRY.

Table 3–28 Creating Nested Groupings, Example 3

Column A Entry	Column B Entry
XDO_REGROUP_?	XDO_REGROUP_?STATEGRP?XDO_CURRGRP_?STATE?

The definition shown in Table 3–28 creates the inner group STATEGRP and it includes a sortByName parameter: STATE.

3.6 Preprocessing the Data Using an XSL Transformation (XSLT) File

For the best performance, you should design the data model to perform as much of the data processing as possible. When it is not possible to get the required output from data engine, you can preprocess the data using an XSLT file that contains the instructions to transform the data. Some sample use cases may be:

- To create groups to establish the necessary hierarchy to support the desired layout

- To add style attributes to data elements
- To perform complex data processing logic that may be impossible in the Excel Template or undesirable for performance reasons

> **Note:** The Template Viewer does not support preview for templates that require XSLT preprocessing.

To use an XSLT preprocess file:

1. Create the file and save as .xsl.
2. Upload the file to the report definition in the BI Publisher repository, as you would a template:
   a. Navigate to the report in the repository.
   b. Click **Edit**.
   c. Click **Layouts**.
   d. Click **Browse** to locate the XSL file and then click **Upload**.
   e. Save the report definition.
3. In the Excel template, on the XDO_METADATA sheet, in the Header section, enter the file name for the **Preprocess XSLT File** parameter. For example: SplitByBrand.xsl

> **Note:** For testing purposes, you can create a Layout for the XSL file to enable you to view the intermediate data when the XSL template is applied to the data. To create the Layout:
>
> 1. With **Layouts** selected in the left pane, click **New**.
> 2. In the **Layout General Settings** pane, enter the following:
>   - Name - enter a name for this layout.
>   - Template - select the XSL template that you uploaded.
>   - Template Type - select XSL Stylesheet (XML)
>   - Output Format - All Formats
> 3. Click **Save**.
> 4. Run the report, applying the Excel template. If the output is not as expected in the Report Viewer, select the XSL template in the Report Viewer to view the preprocessed output.
>
> After testing is complete, delete the XSL layout definition from the report so that users will not see it as an option.

3.7 Debugging a Template Using the Template Viewer

If the template preview is not generating the results expected, then you can use the Template Viewer to enable trace settings to view debug messages. The Template Viewer also enables you to save and view the intermediate XSL file that is generated after the sample data and template are merged in the XSL-FO processor. If you are familiar with XSL, then this can be a very useful debugging tool.

The Template Viewer is installed when you install the Template Builder for Word; see Section 3.1.5, "Desktop Tools" for more information.

> **Note:** The Template Viewer does not support preview for templates that require XSLT preprocessing.

To preview with the Template Viewer and view log messages:

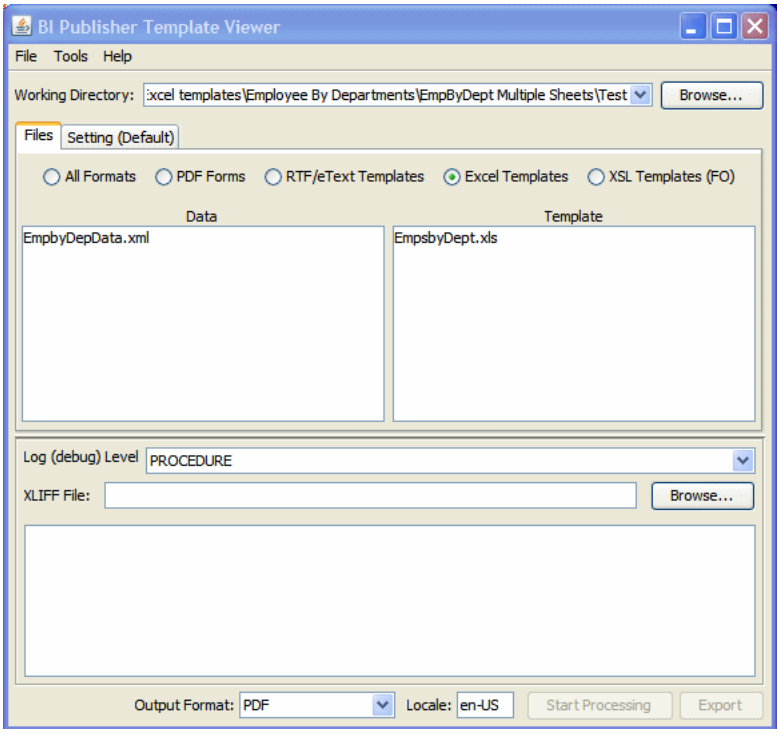
1. Open the Template Viewer:

   From the Windows desktop, click **Start**, then **Programs**, then **Oracle BI Publisher Desktop**, then **Template Viewer**.

2. Click **Browse** to locate the folder that contains the sample data file and template file. The data file and template file must reside in the same folder.

3. Select **Excel Templates**. The **Data** and **Template** regions display all .xml files and all .xls files present in the directory, as shown in Figure 3–31.

Figure 3–31 The Data and Template Regions Showing All .xml and .xls Files



4. Click the appropriate data and template files to select them.

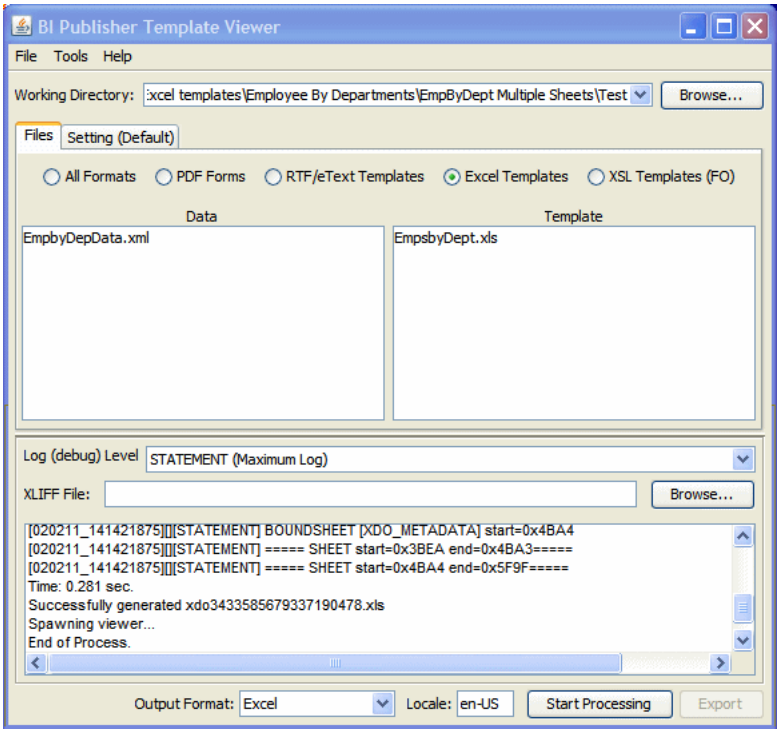
5. Select the log level.

6. From the **Output Format** list, select Excel.

7. Click **Start Processing**.

   The Template Viewer merges the selected data with the selected template and spawn the appropriate viewer. View any log messages in the message box, as shown in Figure 3–32.

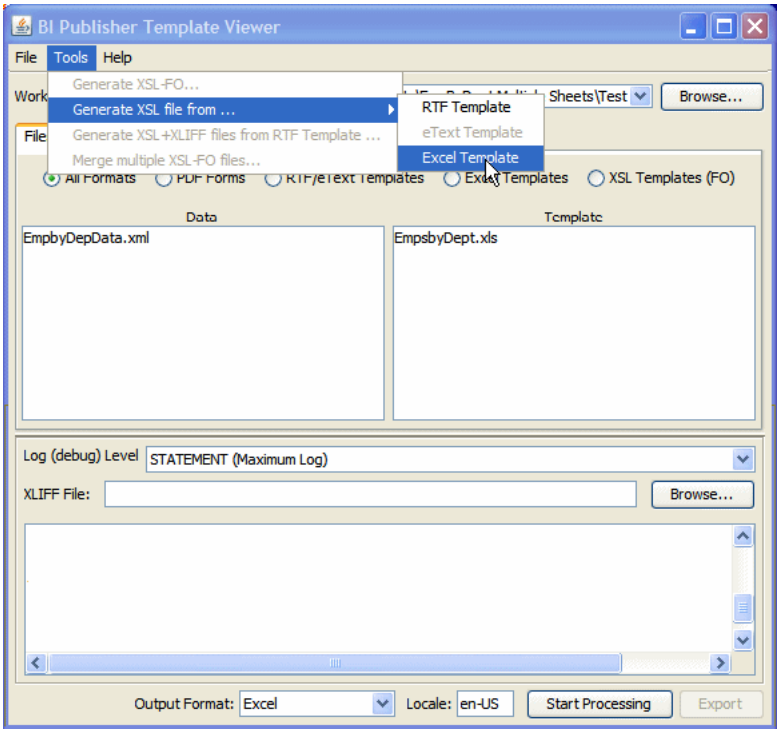
Figure 3–32 Log Messages



To view the generated XSL:

1. In the Template Viewer, select the data and template files and choose Excel output.
2. On the **Tools** menu, select **Generate XSL file from** and then choose **Excel Template**, as shown in Figure 3–33.

Figure 3–33 The Excel Template Option



3. At the prompt, save the generated XSL file.
4. Navigate to the saved location and open the XSL file in an appropriate viewer.