

**Oracle® Fusion Middleware**

Report Designer's Guide for Oracle Business Intelligence Publisher

Release 11g (11.1.1)

**Part No. E13881-02**

April 2011

Oracle Fusion Middleware Report Designer's Guide for Oracle Business Intelligence Publisher, Release 11g (11.1.1)

Part No. E13881-02

Copyright © 2010, 2011, Oracle and/or its affiliates. All rights reserved.

Primary Author: Leslie Grumbach Studdard

Contributing Author: Tim Dexter

Contributor: Oracle Business Intelligence Publisher development, product management, and quality assurance teams

Oracle is a registered trademark of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this software or related documentation is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, the following notice is applicable:

#### U.S. GOVERNMENT RIGHTS

Programs, software, databases, and related documentation and technical data delivered to U.S. Government customers are "commercial computer software" or "commercial technical data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, duplication, disclosure, modification, and adaptation shall be subject to the restrictions and license terms set forth in the applicable Government contract, and, to the extent applicable by the terms of the Government contract, the additional rights set forth in FAR 52.227-19, Commercial Computer Software License (December 2007). Oracle USA, Inc., 500 Oracle Parkway, Redwood City, CA 94065.

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

This software and documentation may provide access to or information on content, products and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third party content, products and services. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third party content, products or services.

---

# Contents

**Send Us Your Comments**

**Preface**

## **Part 1 Introduction**

### **1 Introduction**

Overview of the Oracle Business Intelligence Publisher Report Designer's Guide.....	1-1
New Features for Oracle Business Intelligence Publisher 11g Release 1 (11.1.1) .....	1-4

## **Part 2 Creating Reports and Layouts**

### **2 Creating or Editing a Report**

About Report Components.....	2-1
Process Overview.....	2-2
Launching the Report Editor.....	2-3
Selecting the Data Model.....	2-3
About the Report Editor Interface.....	2-4
Adding Layouts to the Report Definition.....	2-5
Configuring Layouts.....	2-7
Configuring Parameter Settings for the Report.....	2-10
Configuring Report Properties.....	2-11

### **3 Creating BI Publisher Layout Templates**

Introduction.....	3-1
-------------------	-----

Launching the Layout Editor.....	3-3
About the Layout Editor Interface.....	3-6
The Page Layout Tab.....	3-12
Inserting Layout Components.....	3-22
About Layout Grids.....	3-22
About Repeating Sections.....	3-26
About Data Tables .....	3-31
About Charts.....	3-52
About Gauge Charts.....	3-58
About Pivot Tables.....	3-61
About Text Items.....	3-68
About Images.....	3-73
About Lists.....	3-74
Setting Predefined or Custom Formulas.....	3-80
Saving a Layout.....	3-90

## 4 Creating RTF Templates

What Is an RTF Template?.....	4-2
About XSLT Compatibility.....	4-2
Getting Started.....	4-2
Prerequisites.....	4-3
About Adding BI Publisher Code.....	4-3
Key Concepts.....	4-4
Associating the XML Data to the Template Layout.....	4-4
Designing the Template Layout.....	4-7
Adding Markup to the Template Layout.....	4-7
Creating Placeholders.....	4-8
Defining Groups.....	4-12
Defining Headers and Footers.....	4-16
Native Support.....	4-16
Inserting Images and Charts.....	4-18
Images.....	4-18
Chart Support.....	4-19
Drawing, Shape, and Clip Art Support.....	4-30
Supported Native Formatting Features.....	4-41
General Features.....	4-41
Alignment.....	4-42
Tables.....	4-42
Date Fields.....	4-45
Multicolumn Page Support.....	4-46

Background and Watermark Support.....	4-47
<b>Template Features</b> .....	4-49
Page Breaks.....	4-49
Initial Page Number.....	4-50
Last Page Only Content .....	4-51
End on Even or End on Odd Page.....	4-54
Hyperlinks.....	4-55
Table of Contents.....	4-58
Generating Bookmarks in PDF Output.....	4-59
Check Boxes.....	4-60
Drop Down Lists.....	4-61
<b>Conditional Formatting</b> .....	4-64
If Statements.....	4-65
If Statements in Boilerplate Text.....	4-65
If-then-Else Statements.....	4-66
Choose Statements.....	4-67
Column Formatting.....	4-69
Row Formatting.....	4-71
Cell Highlighting.....	4-73
<b>Page-Level Calculations</b> .....	4-75
Displaying Page Totals.....	4-76
Brought Forward/Carried Forward Totals.....	4-78
Running Totals.....	4-82
<b>Data Handling</b> .....	4-84
Sorting.....	4-84
Checking for Nulls.....	4-85
Regrouping the XML Data.....	4-85
<b>Using Variables</b> .....	4-92
<b>Defining Parameters</b> .....	4-93
<b>Setting Properties</b> .....	4-95
<b>Advanced Report Layouts</b> .....	4-97
Batch Reports.....	4-97
Handling No Data Found Conditions.....	4-99
Pivot Table Support.....	4-100
Dynamic Data Columns.....	4-103
<b>Number, Date, and Currency Formatting</b> .....	4-106
<b>Calendar and Timezone Support</b> .....	4-123
<b>Using External Fonts</b> .....	4-125
Custom Barcode Formatting.....	4-128
<b>Controlling the Placement of Instructions Using the Context Commands</b> .....	4-129
<b>Using XPath Commands</b> .....	4-132

Namespace Support.....	4-136
Using XSL Elements.....	4-136
Using FO Elements.....	4-138
Guidelines for Designing RTF Templates for Microsoft PowerPoint Output.....	4-138

## 5 Creating RTF Templates Using the Template Builder for Word

Introduction.....	5-1
Getting Started.....	5-3
Accessing Data for Building Your Template.....	5-7
Inserting Components to the Template.....	5-9
Previewing a Template.....	5-31
Template Editing Tools.....	5-32
Uploading a Template to the BI Publisher Server.....	5-37
Using the Template Builder Translation Tools.....	5-37
Setting Options for the Template Builder.....	5-39
Setting Up a Configuration File.....	5-44
BI Publisher Menu Reference.....	5-44

## 6 Creating Excel Templates

Introduction.....	6-1
Concepts.....	6-3
Building a Simple Template.....	6-4
Formatting Dates.....	6-14
Defining BI Publisher Functions .....	6-18
Reporting Functions.....	6-18
Formatting Functions That Rely on Specific Data Attribute Values.....	6-29
Grouping Functions.....	6-39
Preprocessing the Data Using an XSL Transformation (XSLT) File.....	6-41
Using the Template Viewer to Debug a Template.....	6-42

## 7 Creating PDF Templates

Overview.....	7-1
Requirements.....	7-1
Designing the Template.....	7-2
Adding Markup to the Template.....	7-4
Creating a Placeholder.....	7-5
Defining Groups of Repeating Fields.....	7-8
Adding Page Numbers.....	7-9
Performing Calculations.....	7-13
Completed PDF Layout Example.....	7-14

Runtime Behavior.....	7-15
Creating a Layout from a Predefined PDF Form.....	7-17
Adding or Designating a Field for a Digital Signature.....	7-19
<b>8 Creating Flash Templates</b>	
Introduction.....	8-1
Building a Flash Template.....	8-3
Uploading the Flash Template to the Report Definition.....	8-13
Setting Properties for PDF Output.....	8-13
For More Information.....	8-15
<b>9 Creating eText Templates</b>	
Introduction.....	9-1
Structure of eText Templates.....	9-2
Constructing the Data Tables.....	9-6
Command Rows.....	9-6
Structure of the Data Rows.....	9-12
Setup Command Tables.....	9-16
Creating a Filler Block.....	9-29
Expressions, Control Structures, and Functions.....	9-31
Identifiers, Operators, and Literals.....	9-35
<b>10 Setting Report Processing and Output Document Properties</b>	
Introduction.....	10-1
PDF Output Properties.....	10-2
PDF Security Properties.....	10-4
PDF Digital Signature Properties.....	10-7
RTF Output Properties.....	10-9
HTML Output Properties.....	10-10
FO Processing Properties.....	10-11
RTF Template Properties.....	10-14
PDF Template Properties.....	10-15
Flash Template Properties.....	10-16
CSV Output Properties.....	10-17
Excel 2007 Output Properties.....	10-18
All Outputs.....	10-18
Defining Font Mappings.....	10-19

## **Part 3 Creating and Implementing Style Templates**

### **11 Creating and Implementing Style Templates**

Understanding Style Templates.....	11-1
Creating a Style Template RTF File.....	11-3
Uploading a Style Template File to the Catalog.....	11-5
Assigning a Style Template to a Report Layout.....	11-6
Updating a Style Template.....	11-7
Adding Translations to Your Style Template Definition.....	11-7

## **Part 4 Creating and Implementing Sub Templates**

### **12 Understanding Subtemplates**

What is a Subtemplate.....	12-1
Supported Locations for Subtemplates.....	12-2
Designing a Subtemplate.....	12-2
Testing Subtemplates from the Desktop.....	12-2
Creating the Sub Template Object in the Catalog.....	12-3
Calling a Subtemplate from an External Source.....	12-4

### **13 Designing RTF Subtemplates**

Understanding RTF Subtemplates.....	13-1
Process Overview for Creating and Implementing RTF Subtemplates.....	13-2
Creating an RTF Subtemplate File.....	13-2
Calling a Subtemplate from Your Main Template.....	13-3
When to Use RTF Subtemplates.....	13-5
Adding Translations to an RTF Subtemplate.....	13-9

### **14 Designing XSL Subtemplates**

Understanding XSL Subtemplates.....	14-1
Process Overview for Creating and Implementing XSL Subtemplates.....	14-2
Creating an XSL Subtemplate File.....	14-2
Calling an XSL Subtemplate from Your Main Template.....	14-3
Creating the Sub Template Object in the Catalog.....	14-5
Example Uses of XSL Subtemplates.....	14-6



## Part 5 Translating Reports and Catalog Objects

### 15 Translation Support Overview and Concepts

Translation Support Overview.....	15-1
Working with Translation Files.....	15-2
What Is an XLIFF?.....	15-2
Structure of the XLIFF File.....	15-2
Locale Selection Logic.....	15-5

### 16 Translating Individual Templates

Overview.....	16-1
Types of Translations.....	16-1
Using the XLIFF Option.....	16-3
Using the Localized Template Option .....	16-6

### 17 Translating Catalog Objects, Data Models, and Templates

Overview .....	17-1
What Can Be Translated.....	17-2
Exporting the XLIFF File.....	17-3
Identifying and Updating the Object Tags.....	17-3
Importing the XLIFF File.....	17-3

### A Techniques for Handling Large Output Files

Techniques for Handling Large PDF Output Files.....	A-1
Reusing Static Content.....	A-1
Generating Zipped PDF Output.....	A-4
Implementing PDF Splitting for an RTF Template.....	A-6
Implementing PDF Splitting for a PDF Template.....	A-7

### B Extended Function Support in RTF Templates

Extended SQL and XSL Functions.....	B-1
XSL Equivalents.....	B-18
Using FO Elements.....	B-19

### C Designing Accessible Reports

Introduction.....	C-1
Avoiding Nested Tables or Separated Tables.....	C-1

Defining a Document Title.....	C-3
Defining Alternative Text for an Image.....	C-3
Defining a Table Summary.....	C-4
Defining a Table Column Header.....	C-4
Defining a Table Row Header.....	C-4
Sample Supported Tables.....	C-5

## **D Supported XSL-FO Elements**

Supported XSL-FO Elements.....	D-1
--------------------------------	-----

## **Index**

---

# Send Us Your Comments

## **Oracle Fusion Middleware Report Designer's Guide for Oracle Business Intelligence Publisher, Release 11g (11.1.1)**

**Part No. E13881-02**

Oracle welcomes customers' comments and suggestions on the quality and usefulness of this document. Your feedback is important, and helps us to best meet your needs as a user of our products. For example:

- Are the implementation steps correct and complete?
- Did you understand the context of the procedures?
- Did you find any errors in the information?
- Does the structure of the information help you with your tasks?
- Do you need different information or graphics? If so, where, and in what format?
- Are the examples correct? Do you need more examples?

If you find any errors or have any other suggestions for improvement, then please tell us your name, the name of the company who has licensed our products, the title and part number of the documentation and the chapter, section, and page number (if available).

Note: Before sending us your comments, you might like to check that you have the latest version of the document and if any concerns are already addressed. To do this, access the new Oracle E-Business Suite Release Online Documentation CD available on My Oracle Support and [www.oracle.com](http://www.oracle.com). It contains the most current Documentation Library plus all documents revised or released recently.

Send your comments to us using the electronic mail address: [appsdoc\\_us@oracle.com](mailto:appsdoc_us@oracle.com)

Please give your name, address, electronic mail address, and telephone number (optional).

If you need assistance with Oracle software, then please contact your support representative or Oracle Support Services.

If you require training or instruction in using Oracle software, then please contact your Oracle local office and inquire about our Oracle University offerings. A list of Oracle offices is available on our Web site at [www.oracle.com](http://www.oracle.com).



---

# Preface

## Intended Audience

Welcome to Release 11g (11.1.1) of the *Oracle Fusion Middleware Report Designer's Guide for Oracle Business Intelligence Publisher*.

This book is intended for the following users:

- report designers who will be designing report layouts and creating reports for BI Publisher

See Related Information Sources on page xiv for more Oracle E-Business Suite product information.

## Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at <http://www.oracle.com/us/corporate/accessibility/>

## Structure

- 1 Introduction
- 2 Creating or Editing a Report
- 3 Creating BI Publisher Layout Templates
- 4 Creating RTF Templates
- 5 Creating RTF Templates Using the Template Builder for Word
- 6 Creating Excel Templates
- 7 Creating PDF Templates
- 8 Creating Flash Templates
- 9 Creating eText Templates
- 10 Setting Report Processing and Output Document Properties
- 11 Creating and Implementing Style Templates
- 12 Understanding Subtemplates
- 13 Designing RTF Subtemplates

- 14 Designing XSL Subtemplates**
- 15 Translation Support Overview and Concepts**
- 16 Translating Individual Templates**
- 17 Translating Catalog Objects, Data Models, and Templates**
- A Techniques for Handling Large Output Files**
- B Extended Function Support in RTF Templates**
- C Designing Accessible Reports**
- D Supported XSL-FO Elements**

## Related Information Sources

For more information, see the following documents in the Oracle Business Intelligence Enterprise Edition 11g Release 1 (11.1.1) documentation set:

- The Oracle Business Intelligence chapter in the *Oracle Fusion Middleware Release Notes* for your platform
- *Oracle Fusion Middleware Data Modeling Guide for Oracle Business Intelligence Publisher*
- *Oracle Fusion Middleware User's Guide for Oracle Business Intelligence Publisher*
- *Oracle Fusion Middleware Administrator's Guide for Oracle Business Intelligence Publisher*
- *Oracle Fusion Middleware Developer's Guide for Oracle Business Intelligence Publisher*
- *Oracle Fusion Middleware Upgrade Guide for Oracle Business Intelligence Enterprise Edition*
- *Oracle Fusion Middleware User's Guide for Oracle Business Intelligence Enterprise Edition*  
*Oracle Fusion Middleware Installation Guide for Oracle Business Intelligence*
- *Oracle Fusion Middleware System Administrator's Guide for Oracle Business Intelligence Enterprise Edition*
- *Oracle Fusion Middleware Metadata Repository Builder's Guide for Oracle Business Intelligence Enterprise Edition*

## System Requirements and Certification

Refer to the system requirements and certification documentation for information about hardware and software requirements, platforms, databases, and other information. Both of these documents are available on Oracle Technology Network (OTN).

The system requirements document covers information such as hardware and software requirements, minimum disk space and memory requirements, and required system libraries, packages, or patches:

[http://www.oracle.com/technology/software/products/ias/files/fusion\\_requirements.htm](http://www.oracle.com/technology/software/products/ias/files/fusion_requirements.htm)

The certification document covers supported installation types, platforms, operating systems, databases, JDKs, and third-party products:

[http://www.oracle.com/technology/software/products/ias/files/fusion\\_certification.html](http://www.oracle.com/technology/software/products/ias/files/fusion_certification.html)

## **Do Not Use Database Tools to Modify Oracle E-Business Suite Data**

Oracle **STRONGLY RECOMMENDS** that you never use SQL\*Plus, Oracle Data Browser, database triggers, or any other tool to modify Oracle E-Business Suite data unless otherwise instructed.

Oracle provides powerful tools you can use to create, store, change, retrieve, and maintain information in an Oracle database. But if you use Oracle tools such as SQL\*Plus to modify Oracle E-Business Suite data, you risk destroying the integrity of your data and you lose the ability to audit changes to your data.

Because Oracle E-Business Suite tables are interrelated, any change you make using an Oracle E-Business Suite form can update many tables at once. But when you modify Oracle E-Business Suite data using anything other than Oracle E-Business Suite, you may change a row in one table without making corresponding changes in related tables. If your tables get out of synchronization with each other, you risk retrieving erroneous information and you risk unpredictable results throughout Oracle E-Business Suite.

When you use Oracle E-Business Suite to modify your data, Oracle E-Business Suite automatically checks that your changes are valid. Oracle E-Business Suite also keeps track of who changes information. If you enter information into database tables using database tools, you may store invalid information. You also lose the ability to track who has changed your information because SQL\*Plus and other database tools do not keep a record of changes.





# Part 1

---

## Introduction



---

# Introduction

## Overview of the Oracle Business Intelligence Publisher Report Designer's Guide

Oracle Business Intelligence Publisher is a reporting and publishing application that enables you to extract data from multiple data sources, create layouts for report data, and publish the report to numerous output formats. BI Publisher also enables you to schedule reports and deliver the reports to multiple delivery destinations required by your business.

The *Oracle Fusion Middleware Report Designer's Guide for Oracle Business Intelligence Publisher* describes how to create report layouts and use BI Publisher's report editor to assemble the components of a report. See these other guides for more information about using the product for other business roles:

---

Role	Sample Tasks	Guide
Data Model developer	Fetching and structuring the data to use in reports	<i>Oracle Fusion Middleware Data Modeling Guide for Oracle Business Intelligence Publisher</i>
Administrator	Configuring Security Configuring System Settings Diagnosing and Monitoring System Processes	<i>Oracle Fusion Middleware Administrator's Guide for Oracle Business Intelligence Publisher</i>
Application developer or integrator	Integrating BI Publisher into existing applications using the application programming interfaces	<i>Oracle Fusion Middleware Developer's Guide for Oracle Business Intelligence Publisher</i>

---

<b>Role</b>	<b>Sample Tasks</b>	<b>Guide</b>
Report consumer	Viewing reports Scheduling report jobs Managing report jobs	<i>Oracle Fusion Middleware User's Guide for Oracle Business Intelligence Publisher</i>

## Overview for Report Designers

A report consists of a data model, a layout, and a set of properties. Optionally, a report may also include a style template and a set of translations. A report designer performs the following tasks:

- Design the layout for the report. The layout can be created using a variety of tools. The output and design requirements of a particular report will determine the best layout design tool. Options include the Layout Editor, which is a Web-based layout design tool and enables interactive output, Microsoft Word, Adobe Acrobat, Microsoft Excel, and Adobe Flexbuilder.
- Set runtime configuration properties for the report.
- Design style templates to enhance a consistent look and feel of reports in your enterprise.
- Create sub templates to re-use common functionality across multiple templates
- Enable translations for a report.

## About the Template Types

BI Publisher offers several options for designing layouts for your reports. The following formats are supported. Note that the layout type determines the types of output documents supported.

- BI Publisher layout (XPT)  
BI Publisher's Layout Editor is a Web-based design tool for creating layouts. Layouts created with the Layout Editor support interactive viewing as well as the full range of output types supported by RTF layouts.
- Rich Text Format (RTF)  
BI Publisher provides a plugin utility for Microsoft Word that automates layout design and enables you to connect to BI Publisher to access data and upload templates directly from a Microsoft Word session. The RTF format also supports advanced formatting commands providing the most flexible and powerful of the

layout options. RTF templates support a variety of output types, including: PDF, HTML, RTF, Excel, PowerPoint, zipped PDF, and MHTML.

- **Portable Document Format (PDF)**

PDF templates are used primarily when you need to use a predefined form as a layout for a report (for example, a form provided by a government agency). Because many PDF forms already contain form fields, using the PDF form as a template simply requires mapping data elements to the fields that exist on the form. You can also design your own PDF templates using Adobe Acrobat Professional. PDF templates only support PDF output.

- **Microsoft Excel (XLS)**

Excel templates enable you to map data and define calculations and formatting logic in your Excel workbook. Excel templates support Microsoft Excel (.xls) output only.

If you only need to view report data in Excel, you can also use BI Publisher's Analyzer for Microsoft Excel to download report data to an Excel worksheet. Create a layout for the data in Excel and then upload the spreadsheet back to BI Publisher to save as a report layout.

- **XSL Stylesheet**

Layouts can also be defined directly in XSL formatting language. Specify whether your layout is for FO, HTML, XML, or Text transformation.

- **eText**

These are specialized RTF templates used for creating text output for Electronic Data Interchange (EDI) or Electronic Funds Transfer (EFT) transactions.

- **Flash**

BI Publisher's support for Flash layouts enables you to develop Adobe Flex templates that can be applied to BI Publisher reports to generate interactive Flash output documents.

## **About Setting Runtime Properties**

BI Publisher provides a variety of user-controlled settings that are specified via an easily accessible Runtime Configuration page. These include security settings for individual PDF reports, HTML output display settings, font mapping, currency formatting, and other output-specific settings. For more information see *Setting Report Processing and Output Document Properties*, page 10-1. These settings are also configured at the system-level, but can be customized per report.

## About Translations

BI Publisher provides the ability to create an XLIFF file from your RTF templates. XLIFF is the XML Localization Interchange File Format. It is the standard format used by localization providers. Using BI Publisher's XLIFF generation tool you can generate the standard translation file of your RTF template. You can then translate this file (or send to a translation provider). Once translated, the file can be uploaded to the report definition under the appropriate locale setting so that at runtime the translated report will automatically be run for users selecting the corresponding locale. For more information, see *Translating Reports and Catalog Objects*, page 15-1.

## About Style Templates

A style template is an RTF template that contains style information that can be applied to layout templates. The style information in the style template is applied to report layout templates at runtime to achieve a consistent look and feel across your enterprise reports.

For more information, see *Creating and Implementing Style Templates*, page 11-1.

## About Sub Templates

A Sub Template is a piece of formatting functionality that can be defined once and used multiple times within a single layout template or across multiple layout template files. This piece of formatting can be in an RTF file format or an XSL file format. RTF subtemplates are easy to design as you can use Microsoft Word native features. XSL subtemplates can be used for complex layout and data requirements.

For more information, see *Creating and Implementing Sub Templates*, page 12-1.

## New Features for Oracle Business Intelligence Publisher 11g Release 1 (11.1.1)

This section describes new features in Oracle Business Intelligence Publisher 11g Release 1 (11.1.1). If you are upgrading from a previous release, read the following for information about new features, tools, and procedures.

This section includes the following topics:

- New Features for Oracle BI Publisher 11g Release 1 (11.1.1.5)
- New Features for Oracle BI Publisher 11g Release 1 (11.1.1.3)

### New Features for Oracle BI Publisher 11g Release 1 (11.1.1.5)

New features in Oracle BI Publisher 11g Release 1 (11.1.1.5) include:

## Excel Templates

An Excel template is a report layout that you design in Microsoft Excel for retrieving your enterprise data in Excel. Excel templates provide a set of special features for controlling the display of data and providing specific formatting instructions. Excel templates support the following features:

- Define the structure for your 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 your data to manipulate it prior to rendering
- Use native Excel functionality

For more information, see *Creating Excel Templates*, page 6-1.

## Excel Template Builder

BI Publisher provides a downloadable add-in to Excel that enables you to preview your Excel template with sample data. This facilitates design by enabling you to test and edit your template without having to upload it to the BI Publisher catalog first. The Template Builder for Excel is installed automatically when you install the Template Builder for Word. The tools can be downloaded from the Home page of Oracle Business Intelligence Publisher or Oracle Business Intelligence Enterprise Edition as follows:

Under the **Get Started** region, click **Download BI Publisher Tools**.

For more information, see *Creating Excel Templates*, page 6-1

## New Interactive List Component for BI Publisher Layouts

The list component displays all values of a data element in a vertical or horizontal list. When viewed in interactive mode, clicking an item in the list updates the results shown in the linked components of the report. The list enables the report consumer to quickly see results for each item in the list by clicking the list entry. For more information see *About Lists*, page 3-74.

## Redesigned Formula Dialog for BI Publisher Layout Editor

The formula dialog in the BI Publisher layout editor has been redesigned for better usability. For more information, see *Setting Predefined or Custom Formulas*, page 3-80.

## New Features for Oracle BI Publisher 11g Release 1 (11.1.1.3)

New features in Oracle BI Publisher 11g Release 1 (11.1.1.3) include:

## Major User Interface Improvements

The user interface has undergone major improvements in several areas, including a new Home page and redesigned editors and panes. These improvements are intended to make working with Oracle BI Publisher easier and more consistent. This guide provides detailed information on working with the various pieces of the user interface.

## Shared Oracle BI Presentation Catalog

For installations of BI Publisher with the Oracle BI Enterprise Edition, BI Publisher now shares the same catalog with Oracle BI Presentation services.

## Layout Editor Design Tool

This release introduces a new type of layout template and design tool. The Layout Editor is launched from within BI Publisher and provides an intuitive WYSIWIG drag-and-drop interface for designing report layouts.

## Interactive Viewer

For reports created with the new BI Publisher layout editor, a new interactive output type is available. The interactive viewer enables pop-up chart details, scrollable tables, table filtering, table sorting, and propagated filtering across different components of the report. This interactivity is achieved simply by designing the report in the layout editor, no additional coding is necessary.

## Sub Templates

Previously subtemplates had to be stored outside of the BI Publisher catalog and called at runtime from the external directory. In this release your RTF and XSL subtemplates can be saved and managed as objects in the BI Publisher catalog.

## Style Templates

A style template is an RTF template that contains style information that can be applied to RTF layouts. The style information in the style template is applied to RTF layouts at runtime to achieve a consistent look and feel across your enterprise reports. Style templates are saved and managed in the BI Publisher catalog.

## Zippered PDF Output

This release introduces a feature to split a large PDF output file into smaller, more manageable files, while still maintaining the integrity of the report as one logical unit. When PDF output splitting is enabled for a report, the report is split into multiple files generated in one zip file. The output type is PDFZ. For easy access to the component files, BI Publisher also generates an index file that specifies from and to elements contained in each component PDF file.



# Part 2

---

## Creating Reports and Layouts



---

# Creating or Editing a Report

## About Report Components

A report consists of the following components:

- Data Model
- Layout
- Properties
- Translations

The first step in creating a new report is to select the source of the data for the report. A **Data Model** defines data that is used by a report. A Data Model may contain multiple data sets and it defines how data fields are structured in relation to each other. It may also contain parameters with lists of values, bursting definitions and other structures or properties that determine how data is provided to a report. Creating a data model is documented in the *Oracle Fusion Middleware Data Modeling Guide for Oracle Business Intelligence Publisher*.

The next step is to design a **layout** for the report data. The layout defines how the data is presented in the report. A layout consists of a template file and a set of properties for rendering the template file. BI Publisher supports templates created from a variety of sources including Microsoft Word, Adobe Acrobat, Microsoft Excel, Adobe Flash, and BI Publisher's own layout editor. A report can include multiple layouts.

Next, configure **properties** for the report. The report properties enable you to control many aspects of the report generation, formatting, and display.

Optionally, add **translations** for the report. BI Publisher's translation support enables you to include translations for individual layouts or for all translatable strings in the layout, data model, and the report metadata.

This chapter will describe the process of creating a report by selecting a data model, adding a layout, and configuring properties using the report editor. For more

information about report components see the following:

---

<b>Topic</b>	<b>Where to get more information</b>
Creating a data model	Creating Data Models, <i>Oracle Fusion Middleware Data Modeling Guide for Oracle Business Intelligence Publisher</i>
How to choose a template type	About the Template Types, page 1-2
Creating specific template types	Creating BI Publisher Layout Templates, page 3-1 Creating RTF Templates Using the Template Builder for Word, page 5-1 Creating RTF Templates, page 4-2 Creating Excel Templates, page 6-1 Creating PDF Templates, page 7-1 Creating Templates Using the Excel Analyzer, <i>Oracle Fusion Middleware User's Guide for Oracle Business Intelligence Publisher</i> Creating Flash Templates, page 8-1 Creating eText Templates, page 9-1
Translating reports	Translating Reports and Catalog Objects, page 15-1

---

## Process Overview

Creating a new report consists of the following steps. Each of these steps is discussed in detail in this section.

---

<b>Task</b>	<b>Section</b>
Launch the Report Editor.	Launching the Report Editor, page 2-3
Select the data model.	Selecting the Data Model, page 2-3
Use the Layout Editor to create the layout or upload a template file.	Adding Layouts to the Report Definition, page 2-5
Configure the properties for the layout.	Configuring Layouts, page 2-7

---

Task	Section
Configure parameters for the report.	Configuring Parameter Settings for the Report, page 2-10
Configure report properties.	Configuring Report Properties, page 2-11
Add translations for your layouts.	Translation Support Overview and Concepts, page 15-1
Complete this step if your report requires support for multiple languages.	

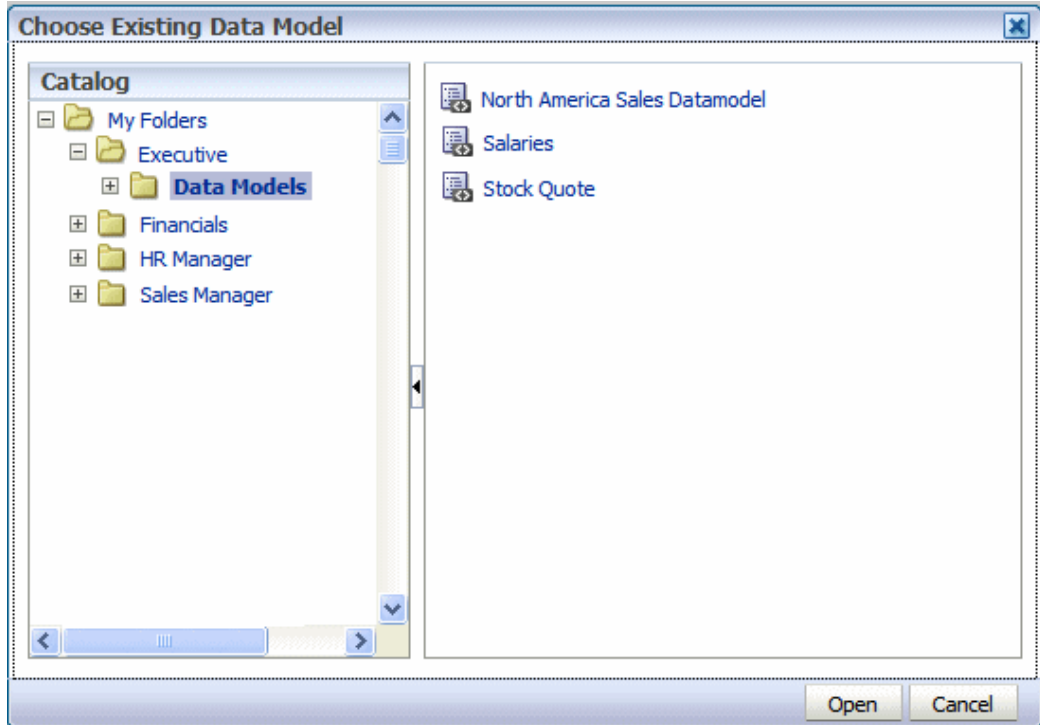
## Launching the Report Editor

Launch the Report Editor in one of the following ways:

- From the global header, click **New** and then click **Report**.
- From the **Home** page, under the **Create** region, click **Report**.

## Selecting the Data Model

After clicking one of the Create New Report selections you are prompted to select an existing data model.



Navigate through the catalog and select the data model for your report.

Click **Open** to add the data model to your report.

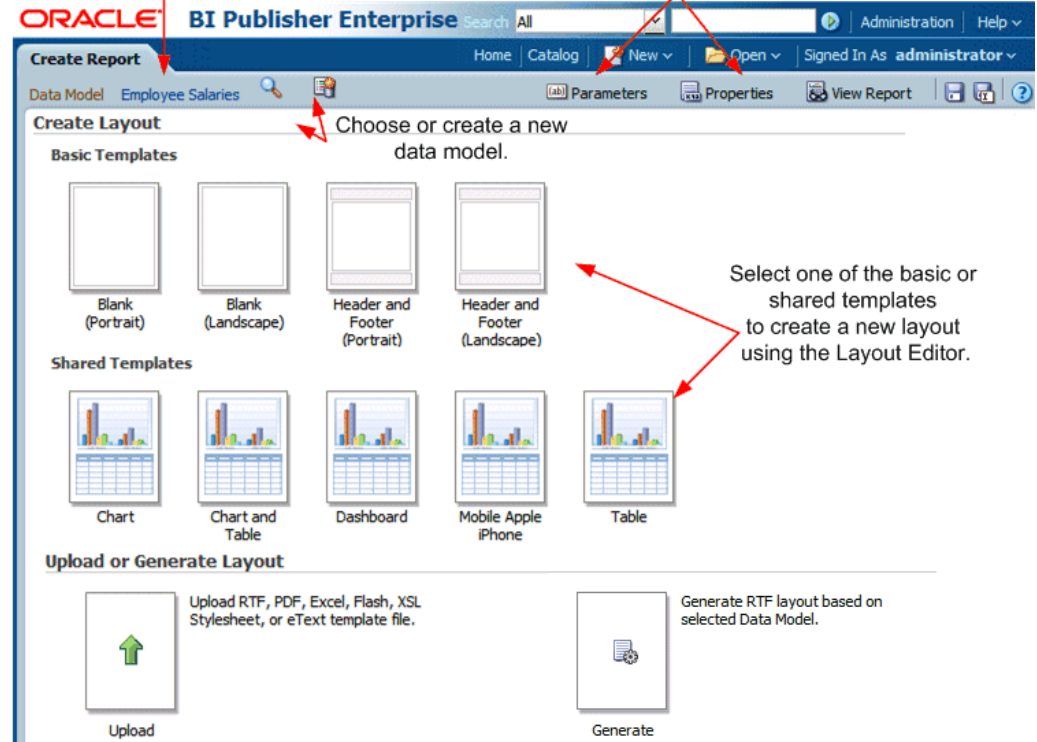
If you have not already created the data model, click **Cancel** to close the **Choose** dialog. From the report editor, click the **Create New Data Model** button (see figure in next section) to launch the Data Model Editor. See *Creating a Data Model, Oracle Fusion Middleware Data Modeling Guide for Oracle Business Intelligence Publisher* for more information on creating data models.

## About the Report Editor Interface

After selecting a data model the Create Report page displays:

The data model is displayed here. Click to open the data model editor.

Choose default parameters and set report properties.



From this point you can:

- add, upload, or generate a layout
- configure the data model parameter defaults for this report
- configure the report properties

This procedure addresses these options in the order listed.

## Adding Layouts to the Report Definition

There are three options for adding a layout to your report:

- **Create Layout** - select one of the basic or shared templates to launch the Layout Editor.
- **Upload Layout** - upload a template file layout that you have designed in one of the supported file types.
- **Generate Layout** - automatically generate a simple RTF layout.

## To Add a Layout Using the Layout Editor

1. Under the **Create Layout** region, click one of the basic or shared templates to launch the Layout Editor.
2. Design the template. See *Creating BI Publisher Layout Templates*, page 3-1 for information on using the Layout Editor.
3. When finished, click **Save**. In the **Save Template** dialog enter a name for this layout and select a locale. Click **Save**.
4. Click **Return** to return to the Report Editor.
5. Configure the settings for the layout. See *Configuring Layouts*, page 2-7.

## To Add a Layout by Uploading a Template File

Note that uploading a template file assumes that you have followed the instructions in this guide for creating a template file (RTF, PDF, Excel, Flash, or eText).

1. Under the **Upload or Generate Layout** region, click the **Upload** icon.
2. In the **Upload** dialog, perform the following:
  - Enter a **Layout Name**.
  - Click **Browse** to locate the **Template File** in your local file system.
  - Select the **Template Type** from the list.
  - Select the **Template Locale** from the list.
  - Click **Upload**.

If you are connected to BI Publisher through the Template Builder or Excel Analyzer, you can upload the layout file directly from either tool. See *Creating RTF Templates Using the Oracle BI Publisher Template Builder for Word*, page 5-1 or *Using the Analyzer for Excel*, *Oracle Fusion Middleware User's Guide for Oracle Business Intelligence Publisher* for more information.

## To Add a Layout by Generating a Template File

1. Under the Upload or Generate Layout region, click the Generate icon.
2. In the **Autogenerate Layout** dialog, perform the following:
  - Enter a **Template Name** for the layout.



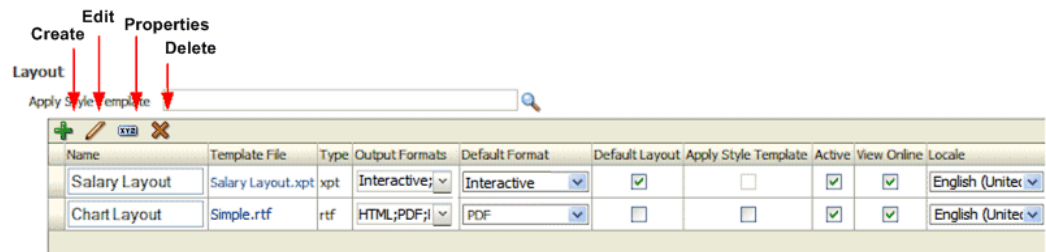
- Click **Generate**.

The autogenerate feature creates a simple table-based RTF layout that includes all the fields in your data model.

**Important:** The autogenerate layout feature can only be used with data sets for which metadata is available. Therefore this feature cannot be used with data sets generated from stored XML files, HTTP feeds, Web services, or migrated data templates.

## Configuring Layouts

After creating or uploading the layouts for your report, you can configure settings for the layout from the **List View**. The **List View** is shown in the following figure:



## Applying a Style Template to the Layout

A style template contains style definitions that are applied to the paragraphs, headings, tables, and headers and footers of a report. A style template is optional and can only be applied to an RTF template file. For more information on creating a style template, see *Creating and Implementing Style Templates*, page 11-1.

If you wish to apply a style template to this layout, click **Choose** to browse for and select the style template. To then apply the style template to an individual layout in the list, select the Apply Style Template box for that layout in the list of properties.

## About the Layouts Toolbar

Use the toolbar buttons to perform the following:

Toolbar Button	Description
Create	Launches the add layout page to upload or create a new layout.

<b>Toolbar Button</b>	<b>Description</b>
<b>Edit</b>	<p>Launches the Layout Editor for the selected layout.</p> <p>This button is enabled for BI Publisher layouts (.xpt) only.</p>
<b>Properties</b>	<p>Launches the Properties page to enable the upload of localized templates and XLIFF files to associate with this layout.</p> <p>This button is enabled for RTF (.rtf) and BI Publisher layouts (.xpt) only.</p> <p>For more information on localizing templates, see Translation Support Overview and Concepts, page 15-1.</p>
<b>Delete</b>	<p>Deletes the selected layout.</p>

## Configuring the Layout Settings

The **List View** enables you to configure the following settings for your layout:

<b>Setting</b>	<b>Description</b>
<b>Name</b>	<p>Place your cursor in the text box to enter a new name for the layout.</p>
<b>Template File</b>	<p>Displays the name of the file that was saved to the report definition. Click the template file name to download it.</p>
<b>Type</b>	<p>Displays the template file type.</p>
<b>Output Formats</b>	<p>Select the output types to be enabled for this layout. By default, all valid output types for a layout are enabled. The layout type determines the output types available.</p> <p>See Valid Output Types for Layout Types, page 2-9 for the complete list.</p>
<b>Default Format</b>	<p>Select the default output format for this layout when viewed or scheduled.</p>
<b>Default Layout</b>	<p>Select the layout that this report will use by default when viewed online or scheduled. Only one box in this column can be checked.</p>
<b>Apply Style Template</b>	<p>Select this box to apply the style template to this layout. Note that a style template can only be applied to RTF template files. For more information, see Applying a Style Template to a Layout, page 2-7.</p>

Setting	Description
<b>Active</b>	By default a layout is active.  Clear this box when you want to keep the layout as part of the report definition, but no longer make it available. When a layout is inactive it will not display in the report viewer or the scheduler.
<b>View Online</b>	By default, a layout is available for report consumers who open the report in the Report Viewer. If this layout is for scheduled reports only, clear this box.
<b>Locale</b>	Displays the locale selected when the layout was uploaded. This field is not updateable.

## Valid Output Types for Layout Types

Layout Type	Valid Output Types
PDF	PDF, PDFZ, CSV, Data
RTF	HTML, PDF, PDFZ, RTF, Excel, Excel2000, PowerPoint, PowerPoint2007, MHTML, CSV, FO, Data
XPT	Interactive, HTML, PDF, PDFZ, RTF, Excel, Excel2000, PowerPoint, PowerPoint2007, MHTML, CSV, FO, Data
XLS	Excel, Excel2000, CSV, Data
Flash	Flash, PDF, MHTML, CSV, Data
XSL Stylesheet (FO)	Same outputs as RTF
XSL Stylesheet (HTML XML/Text)	HTML, XML, Text, Data
eText	Text, CSV, Data

## Editing a Layout

To edit a BI Publisher layout (.xpt file type) select the report from the list and click Edit.

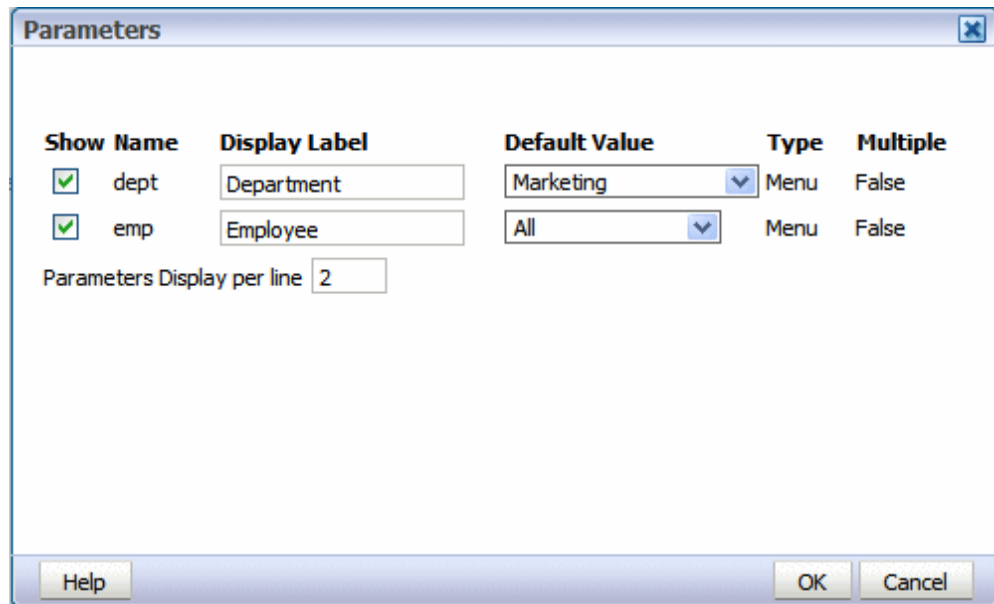
To edit any other template type, click the File name link to download the layout to a local computer for editing.

## Configuring Parameter Settings for the Report

Parameters are defined in the data model, but the report editor enables you to configure the parameter settings specifically for each report that uses the data model.

To configure the parameters for this report:

1. On the Report Editor page, click **Parameters**. The **Parameters** dialog is shown in the following figure:



2. Customize the parameter settings for this report as follows:

Property	Description
<b>Show</b>	This property controls whether the parameter is displayed to the user.  Disable the <b>Show</b> property If you do not want the user to see or change the parameter values that are passed to the data model.
<b>Display Label</b>	This property enables you to reset the display labels shown for each parameter. The default values are defined in the data model.

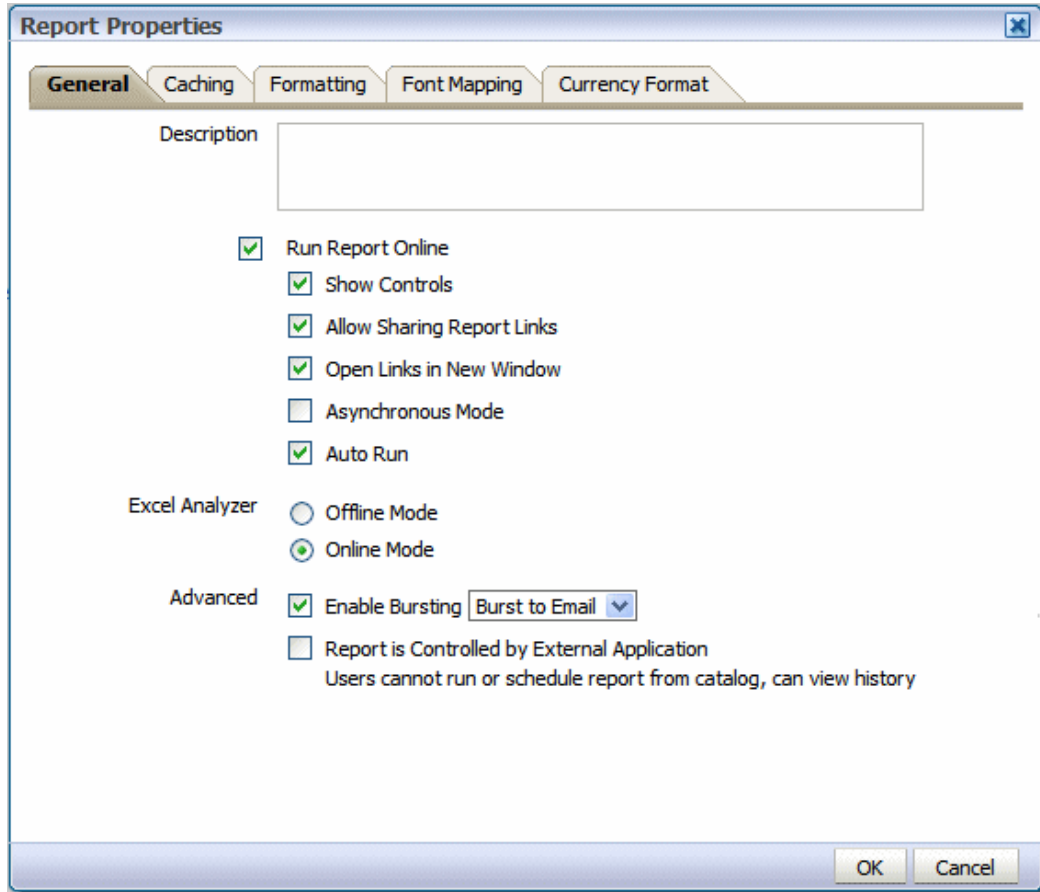
---

<b>Property</b>	<b>Description</b>
<b>Default Value</b>	This property enables you to configure the default value for the parameter specifically for this report. The initial default values are defined in the data model.
<b>Parameters Display per Line</b>	This property controls how many parameters are displayed on one line in the report viewer.

---

## Configuring Report Properties

Launch the **Report Properties** dialog by clicking **Properties** in the page header.



The **Report Properties** dialog has the following option sets:

- **General** - set general properties for your report.
- **Caching** - specify caching options for this report.
- **Formatting** - set the runtime configuration properties for the report. See *Setting Report Processing and Output Document Properties*, page 10-1 for information on setting these properties.
- **Font Mapping** - create font mappings for this report.
- **Currency Format**- define currency formats for this report.

## Setting the General Properties

Set the properties on the General tab as follows:

### Description

(Optional) Enter a description to display with the report in the catalog. This text is

translatable.

## Run Report Online

Disable this property if you do not want users to view this report in the online Report Viewer. When disabled, users will be able to Schedule the report only. For most reports you will keep this enabled. Disable it for long-running, batch, or other reports for which online viewing is not appropriate. When this property is enabled, you can also set the following properties:

---

Property	Description
<b>Show controls</b>	Default: Enabled  This property controls the display of the control region of the report. The Control region consists of the Template list, Output list, and Parameter lists. Disable this property if you do not want users to view and update these options.
<b>Allow Sharing Report Links</b>	Default: Enabled  The <b>Actions</b> menu of the Report Viewer includes the option <b>Share Report Link</b> , which enables users to display the URL for the current report. Disable this property if you do not want users to see and copy the report link.
<b>Open Links in New Window</b>	Default: Enabled  This property controls how links contained within a report are opened. By default links will open in a new browser window. Disable this property to open links in the same browser window.
<b>Asynchronous Mode</b>	Default: Not enabled.  Reports run in asynchronous mode use a unique thread to execute the report when run in the report viewer. This allows BI Publisher to cleanly terminate the thread if a user cancels the report execution. Note that there are performance implications when enabling this property.

---

Property	Description
<b>Auto Run</b>	<p>Default: Enabled</p> <p>When this property is enabled the report will automatically run when the user selects the <b>Open</b> link for the report. When Auto Run is disabled, selecting the <b>Open</b> link for the report displays the online viewer but does not run the report. The user must select an output type from the <b>View Report</b> menu to run the report.</p>

## Excel Analyzer Options

Default: **Online Mode**

This property controls the method by which report data is downloaded to Excel and also impacts the ability to interact with the BI Publisher server from Microsoft Excel. Using **Offline Mode** has the following effects:

- Your report data downloads faster and large data sets are handled more efficiently
- You do not have to enable macros
- You can enable your own custom macros
- You cannot log in or connect to the BI Publisher server from your Microsoft Excel session. Therefore you cannot upload a template directly from Excel, nor can you update the report parameters or apply a new template.

The online mode cannot process data sets that are larger than 5 megabytes. You would therefore enable this property for reports that generate very large data sets that you wish to manipulate in Excel. Note that the Offline Mode also requires the data to be in <ROWSET><ROW>...</ROWS>...</ROWSET> format.

The following table details the differences between the online and offline modes:

Consideration	Offline Mode	Online Mode
Performance	Data is downloaded faster to Excel and large data sets are handled more efficiently	Download is slower and very large data sets can impact the functioning of the Add-in. Online mode cannot process files larger than 5 megabytes.



Consideration	Offline Mode	Online Mode
Macros	You do not have to enable macros to use the Excel Analyzer in this mode. You can also create your own custom macros to use with the Excel Analyzer.	You must enable macros to use the Excel Analyzer in this mode. Custom macros are not supported in this mode.
Connection with BI Publisher	No connection after data is downloaded. You cannot upload templates directly from Excel, change parameters, or apply new templates to the data.	You can connect to the BI Publisher server from your Excel session. You can directly upload templates to the report, update the report parameters, and apply new templates from within your Excel session.
Data set Structure	Data must be in <ROWSET> <ROW> ...</ROWS>.. </ROWSET> format	It is recommended that data is in <ROWSET> <ROW>... </ROWS>.. </ROWSET> format.

For more information about the Analyzer for Excel, see *Using the BI Publisher Analyzer for Excel*, *Oracle Fusion Middleware User's Guide for Oracle Business Intelligence Publisher*.

## Advanced Options

### Enable Bursting

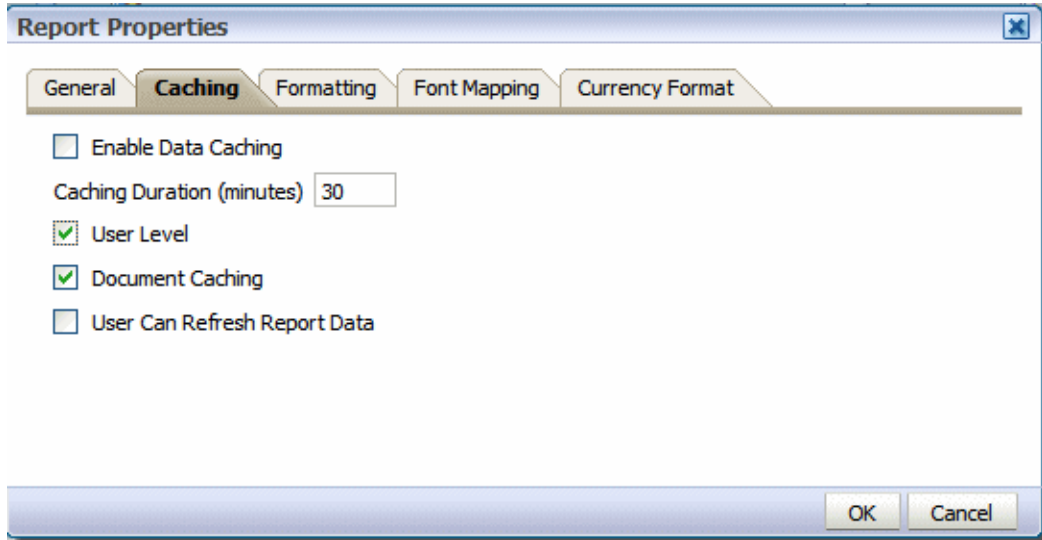
If this report requires bursting, select this box and then select the appropriate bursting definition from the list. When a user schedules the report, he can choose to use the bursting definition to format and deliver the report.

The bursting definition is a component of the data model. For more information, see *Adding Bursting Definitions*, *Oracle Fusion Middleware Data Modeling Guide for Oracle Business Intelligence Publisher*.

### Report is Controlled by External Application. Users cannot run or schedule report from catalog, can view history

If BI Publisher is integrated with another application that controls the running of this report and you do not want users to run and view this report directly from the catalog, enable this option. Reports run by BI Publisher will be stored in the BI Publisher history tables and users can view completed reports from the Report Job History page.

## Setting the Caching Properties



Set the properties on the Caching tab as follows:

### Enable Data Caching

Default: Not enabled

When this property is enabled, the data generated by the online submission of this report will be stored in cache. Subsequent requests to run this report with the same parameter selections will display the report using the data stored in the cache.

This setting enhances performance by using stored data to generate a report rather than regenerating the data from the source. The data will remain in the cache according to the time limit specified in the Cache Duration property.

You can control whether the cache for the report is shared by users by setting the User Level property.

When data caching is enabled, you have the options of setting the following properties:

### Caching Duration (Minutes)

Default: 30 minutes

Enter the time limit for a report data set or document to remain in cache. Once the time limit has expired, the next request for the same report will generate a fresh data set.

### User Level

Default: Enabled

This property stores a separate cache for each user. The report data shown to each user comes only from the private

cache. When enabled, this property ensures that each user can only see data that they are authorized to view. However, user-level cache has less efficient performance. If the report data is not user sensitive, you can disable this property to enhance performance.

#### **Document Caching**

Default: Enabled

Enable this property to cache the report document. With document cache enabled, when a user views the report online, the document (data plus layout) will be placed in cache. When any other user (unless User Level is enabled) uses the online viewer to view the exact same report (same layout, same output type, same parameter selections) the document will be retrieved from cache. The document will remain in cache according to the caching duration specified. Note that scheduled reports do not use document cache.

#### **User Can Refresh Report Data**

Default: Not Enabled

When this property is enabled, the user can choose to refresh the data on demand. When the user clicks **Refresh** in the report viewer, BI Publisher will generate a fresh data set for the report.

## **Setting the Formatting Properties**

The Formatting properties tab enables you to set runtime properties at the report level. These same properties can also be set at the system level, from the Administration page. The Formatting properties tab displays both the system-level setting and the report-level setting for each property. If different values are set at each level, the report level will take precedence.

For a full description of each property, see *Setting Report Processing and Output Document Properties*, page 10-1.

## **Configuring Font Mapping**

BI Publisher's font mapping feature enables you to map base fonts in RTF or PDF templates to target fonts to be used in the published document. Font mappings can be set at the report level or the system level. When you view the report properties Font Mapping tab, any system level settings will be displayed. To change the settings for this report, edit the font mappings here.

For detailed information on font mapping, see *Defining Font Mappings*, page 10-19.

**To create a Font Mapping:**

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

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

For a list of the predefined fonts see BI Publisher's Predefined Fonts, page 10-20.

## Configuring Currency Formats

The Currency Formats tab enables you to map a number format mask to a specific currency so that your reports can display multiple currencies with their own corresponding formatting. Currency formatting is only supported for RTF and XSL-FO templates.

Currency formats can be set at the report level or the system level. When you view the report properties Currency Formats tab, any system level settings will be displayed. To change the settings for this report, edit the currency formats here.

To apply these currency formats in your RTF template, you must use the `format-currency` function. See Currency Formatting, page 4-121 for detailed procedures.

**To add a currency format:**

1. Click the Add icon.
2. Enter the ISO currency code, for example: USD, JPY, EUR, GBP, INR.
3. Enter the format mask to apply for this currency.

The Format Mask must be in the Oracle number format. The Oracle number format uses the components "9", "0", "D", and "G" to compose the format, for example: 9G999D00

where

9 represents a displayed number only if present in data

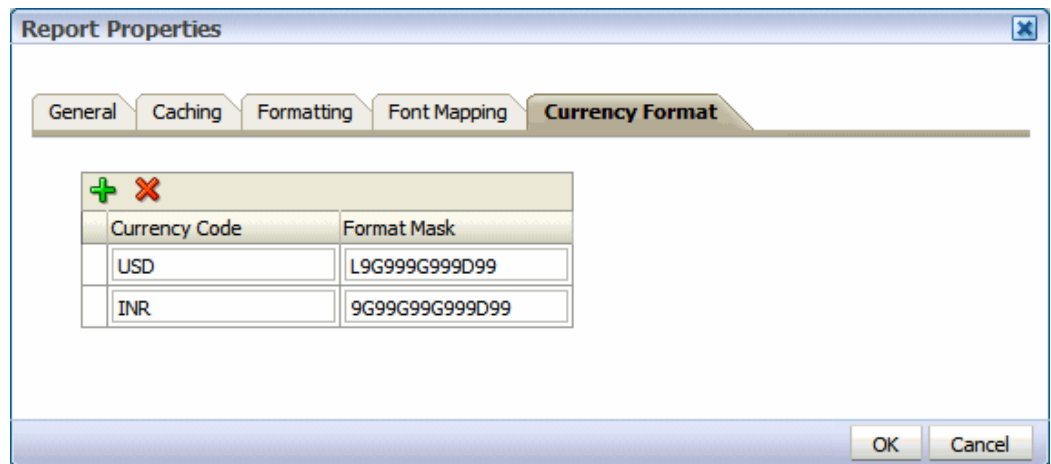
G represents the group separator

D represents the decimal separator

0 represents an explicitly displayed number regardless of incoming data

See Using the Oracle Format Mask, page 4-110 for more information about these format mask components.

The following figure shows sample currency formats:





---

# Creating BI Publisher Layout Templates

This chapter covers the following topics:

- Introduction
- Launching the Layout Editor
- About the Layout Editor Interface
- The Page Layout Tab
- Inserting Layout Components
- About Layout Grids
- About Repeating Sections
- About Data Tables
- About Charts
- About Gauge Charts
- About Pivot Tables
- About Text Items
- About Images
- About Lists
- Setting Predefined or Custom Formulas
- Saving a Layout

## Introduction

Release 11g of Oracle BI Publisher introduces a new type of layout template. The BI Publisher Layout template enables end users to:

- view Dynamic HTML output and perform lightweight interaction with their report data from within a browser

- generate high fidelity, pixel perfect reports to PDF, RTF, Excel, PowerPoint, and static HTML

BI Publisher Layout Templates are created using the BI Publisher Layout Editor - a design tool that provides a WYSIWIG, drag and drop interface for creating pixel perfect reports in PDF, RTF, Excel, PowerPoint, and HTML. It also provides dynamic HTML output that supports lightweight interaction through a browser. This interactive output is featured in the figure below:



**Employees by Department**

Department: Sales

Manager	Name	Title	Hire Date	Annual Salary
	Alberto Errazuriz	Sales M...	Mar 9, 1997	\$144,000.00
	Karen Partners	Sales M...	Jan 4, 1997	\$162,000.00
	John Russell	Sales M...	Sep 30, 1996	\$168,000.00
	Eleni Zlotkey	Sales M...	Jan 28, 2000	\$126,000.00
	Gerald Cambrault	Sales M...	Oct 14, 1999	\$132,000.00
	Oliver Tuvault	Sales R...	Nov 22, 1999	\$84,000.00
	Nanette Cambrault	Sales Representative	Dec 8, 1998	\$90,000.00
	Christopher Olsen	Sales Representative	Mar 29, 1998	\$96,000.00
	Peter Hall	Sales Representative	Aug 19, 1997	\$108,000.00
				\$3,654,000.00

Department: Sales

- All
- Accounting
- Administration
- Executive
- Finance
- Human Resources
- IT
- Marketing
- Public Relations
- Purchasing
- Sales
- Shipping

Sort Ascending  
Sort Descending  
 (Select All)  
 Steven King  
 John Russell  
 Karen Partners  
 Eleni Zlotkey  
 Alberto Errazuriz  
 Gerald Cambrault

OK Cancel

Notice the following features:

- Pop-up chart details – pause your cursor over chart items to display details of data.
- Group filtering – grouped regions can be filtered by the grouping element.



- Scrollable tables – table data can be scrolled while maintaining display of the headers and totals.
- Table column sorting – table data can be sorted by different columns from within the viewer.
- Table column filtering – table data can be filtered by values in different columns from within the viewer.
- Automatic table totaling – table data totals are automatically added to the layout.
- Propagated filtering - filter other components by clicking on chart areas or by clicking on pivot table header, column, or elements
- Collapse and expand areas of the document

## When to Use a BI Publisher Layout

BI Publisher layouts are best suited for reports of simple to medium complexity that do not require custom coding. Because the dynamic HTML view is only available for BI Publisher layouts, BI Publisher layouts must be used when there is a requirement to enable a report consumer to interact with the report (change sorting, apply filters, and so on).

## Prerequisites and Recommendations

- To use the layout editor your account must be granted a role that includes the appropriate permissions for accessing report layout tools.
- You must attach sample data to your data model before you create a new layout. For information on adding sample data to the data model, see *Testing Data Models and Generating Sample Data, Oracle Fusion Middleware Data Modeling Guide for Oracle Business Intelligence Publisher*.
- For optimum viewing, set your display resolution to 1024 x 768 or higher.
- For enabling interactive sorting and filtering the complete data set that is passed to the layout should be limited to several thousand rows to achieve a responsive report. BI Publisher layouts have a comparable performance to RTF layouts for generating static output such as PDF or RTF documents.

## Launching the Layout Editor

Launch the layout editor in one of the following ways:

## When Creating a New Report:

- After selecting the data model for a new report, the report editor displays the **Add Layout** page.

From the **Create Layout** region, click a predefined template to launch the layout editor.

## When Editing a Report:

1. In the Report Editor:

From the **Thumbnail** view, click **Add New Layout**.

or

From the **List** view, click the **Create** button on the layouts table toolbar.

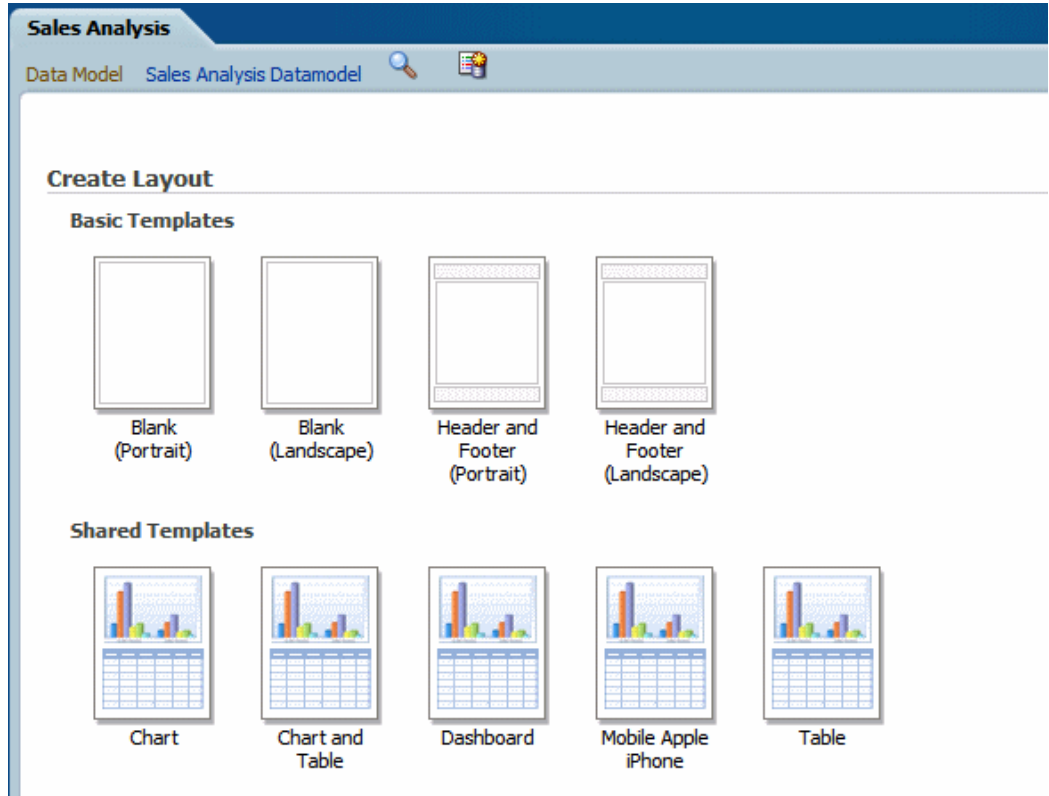
2. From the **Create Layout** region, click a predefined template to use to launch the layout editor.

## When Viewing a Report:

You can also access the Layout Editor when viewing a report. Click **Actions** and then click **Edit Layout**. Note that the layout must have been created in the layout editor.

## Selecting a Predefined Layout

When you creating a new layout, you are given the option of selecting a predefined layout to help you get started.



The Basic and Shared Templates offer common layout structures with specific components already added. Choosing one of the predefined layouts is optional, but can facilitate layout design. If your enterprise utilizes a common design that is not available here, you can add predefined layouts for your own use, or your Administrator can add more for all users.

### Adding Shared Templates for All Users

To add predefined layout files to the shared directory for all users to access:

1. Log in with Administrator privileges and navigate to the Catalog.
2. In the **Shared Folders** directory, open the **Components** folder.
3. Locate the **Boilerplates** report and click **Edit**.
4. Click **Add New Layout**.
5. Design or upload the layout.

To design the layout: Click an existing boilerplate (or blank) to launch the layout editor. Insert the components to the layout. When finished, click **Save** and give your boilerplate a name. This layout will now display to all users in the **Shared Templates** region.

To upload a layout: Click **Upload** to upload a predefined BI Publisher Template (.xpt file).

6. Save the report.

Any BI Publisher Templates (.xpt) added to this report will be displayed to all users as a Shared Template.

### Adding Personal Predefined Layouts

To add predefined layouts that are available to your account user only:

1. Navigate to My Folders.
2. Create a new report called "Boilerplates". This report will not have a data model.
3. Click **Add New Layout**.
4. Design or upload the layout.

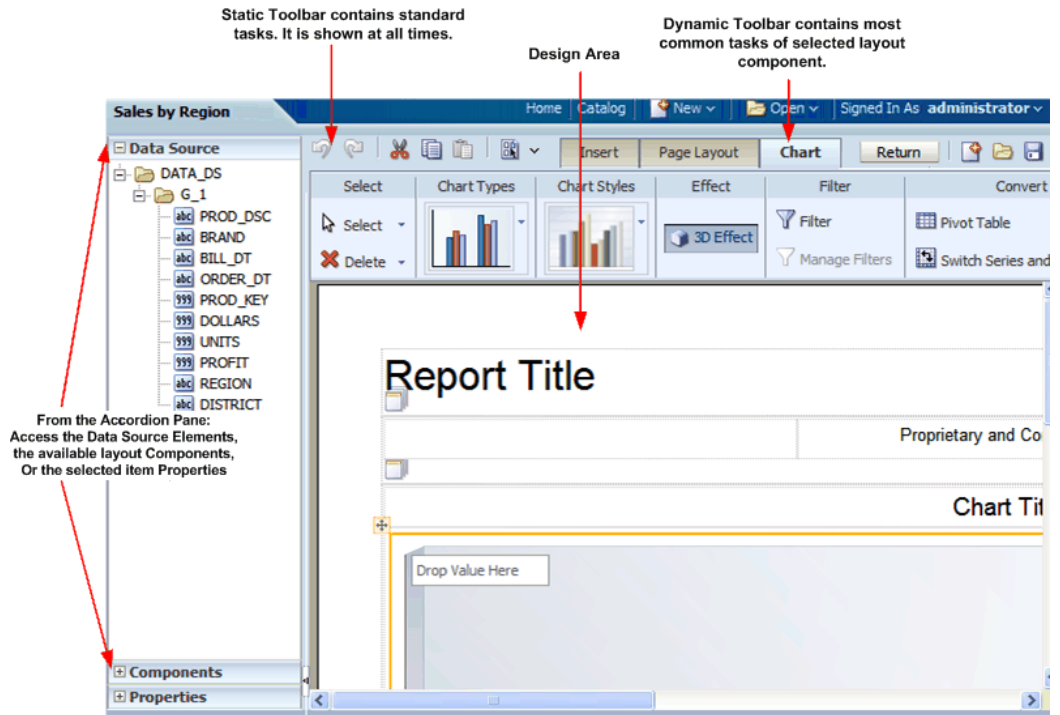
To design the layout: Click an existing boilerplate (or blank) to launch the layout editor. Insert the components to the layout. When finished, click **Save** and give your boilerplate a name.

To upload a layout: Click **Upload** to upload a predefined BI Publisher Template (.xpt file).

These layouts will be presented in the **My Templates** region when you create a new layout.

## About the Layout Editor Interface

The following figure shows the Layout Editor:



The Layout Editor interface comprises the following:

- The top of the Layout Editor contains two toolbars:
  - The **Static toolbar** is always available and contains common commands such as save and preview. See About the Static Toolbar, page 3-10.
  - The **Tabbed toolbar** includes the Insert tab, the Page Layout tab, and a dynamic tab that shows the most commonly used actions and commands for the selected layout component. You can collapse this toolbar to make more room to view your design area. See About the Tabbed Toolbar, page 3-11.
- The accordion pane on the left contains the following:
  - Use the **Data Source** pane to select the data fields to drag to the layout components.
  - Use the **Components** pane to select layout components and drag them to the design area. You can also use the **Insert** tab to insert components when this pane is collapsed.
  - Use the **Properties** pane to modify properties for the selected layout component.

You can expand and display each control by clicking the title of the control or the plus sign next to the title of the control. You can collapse the entire accordion pane

to allow more room to view the layout.

- The lower right region is the design area for building your layout.

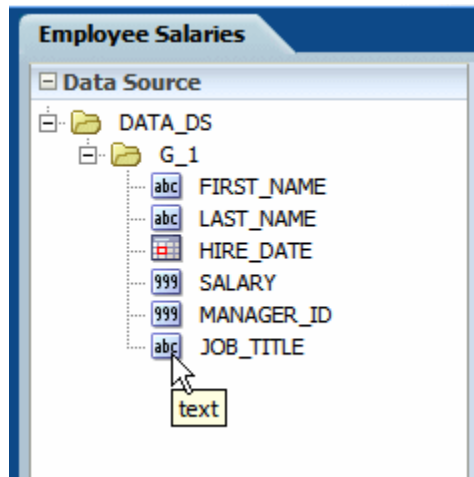
## About the Data Source Pane

The Data Source pane displays the structure of your data model and the data elements that are available to insert into your layout.

To insert a data element, select and drag it from the Data Source pane to the component in the layout.

The data type for each field is represented by an appropriate icon: number, date, or text.

The following figure shows the data source pane. Note that the icon beside each element indicates the data type:



The JOB\_TITLE element is shown as text, the SALARY element is shown as a number, and the HIRE\_DATE element is shown as a date data type.

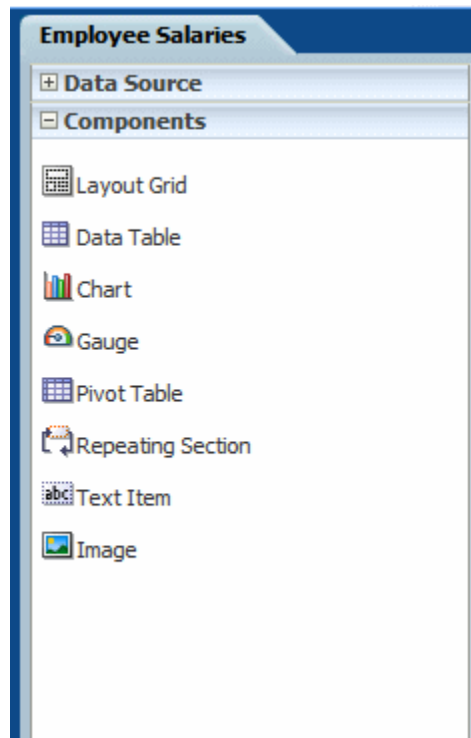
**Note:** When you enter dates in the Layout Editor (such as a data comparison for a filter or for conditional formatting), use one of the following XSL date or time formats: YYYY-MM-DD or YYYY-MM-DDTHH:MM:SS.

## About the Components Pane

The Components pane contains the layout components that you can insert into a report. These components include charts, pivot tables, and images. To insert a component, simply drag and drop it to the layout.

You can also use the Insert menu to add components to your layout.

The following figure shows the Components pane:



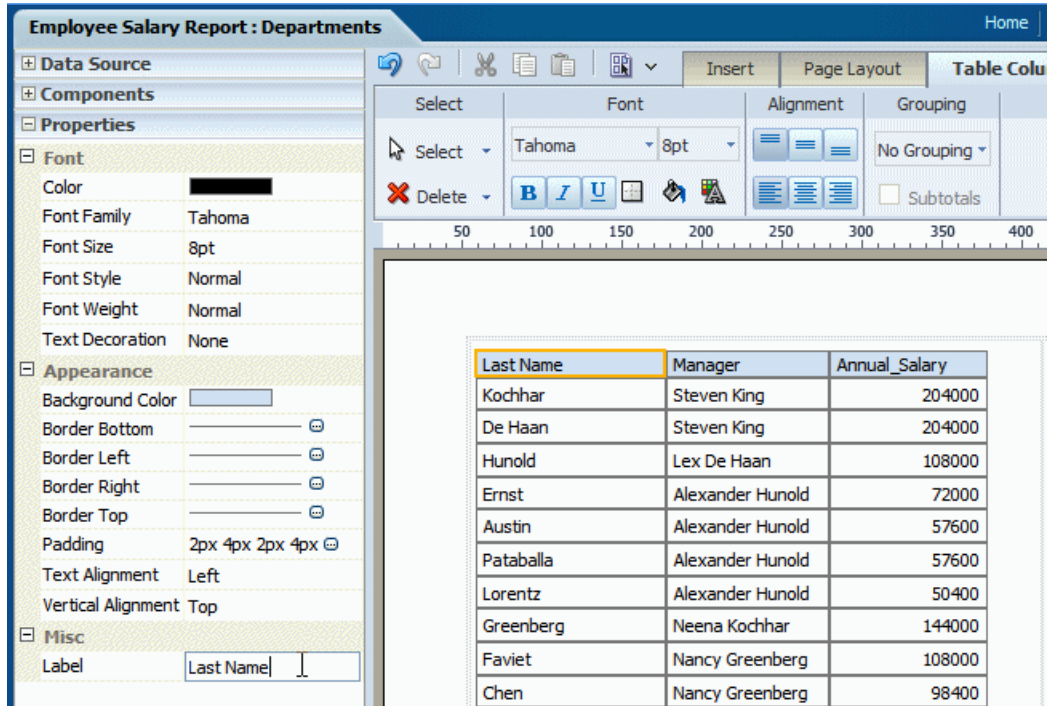
## About the Properties Pane

The Properties pane displays the properties for the selected component. The properties displayed are determined by the selected component. Some of the properties available in the Properties pane are also editable in the dynamic tab for the component.

Click a property value to edit it. The change is applied to the component when you move your cursor out of the field. Collapse or expand a property group by clicking the plus or minus signs beside the group name.

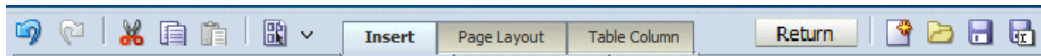
The properties available for each component are discussed in detail in the corresponding section for that component in this chapter. Note that if a property field is blank, the default is used.

The following figure shows a sample Properties pane for a table column header:



## About the Static Toolbar

The Static toolbar extends on either side of the tabbed toolbar and is shown in the following figure:



Use it to perform the following functions:

- Undo and redo operations.
- Cut, copy and paste items.
- Preview as Interactive, HTML, PDF, Microsoft Word (RTF), Microsoft Excel, Microsoft PowerPoint, or Microsoft PowerPoint 2007.
- Return to the previous page.
- Create a new layout.
- Open a layout.
- Save the layout.
- Save the layout as a new copy under a different name.



## About the Tabbed Toolbar

The Tabbed toolbar contains the following tabs:

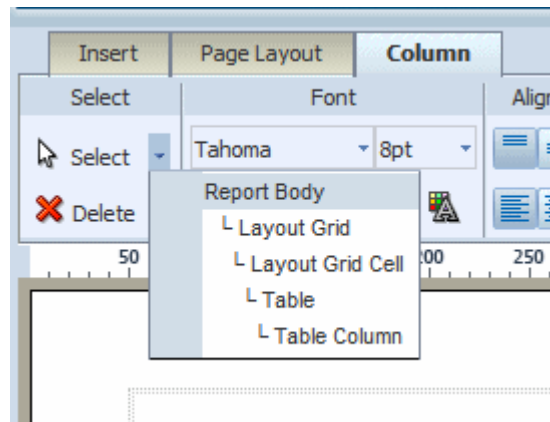
- The **Insert** tab provides the components and page elements that can be placed on a layout. See *Inserting Layout Components*, page 3-22.
- The **Page Layout** tab provides common page-level tools and commands. See *About the Page Layout Menu*, page 3-12.
- The component-specific tab provides the most commonly used commands and properties for the component that is selected in the layout. For example, when you select a chart, the Chart tab displays. See the section on a specific component for details on the commands.

To set or control more properties for the selected component, open the Properties pane in the accordion pane, as described in *About the Properties Pane*, page 3-9.

## Selecting and Deleting Layout Objects

Each of the component-specific tabs include the **Select** region.

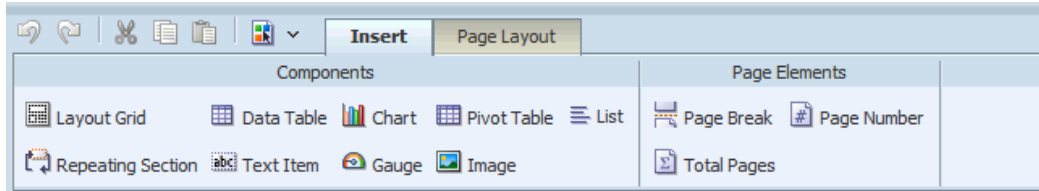
- The **Select** tool enables you to control precisely which component on the layout has focus. This ability is particularly helpful when working with a complex layout where components overlap. For example, to select a table, it is sometimes difficult to click the correct spot to select the table and not a column, or header cell. To avoid unnecessary clicking, use the Select tool to precisely select the Table component from the list.



- The **Delete** tool provides a similar function to the **Select** tool to enable you to precisely select the component to delete.

## About the Insert Tab

Use the **Insert** tab to insert report components and page elements. The following figure shows the **Insert** tab:



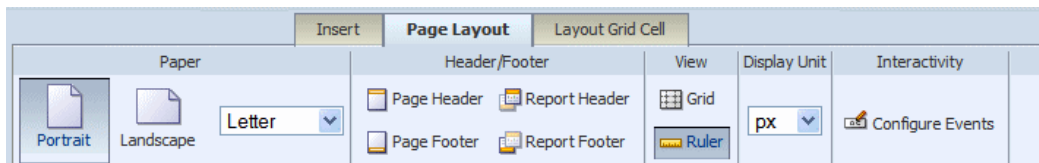
The **Components** group displays the report components that you can insert into your layout. To insert a component, select and drag the item to the desired location in the design area. For more information about each component, see its corresponding section in this chapter.

The **Page Elements** group contains page-level elements for your report. To insert a page break, the page number, or the total page number calculation, select and drag the component to the desired position in the layout.

**Note:** Page elements are intended for paginated output types, such as PDF and RTF. Using them in interactive or HTML output may have unexpected results.

## The Page Layout Tab

The Page Layout tab is shown in the following figure:



The Page Layout tab contains commands to set up your layout.

## Paper Options

Option	Description
Orientation	Choose <b>Portrait</b> or <b>Landscape</b> .

Option	Description
Paper Size	Select from the following paper size options: Letter, Legal, A4, A3, Executive, B5, Com-10, Monarch DL, or C5. Note that the paper size will determine the dimensions of the layout area.

## Header/Footer Options

Option	Description
Page Header	<p>Click to insert a page header in your layout. By default, the page header appears on every page of a printed report, but can be configured to skip the first page.</p> <p>To remove the page header, click <b>Page Header</b> again.</p>
Page Footer	<p>Click to insert a page footer in your layout. By default, the page footer appears on every page of a printed report, but can be configured to skip the last page.</p> <p>To remove the page footer, click <b>Page Footer</b> again.</p>
Report Header	<p>Click to insert a report header to your layout. The report header appears only once at the beginning of the report.</p> <p>To remove the report header, click <b>Report Header</b> again.</p>
Report Footer	<p>Click to insert a report footer to your layout. The report footer appears only once at the end of the report.</p> <p>To remove the report footer, click <b>Report Footer</b> again.</p>

## Setting Properties for Headers and Footers

The **Properties** pane enables you to set the following properties for headers and footers. To access the **Properties** pane, select the header or footer in the design region, then click Properties from the accordion pane on the left of the page.

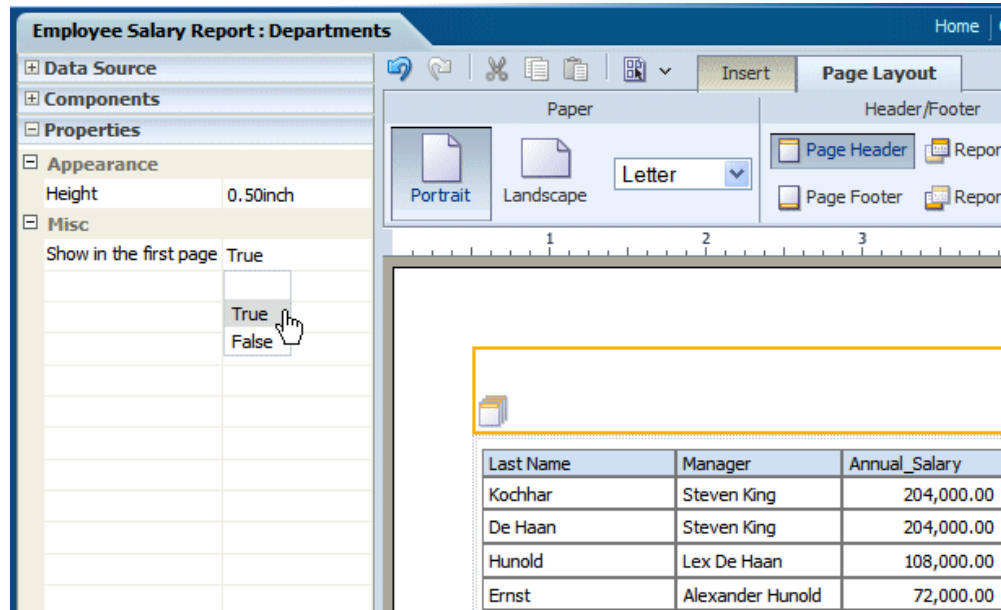
For all report and page headers and footers:

- **Height** - set the height of the header region in pixels, points, centimeters, or inches

For headers:

- **Show in the first page** - select True to show the header in the first page. Select False to suppress the header from the first page.

The following figure shows the Properties for a report header:



For footers:

- **Show in the last page** - select True to show the footer in the last page. Select False to suppress the footer from the last page.

## View Options

Option	Description
<b>Grid</b>	Click to insert gridlines in the layout design area. The grid unit size will depend on the <b>Display Unit</b> selected. To remove the gridlines, click <b>Grid</b> again.
<b>Ruler</b>	Click to insert a display ruler across the top of the layout design area. The ruler units will depend on the <b>Display Unit</b> . To remove the ruler, click <b>Ruler</b> again.

## Display Unit

Select the unit of measure to display. This unit is used for the ruler and grid view options, as well as for any other function that displays a measurement, such as setting

border widths and sizing grid cells. Options are: inch, px (pixel), cm (centimeter), and point (pt).

## Interactivity: Event Configuration

The **Configure Events** feature enables you to configure how components of your layout respond to events triggered by a user when viewing the report in interactive mode.

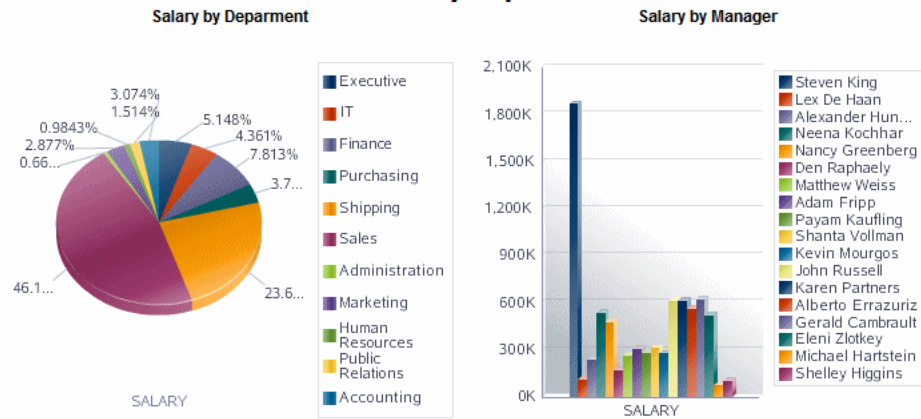
The two types of events are:

- **Filter** - if you click an element in a list, chart, or pivot table, that element will be used to dynamically filter other components defined as targets in the report. The component being clicked does not change.
- **Show Selection Only** - if you click an element of a list, chart, or pivot table, the chart or pivot table (being clicked) will show the results for the selected element only. This action does not affect other components of the report.

## Example of Filter Event Configuration

In this example the layout contains two charts and a table. The first chart shows salary totals by department in a pie chart. The second chart shows salary totals by manager in a bar chart. The table displays a list of employees and their salaries. This example is shown in the following figure:

## Salary Report



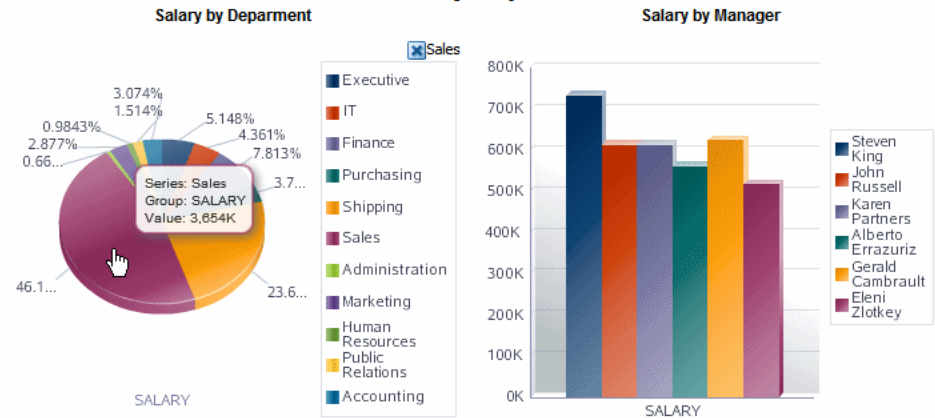
Employee List

Name	Title	Manager	Salary
Neena Kochhar	Administration Vice President	Steven King	204,000.00
Lex De Haan	Administration Vice President	Steven King	204,000.00
Alexander Hunold	Programmer	Lex De Haan	108,000.00
Bruce Ernst	Programmer	Alexander Hunold	72,000.00
David Austin	Programmer	Alexander Hunold	57,600.00
Valli Pataballa	Programmer	Alexander Hunold	57,600.00
Diana Lorentz	Programmer	Alexander Hunold	50,400.00

In this report, if a user clicks on a value in the Salary by Department chart, you want the Salary by Manager chart and the Employees table to automatically filter to show only the managers and employees in the selected department.

The following figure shows the automatic filtering that occurs when a user clicks the Sales department section of the Salary by Department pie chart. The Salary by Manager chart automatically filters to display only the managers belonging to the sales department. The Employee table automatically filters to display only the employees in the sales department.

## Salary Report

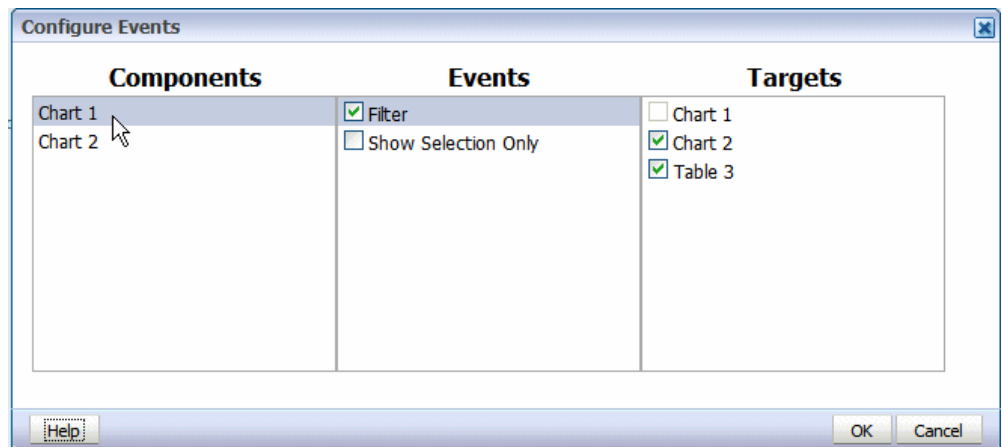


**Employee List**

Name	Title	Manager	Salary
John Russell	Sales Manager	Steven King	168,000.00
Karen Partners	Sales Manager	Steven King	162,000.00
Alberto Errazuriz	Sales Manager	Steven King	144,000.00
Gerald Cambrault	Sales Manager	Steven King	132,000.00
Eleni Zlotkey	Sales Manager	Steven King	126,000.00
Peter Tucker	Sales Representative	John Russell	120,000.00
David Bernstein	Sales Representative	John Russell	114,000.00

### To Configure Automatic Filtering:

1. On the **Page Layout** tab, click **Event Configuration** to display the **Configure Events** dialog.



2. In the **Components** column, click the layout component (lists, charts, and pivot tables are available to configure).
3. Select **Filter** to enable automatic filtering in other report components.

4. Select the report components in the **Targets** column to enable the automatic filtering based on interactive events in the selected component. To disable the automatic filtering for a target component, clear the box.

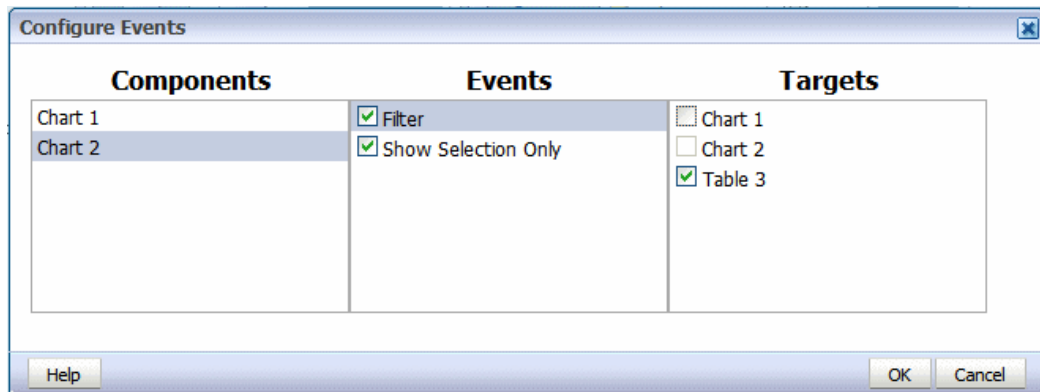
The preceding figure shows that the Filter event is enabled for Chart 1 in the layout. Chart 2 and Table 3 are selected as targets to enable automatic filtering when a selection event occurs in Chart 1.

Note that **Show Selection Only** is not enabled for Chart 1. That means that Chart 1 will continue to display all values when one its elements is selected.

### Example: Show Selection Only

The Show Selection Only event displays only the value of the selected element within the chart or pivot table (being acted on).

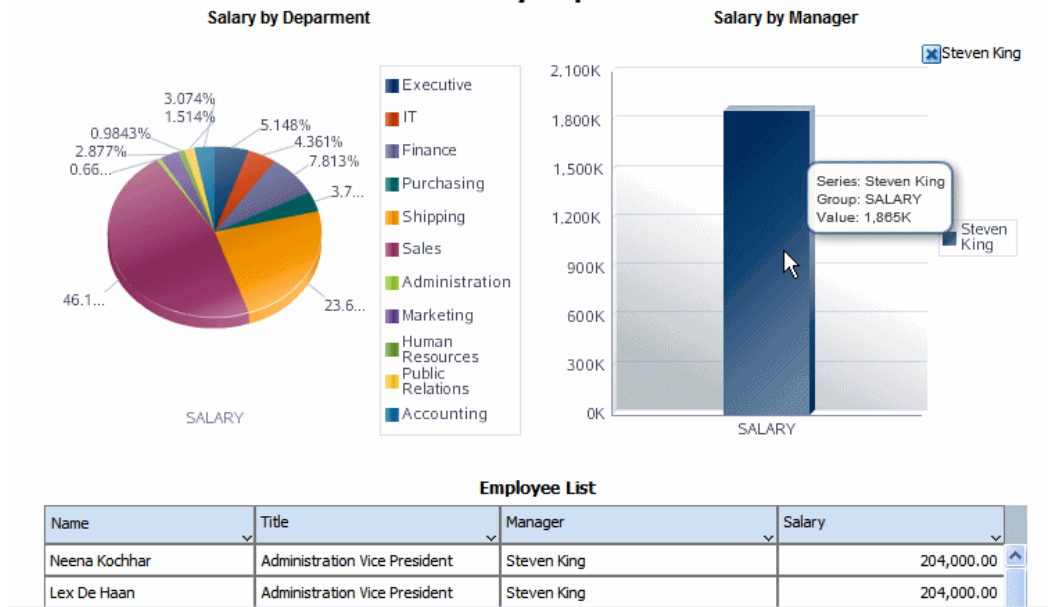
In this example, Chart 2 is configured with **Show Selection Only** enabled and **Filter** enabled with Table 3 as the Target, as shown:



This configuration will result in the output shown below. When the user clicks on Chart 2, only the selected value will be shown in Chart 2. Because the Filter event is enabled for Table 3, the selection is applied as a filter to Table 3.



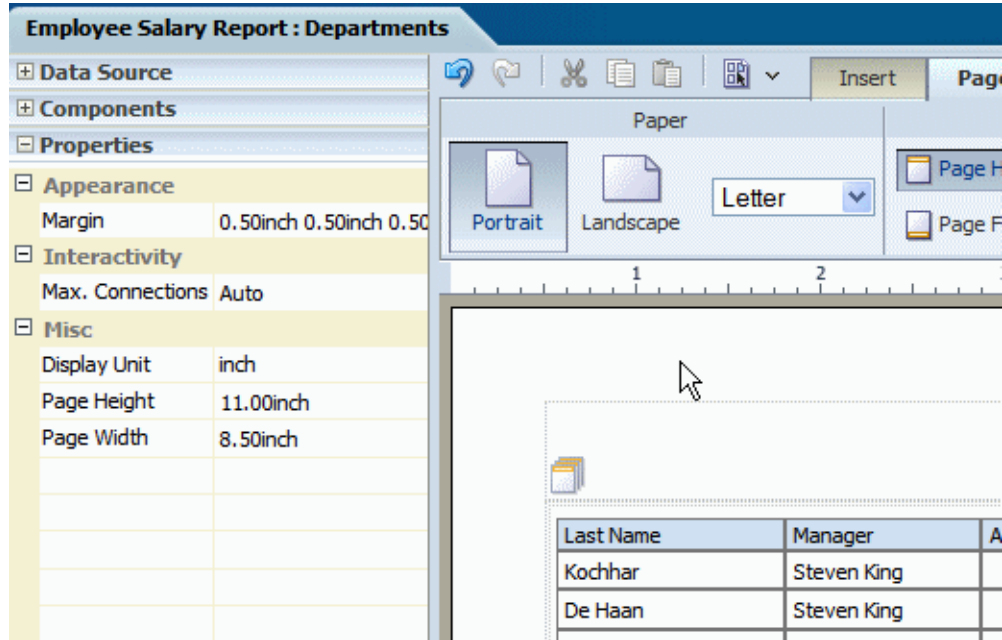
## Salary Report



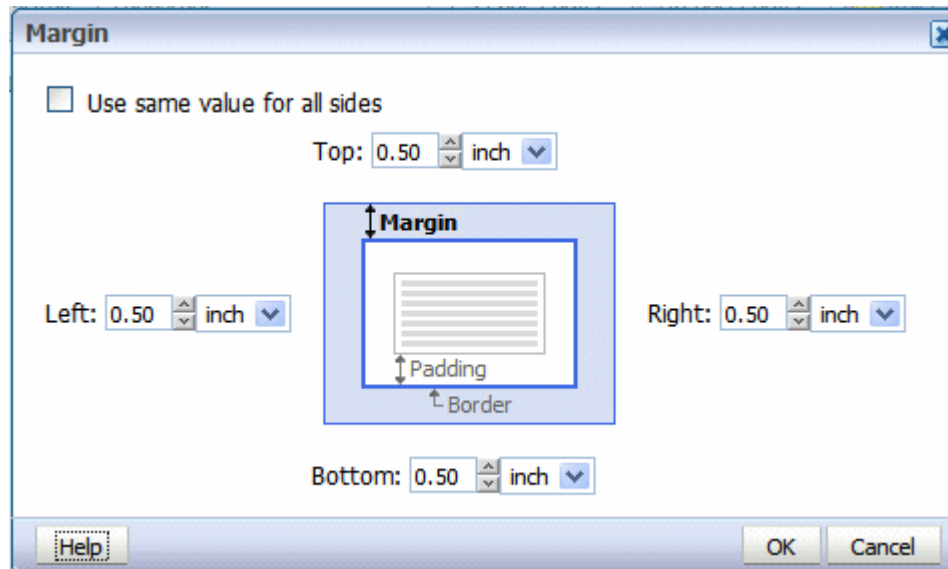
## Setting Page Margins

To set the page margins for your report:

1. Click anywhere in the design area outside of an inserted component.
2. Click the **Properties** pane in the lower left of the Layout Editor. The following figure shows the Properties for the page:



3. Click the value shown for **Margin** to launch the **Margin** dialog. The **Margin** dialog is shown in the following figure:



4. Select the desired size for the margin. Enter the value for the Top, Left, Right, and Bottom margins.

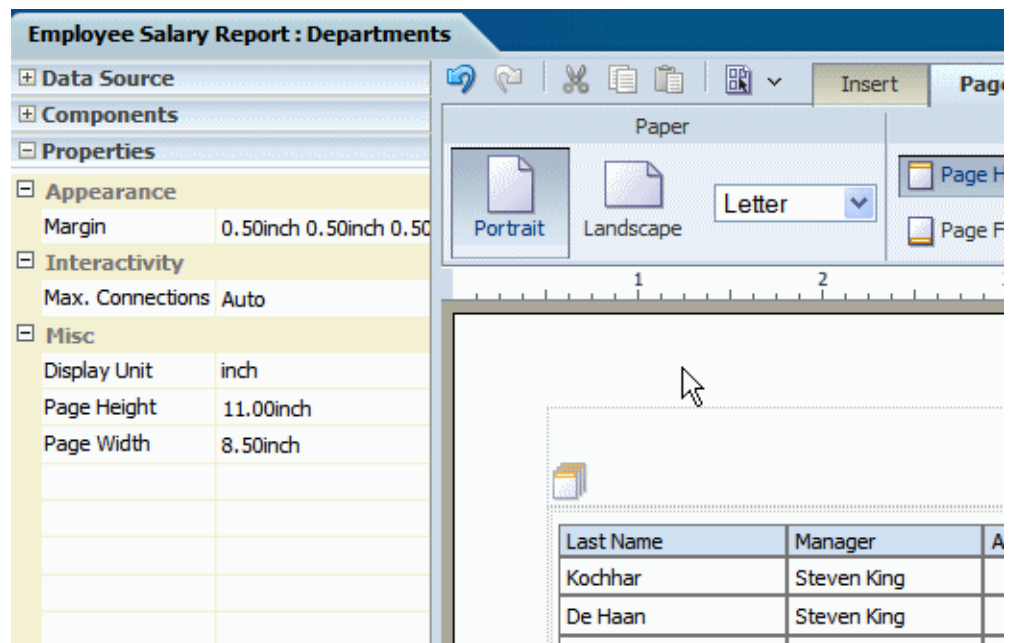
To automatically set the same value for all sides, select the box: **Use same value for all sides**. This action will disable all but the Top margin entry. Enter the value in the Top to apply to all sides.

## Setting Maximum Connections for an Interactive Report

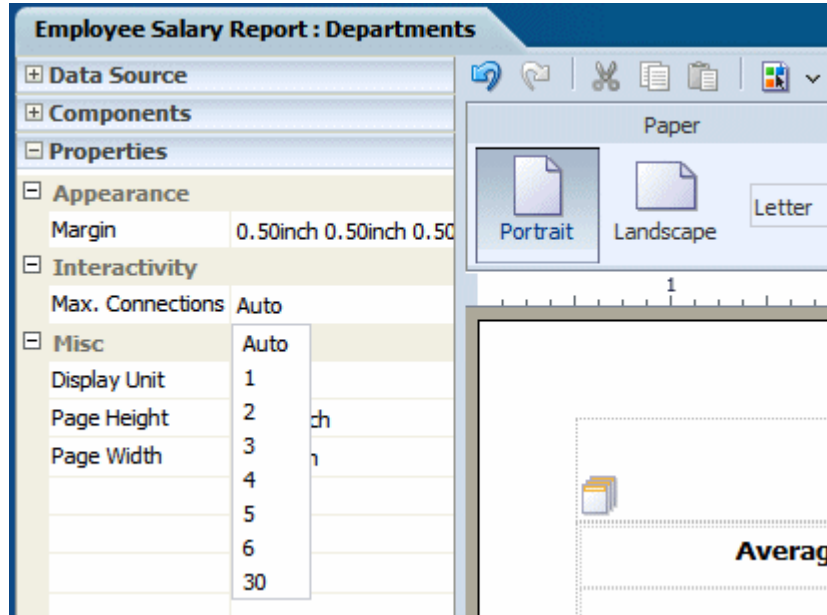
You can limit the connections from the browser to the server for the interactive viewer. More connections are faster but increase server load. The default is six connections. Reduce the number to reduce the load on your server for large reports.

To set the maximum connections for this layout:

1. Click anywhere in the design area outside of an inserted component.
2. Click the **Properties** pane in the lower left of the Layout Editor. The following figure shows the Properties for the page:



3. Click the value shown for **Max. Connections** and select the desired value from the list.:



## Inserting Layout Components

The layout editor supports components that are typically used in reports and other business documents. The following components are described in these sections:

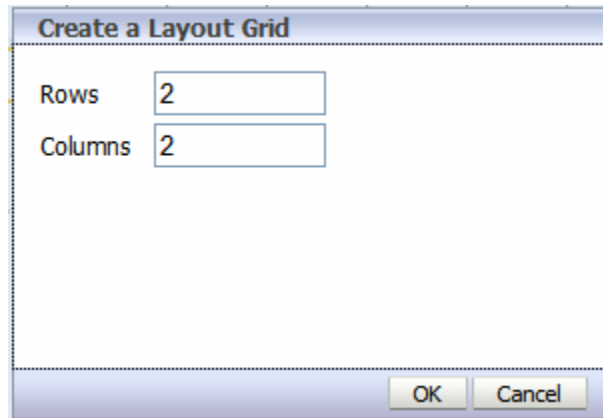
- Layout Grid, page 3-22
- Data Table, page 3-31
- Chart, page 3-52
- Gauge, page 3-58
- Pivot Table, page 3-61
- Repeating Section, page 3-26
- Text Item, page 3-68
- Image, page 3-73
- List, page 3-74

## About Layout Grids

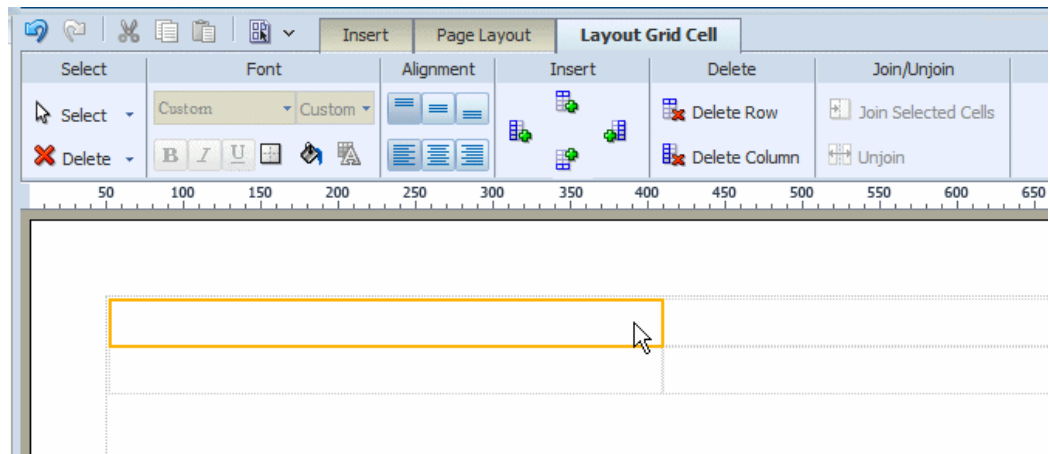
The layout grid provides a way to divide a layout into sections. It functions similarly to a table in HTML or Word documents to create forms or to provide sophisticated

layouts. Use a layout grid to control the exact placement of all other components in the layout.

To create a layout grid, select and drag the Layout Grid component to the design area.



In the dialog, enter the number of rows and columns for the grid and click **OK** to insert the grid to the design area as shown in the following figure:



Note the following about a layout grid:

- The grid is created with equidistant columns, and the row size defaults to a minimum of one row of text.
- Although Font properties are not enabled for a layout grid cell (set font properties using the individual component properties), the background color and border properties are enabled.
- When you insert a component to a grid cell, it automatically resizes to accommodate the component.
- Adjust the column width and height by either positioning the mouse pointer over

the border and dragging the blue bar, or by changing the grid column properties in the Properties pane.

- The grid supports merging of cells.
- You can insert a grid inside a grid.
- Similar to Microsoft Word, the grid uses a flow layout that is very convenient for designing business documents. Components that do not occupy a full paragraph or block are positioned top-down and left to right.

## Adding a Border or Background Color

By default, the gridlines are displayed in the design area only and are not shown during runtime. If you wish to display the gridlines in your finished report, select the grid cell and click the **Set Border** command button to launch the **Border** dialog.

To add a background color to a cell, click the Background Color command button to launch the Color Picker.

## About the Insert Options

Once you have inserted a layout grid, you can add additional rows or columns. Select the layout grid cell that is the focal point, then click the appropriate command button:

- Add a Row above
- Add a Column to the right
- Add a Row below
- Add a Column to the left

## About the Join/Unjoin Options

To join cells horizontally or vertically, select multiple adjacent cells by holding down the Ctrl key and clicking each grid cell. Then click the **Join** command button.

To unjoin cells that have been joined, select the joined cell and click the **Unjoin** button.

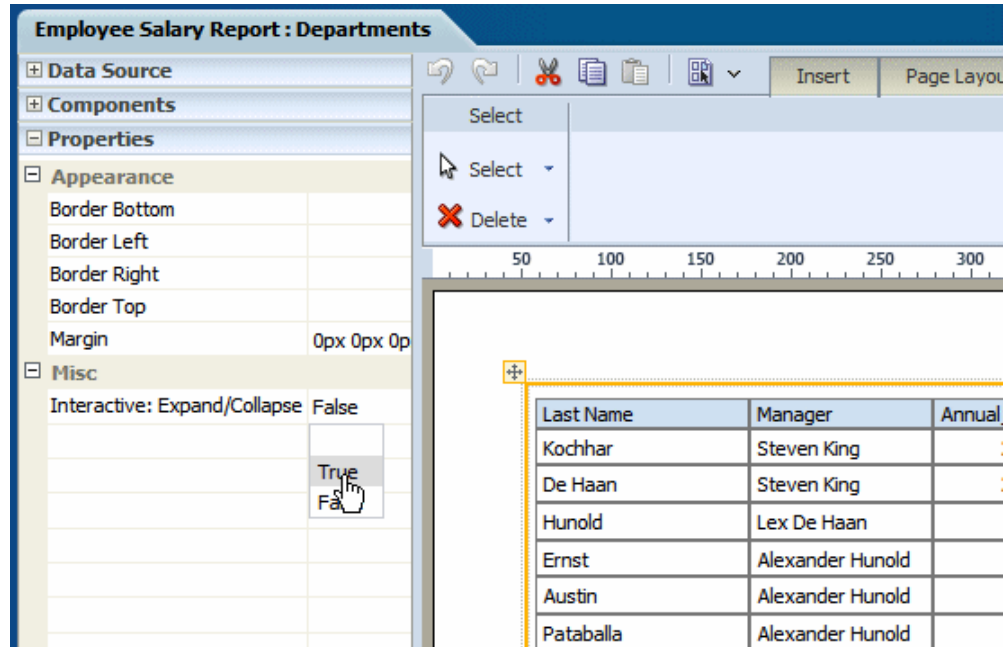
## Adding an Expand and Collapse Option

When viewing a report in interactive mode, expand and collapse of a layout grid are supported. Expand and Collapse are supported at the grid level, (not the cell-level) therefore ensure to insert grids appropriately. For example, if your report contains a chart in the top portion of the layout and a table in the bottom and you want to be able to collapse the chart display, you must insert one layout grid to contain the chart and a

second layout grid beneath the first to contain the table. Do not insert one grid with two rows.

To enable the expand and collapse option:

1. Select the layout grid.
2. Open the **Properties** pane.
3. Set the **Interactive: Expand/Collapse** property to True. The following figure shows this option on the Properties pane.



The following figures demonstrate the expand and collapse behavior when the report is viewed in interactive mode. Note the collapse icon in the upper right area of the report. Click the icon to collapse the grid. The second figure shows the report with the region collapsed.



The screenshot shows the Oracle BI Publisher Enterprise interface for the same 'Employee Salary Report', but with the 'Employee List' table displayed. The table lists employee details including Name, Title, Manager, and Salary.

Name	Title	Manager	Salary
Neena Kochhar	Administration Vice President	Steven King	204,000.00
Lex De Haan	Administration Vice President	Steven King	204,000.00
Alexander Hunold	Programmer	Lex De Haan	108,000.00
Bruce Ernst	Programmer	Alexander Hunold	72,000.00
David Austin	Programmer	Alexander Hunold	57,600.00
Vali Pataballa	Programmer	Alexander Hunold	57,600.00
Diana Lorentz	Programmer	Alexander Hunold	50,400.00
Nancy Greenberg	Finance Manager	Neena Kochhar	144,000.00
Daniel Faviet	Accountant	Nancy Greenberg	108,000.00
John Chen	Accountant	Nancy Greenberg	98,400.00
			7,924,800.00

## About Repeating Sections

Repeating sections repeat the components within the section of the layout based on the occurrence of an element in the data. Repeating sections are used to create classic banded reports, as well as repeating pages or sections for different data elements (such as Group Above/Outline).

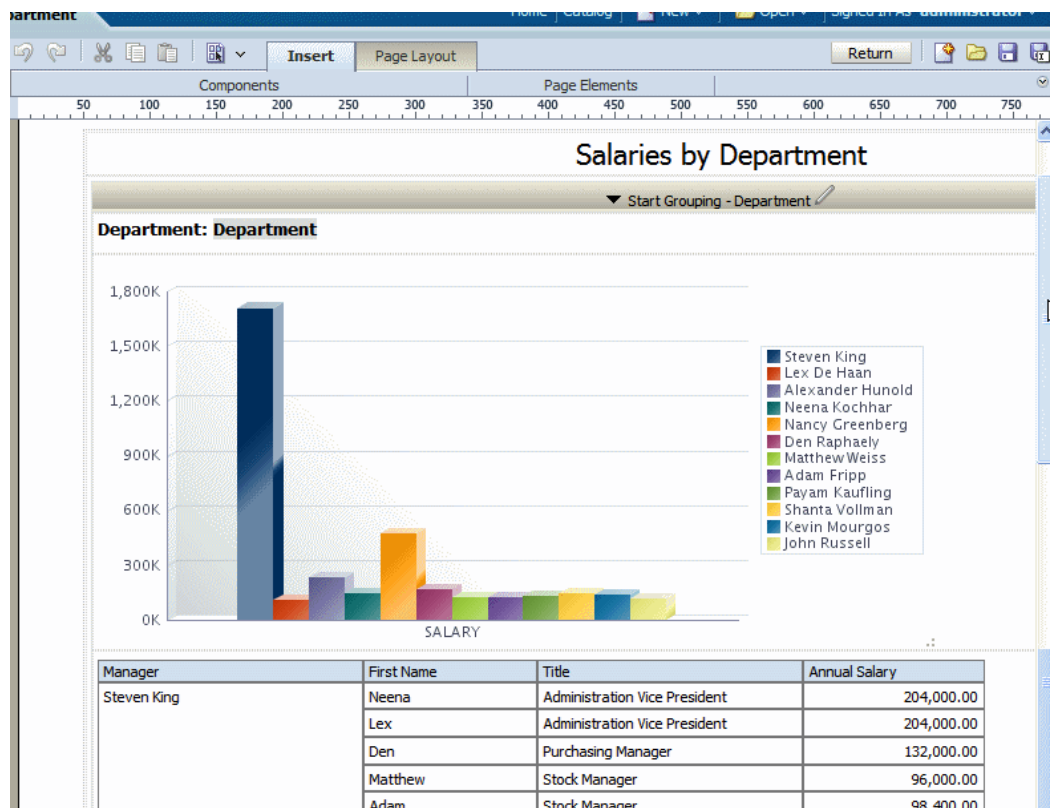
**To create a repeating section:**

1. Drag and drop the repeating section component to the layout.
2. In the **Repeating Section** dialog, select one of the following:



- **Element:** Specify the element for which the section repeats. For example, if your dataset contains sales information for several countries. If you select COUNTRY as the repeat-by element, then the section of the layout repeats for each unique country occurring in the dataset.
- **Group Detail:** If you have nested sections, then select this option. To continue the previous example, assuming there are unique data rows for each city and grouping by country, then this option creates a section that repeats for each city.

The following example shows a layout that has a repeating section defined for the element Department. Within the repeating section are a chart that shows salaries by manager and a table that shows all employee salaries. So for each occurrence of department in the dataset, the chart and table will be repeated:



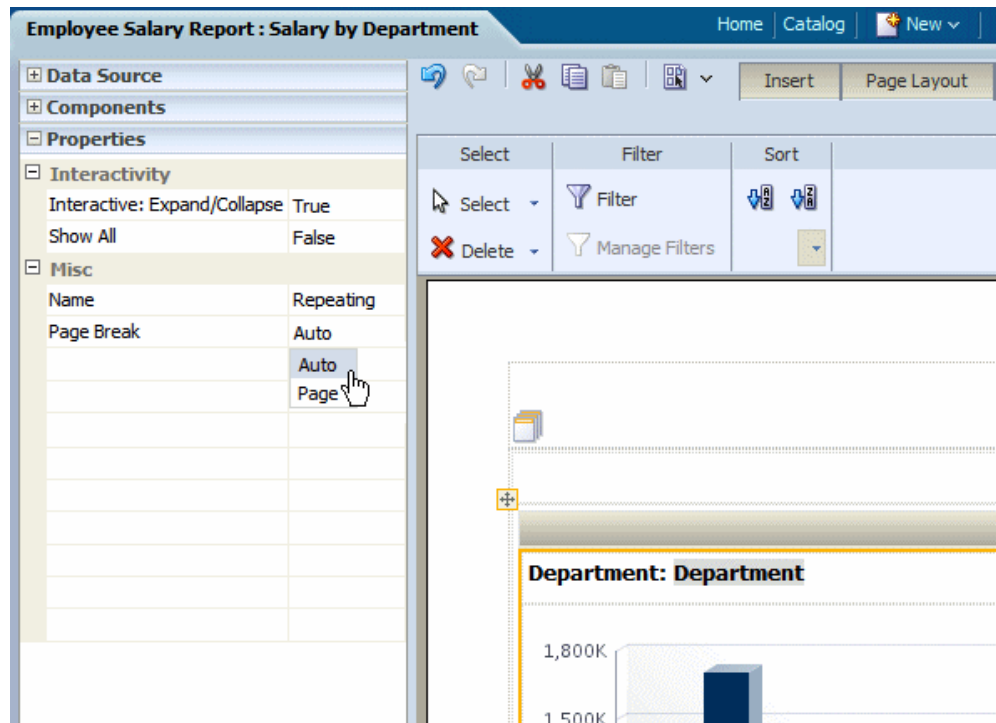
## Setting Page Break Options for a Repeating Section

By default, for paginated output types, the page will break automatically according to the amount of content that will fit on a page. It is frequently desirable to have the report break after each occurrence of the repeated content.

Using the preceding example, it would be desirable for the PDF output of this report to break after each department. To create a break in the report after each occurrence of the repeating section:

1. Select the repeating section component.
2. Open the **Properties** pane.
3. Set the **Page Break** property to **Page**.

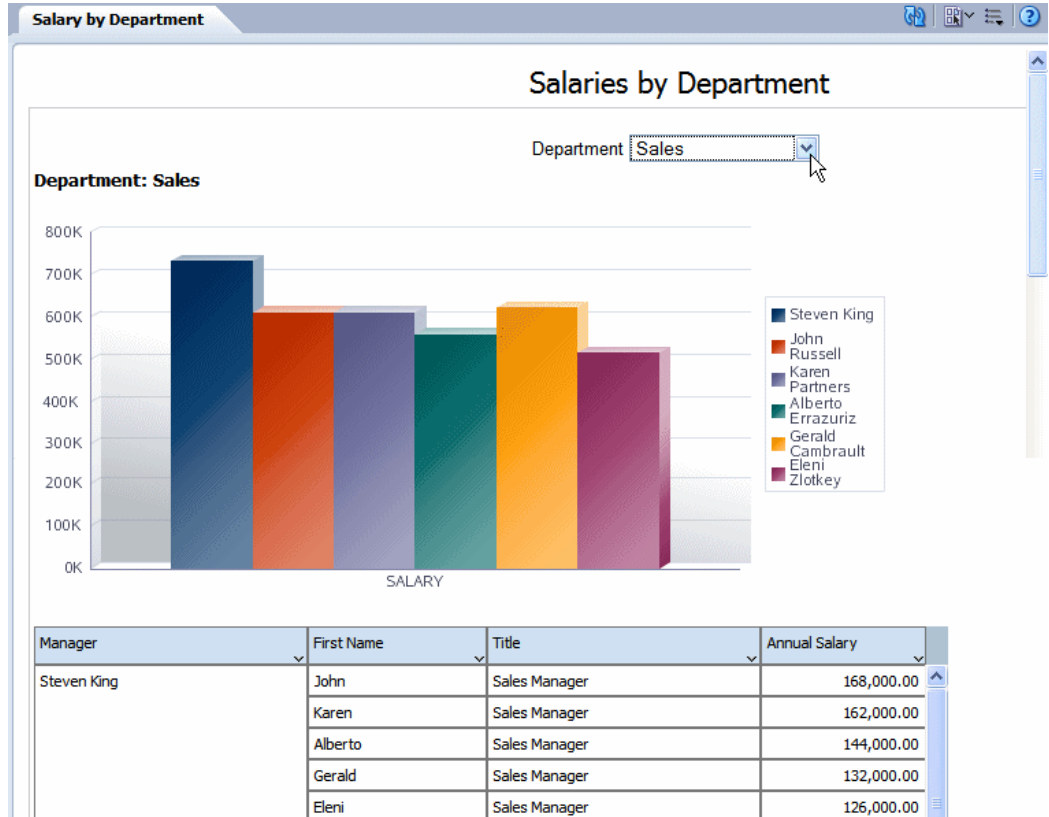
The following figure displays the Properties for a repeating section:



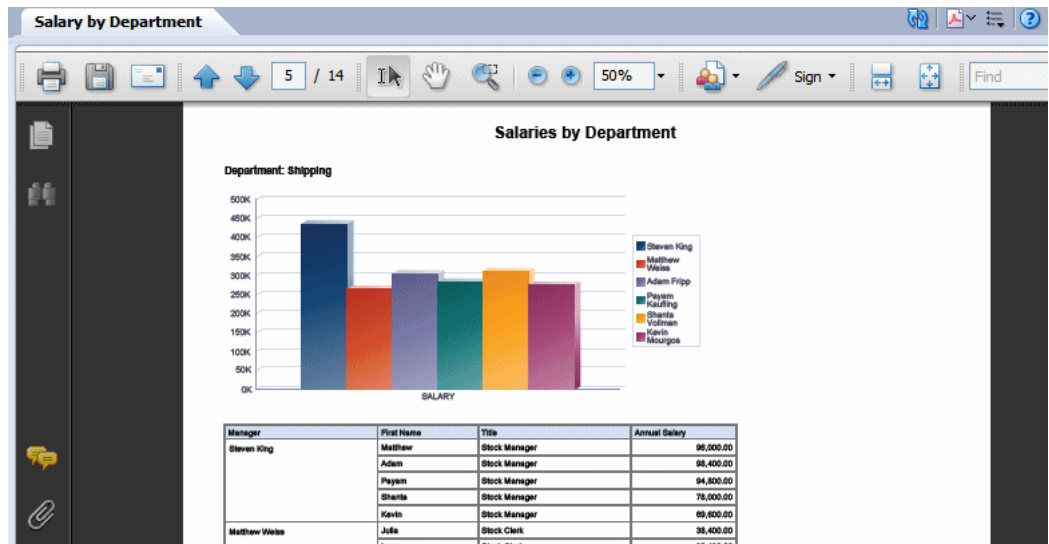
## How Repeating Sections Display in Interactive Mode

In interactive mode, the values for the repeat by element are displayed as a list of values. This enables the enable the report consumer to dynamically select and view the results.

Shown in the example below, the repeat by element Department is displayed in a list of values:



By contrast, note the same layout displayed in PDF. In this example the page break option is set so that each new department begins the repeating section on a new page:



## Showing All Values in a Repeating Section

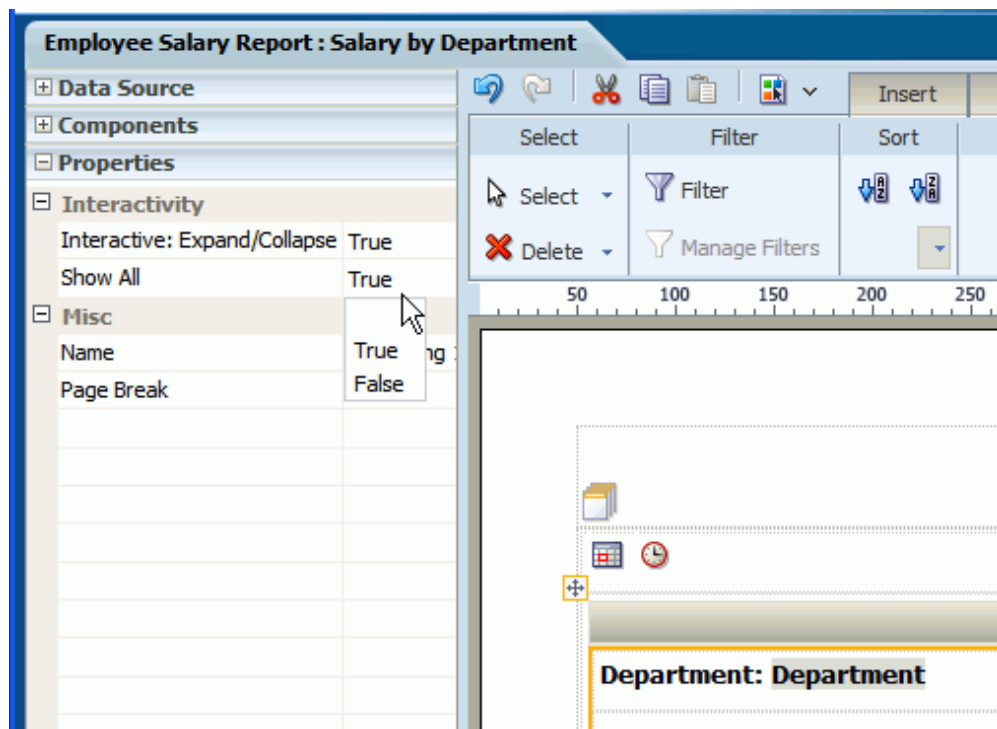
In interactive mode, the values for the repeat by element are displayed as a list of values. By default, this list includes only the values present for the element in the data. Therefore, a report consumer can view results for only one item at a time.

To enable a report consumer to view the results in the repeating section for all values of the element, the Repeating Section component provides the property: Show All. When this property is set to true, the value "All" is added to the list to enable the display of results for all values.

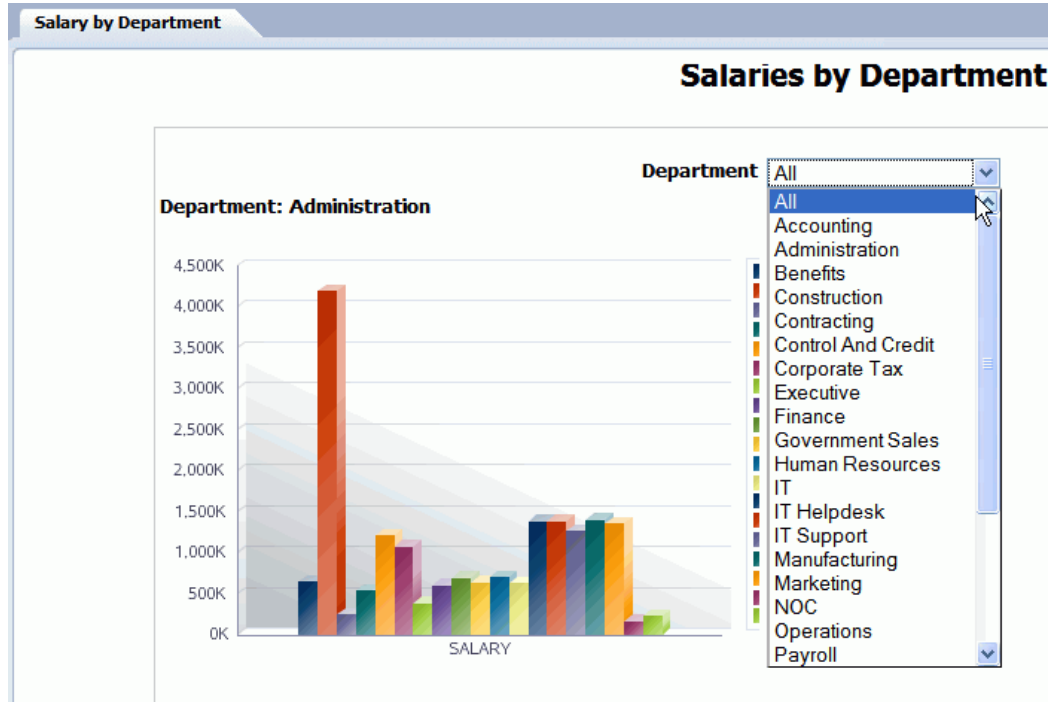
To enable Show All:

1. Select the repeating section component.
2. Open the **Properties** pane.
3. Set the **Show All** property to **True**.

The following figure displays the **Show All** setting in the **Properties** pane:



When you view the report, the option All is added to the menu of values:



## About Data Tables

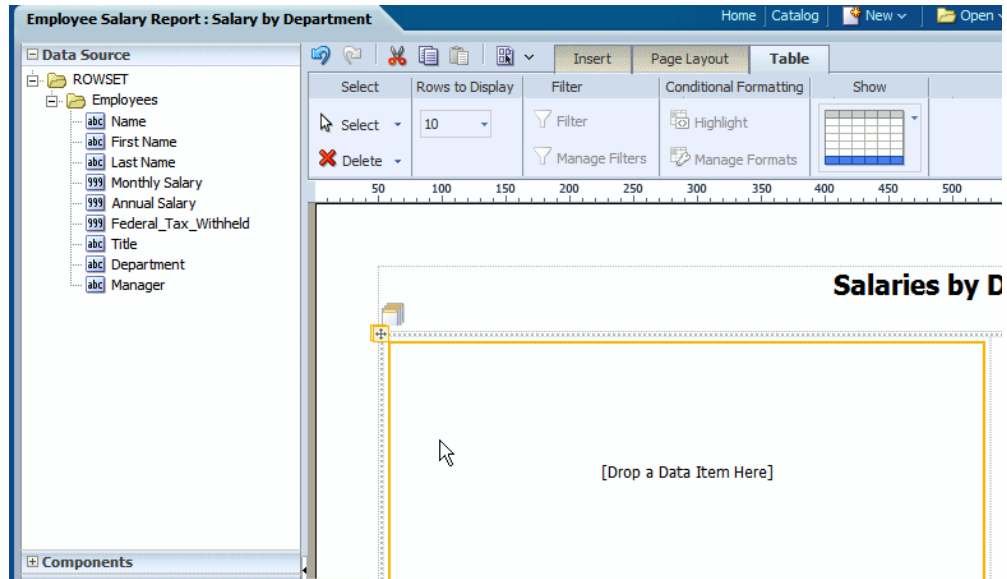
The data table is a standard table that is shown in many layouts. It contains a header, data columns, and a total row. The table supports "group left" functionality (outlines) that merges fields with the same values as well as subtotals, grand totals, custom calculations, and running totals.

Once inserted, you can edit the table properties using the dynamic tabs or the **Properties** pane. The following dynamic tabs are available for the table components:

- Table
- Table Column Header
- Column
- Total Cell

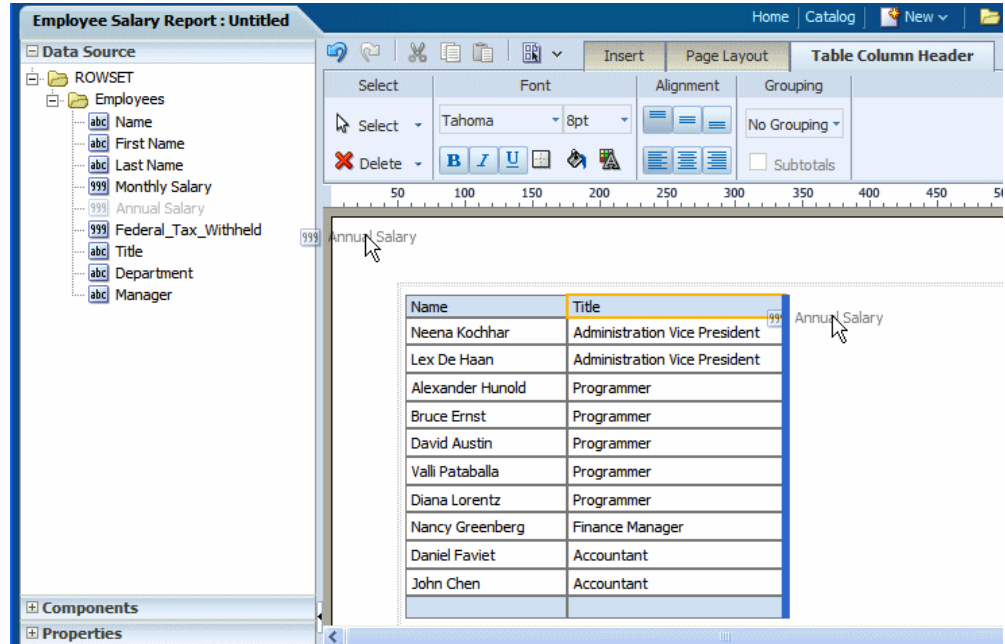
## Inserting a Data Table

1. From the **Insert** tab, select and drag the **Data Table** component to the design area.  
The following figure shows an inserted, empty data table. Notice that the **Table** tab is now displayed.



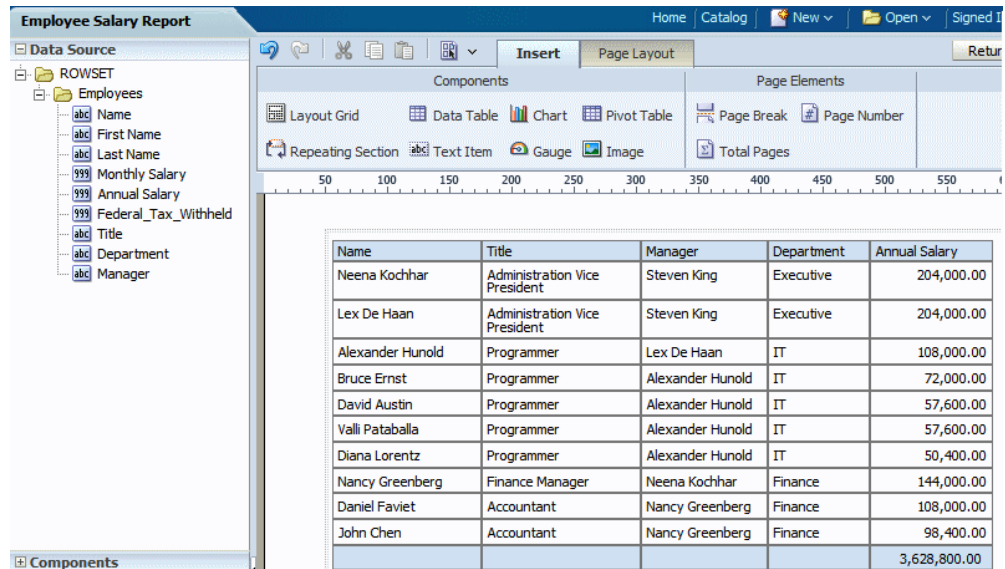
2. To add data columns to your table, select an element from the **Data Source** pane and drag it to the table in the layout.

The following figure shows the columns being added to the table. Notice that when you drop a column on the table your sample data is immediately displayed.



- Continue to drag the elements from the **Data Source** pane to form the columns of your table. If you need to reposition a column that you have already added, select it and drag it to the correct position.

The following figure shows a completed data table.



Notice the following default behavior:

- A total row is automatically inserted. By default it calculates the sum of the items in the column. You can remove this row or edit the display and

calculation applied. See About the Total Cell Tab, page 3-51.

- Default date formatting is applied. To change the default formatting, see About the Column Tab, page 3-47.
- Default number formatting and alignment is applied. To change the default formatting, see About the Column Tab, page 3-47.

## Setting Alternating Row Colors

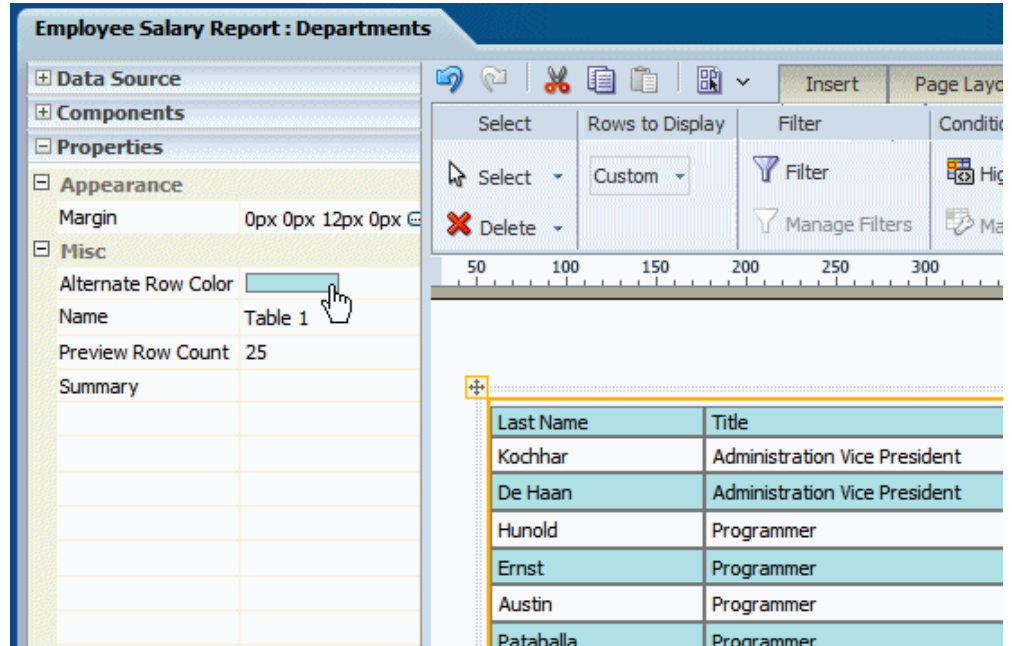
Some data tables are easier to read when the rows display alternating colors, as shown in the following figure:

Last Name	Title	Manager	Department	Monthly_Salary	Annual_Salary
Kochhar	Administration Vice President	Steven King	Executive	17000	204,000.00
De Haan	Administration Vice President	Steven King	Executive	17000	204,000.00
Hunold	Programmer	Lex De Haan	IT	9000	108,000.00
Emst	Programmer	Alexander Hunold	IT	6000	72,000.00
Austin	Programmer	Alexander Hunold	IT	4800	57,600.00
Pataballa	Programmer	Alexander Hunold	IT	4800	57,600.00
Lorentz	Programmer	Alexander Hunold	IT	4200	50,400.00
Greenberg	Finance Manager	Neena Kochhar	Finance	12000	144,000.00
Faviet	Accountant	Nancy Greenberg	Finance	9000	108,000.00
Chen	Accountant	Nancy Greenberg	Finance	8200	98,400.00
Seyoum	Accountant	Nancy Greenberg	Finance	7700	92,400.00

To set an alternating row color:

1. Select the table.
2. Open the **Properties** pane.
3. Click the value shown for **Alternate Row Color** to launch the color picker.



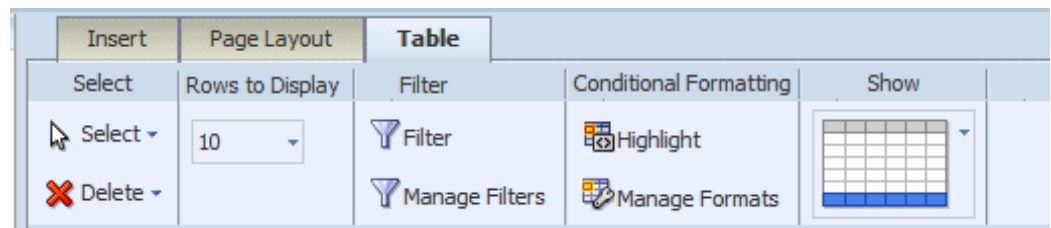


4. Choose a color and click OK.

## About the Table Tab

The Table Tab enables you to perform the following:

- Set the number of rows displayed
- Define filters for the data displayed in the table
- Define conditions and formats to apply to rows that meet the conditions
- Show or hide the total row for the table



## Setting the Rows to Display Option

The **Rows to Display** property controls the number of rows of data displayed as follows:

- When designing the layout, this property sets the number of rows that will be displayed for the table within the layout editor.
- When viewing this layout in the report viewer in interactive mode, this property sets the size of the scrollable region for the table.

The default is 10 rows of data. You can select 10, 20, 30, 40, or All rows of data to be displayed. To set a custom value, open the **Properties** pane and enter the custom value for the **Rows to Display** property.

**Note:** Displaying more rows of data could impact performance of the Layout Editor.

## About Filters

A filter refines the displayed items by a condition. This is a powerful feature that enables you to display only desired elements in your table without having to perform additional coding. For example, you could add a filter to meet some of the following report conditions:

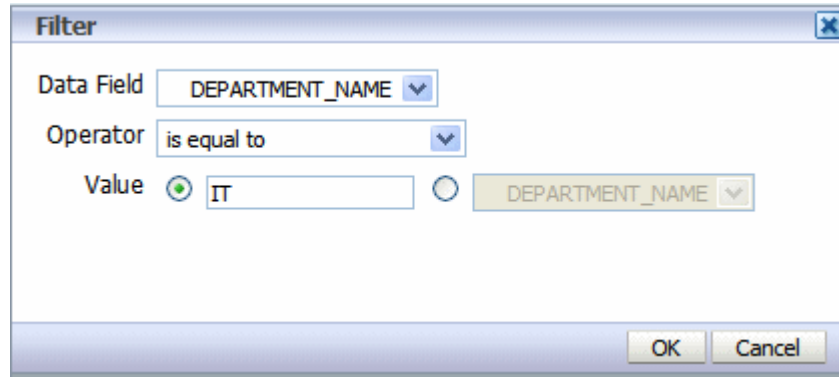
- Display only the top 10 salaries
- Display only the bottom 25 store sales
- Display only employees in the IT department
- Display only sales that are between \$10,000 and \$20,000 and in the Southern region

You can add multiple filters and manage the order in which they are applied to your table data.

## Setting Filters for a Table

To set a filter:

1. Click the **Filter** toolbar button. This launches the **Filter** dialog, shown in the following figure:



2. Enter the fields to define a filter:

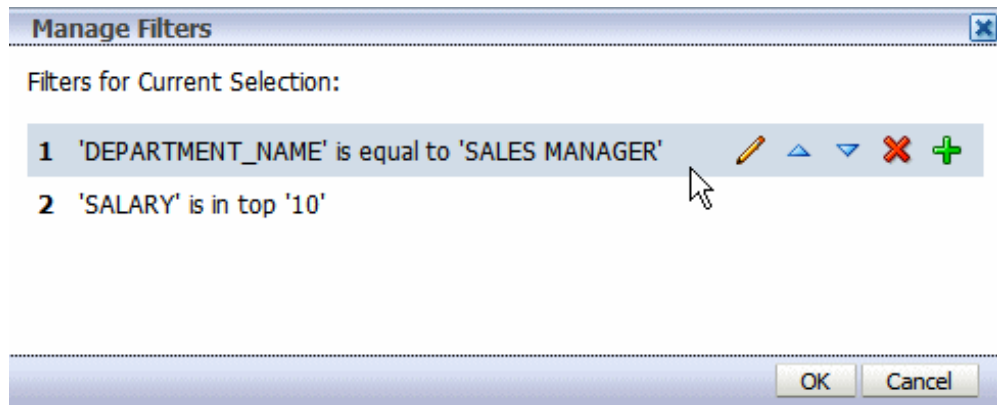
Field	Description
<b>Data Field</b>	Choose the data field to filter the table data by. All elements are available regardless of whether they are included as table columns.
<b>Operator</b>	Select from the following operators: is equal to is not equal to is less than is greater than is less than or equal to is greater than or equal to is between is in top is in bottom
<b>Value</b>	Enter the value or values appropriate for the operator selected. The value can be either a text entry, or an element from the data.

### Managing Filters

After you have added filters, use the Manage Filters feature to edit, delete, or change the order that the filters are applied.

To manage filters:

1. Click the **Manage Filters** toolbar button to launch the **Manage Filters** dialog shown in the following figure:



2. Pause your cursor over the filter to display the actions toolbar. Use the toolbar buttons to edit the filter, move the filter up or down in the order of application, delete, or add another filter.

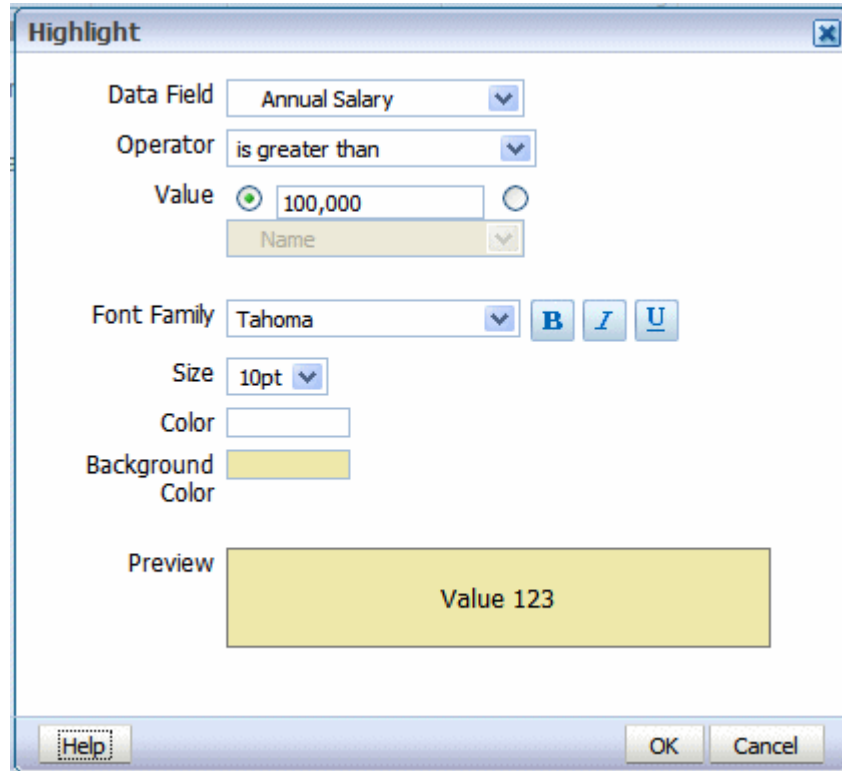
## About Conditional Formats

A conditional format changes the formatting of an element in your table based on a condition. This feature is extremely useful for highlighting target ranges of values in your table. For example, you could create a set of conditional formats for your table that display rows in different colors depending on threshold values.

### Applying Conditional Formats to a Table

To apply a conditional format:

1. Click the **Highlight** button. This launches the **Highlight** dialog, shown in the following figure:



2. Enter the fields to define a condition and format to apply:

Field	Description
<b>Data Field</b>	Choose the data field to apply the condition to. All elements are available regardless of whether they are included as table columns. For example, you may want to highlight in red all employees with salaries greater than \$10,000, but not actually include the salary element in the table.
<b>Operator</b>	Select from the following operators: is equal to is not equal to is less than is greater than is less than or equal to is greater than or equal to is between

Field	Description
Value	<p>Enter the value or values appropriate for the operator selected. The value can be either a text entry, or an element from the data.</p> <p><b>Important:</b> If entering a date value, use on of the following XSL date or time formats: YYYY-MM-DD or YYYY-MM-DDTHH:MM:SS.</p>
Font Family	<p>Select the font to apply to the row of data that meets the condition. You can also apply bold, italic, or underline emphasis.</p>
Size	<p>Select the size of the font to apply to the row of data that meets the condition.</p>
Color	<p>Click the color box to open the <b>Color Picker</b>. Choose one of the predefined colors or click <b>Custom Color</b> to define your own color to apply to the font.</p>
Background Color	<p>Click the color box to open the <b>Color Picker</b>. Choose one of the predefined colors or click <b>Custom Color</b> to define the background color to apply to the row.</p>

The following figure shows the table in the layout with the condition applied:

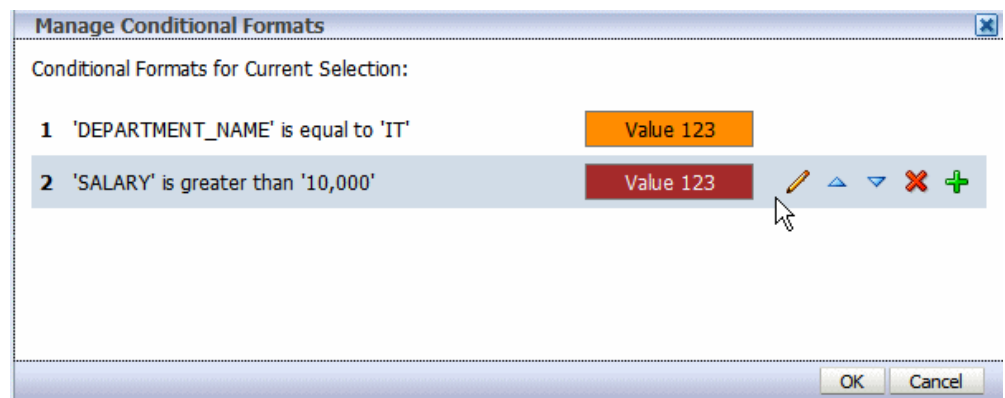
Name	Title	Manager	Department	Annual Salary
Neena Kochhar	Administration Vice President	Steven King	Executive	204,000.00
Lex De Haan	Administration Vice President	Steven King	Executive	204,000.00
Alexander Hunold	Programmer	Lex De Haan	IT	108,000.00
Bruce Ernst	Programmer	Alexander Hunold	IT	72,000.00
David Austin	Programmer	Alexander Hunold	IT	57,600.00
Valli Pataballa	Programmer	Alexander Hunold	IT	57,600.00
Diana Lorentz	Programmer	Alexander Hunold	IT	50,400.00
Nancy Greenberg	Finance Manager	Neena Kochhar	Finance	144,000.00
Daniel Faviet	Accountant	Nancy Greenberg	Finance	108,000.00
John Chen	Accountant	Nancy Greenberg	Finance	98,400.00
				3,628,800.00

## Managing Formats

After you have added conditional formats, use the **Manage Formats** command to edit or delete a format.

To manage formats:

1. Click the **Manage Formats** button to launch the **Manage Conditional Formats** dialog shown in the following figure:

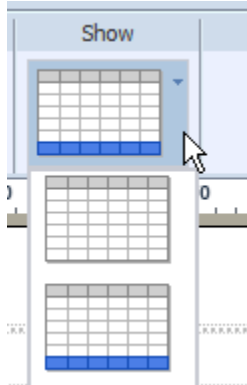


2. Pause your cursor over an item to display the actions toolbar. Use the toolbar buttons to edit the format, move the format up or down in the order of application,

delete, or add another format. Note that the order of the conditions is important because only the first condition met will be applied.

### Controlling the Display of the Total Row

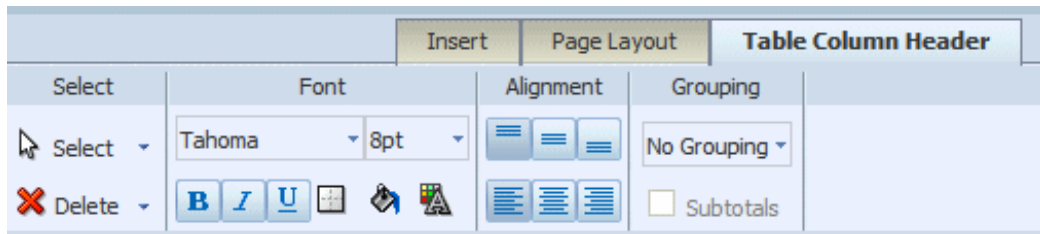
By default, the layout editor inserts a total row in your table that sums numeric columns. To remove the total row, click the **Show** menu and select the table view without the highlighted total row. The **Show** menu options are shown in the following figure:



The total row can be further customized using the **Total Cell** tab and the **Properties** pane. For more information see About the Total Cell Tab, page 3-51.

### About the Table Column Header Tab

The **Table Column Header** tab is shown in the following figure:



The Table Column Header tab enables you to perform the following:

- Edit the font properties of the table header column
- Edit the cell properties of the table header including border weight, style, and color and background fill color
- Set the vertical and horizontal alignment of the table header
- Apply grouping



## About Grouping

"Grouping" groups together elements in the data of the same value. In a table, applying grouping can make your table easier to read.

The Grouping option enables you to choose between "Group Left" or "Group Above". Group left maintains the "group by" element within the table. The following figure shows a table that has been grouped by Manager using **Group Left**:

Manager	Employee	Title	Hire Date	SALARY
Zlotkey	Abel	Sales Representative	May 10, 1996	11,000.00
	Hutton	Sales Representative	Mar 18, 1997	8,800.00
	Taylor	Sales Representative	Mar 23, 1998	8,600.00
	Livingston	Sales Representative	Apr 22, 1998	8,400.00
	Johnson	Sales Representative	Jan 03, 2000	6,200.00
Weiss	Landry	Stock Clerk	Jan 13, 1999	2,400.00
	Markle	Stock Clerk	Mar 07, 2000	2,200.00
	Mikkilineni	Stock Clerk	Sep 27, 1998	2,700.00
Vollman	Ladwig	Shipping Clerk	Jul 13, 1995	3,600.00
	Stiles	Shipping Clerk	Oct 25, 1997	3,200.00
	Seo	Shipping Clerk	Feb 11, 1998	2,700.00
	Patel	Stock Clerk	Apr 05, 1998	2,500.00

Group above inserts a Repeating Section component, and extracts the grouping element from the table. The grouping element is instead displayed above the table and a separate table is displayed for each occurrence of the grouping element. The following figure shows a table that has been grouped by Manager using **Group Above**:

**Manager Zlotkey**

Employee	Title	Hire Date	SALARY
Abel	Sales Representative	May 10, 1996	11,000.00
Hutton	Sales Representative	Mar 18, 1997	8,800.00
Johnson	Sales Representative	Jan 03, 2000	6,200.00
Livingston	Sales Representative	Apr 22, 1998	8,400.00
Taylor	Sales Representative	Mar 23, 1998	8,600.00
		Grand Total	43,000.00

**Manager Weiss**

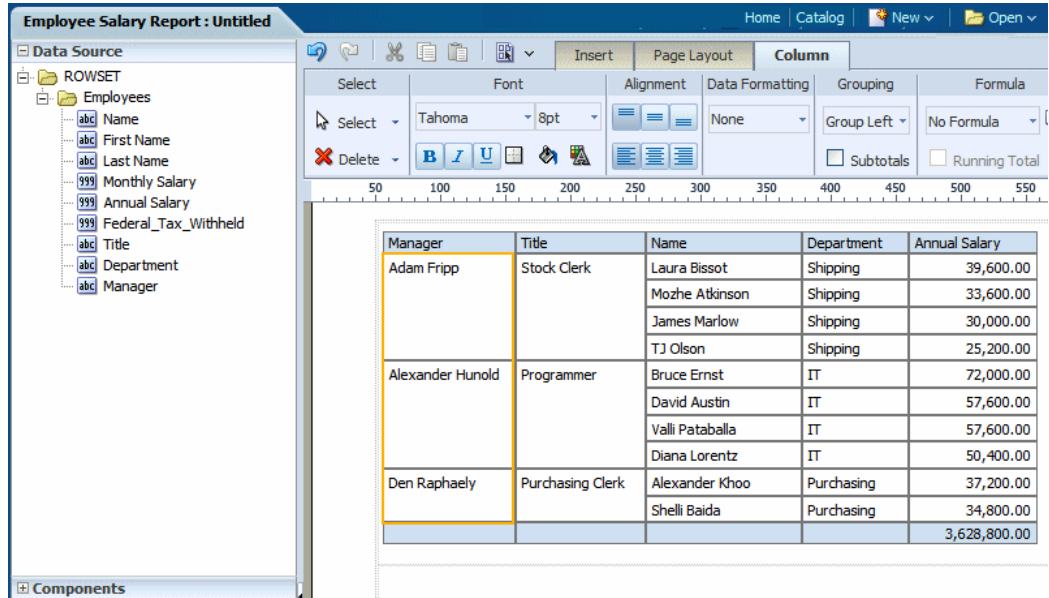
Employee	Title	Hire Date	SALARY
Fleur	Shipping Clerk	Feb 22, 1998	3,100.00
Geoni	Shipping Clerk	Feb 02, 2000	2,800.00
Landry	Stock Clerk	Jan 13, 1999	2,400.00
Markle	Stock Clerk	Mar 07, 2000	2,200.00
Mikkilineni	Stock Clerk	Sep 27, 1998	2,700.00
Nayer	Stock Clerk	Jul 15, 1997	3,200.00
		Grand Total	22,100.00

**Manager Vollman**

Employee	Title	Hire Date	SALARY
Bell	Shipping Clerk	Feb 03, 1996	4,000.00
Everett	Shipping Clerk	Mar 02, 1997	3,900.00

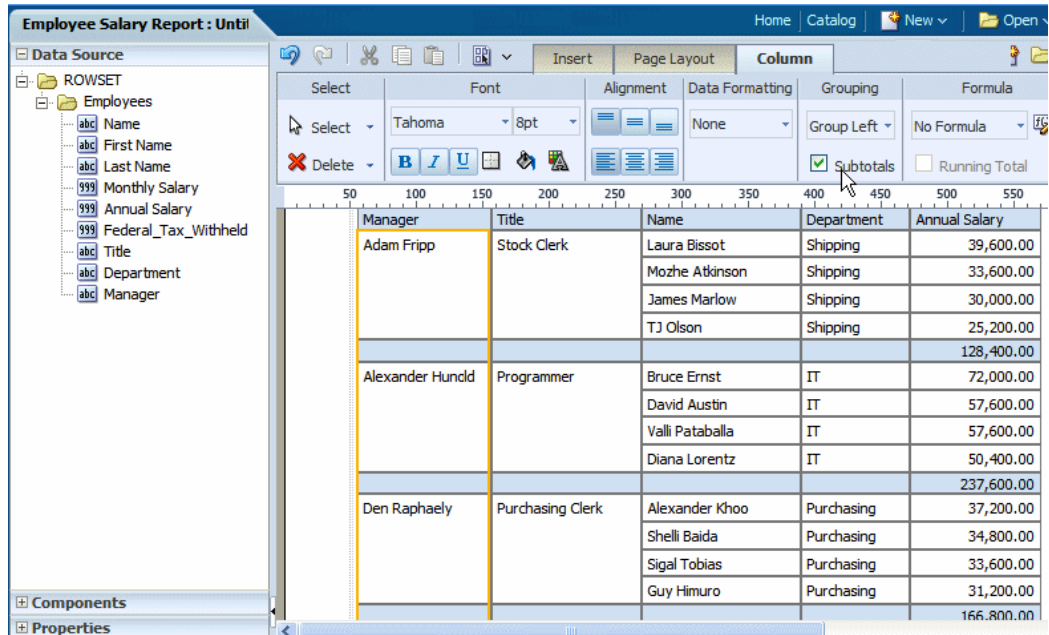
**Example: Group Left**

In the following example, the table data has been grouped by the elements of the first two columns, Manager and Title. Notice that there is only one entry per manager name and one entry for each job title under that manager name. This organizes the data rows more cleanly in the table.



### Applying Subtotals

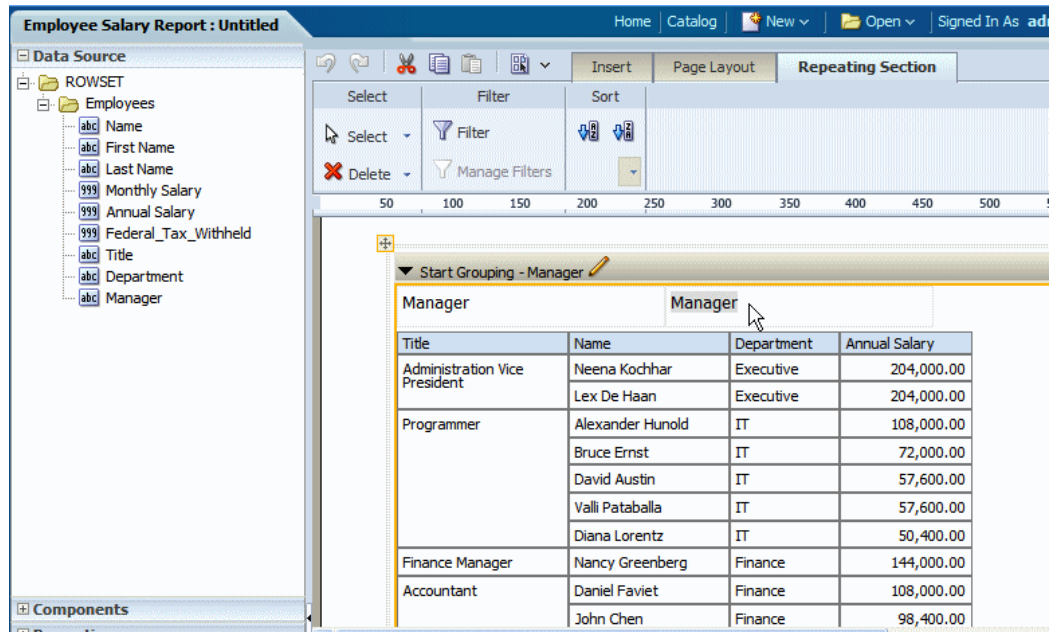
To further enhance your table, you can add a subtotal row to display for each grouped occurrence of the element. The figure below shows the same table with the **Subtotals** box checked. Notice that for each manager a subtotal row has been inserted.



### Example: Group Above

In the following example, the table data has been grouped by Manager. Notice that in the design pane, the Data Table component has been replaced with a Repeating Element component that contains the data table. The Manager element is inserted above the

table with a label.



The label is a text item. Edit the text by double-clicking the item to select it, then single-clicking to edit.

When you run the report, a separate table is created for each occurrence of the grouping element. In Interactive output mode, the grouping element displayed at the top of the table is displayed as a filter. Choose the value you wish to view from the list as shown in the following figure:

**Employee Salary Report** Home Catalog New Open

Department: All Employee: All Apply

**Employee by Manager**

Manager: Adam Fripp

Title	Name	Department	Annual Salary
Stock Clerk	Laura Bissot	Shipping	39,600.00
	Mozhe Atkinson	Shipping	33,600.00
	James Marlow	Shipping	30,000.00
	TJ Olson	Shipping	25,200.00
Shipping Clerk	Nandita Sarchand	Shipping	50,400.00
	Alexis Bull	Shipping	49,200.00
	Julia Dellinger	Shipping	40,800.00
	Anthony Cabrio	Shipping	36,000.00
			304,800.00

## About the Column Tab

The Column tab is enabled when you select a specific column in a table.

Insert Page Layout **Column**

Select Font Alignment Data Formatting Grouping Formula Sort Conditional Formatting

Select Delete Font: Tahoma 8pt Alignment: None Data Formatting: None Grouping: No Grouping Formula: No Formula Sort: Highlight

Subtotals Running Total Manage Formats

It enables you to perform the following:

- Edit the font properties of the column including style, size, and color
- Edit the cell properties of the column including border weight, style, and color and background fill color
- Set the vertical and horizontal alignment of the column contents
- Apply formatting to the column data (options depend on the data type)
- Apply grouping
- Apply a running total (or other formula) to the data
- Apply sorting and sort precedence

- Apply conditional formatting to the column

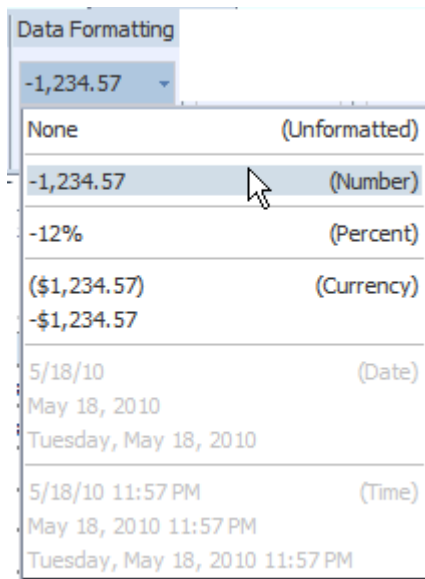
### About the Data Formatting Options for Columns

The options available from the Data Formatting region of the tab depend on the data type of the column selected. The tab provides common options to choose from.

#### Applying Formatting to Numeric Data Columns

If the column contains numeric data, the following formatting options are available:

- **Format** - select one of the common number formats from the list. The format is applied immediately to the table column. The formats are categorized by Number, Percent, and Currency as shown in the following figure.



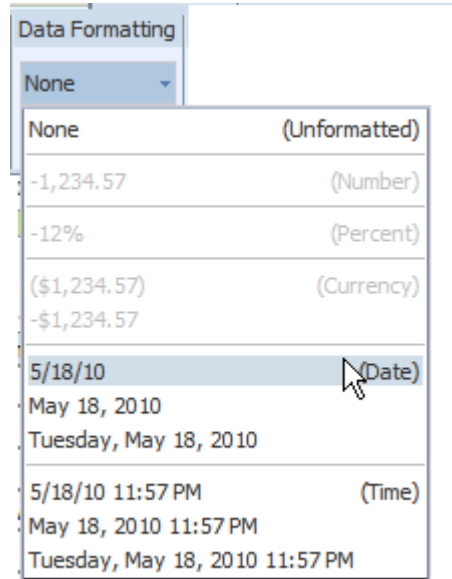
To apply a format not available from this list, see About Custom Formats.

- **Decimal position** - click the **Move Left** or **Move Right** to increase or decrease the decimal positions displayed.
- **Show/Hide Grouping Separator** - click this button to hide the grouping separator (for example, 1,234.00 will display as 1234.00). To show the grouping separator, click the button again.

#### Applying Formatting to Date Type Data Columns

If the column contains dates, the following formatting options are available:

- **Format** - select one of the common date formats from the list. The format is applied immediately to the table column. The formats are categorized by Date and Time as shown in the following figure.



### About the Formula Option

The options available from the **Formula** region of the column tab depend on the data type of the column.

For more information about applying formulas, see Setting Predefined or Custom Formulas, page 3-80.

### About the Sort Option

To sort the data in a column, select the column, then under the **Sort** group click **Ascending Order** or **Descending Order**.

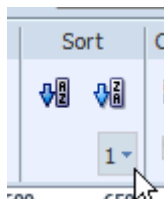
To sort by more than one column, select the column, the sort order, and then assign a **Priority** to each column. The priority list is a list of values beneath the sort order commands.

For example, in the following employee salary table, assume you want to sort ascending first by Title then sort descending by Annual Salary:

Name	Hire Date	Title	Annual Salary
Neena Kochhar	9/21/89	Administration Vice President	\$204,000.00
Lex De Haan	1/13/93	Administration Vice President	\$204,000.00
Alexander Hunold	1/3/90	Programmer	\$108,000.00
Bruce Ernst	5/21/91	Programmer	\$72,000.00
David Austin	6/25/97	Programmer	\$57,600.00
Valli Pataballa	2/5/98	Programmer	\$57,600.00
Diana Lorentz	2/7/99	Programmer	\$50,400.00
Nancy Greenberg	8/17/94	Finance Manager	\$144,000.00
Daniel Faviet	8/16/94	Accountant	\$108,000.00
John Chen	9/28/97	Accountant	\$98,400.00
			7,411,200.00

To apply the sort order to this table:

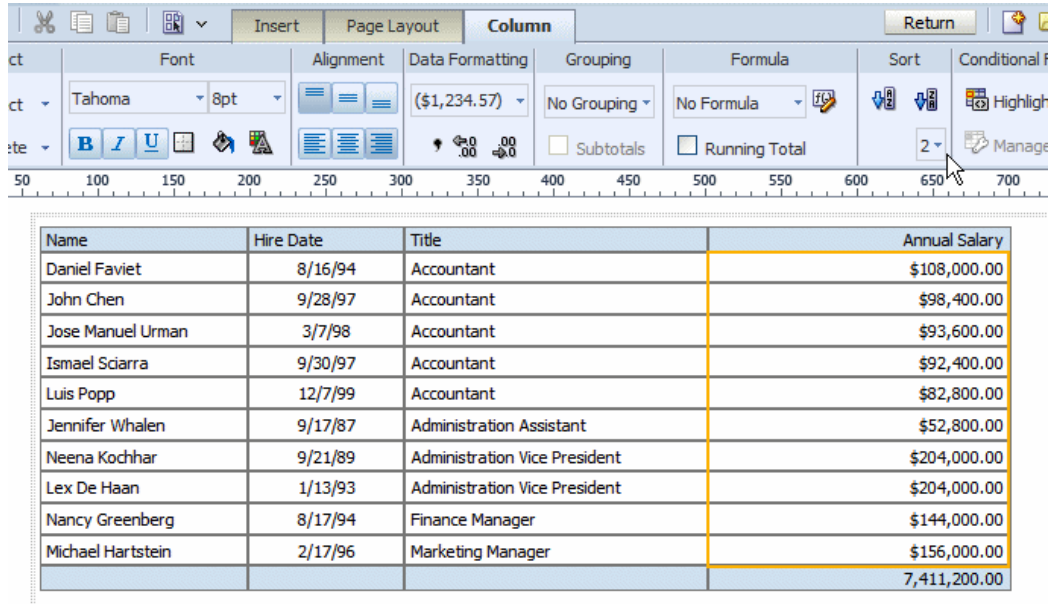
1. Select the Title column.
2. On the **Column** tab, under **Sort**, click the **Ascending Order** button.
3. From the **Priority** list, select 1.



4. Next select the Annual Salary column.
5. On the **Column** tab, under **Sort**, click the **Descending Order** button.
6. From the **Priority** list, select 2.

The sorted table is shown in the following figure:





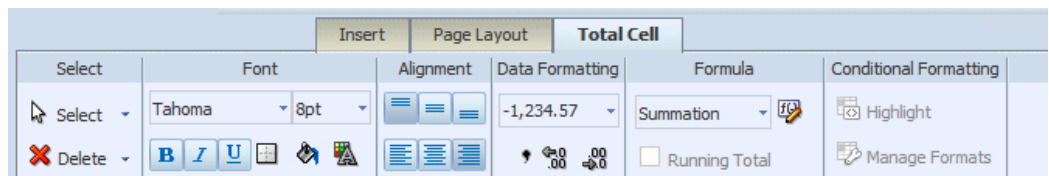
## Removing a Sort Order

To remove a sort order applied to a column:

1. Select the column.
2. From the **Sort** region on the **Column** tab, click the appropriate button of the sort order that has been applied. For example, to deselect the ascending order, click the **Ascending Order** button to undo the sort.

## About the Total Cell Tab

The Layout Editor automatically inserts a grand total row when you insert a data table to your layout. As shown in the section on grouping, you can also insert subtotal rows within your table based on a grouping element. To edit the attributes of the cells in a grand total or subtotal row, select the cell and use the options in the Total Cell tab shown in the following figure:



The Total Cell tab enables you to perform the following:

- Edit the font properties of the total cell

- Edit the cell properties of the total cell including border weight, style, and color and background fill color
- Set the vertical and horizontal alignment of the table header
- Apply formatting to the cell data
- Apply a formula to the cell
- Apply conditional formatting to the cell

### Applying Data Formatting to a Total Cell

See About the Data Formatting Options for Columns, page 3-48 under the Column tab section.

### Applying a Formula

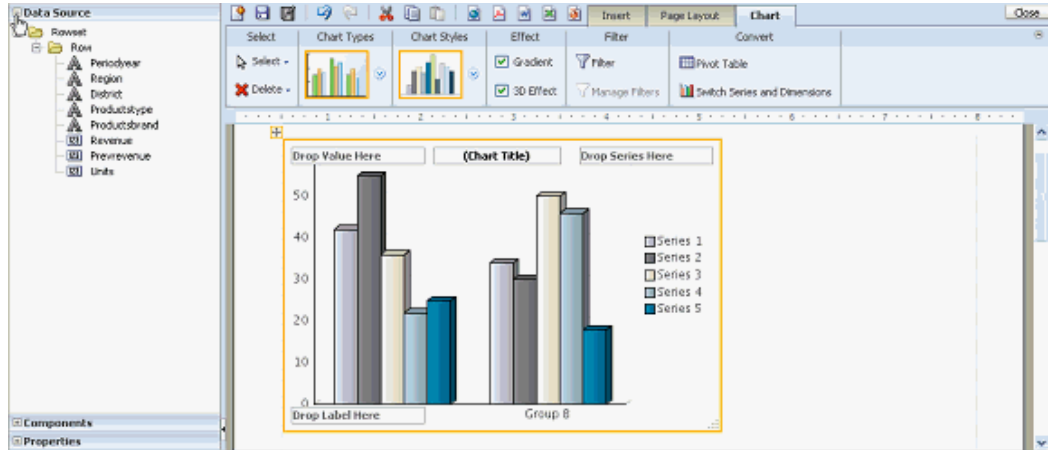
By default, the formula applied to a Total Cell within a numeric column is a sum of the column items. The Formula option enables you to apply a different formula.

Not all options available from the **Formula** region of the column tab are applicable to a Total Cell.

For more information about applying formulas, see Setting Predefined or Custom Formulas, page 3-80.

## About Charts

The layout editor supports a variety of chart types and styles to graphically present data in your layout. The following figure shows side-by-side vertical bar and pie charts in the layout editor.



Once inserted, you can edit the chart properties using the dynamic toolbars or the Properties pane. The Properties pane extends the options from the Chart tab and enables you to enter very specific custom settings for the following:

- Chart Effect
- Chart Legend
- Chart Plot Area
- Chart Title
- Chart Label

**Note:** The Chart Label properties Title Font, Title Horizontal Align, Title Text, and Title Visible apply to Scatter and Bubble chart types only.

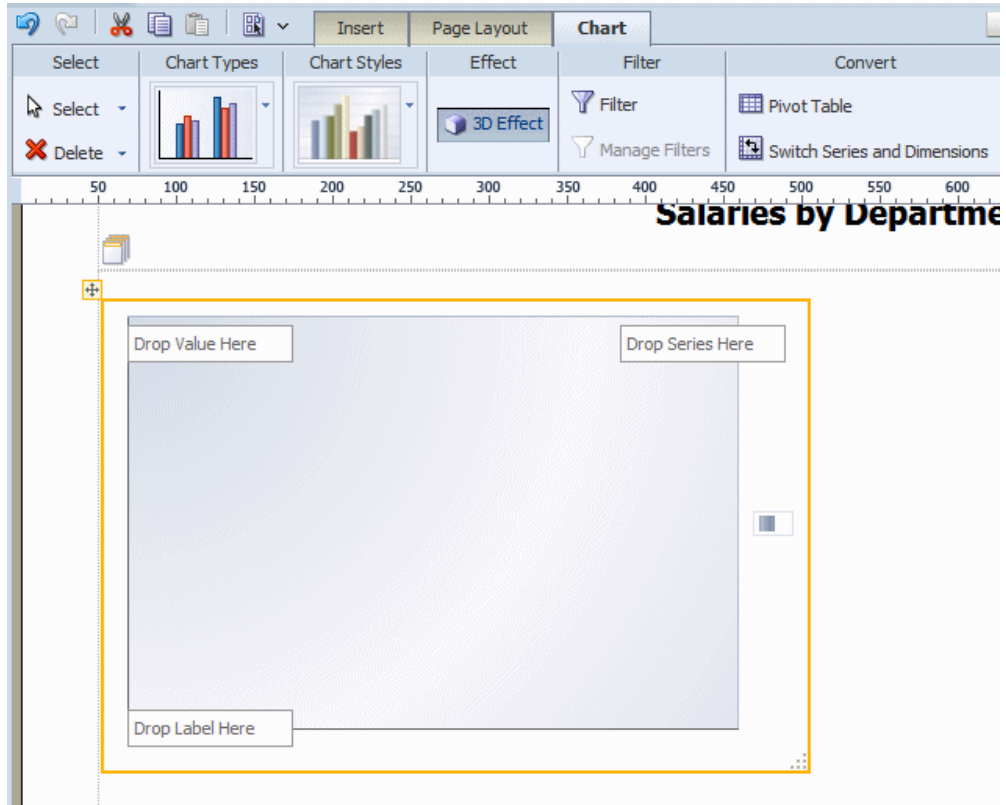
- Chart Values

**Note:** Some font effects such as underline, italic, and bold may not render in PDF output.

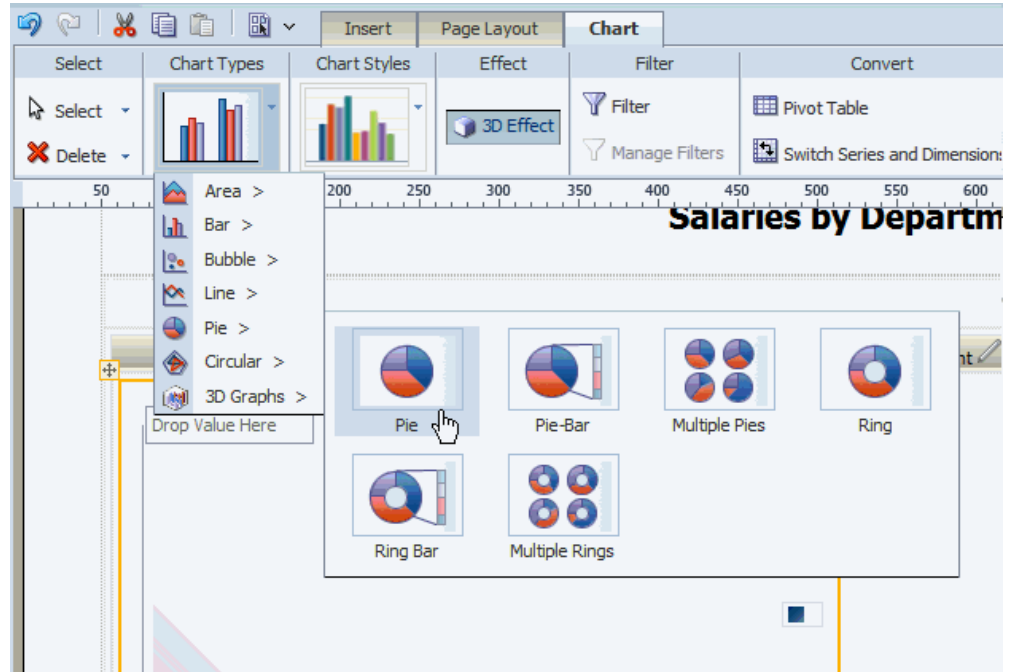
## Inserting a Chart

1. From the **Insert** menu, select and drag the Chart component to the layout.

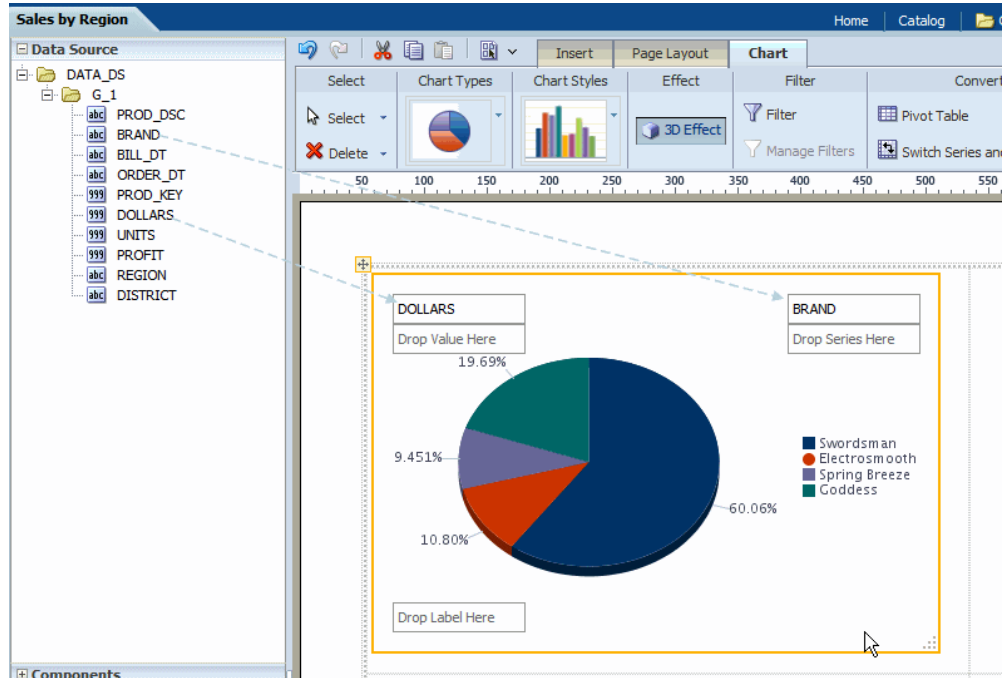
By default an empty vertical bar chart is inserted and the **Chart** dynamic tab is displayed, as shown in the following figure:



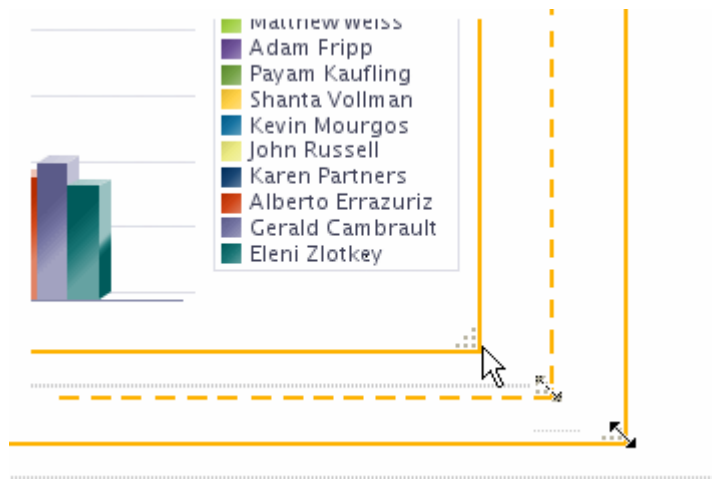
2. To change the chart type, click the Chart Type list to select a different type. In the following figure the chart type is changed to Pie.



3. Select and drag the data fields from the Data Source pane to the appropriate areas in the chart. The chart will immediately update with the preview data.



- To resize the chart, drag and drop the resize handler on the lower right corner of the chart, as shown in the following figure:



### About the Chart Tab

The Chart tab enables you to perform the following:

- select a different Chart Type
- apply a different Chart Style

- enable 3-D effects
- filter the data that is displayed in the chart
- manage multiple filters
- convert the chart to a pivot table or switch the series and dimensions values

### **Applying and Managing Filters**

See About Filters, page 3-36 for information on how to apply and manage filters.

### **Converting a Chart to a Pivot Table**

To convert a chart to a pivot table:

1. Select the chart.
2. In the **Convert** group, click **Pivot Table**.

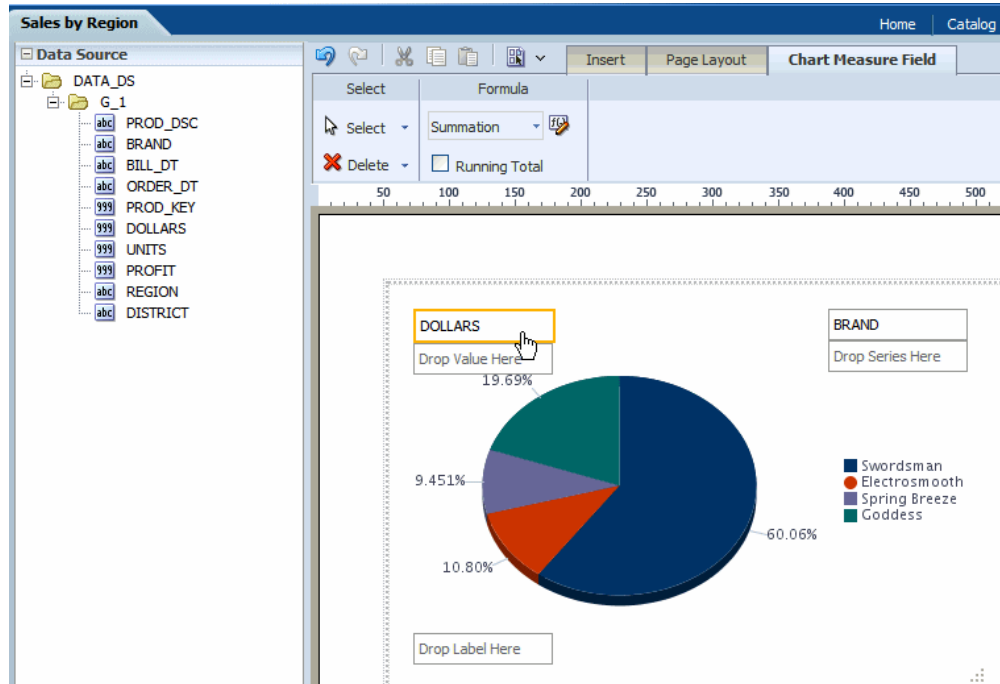
The layout editor will convert the label, series, and value elements of the chart into the appropriate rows, columns, and data elements of a pivot table.

### **Changing the Formula Applied to a Chart Measure Field**

By default, the chart displays a sum of the values of the chart measure. You can change the formula applied to a chart measure field by selecting an option from the **Chart Measure Field** tab.

#### **Change the Formula Using the Tab**

1. Select the measure field in the chart. This displays the **Chart Measure Field** tab as shown in the following figure:



2. Select from the following options available from the **Formula** list:
  - Count
  - Sum
  - Running Total

## Sorting a Chart Field

To sort a field in your chart:

1. Select the field to display the **Chart Field** tab.
2. On the **Chart Field** tab select **Sort Ascending** or **Sort Descending**.
3. To sort by multiple fields, apply a **Priority** to each sort field to apply the sort in the desired order.

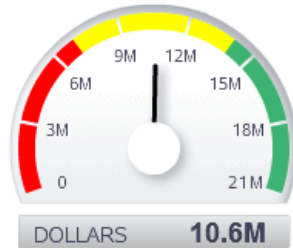
## About Gauge Charts

A gauge chart is a useful way to illustrate progress or goals. For example, the following figure shows a report with three gauges to indicate the status of regional sales goals:



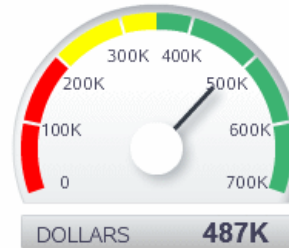
## Regional Sales

South



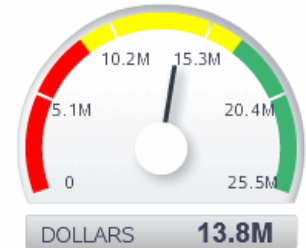
**Southern Region Goal:**  
**\$15,000,000**

Web Direct



**Web Direct Region Goal:**  
**\$400,000**

East

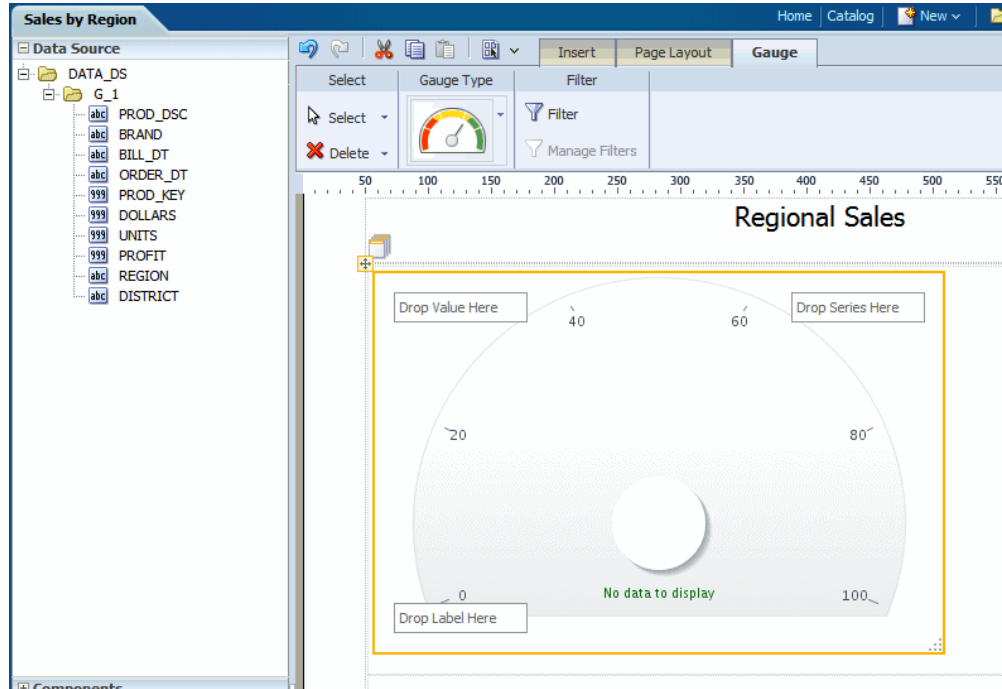


**Eastern Region Goal:**  
**\$18,000,000**

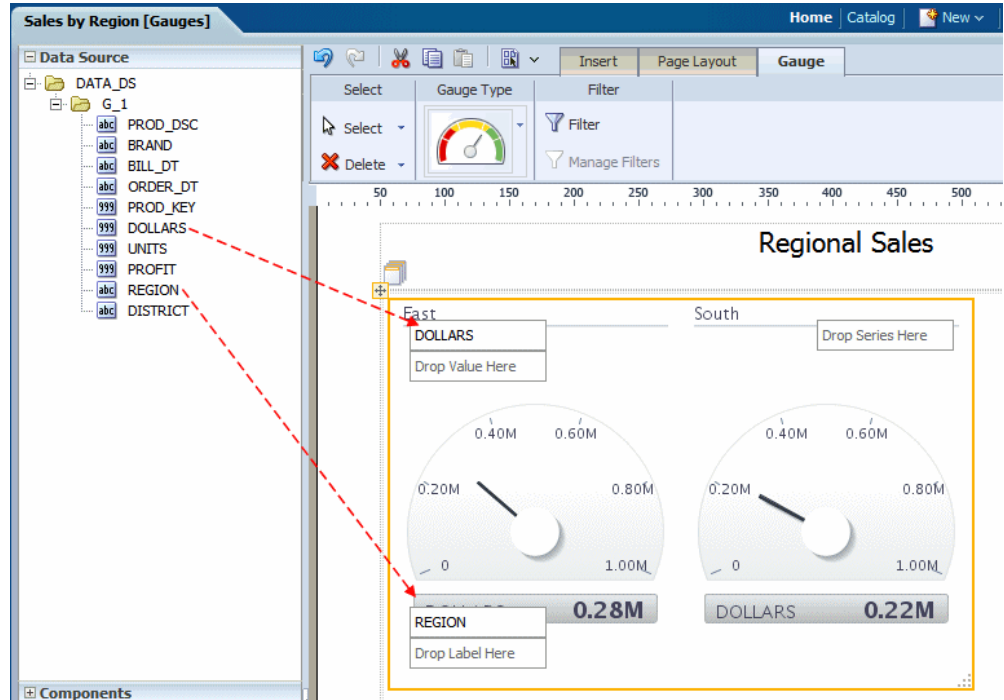
To insert a gauge chart in your layout:

### Inserting a Gauge Chart

1. From the **Insert** menu, select and drag the **Gauge** component to the layout. This inserts an empty gauge chart.



2. Select and drag the data fields from the Data Source pane to the Label, Value, and Series areas of the chart. The chart will immediately update with the preview data. In the example figure, drag REGION to the Label area and DOLLARS to the Value area:



Note the following:

- A separate gauge is created for each occurrence of the Label (that is, each REGION). One set of properties will apply to each occurrence. To apply different properties to each gauge, see [Creating Filtered Gauges](#).
- By default, the Value field is a sum. You can change the expression applied to the value field. See [Changing the Formula Applied to a Chart Measure Field](#), page 3-57.
- You can apply a sort to the other gauge chart fields.

## Setting the Properties for a Gauge Chart

Use the Properties Pane to set detailed options for your gauge chart.

## Applying and Managing Filters

See [About Filters](#), page 3-36 for information on how to apply and manage filters.

## About Pivot Tables

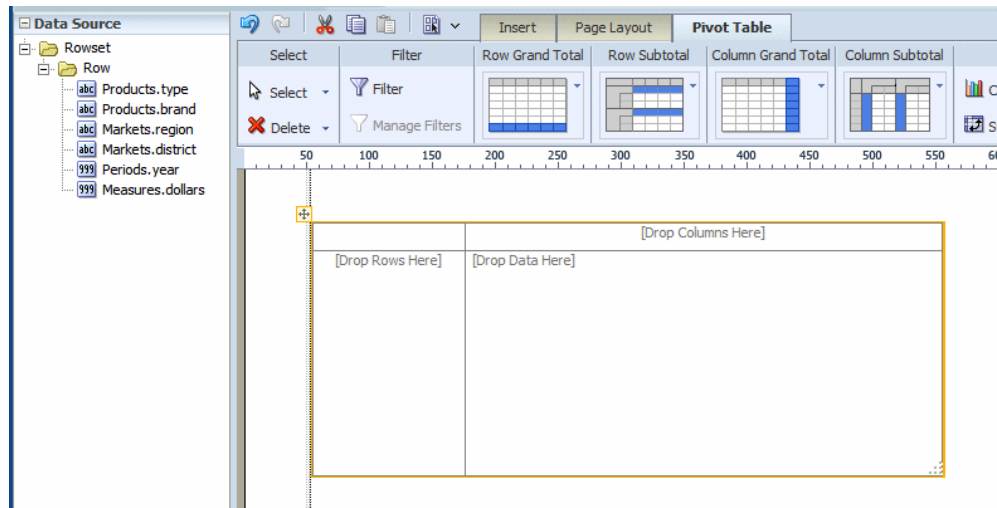
The pivot table provides views of multidimensional data in tabular form. It supports multiple measures and dimensions and subtotals at all levels. The following figure shows a pivot table:

		Magicolor		Subtotal		Total	
		PrevRevenue	Revenue	PrevRevenue	Revenue	PrevRevenue	Revenue
		CENTRAL REGION	CHICAGO DISTRICT	5,587.00	3,741.00	5,587.00	3,741.00
CINCINNATI DISTRICT	9,875.00		38,054.00	9,875.00	38,054.00	9,875.00	38,054.00
DETROIT DISTRICT	5,706.00		2,149.00	5,706.00	2,149.00	5,706.00	2,149.00
KANSAS CITY DISTRICT	9,467.00		55,519.00	9,467.00	55,519.00	9,467.00	55,519.00
MINNEAPOLIS DISTRICT	140.00		231.00	140.00	231.00	140.00	231.00
<b>Subtotal</b>		30,775.00	99,694.00	30,775.00	99,694.00	30,775.00	99,694.00
<b>Total</b>		30,775.00	99,694.00	30,775.00	99,694.00	30,775.00	99,694.00

## Inserting a Pivot Table

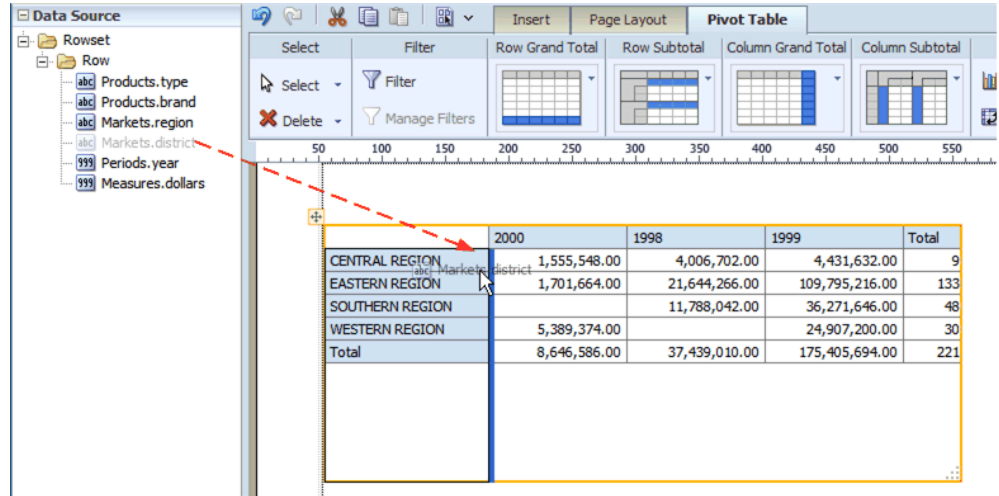
To insert a pivot table:

1. From the **Insert** tab, select and drag the **Pivot Table** component to the layout. The following figure shows the empty pivot table structure:

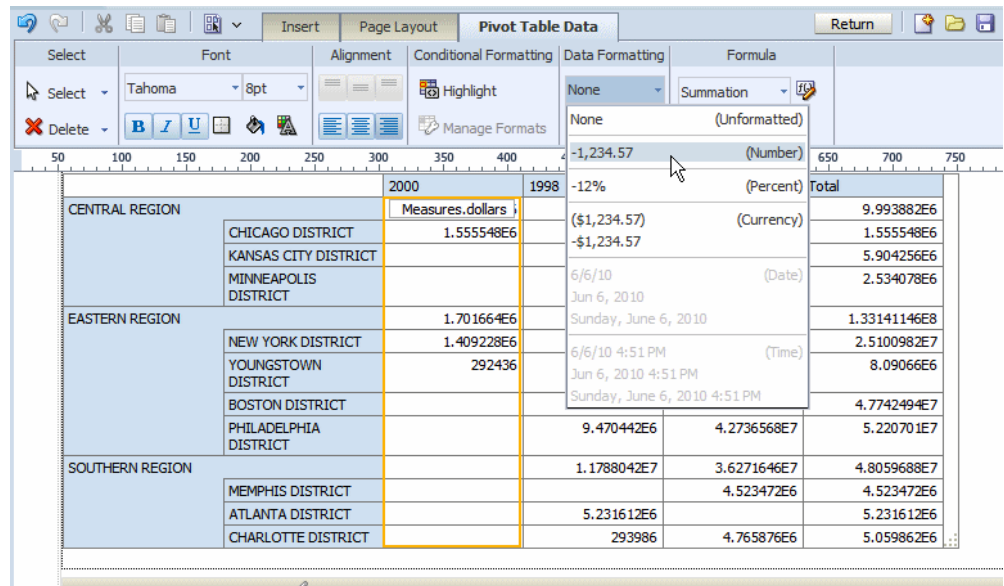


2. Drag and drop data fields from the Data Source pane to the row, column, and data positions.

Drag multiple fields to the pivot table and place them precisely to structure your pivot table.



- By default the pivot table is inserted with no data formatting applied. To apply a format to your data, click the first column of data to enable the Pivot Table Data toolbar. On the Data Formatting group, select the appropriate format as shown in the following figure:



- Optionally resize the pivot table by clicking and dragging the handler in the lower right corner of the pivot table.

	1998	1999	2000	Total
CENTRAL REGION	4,006,702.00	4,431,632.00	1,555,548.00	9,993,882.00
CHICAGO DISTRICT			1,555,548.00	1,555,548.00
KANSAS CITY DISTRICT	4,006,702.00	1,897,554.00		5,904,256.00
MINNEAPOLIS DISTRICT		2,534,078.00		2,534,078.00
EASTERN REGION	21,644,266.00	109,795,216.00	1,701,664.00	133,141,146.00
NEW YORK DISTRICT		23,691,754.00	1,409,228.00	25,100,982.00
YOUNGSTOWN DISTRICT	7,798,224.00		292,436.00	8,090,660.00
BOSTON DISTRICT	4,375,600.00	43,366,894.00		47,742,494.00
PHILADELPHIA DISTRICT	9,470,442.00	42,736,568.00		52,207,010.00
SOUTHERN REGION	11,788,042.00	36,271,646.00		48,059,688.00
MEMPHIS DISTRICT		4,523,472.00		4,523,472.00
ATLANTA DISTRICT	5,231,612.00			5,231,612.00
CHARLOTTE DISTRICT	293,986.00	4,765,876.00		5,059,862.00

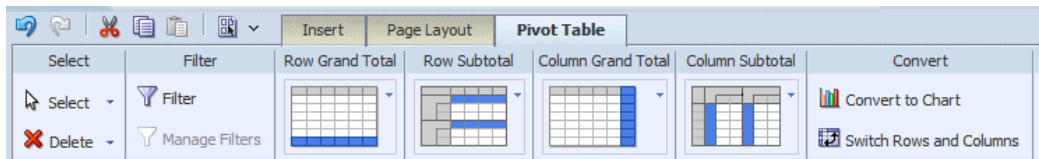
## Customizing a Pivot Table Menu

After you insert a pivot table customize the appearance and layout using the following dynamic tabs:

- Pivot Table tab
- Pivot Table Header tab
- Pivot Table Data tab

## About the Pivot Table Tab

The following figure shows the Pivot Table tab:



## Applying Filters

See About Filters, page 3-36 for a description of the **Filter** and **Manage Filters** features.

## Customizing the Display of Totals

The Pivot Table tab enables you to quickly customize the display of grand total and

subtotal rows.

By default, the layout editor inserts the pivot table with the total and subtotal displays as shown in the tab:

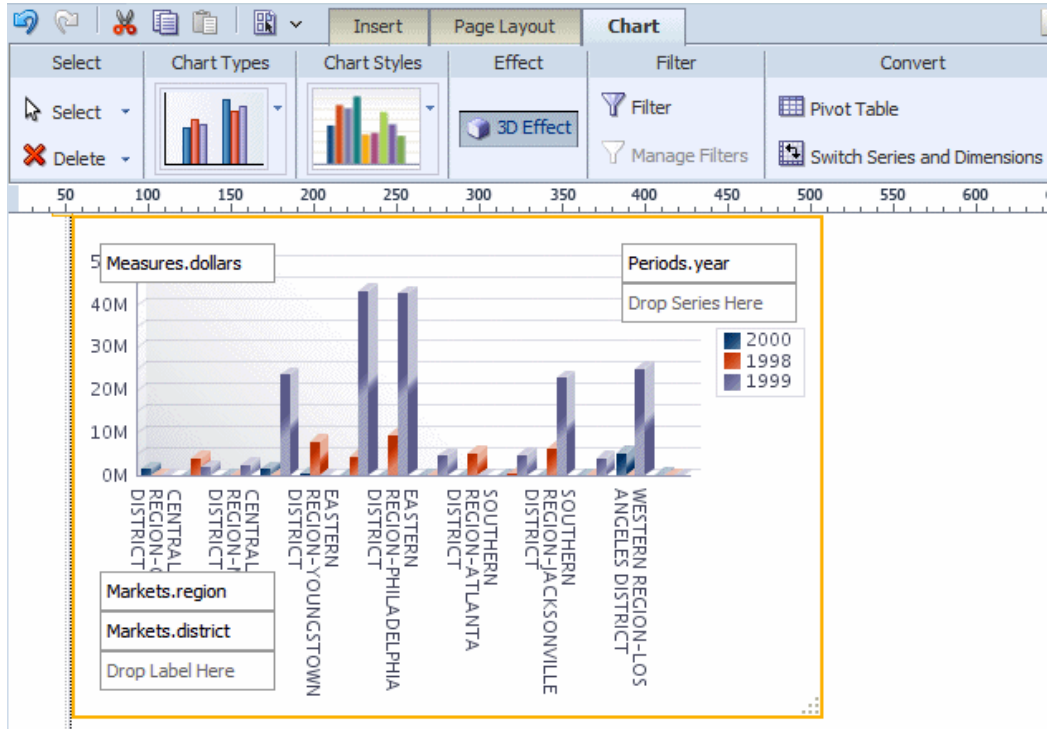
- **Row Grand Total** - inserted at bottom of table
- **Row Subtotal** - inserted at top of each subgroup, with no row header
- **Column Grand Total** - inserted at the far right
- **Column Subtotal** - inserted to the left of each column subgrouping, with no header

Change the positioning and display of totals and subtotals by clicking the appropriate group in the tab and selecting the desired layout pattern from the menu.

### **Converting a Pivot Table to a Chart**

The **Convert Pivot Table to a Chart** command converts the pivot table to a default vertical bar chart. After conversion, customize the table as described in *About Charts*, page 3-52.

The following figure shows the pivot table created in the preceding step converted to a vertical bar chart:



### Switching Rows and Columns

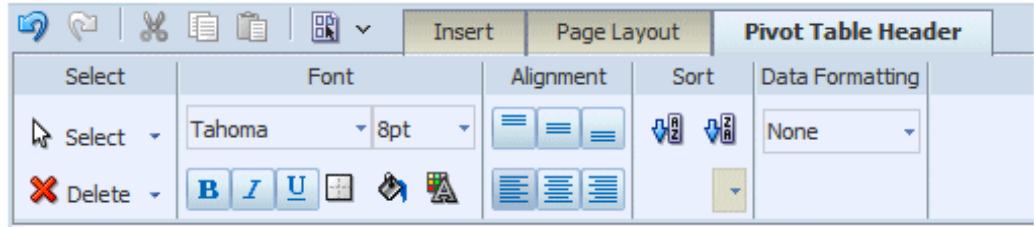
Use the Switch Rows and Columns command to see a different view of the same data. The following figure shows the pivot table created in the previous step with rows and columns switched:

	CENTRAL REGION			EASTERN REGION	
		CHICAGO DISTRICT	KANSAS CITY DISTRICT	MINNEAPOLIS DISTRICT	NEV DIS
2000	1,555,548.00	1,555,548.00			1,701,664.00
1998	4,006,702.00		4,006,702.00		21,644,266.00
1999	4,431,632.00		1,897,554.00	2,534,078.00	109,795,216.00
Total	9,993,882.00	1,555,548.00	5,904,256.00	2,534,078.00	133,141,146.00

### Customizing the Pivot Table Headers

The Pivot Table Header tab is shown in the following figure:



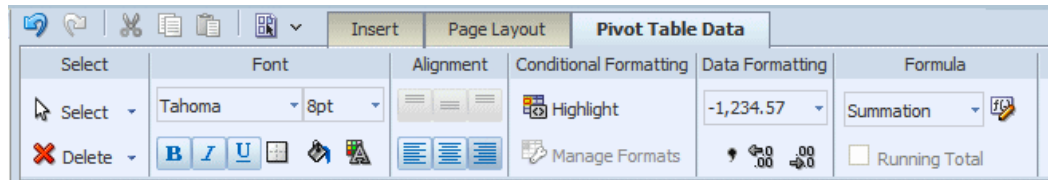


Select the column or row header of the pivot table and use the Pivot Table Header tab to perform the following:

- customize the fonts, colors, alignment and other display features of the header
- apply a sort order (for more information see About the Sort Option, page 3-49)
- apply data formatting (if the data type is number or date)

## Customizing the Pivot Table Data

The **Pivot Table Data** tab is shown in the following figure:



Select the data area of the pivot table and use the Pivot Table Data tab to perform the following:

**Note:** The commands in the Pivot Table Data tab are the same as the corresponding commands in the table Column tab. See the references for more information on their use.

- customize the fonts, colors, alignment and other display features of the data
- apply conditional formatting to the data for more information (see About the Conditional Formatting, page 3-38)
- apply data formatting (see About the Data Formatting Options for Columns, page 3-48)
- apply a formula (see Applying a Formula, page 3-52)

## About Text Items

The text item component allows you to enter free-form text in the layout.

To create a text item component:

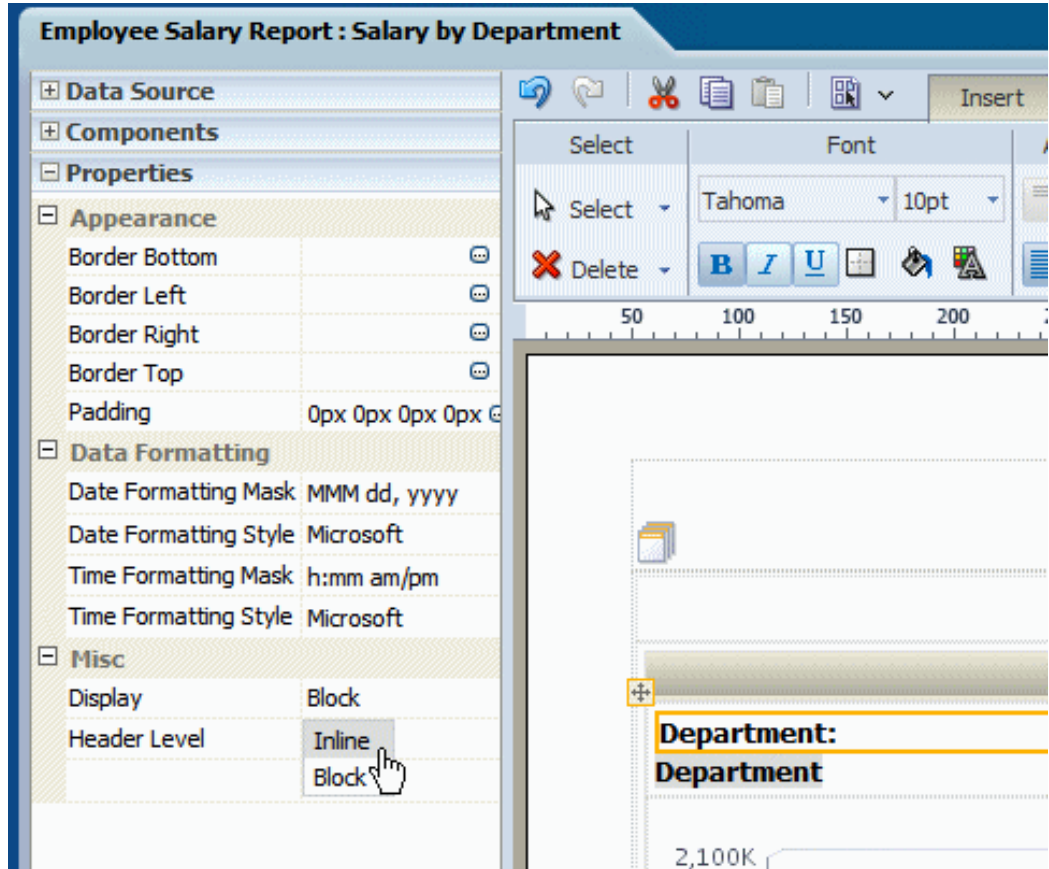
1. Drag and drop the text item component to the layout.
2. Double-click the text to enter text editor mode. Select parts of the text to apply different formatting to different parts.

## Displaying a Data Field Side by Side with a Text Item

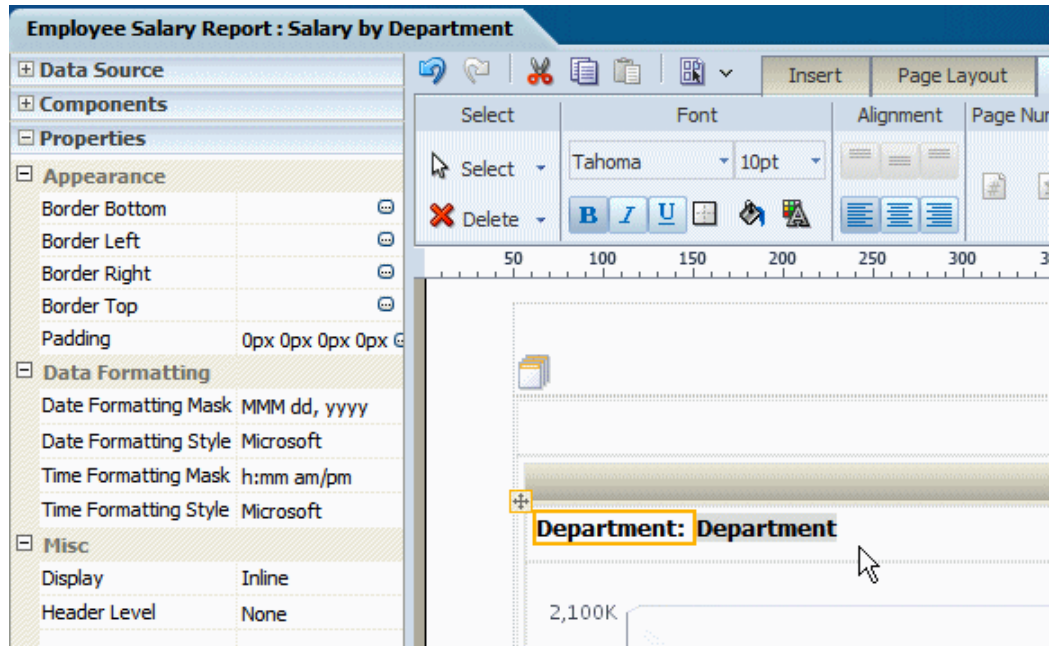
By default, the text item always spawns a complete paragraph. Inserting a data field next to the text field will place the data field beneath the text field as shown in the following figure:



To display the data field inline with the text item, set the Display property to **Inline** in the **Properties** pane:

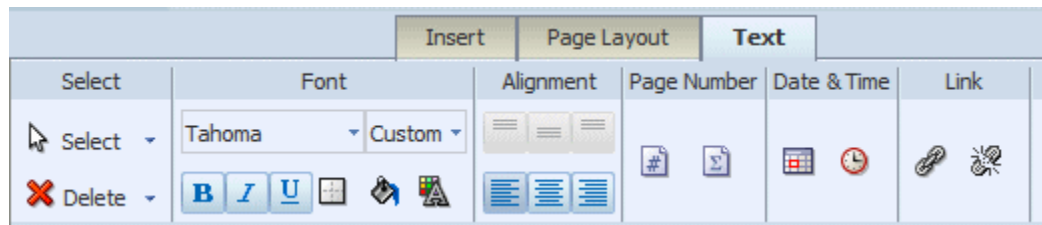


This setting enables the positioning of text items and data fields into a single line as shown in the following figure:



## About the Text Toolbar

The Text Tab is shown in the following figure:



The Text tab enables you to perform the following:

- Set the font properties
- Set alignment of the text in the grid cell
- Insert predefined text items: page number, date, and time
- Insert a hyperlink

## Editing Font Properties

Use the Font group of commands to set the following:



- select a font style

- select a font size
- apply emphasis (bold, italic, or underline)
- insert a border around the text item
- apply a background color
- apply a font color

### Inserting Page Numbers

Drag and drop the page number component to the design area.

To create the following Page # of *N* construction

Page  of 

perform the following:

1. From the **Insert** tab drag and drop a **Text Item** to the design area where you want the page numbers to display.
2. Double-click the inserted text to select the text item for editing. Type "Page ".
3. From the **Text** dynamic tab, drag and drop the **Page Number** component.
4. Enter a space, and type "of ".
5. From the **Text** dynamic tab, drag and drop the **Page Total** component.

### Inserting the Date and Time

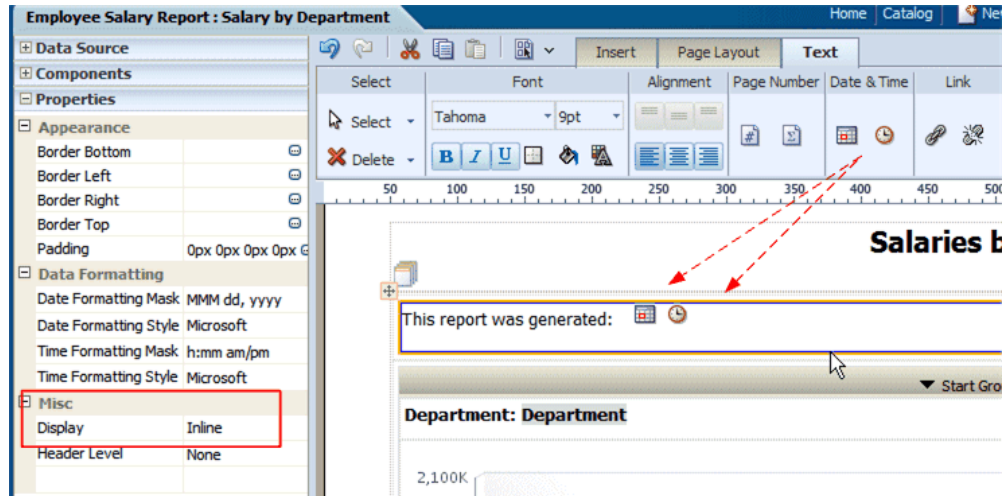
To insert the date and time in your report:

1. From the **Insert** tab drag and drop a **Text Item** to the design area where you want the date and time to display.
2. Double-click the inserted text to select the text item for editing.
3. Click the Date icon to insert the date icon in the text item. Click the Time icon to insert the time icon in the text item.

**Note:** To display the items side-by-side ensure to set the Text Item

property to "Inline".

This is shown in the following figure:



When this report is viewed, the date and time will be displayed according to the server time zone if viewed online, or for scheduled reports, the time zone selected for the schedule job.



### Inserting a Hyperlink

To insert a hyperlink in your report:

1. From the **Insert** tab drag and drop a **Text Item** to the design area where you want the date and time to display.

2. Double-click the inserted text to select the text item for editing. Enter the text which you want to convert to a link.
3. Select the text, then click the **Link** button.
4. In the dialog enter the URL.

## About Images

The image component enables you to include a graphic in the layout. BI Publisher supports the following methods for including an image:

- **Static image:** Upload a static image that is saved in the report file. An uploaded image file must be in one of the following graphic file formats: GIF, JPEG, PNG, or BMP. The image file cannot be larger than 500 KB.
- **Static URL:** Specify a static link to a URL where an image is stored.
- **Dynamic URL:** Include the image URL in an element of your data. The value of the element will be evaluated at runtime enabling dynamic insertion of images.

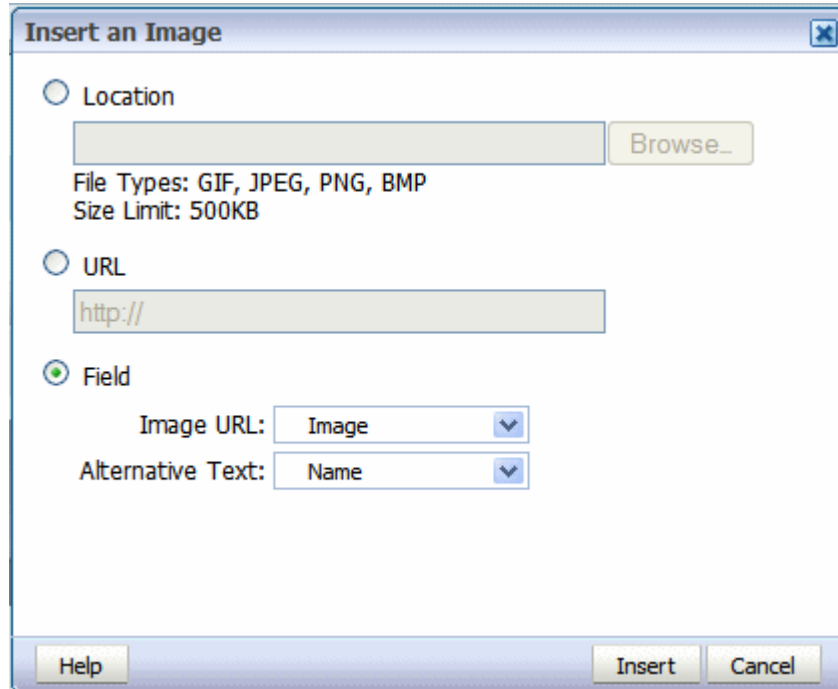
To insert an image:

1. Drag and drop the image component to the layout.
2. In the **Insert an Image** dialog, specify one of the following sources for the image:
  - **Location:** Click **Browse** to specify the file name and directory of the image on a local or mapped drive to upload the image.
  - **URL:** Enter the URL where the image is stored.
  - **Field:**

**Image URL:** Select the field from your data that contains a URL to an image.

**Alternative Text:** If your data includes a field that contains alternative text for the image, select that field to display alternative text when the report is viewed as HTML.

The following figure shows the **Insert an Image** dialog set up to retrieve an image URL dynamically from the "Image" data element. The value of the "Name" element will be used as alternative text.



3. Optionally resize the image in one of these ways:
  - Drag the right bottom corner of the image. To preserve the aspect ratio when resizing an image, press and hold the Shift key before starting to drag the corner.
  - Modify the width and height in the Properties pane.

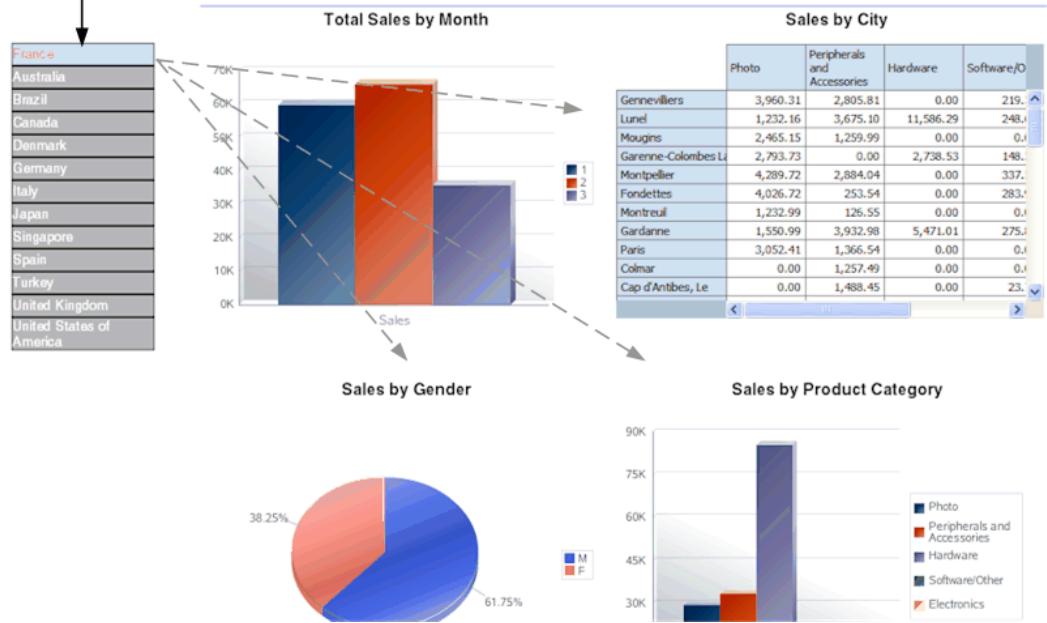
## About Lists

The list component displays all values of a data element in a vertical or horizontal list. When viewed in interactive mode, clicking an item in the list updates the results shown in the linked components of the report. The following example shows a report that displays multiple charts based on sales data. The list component displays each country for which there is sales data. The list enables the report consumer to quickly see results for each country in the list by clicking the entry in the list.



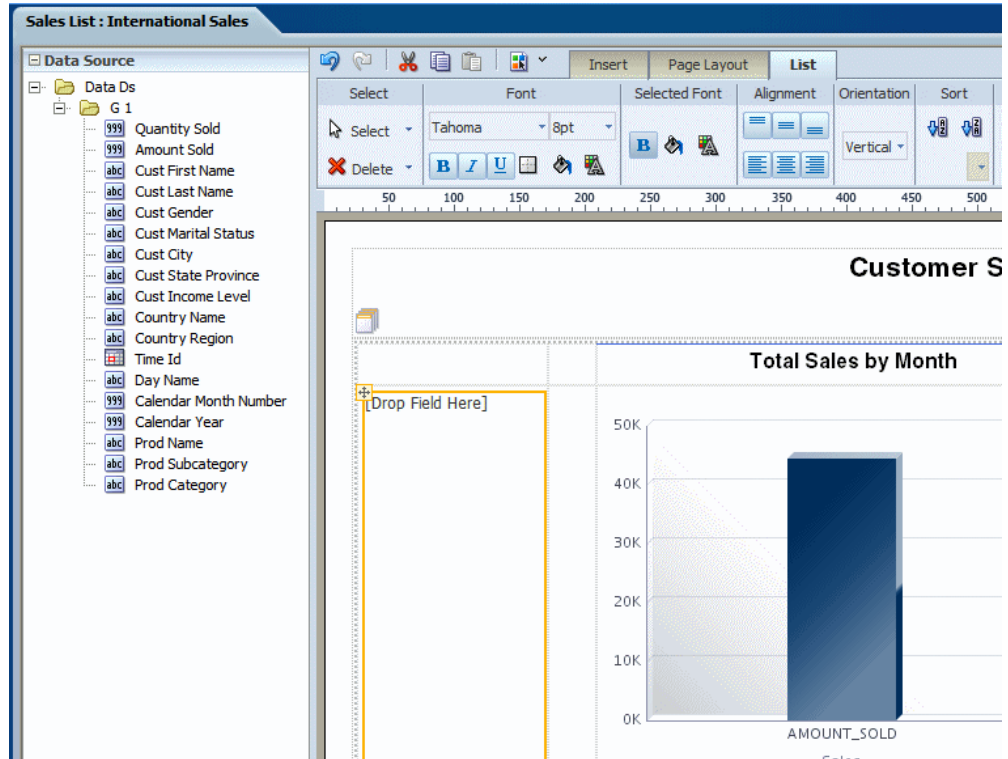
Click item in list to update results shown in other layout components

### Customer Sales Analysis



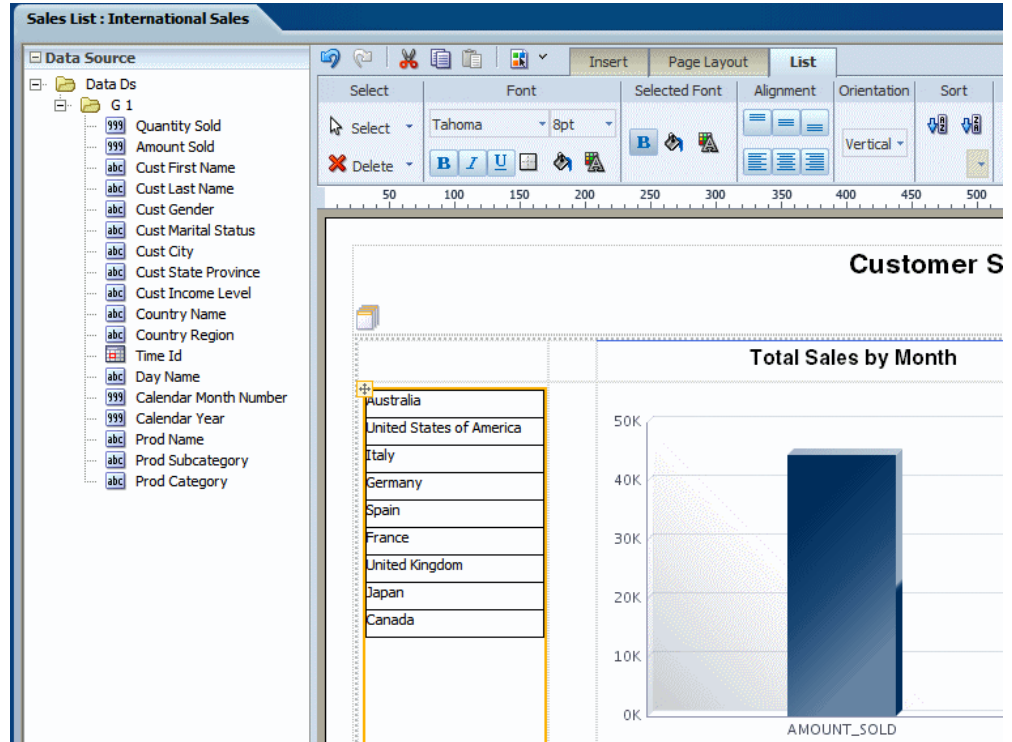
## Inserting a List

1. From the **Insert** tab, select and drag the **List** component to the design area.  
The following figure shows an inserted, empty list.



2. To create the list, select an element from the **Data Source** pane and drag it to the empty list in the layout.

The following figure shows the list component after dragging the element Country Name to it.

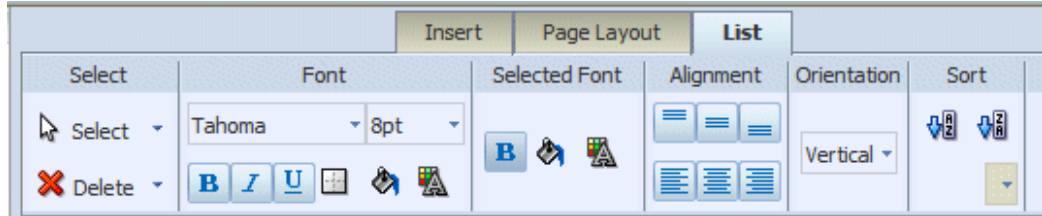


3. Customize the appearance of the list. See *Customizing a List*, page 3-77.
4. Configure linked components using the **Configure Events** command. By default, all other tables and charts in the layout are configured to filter their results based on the user selections in the list component. To change this default behavior, see *Event Configuration*, page 3-15.

## Customizing a List

Use the **List Tab** to:

- Edit the font size, style, and color
- Define borders for the list
- Set the background color
- Edit the font color and background color for the display of selected items
- Set the orientation of the list
- Specify the sort order



### Customizing the Font Style and the Selected Font Style Commands

The list on the left shows the default format of the list. The list on the right shows the **Selected Font** default format:

Australia
Brazil
Canada
Denmark
France
Germany
Italy
Japan
Singapore
Spain
Turkey
United Kingdom
United States of America

Default format of list with no items selected

France
Australia
Brazil
Canada
Denmark
Germany
Italy
Japan
Singapore
Spain
Turkey
United Kingdom
United States of America

Default format with item selected

Edit the font settings by selecting a font family from the list and adjusting the point size.

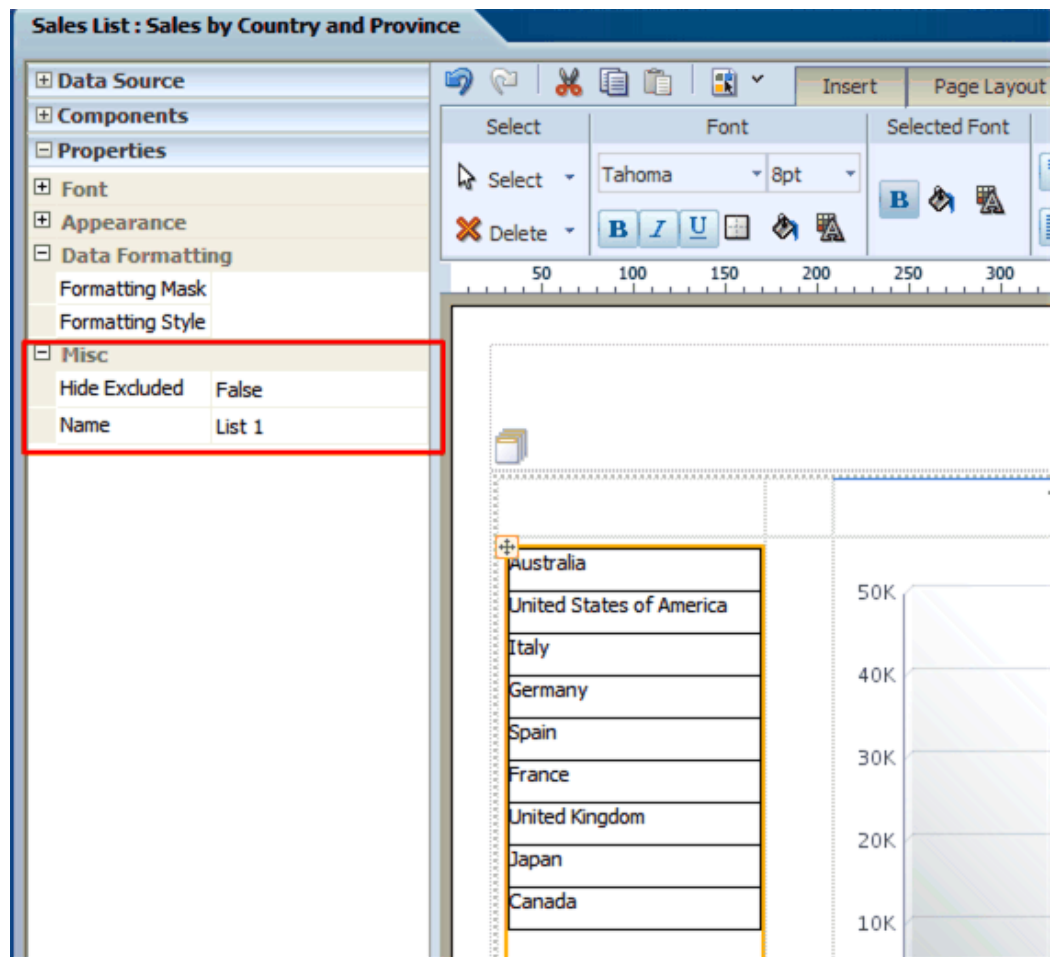
By default, the list displays with one point black gridlines. Click the **Set Border** to adjust the default borders of the list. Use the **Background Color** and **Font Color** commands to customize the colors.

The **Selected Font** commands edit the appearance of the item in a list when it is selected. By default, the selected element is moved to the top of the list, and the background is changed to light blue. You can edit the font weight, background color, and font color that are displayed for selected items.

## Customizing Behavior of Selected Items

By default, the selected items move to the top of the list and the nonselected items are "hidden" by a gray fill. You also have the option of not applying this behavior by setting the property "Hide Excluded".

This property is available from the **Properties** pane when the List component is selected. The **Hide Excluded** property is highlighted in the following figure:



The following figure shows the difference in the display depending on the setting of the property:

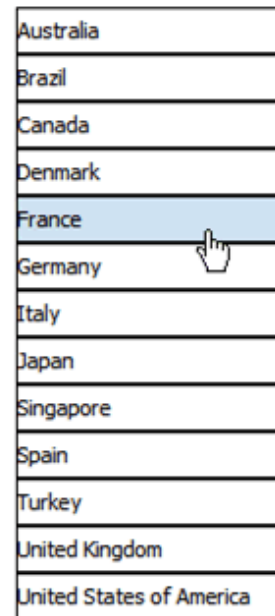
Display of selected item when "Hide Excluded" set to False



France
Australia
Brazil
Canada
Denmark
Germany
Italy
Japan
Singapore
Spain
Turkey
United Kingdom
United States of America

Selected item moved to top of list; other items grayed out

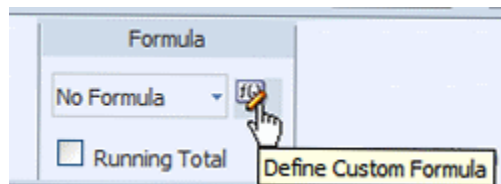
Display of selected item when "Hide Excluded" set to True



Australia
Brazil
Canada
Denmark
France
Germany
Italy
Japan
Singapore
Spain
Turkey
United Kingdom
United States of America

Selected item not moved; other items unchanged

## Setting Predefined or Custom Formulas



The Formula group of commands is available from the following tabs:

- Column tab
- Total Cell tab
- Chart Measure Field tab
- Pivot Table Data tab

Note that not all options are applicable to each component type.

## About the Predefined Formulas

The menu provides the following predefined formulas:

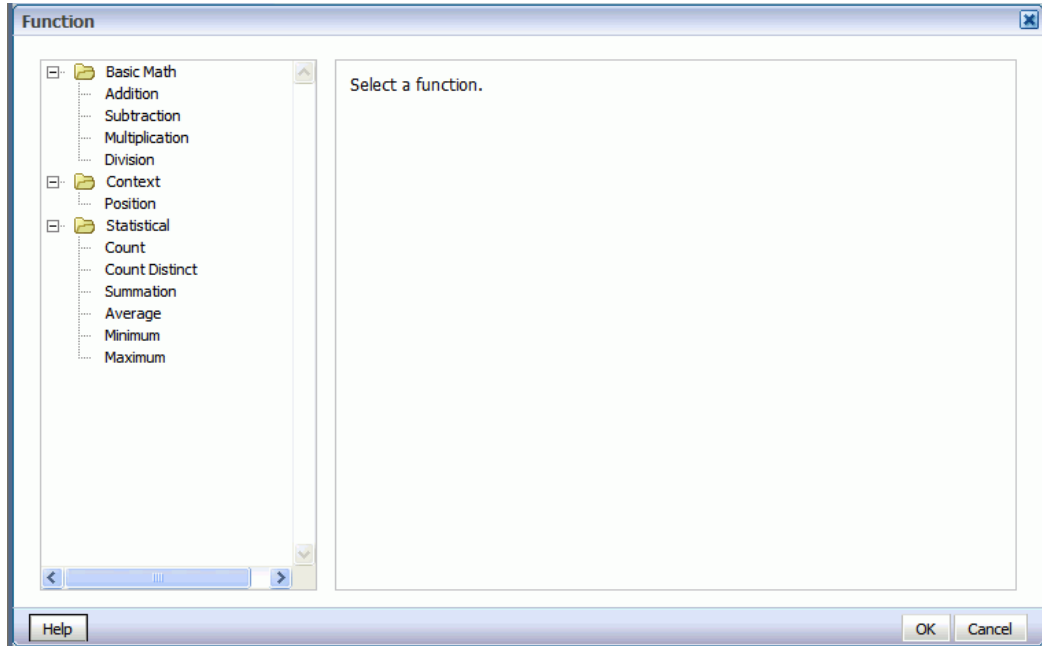
Formula	Description
No Formula	Removes any mathematical formula from a numeric column.
Blank Text	Removes all data and inserts blank text.
Count	Returns the count of the number of occurrences of the element in the current . group.
Count Distinct	Returns a count of the distinct values of an element in the current group.
Summation	Sums the values of the element in the current group.
Average	Displays the average of the values in the current group.
Maximum	Displays the highest value of all occurrences in the current group.
Minimum	Displays the lowest value of all occurrences in the current group.

For non-numeric data, only the following formula options are supported:

- Blank Text
- Count
- Count Distinct

## Applying a Custom Formula

Click **Define Custom Formula** to define your own formula for a component. The **Function** dialog enables you to define Basic Math, Context, and Statistical functions in your layout.

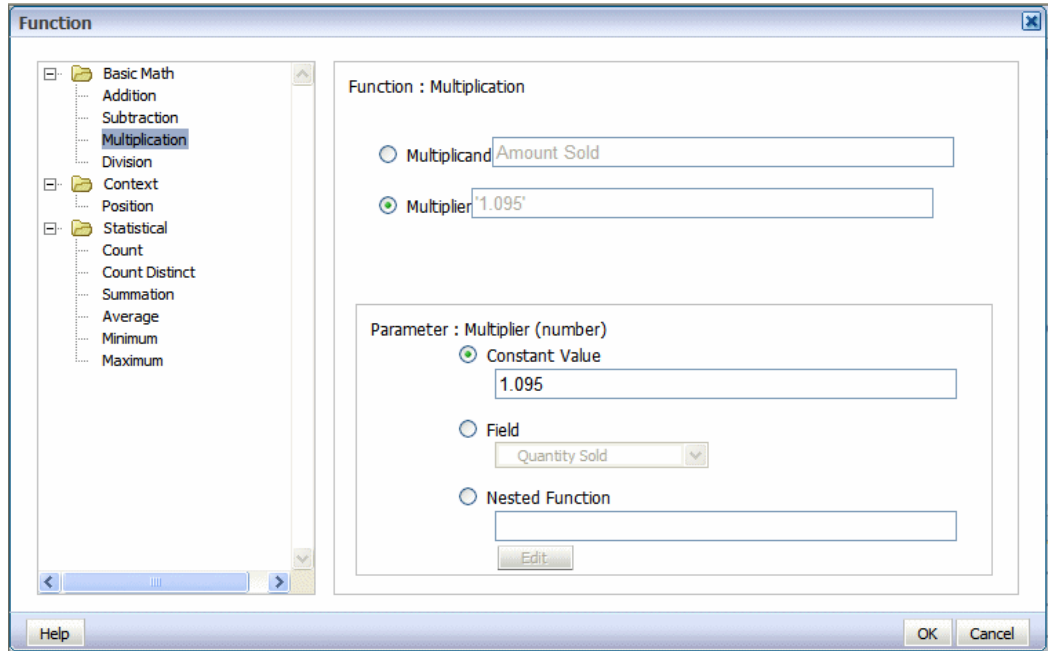


### About the Basic Math Functions

When you click one of the basic math functions you are prompted to define the appropriate parameters for the function. You can enter a constant value, select a field from the data, or create a nested function to supply the value.

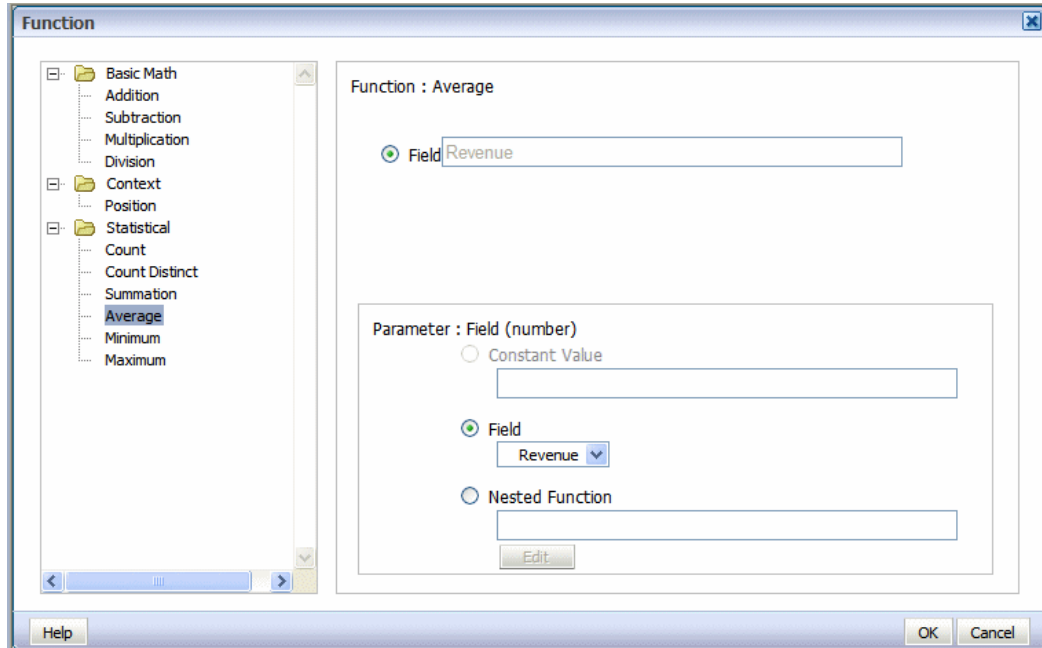
In the following example, clicking the Multiplication function displays prompts to enter the multiplicand and the multiplier. The example shows that the multiplicand is the value of the Amount Sold field. The multiplier is the constant value.





### About the Statistical Math Functions

When you click one of the statistical math functions you are prompted to define the appropriate parameter for the function. You can select a field from the data, or create a nested function to supply the values. In the following example, clicking the Average function displays prompts for you to specify the source of the values for which to calculate the average.



## Applying a Custom Formula: Examples

### Example 1: Subtraction

The following table shows data for Revenue and Cost for each Office:

Office	Revenue	Cost
Whalen	\$4,400.00	\$1,100.00
Hartstein	\$13,000.00	\$3,250.00
Fay	\$6,000.00	\$1,500.00
Raphaely	\$11,000.00	\$2,750.00
Khoo	\$3,100.00	\$775.00
Baida	\$2,900.00	\$725.00
Tobias	\$2,800.00	\$700.00
Himuro	\$2,600.00	\$650.00
Colmenares	\$2,500.00	\$625.00
Mavris	\$6,500.00	\$1,625.00

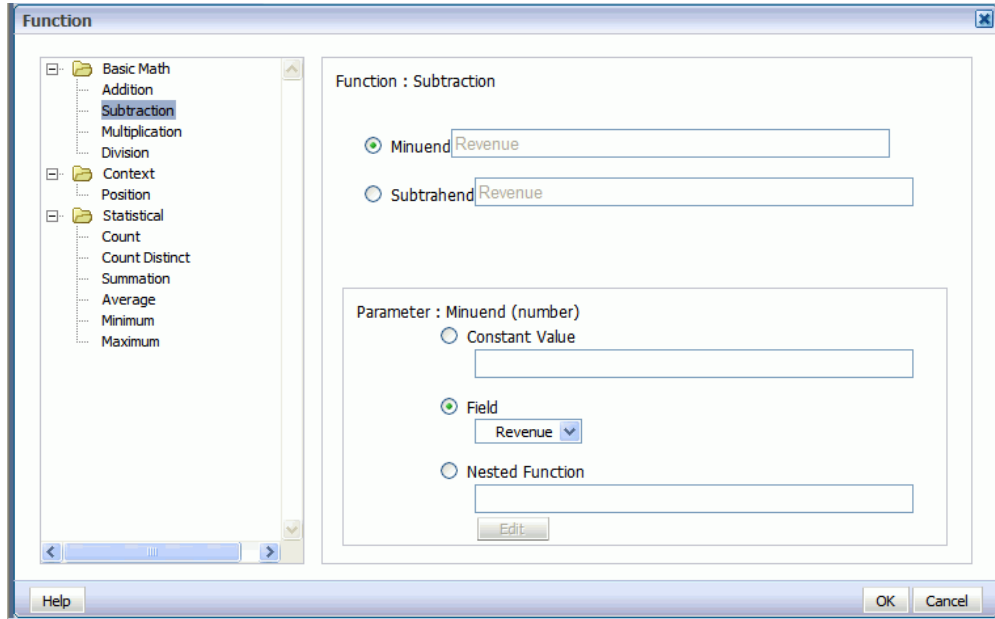
Using a custom formula, you can add a column to this table to calculate Profit (Revenue - Cost).

1. Add another numeric data column to the table. For example, drag another instance

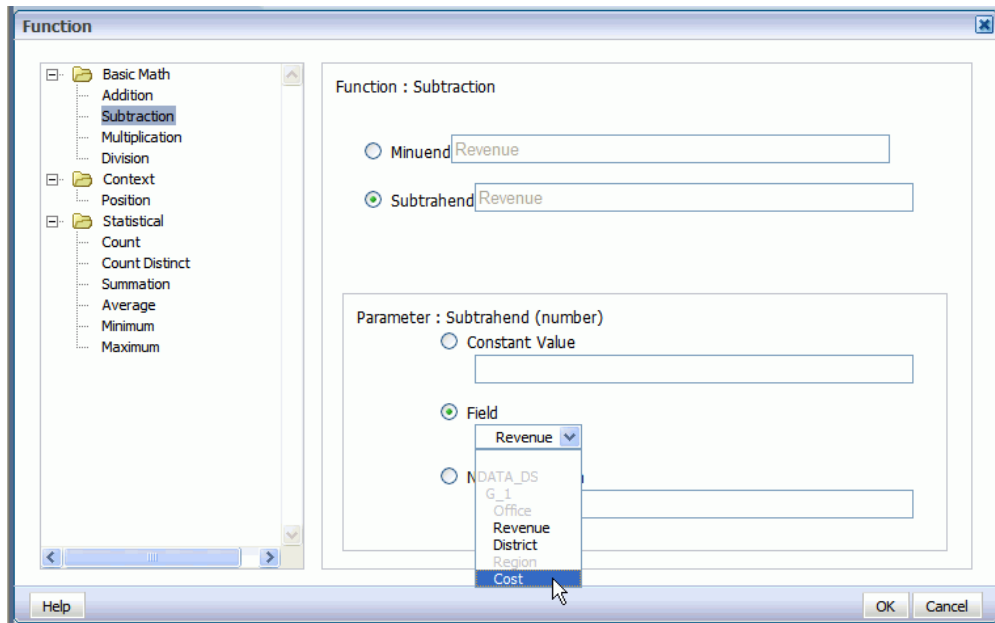
of Revenue to the table:

Office	Revenue	Cost	Revenue
Whalen	\$4,400.00	\$1,100.00	\$4,400.00
Hartstein	\$13,000.00	\$3,250.00	\$13,000.00
Fay	\$6,000.00	\$1,500.00	\$6,000.00
Raphaely	\$11,000.00	\$2,750.00	\$11,000.00
Khoo	\$3,100.00	\$775.00	\$3,100.00
Baida	\$2,900.00	\$725.00	\$2,900.00
Tobias	\$2,800.00	\$700.00	\$2,800.00
Himuro	\$2,600.00	\$650.00	\$2,600.00
Colmenares	\$2,500.00	\$625.00	\$2,500.00
Mavris	\$6,500.00	\$1,625.00	\$6,500.00

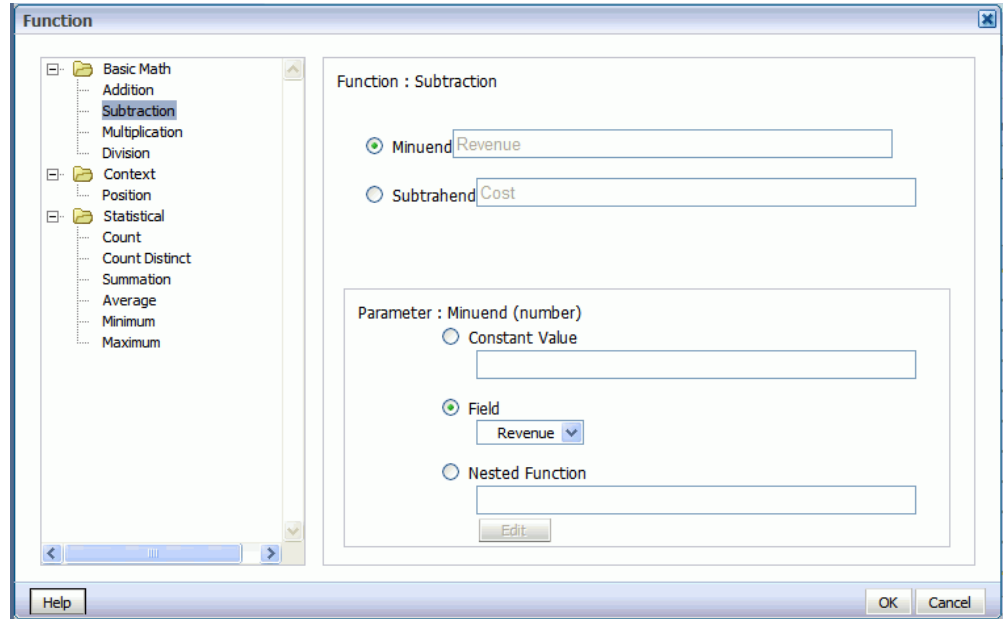
2. With the table column selected, click **Define Custom Formula**.
3. In the **Function** dialog select **Subtraction** from the list. Because the source data for the column is Revenue, by default the Minuend and the Subtrahend will both show the Revenue element



4. Select Subtrahend, then in the Parameter region, select **Field** and choose the Cost element.



The dialog will update to show that your formula is now Revenue minus Cost.



5. Click OK to close the dialog.
6. The table column displays the custom formula. Edit the table column header title, and now your table has a Profit column:

Office	Revenue	Cost	Profit
Whalen	\$4,400.00	\$1,100.00	\$3,300.00
Hartstein	\$13,000.00	\$3,250.00	\$9,750.00
Fay	\$6,000.00	\$1,500.00	\$4,500.00
Raphaely	\$11,000.00	\$2,750.00	\$8,250.00
Khoo	\$3,100.00	\$775.00	\$2,325.00
Baida	\$2,900.00	\$725.00	\$2,175.00
Tobias	\$2,800.00	\$700.00	\$2,100.00
Himuro	\$2,600.00	\$650.00	\$1,950.00
Colmenares	\$2,500.00	\$625.00	\$1,875.00
Mavris	\$6,500.00	\$1,625.00	\$4,875.00

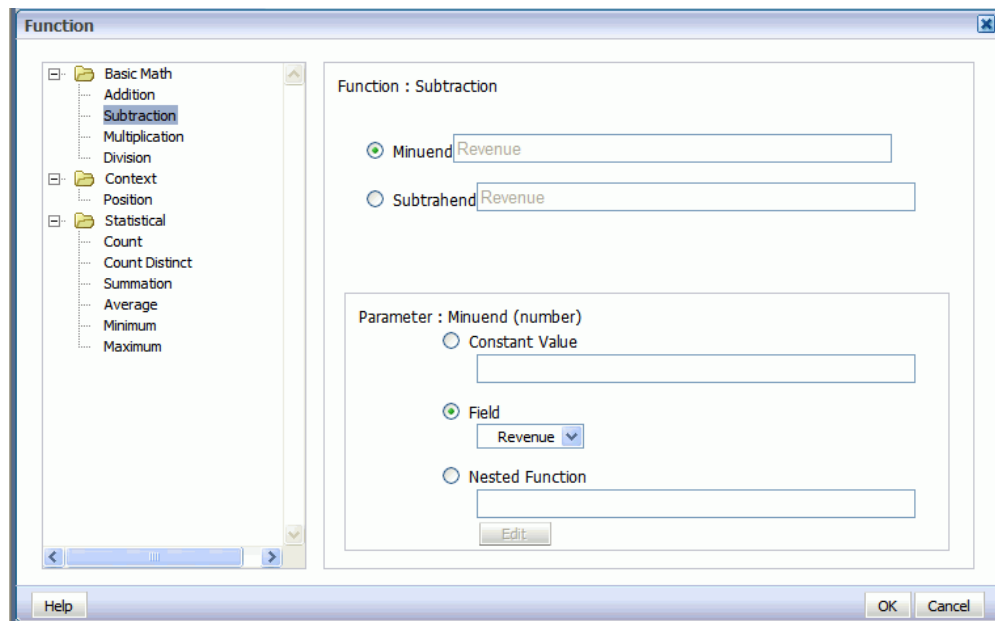
### Example 2: Nested Function

This example will use a nested function to create a column that shows Revenue less taxes.

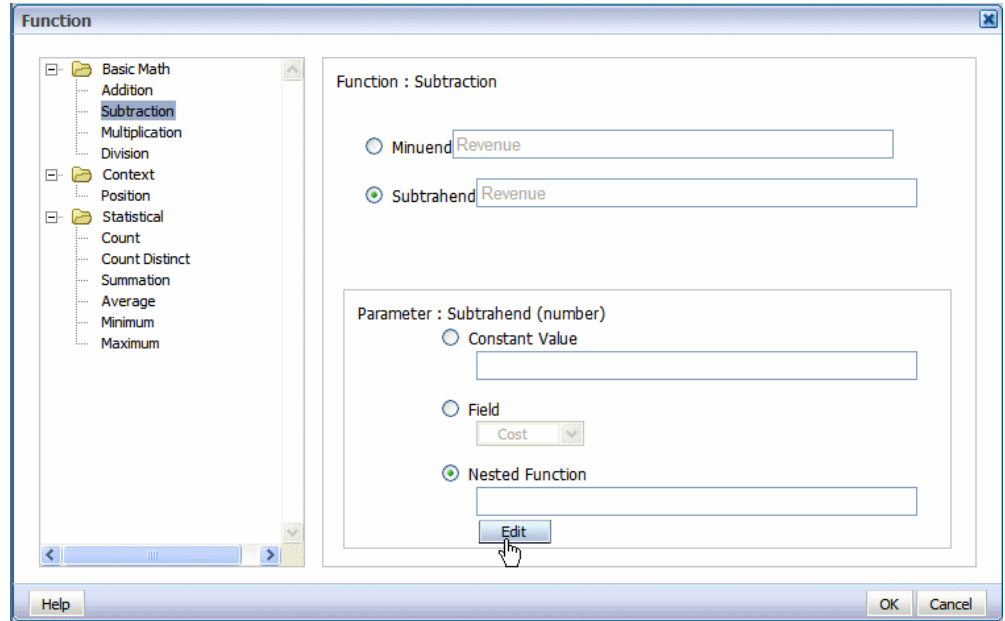
1. Add another numeric data column to the table. For example, drag another instance of Revenue to the table:

Office	Revenue	Revenue
Whalen	\$4,400.00	\$4,400.00
Hartstein	\$13,000.00	\$13,000.00
Fay	\$6,000.00	\$6,000.00
Raphaely	\$11,000.00	\$11,000.00
Khoo	\$3,100.00	\$3,100.00
Baida	\$2,900.00	\$2,900.00
Tobias	\$2,800.00	\$2,800.00
Himuro	\$2,600.00	\$2,600.00
Colmenares	\$2,500.00	\$2,500.00
Mavris	\$6,500.00	\$6,500.00

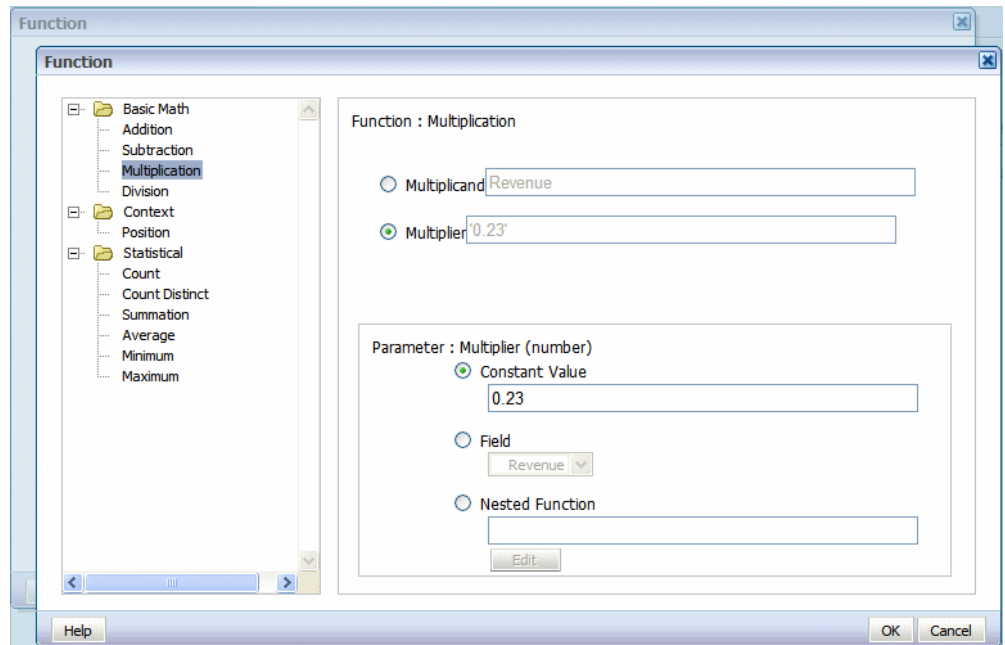
2. With the table column selected, click **Define Custom Formula**.
3. In the **Function** dialog select **Subtraction** from the list. Because the source data for the column is Revenue, by default the Minuend and the Subtrahend will both show the Revenue element



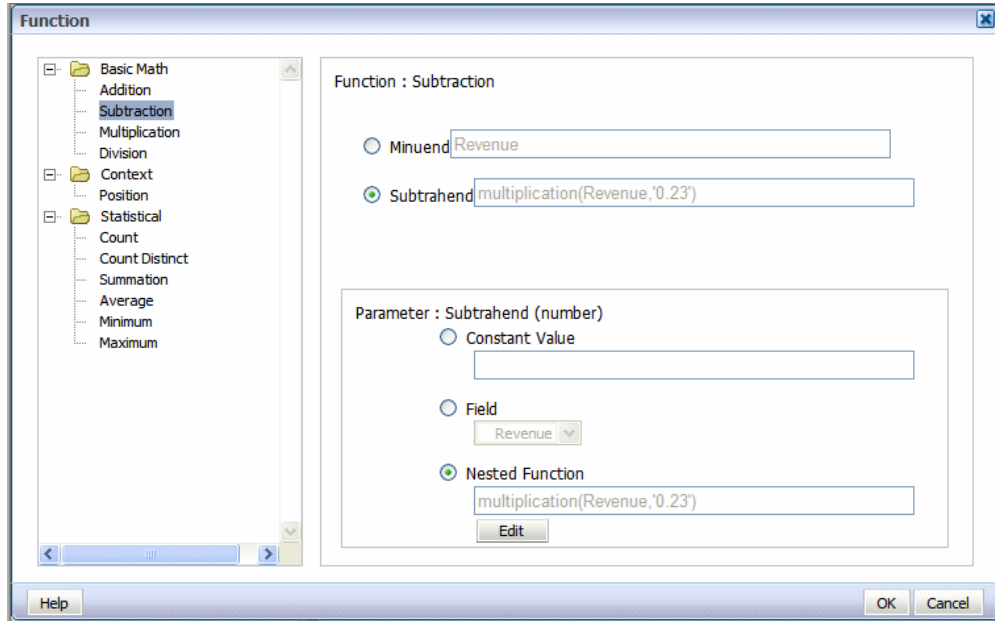
4. Select Subtrahend, then in the Parameter region, select **Nested Function** and click **Edit**.



A second Function dialog will launch to enable you to define the nested function. In this case the nested function is Revenue times a constant value (tax rate of .23).



5. Click OK to close the dialog. The primary Function dialog now shows the nested function as the source of the subtrahend:



- Click OK to close the Function dialog. The table column displays the custom formula. Edit the table column header label, and now your table shows your custom function:

Office	Revenue	Revenue less tax (23%)
Whalen	4,400.00	3,388.00
Hartstein	13,000.00	10,010.00
Fay	6,000.00	4,620.00
Raphaely	11,000.00	8,470.00
Khoo	3,100.00	2,387.00
Baida	2,900.00	2,233.00
Tobias	2,800.00	2,156.00
Himuro	2,600.00	2,002.00
Colmenares	2,500.00	1,925.00
Mavris	6,500.00	5,005.00
	633,500.00	487,795.00

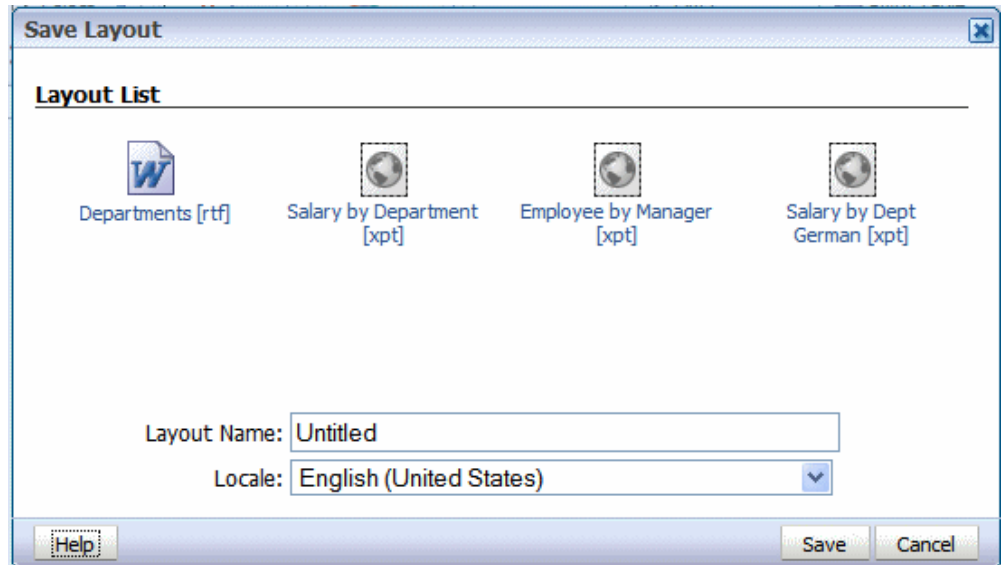
## Saving a Layout

To save your layout to the report definition:

- Click the **Save** or **Save As** toolbar button



2. The Save Layout dialog displays the list of layouts defined for the report definition as shown in the following figure:



3. Enter a unique name for this layout.
4. Select a **Locale**.

**Important:** Once you have saved the layout, the Locale cannot be updated.



---

## Creating RTF Templates

This chapter covers the following topics:

- What Is an RTF Template?
- About XSLT Compatibility
- Getting Started
- Key Concepts
- Designing the Template Layout
- Adding Markup to the Template Layout
- Defining Headers and Footers
- Inserting Images and Charts
- Drawing, Shape, and Clip Art Support
- Supported Native Formatting Features
- Template Features
- Conditional Formatting
- Page-Level Calculations
- Data Handling
- Using Variables
- Defining Parameters
- Setting Properties
- Advanced Report Layouts
- Number, Date, and Currency Formatting
- Calendar and Timezone Support
- Using External Fonts
- Controlling the Placement of Instructions Using the Context Commands

- Using XPath Commands
- Namespace Support
- Using XSL Elements
- Using FO Elements
- Guidelines for Designing RTF Templates for Microsoft PowerPoint Output

## What Is an RTF Template?

Rich Text Format (RTF) is a specification used by common word processing applications, such as Microsoft Word. When you save a document, RTF is a file type option.

BI Publisher converts documents saved as the RTF file type to XSL-FO enabling you to create report layouts using many standard word processing application features.

During design time, you add data fields and other markup to your template using BI Publisher's simplified tags for XSL expressions. These tags associate the XML report data to your report layout and include other processing instructions.

In addition to your word processing application's formatting features, BI Publisher supports other advanced reporting features such as conditional formatting, dynamic data columns, running totals, and charts.

If you are familiar with XSL and prefer not to use the simplified tags, BI Publisher also supports the use of pure XSL elements in the template. If you wish to include code directly in your template, you can include *any* XSL element, many FO elements, and a set of SQL expressions extended by BI Publisher.

## About XSLT Compatibility

BI Publisher uses the XSLT processor provided by Oracle XDK 11.1.0.7.0, which supports the W3C XSL Transformations 1.0 recommendation. The processor also implements the current working drafts of the XSLT and XPath 2.0 standards. For more information about Oracle XDK see *Oracle XML Developer's Kit Programmer's Guide 11g*.

By default, BI Publisher is compatible with XSLT 1.0. If you wish to use XSLT and XPath 2.0 features in your template you must disable XSLT 1.0 compatibility. This configuration is performed at the template level. The template-level setting will override the server setting.

XSLT compatibility is set as a Build Option in the Template Builder for Word. See *Setting Build Options*, page 5-39.

## Getting Started

This chapter describes the concepts of associating XML data to layout elements in your

report template. It describes basic techniques as well as advanced techniques for creating complex and highly conditionalized report formats.

If you are using Microsoft Word to create RTF templates, see *Creating RTF Templates Using the Template Builder for Word*, page 5-1 before reading this chapter. The demos and samples provided in the Template Builder installation will help orient you to the process of creating templates in Microsoft Word.

It is not required to have Microsoft Word or the Template Builder to create RTF templates and this chapter describes how to add components without using the Template Builder. Many of the layout components described in this chapter can also be inserted to your template using the Template Builder.

## Prerequisites

Before you design your template, you must:

- Know the business rules that apply to the data from your source report.
- Generate sample data from the report data model.

For information on generating sample data from a data model, see *Testing Data Models and Generating Sample Data, Oracle Fusion Middleware Data Modeling Guide for Oracle Business Intelligence Publisher*.

- Be familiar with the formatting features of your word processing application.

## About Adding BI Publisher Code

When you create an RTF template you add BI Publisher code to your RTF document. BI Publisher supports the following methods for adding code:

- Basic RTF Method

Use any word processing application that supports RTF version 1.6 writer (or later) to design a template using BI Publisher's simplified syntax.

- Form Field Method

Using Microsoft Word's form field feature allows you to place the syntax in hidden form fields, rather than directly into the design of your template.

**Note:** If you use XSL or XSL:FO code rather than the simplified syntax, you must use the form field method.

This chapter describes how to create RTF templates using the preceding methods.

If you are using Microsoft Word, you can use the BI Publisher Template Builder for Word to facilitate inserting BI Publisher code fields. For detailed information, see *Creating RTF Templates Using the Template Builder for Word*, page 5-1.

## Key Concepts

When you design your template layout, you must understand how to associate the XML input file to the layout. This chapter presents a sample template layout with its input XML file to illustrate how to make the proper associations to add the markup tags to the template.

## Associating the XML Data to the Template Layout

The following is a sample layout for a Payables Invoice Register:

### Sample Template Layout



### Payables Invoice Register

Page 1 of 1  
25<sup>th</sup> July 2003

Supplier:

Invoice Num	Invoice Date	GL Date	Curr	Entered Amt	Accounted Amt
Total for Supplier:					

Company Confidential

Note the following:

- The data fields that are defined on the template  
For example: Supplier, Invoice Number, and Invoice Date
- The elements of the template that will repeat when the report is run.  
For example, all the fields on the template will repeat for each Supplier that is reported. Each row of the invoice table will repeat for each invoice that is reported.

## XML Input File

Following is the XML file that will be used as input to the Payables Invoice Register report template:

**Note:** To simplify the example, the XML output shown below has been modified from the actual output from the Payables report.

```
<?xml version="1.0" encoding="WINDOWS-1252" ?>
- <VENDOR_REPORT>
- <LIST_G_VENDOR_NAME>
- <G_VENDOR_NAME>
  <VENDOR_NAME>COMPANY A</VENDOR_NAME>
- <LIST_G_INVOICE_NUM>
- <G_INVOICE_NUM>
  <SET_OF_BOOKS_ID>124</SET_OF_BOOKS_ID>
  <GL_DATE>10-NOV-03</GL_DATE>
  <INV_TYPE>Standard</INV_TYPE>
  <INVOICE_NUM>031110</INVOICE_NUM>
  <INVOICE_DATE>10-NOV-03</INVOICE_DATE>
  <INVOICE_CURRENCY_CODE>EUR</INVOICE_CURRENCY_CODE>
  <ENT_AMT>122</ENT_AMT>
  <ACCTD_AMT>122</ACCTD_AMT>
  <VAT_CODE>VAT22%</VAT_CODE>
</G_INVOICE_NUM>
</LIST_G_INVOICE_NUM>
<ENT_SUM_VENDOR>1000.00</ENT_SUM_VENDOR>
<ACCTD_SUM_VENDOR>1000.00</ACCTD_SUM_VENDOR>
</G_VENDOR_NAME>
</LIST_G_VENDOR_NAME>
<ACCTD_SUM_REP>108763.68</ACCTD_SUM_REP>
<ENT_SUM_REP>122039</ENT_SUM_REP>
</VENDOR_REPORT>
```

XML files are composed of elements. Each tag set is an element. For example `<INVOICE_DATE></INVOICE_DATE>` is the invoice date element. "INVOICE\_DATE" is the tag name. The data between the tags is the value of the element. For example, the value of INVOICE\_DATE is "10-NOV-03".

The elements of the XML file have a hierarchical structure. Another way of saying this is that the elements have parent-child relationships. In the XML sample, some elements are contained within the tags of another element. The containing element is the parent and the included elements are its children.

Every XML file has only one root element that contains all the other elements. In this example, VENDOR\_REPORT is the root element. The elements LIST\_G\_VENDOR\_NAME, ACCTD\_SUM\_REP, and ENT\_SUM\_REP are contained between the VENDOR\_REPORT tags and are children of VENDOR\_REPORT. Each child element can have child elements of its own.

## Identifying Placeholders and Groups

Your template content and layout must correspond to the content and hierarchy of the input XML file. Each data field in your template must map to an element in the XML file. Each group of repeating elements in your template must correspond to a parent-child relationship in the XML file.

To map the data fields you define *placeholders*. To designate the repeating elements, you define *groups*.

**Note:** BI Publisher supports regrouping of data if your report requires grouping that does not follow the hierarchy of your incoming XML data. For information on using this feature, see Regrouping the XML Data, page 4-85.

## Placeholders

Each data field in your report template must correspond to an element in the XML file. When you mark up your template design, you define placeholders for the XML elements. The placeholder maps the template report field to the XML element. At runtime the placeholder is replaced by the value of the element of the same name in the XML data file.

For example, the "Supplier" field from the sample report layout corresponds to the XML element `VENDOR_NAME`. When you mark up your template, you create a placeholder for `VENDOR_NAME` in the position of the Supplier field. At runtime, this placeholder will be replaced by the value of the element from the XML file (the value in the sample file is **COMPANY A**).

## Identifying the Groups of Repeating Elements

The sample report lists suppliers and their invoices. There are fields that repeat for each supplier. One of these fields is the supplier's invoices. There are fields that repeat for each invoice. The report therefore consists of two groups of repeating fields:

- Fields that repeat for each supplier
- Fields that repeat for each invoice

The invoices group is nested inside the suppliers group. This can be represented as follows:

### Suppliers

- Supplier Name
- Invoices
  - Invoice Num
  - Invoice Date
  - GL Date
  - Currency
  - Entered Amount



- Accounted Amount
- Total Entered Amount
- Total Accounted Amount

Compare this structure to the hierarchy of the XML input file. The fields that belong to the Suppliers group shown above are children of the element G\_VENDOR\_NAME. The fields that belong to the Invoices group are children of the element G\_INVOICE\_NUM.

By defining a group, you are notifying BI Publisher that *for each* occurrence of an element (parent), you want the included fields (children) displayed. At runtime, BI Publisher will loop through the occurrences of the element and display the fields each time.

## Designing the Template Layout

Use your word processing application's formatting features to create the design.

For example:

- Select the size, font, and alignment of text
- Insert bullets and numbering
- Draw borders around paragraphs
- Include a watermark
- Include images (jpg, gif, or png)
- Use table autoformatting features
- Insert a header and footer

For additional information on inserting headers and footers, see *Defining Headers and Footers*, page 4-16.

For a detailed list of supported formatting features in Microsoft Word, see *Supported Native Formatting Features*, page 4-41. Additional formatting and reporting features are described at the end of this section.

## Adding Markup to the Template Layout

BI Publisher converts the formatting that you apply in your word processing application to XSL-FO. You add markup to create the mapping between your layout and the XML file and to include features that cannot be represented directly in your format.

The most basic markup elements are placeholders, to define the XML data elements; and groups, to define the repeating elements.

BI Publisher provides tags to add markup to your template.

**Note:** For the XSL equivalents of the BI Publisher tags, see XSL Equivalent Syntax, page B-18.

## Creating Placeholders

The placeholder maps the template field to the XML element data field. At runtime the placeholder is replaced by the value of the element of the same name in the XML data file.

Enter placeholders in your document using the following syntax:

```
<?XML element tag name?>
```

**Note:** The placeholder must match the XML element tag name exactly. It is case sensitive.

There are two ways to insert placeholders in your document:

1. Basic RTF Method: Insert the placeholder syntax directly into your template document.
2. Form Field Method: (Requires Microsoft Word) Insert the placeholder syntax in Microsoft Word's Text Form Field Options window. This method allows you to maintain the appearance of your template.

Also see Inserting a Field, page 5-9 in the chapter "Creating RTF Templates Using the Template Builder for Word."

### Basic RTF Method

Enter the placeholder syntax in your document where you want the XML data value to appear.

Enter the element's XML tag name using the syntax:

```
<?XML element tag name?>
```

In the example, the template field "Supplier" maps to the XML element `VENDOR_NAME`. In your document, enter:

```
<?VENDOR_NAME?>
```

The entry in the template is shown in the following figure:

Supplier: <?VENDOR\_NAME?>

Invoice Num	Invoice

Total for Supplier:

## Form Field Method

Use Microsoft Word's **Text Form Field Options** window to insert the placeholder tags:

1. Enable the **Forms** toolbar in your Microsoft Word application.
2. Position your cursor in the place you want to create a placeholder.
3. Select the **Text Form Field** toolbar icon. This action inserts a form field area in your document.
4. Double-click the form field area to invoke the **Text Form Field Options** dialog box.
5. (Optional) Enter a description of the field in the **Default text** field. The entry in this field will populate the placeholder's position on the template.  
For the example, enter "Supplier 1".
6. Select the **Add Help Text** button.
7. In the help text entry field, enter the XML element's tag name using the syntax:

`<?XML element tag name?>`

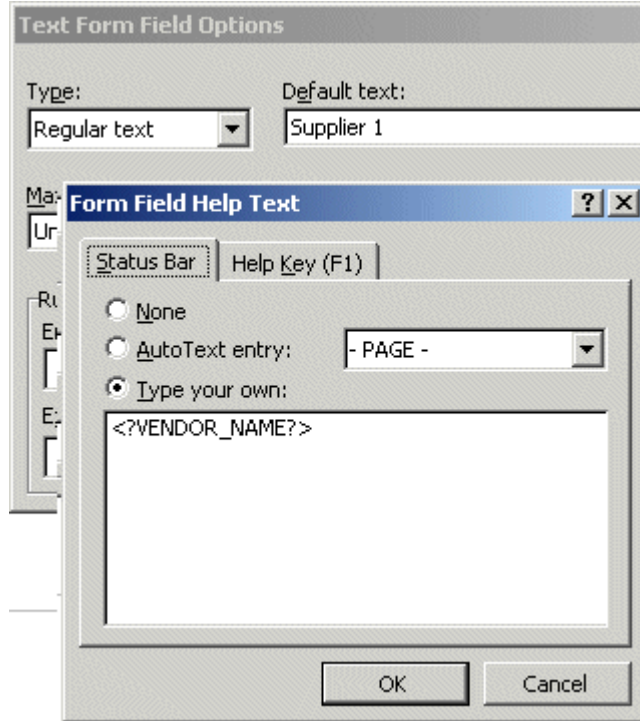
You can enter multiple element tag names in the text entry field.

In the example, the report field "Supplier" maps to the XML element VENDOR\_NAME. In the **Form Field Help Text** field enter:

`<?VENDOR_NAME?>`

The following figure shows the **Text Form Field Options** dialog box and the **Form Field Help Text** dialog box with the appropriate entries for the Supplier field.

**Tip:** For longer strings of BI Publisher syntax, use the Help Key (F1) tab instead of the Status Bar tab. The text entry field on the Help Key (F1) tab allows more characters.



8. Select **OK** to apply.

The **Default text** is displayed in the form field on your template.

The figure below shows the Supplier field from the template with the added form field markup.

Supplier:

Invoice Num	Inv
Total	

### Complete the Example

The following table shows the entries made to complete the example. The Template

Field Name is the display name from the template. The Default Text Entry is the value entered in the Default Text field of the Text Form Field Options dialog box (form field method only). The Placeholder Entry is the XML element tag name entered either in the Form Field Help Text field (form field method) or directly on the template.

Template Field Name	Default Text Entry (Form Field Method)	Placeholder Entry (XML Tag Name)
Invoice Num	1234566	<?INVOICE_NUM?>
Invoice Date	1-Jan-2004	<?INVOICE_DATE?>
GL Date	1-Jan-2004	<?GL_DATE?>
Curr	USD	<?INVOICE_CURRENCY_CODE?>
Entered Amt	1000.00	<?ENT_AMT?>
Accounted Amt	1000.00	<?ACCTD_AMT?>
(Total of Entered Amt column)	1000.00	<?ENT_SUM_VENDOR?>
(Total of Accounted Amt column)	1000.00	<?ACCTD_SUM_VENDOR?>

The following figure shows the Payables Invoice Register with the completed form field placeholder markup.

See the Payables Invoice Register with Completed Basic RTF Markup, page 4-13 for the completed basic RTF markup.

Supplier: **Supplier 1**

Invoice Num	Invoice Date	GL Date	Curr	Entered Amt	Accounted Amt
1234566	1-Jan-2004	1-Jan-2004	USD	1000.00	1000.00
Total for Supplier: Supplier1				1000.00	1000.00

Company Confidential

## Defining Groups

**Note:** Also see Inserting a Repeating Group, page 5-25 in the chapter "Creating RTF Templates Using the Template Builder for Word."

By defining a group, you are notifying BI Publisher that *for each* occurrence of an element, you want the included fields displayed. At runtime, BI Publisher will loop through the occurrences of the element and display the fields each time.

In the example, for each occurrence of G\_VENDOR\_NAME in the XML file, we want the template to display its child elements VENDOR\_NAME (Supplier Name), G\_INVOICE\_NUM (the Invoices group), Total Entered Amount, and Total Accounted Amount. And, for each occurrence of G\_INVOICE\_NUM (Invoices group), we want the template to display Invoice Number, Invoice Date, GL Date, Currency, Entered Amount, and Accounted Amount.

To designate a group of repeating fields, insert the grouping tags around the elements to repeat.

Insert the following tag before the first element:

```
<?for-each:XML group element tag name?>
```

Insert the following tag after the final element:

```
<?end for-each?>
```

## Grouping scenarios

Note that the group element must be a parent of the repeating elements in the XML input file.

- If you insert the grouping tags around text or formatting elements, the text and formatting elements between the group tags will be repeated.
- If you insert the tags around a table, the table will be repeated.
- If you insert the tags around text in a table cell, the text in the table cell between the tags will be repeated.
- If you insert the tags around two different table cells, but in the same table row, the single row will be repeated.
- If you insert the tags around two different table rows, the rows between the tags will be repeated (this does not include the row that contains the "end group" tag).

### Basic RTF Method

Enter the tags in your document to define the beginning and end of the repeating element group.

To create the Suppliers group in the example, insert the tag

```
<?for-each:G_VENDOR_NAME?>
```

before the Supplier field that you previously created.

Insert `<?end for-each?>` in the document after the summary row.

The following figure shows the Payables Invoice Register with the basic RTF grouping and placeholder markup:

<?for-each:G\_VENDOR\_NAME?> <?sort:VENDOR\_NAME?>

Supplier: <?VENDOR\_NAME?>

Invoice Num	Invoice Date	GL Date	Curr	Entered Amt	Accounted Amt
<?for-each: G_INVOICE_NUM?> <?INVOICE_NUM?>	<?INVOICE_D ATE?>	<?GL_DATE?>	<?INV: OICE CUR REN CY_C ODE? >	<?ENT_AMT?>	<?ACCTD_AMT?> <?end for-each?>
Total for Supplier: <?VENDOR_NAME?>				<?ENT_SUM_VEND OR?>	<?ACCTD_SUM_V ENDOR?>

<?end for-each?>

Company Confidential

## Form Field Method

1. Insert a form field to designate the beginning of the group.

In the help text field enter:

<?for-each:group element tag name?>

To create the Suppliers group in the example, insert a form field before the Suppliers field that you previously created. In the help text field enter:

<?for-each:G\_VENDOR\_NAME?>

For the example, enter the Default text "Group: Suppliers" to designate the beginning of the group on the template. The Default text is not required, but can make the template easier to read.

2. Insert a form field after the final placeholder element in the group. In the help text field enter <?end for-each?>.

For the example, enter the Default text "End: Suppliers" after the summary row to designate the end of the group on the template.

The following figure shows the template after the markup to designate the Suppliers group was added.



Group: Suppliers

Supplier: Supplier 1

Invoice Num	Invoice
1234566	1-Jan-

End:Suppliers

### Complete the Example

The second group in the example is the invoices group. The repeating elements in this group are displayed in the table. For each invoice, the table row should repeat. Create a group within the table to contain these elements.

**Note:** For each invoice, only the table *row* should repeat, not the entire table. Placing the grouping tags at the beginning and end of the table row will repeat only the row. If you place the tags around the table, then for each new invoice the entire table with headings will be repeated.

To mark up the example, insert the grouping tag `<?for-each:G_INVOICE_NUM?>` in the table cell before the Invoice Num placeholder. Enter the Default text "Group:Invoices" to designate the beginning of the group.

Insert the end tag inside the final table cell of the row after the Accounted Amt placeholder. Enter the Default text "End:Invoices" to designate the end of the group.

The following figure shows the completed example using the form field method:

Group: Suppliers    Sort by Supplier

Supplier: **Supplier 1**

Invoice Num	Invoice Date	GL Date	Curr	Entered Amt	Accounted Amt
Group:Invoices 1234566	1-Jan-2004	1-Jan-2004	USD	1000.00	1000.00
					End:Invoices
Total for Supplier: <b>Supplier1</b>				<b>1000.00</b>	<b>1000.00</b>

End:Suppliers

Company Confidential

## Defining Headers and Footers

### Native Support

BI Publisher supports the use of the native RTF header and footer feature. To create a header or footer, use the your word processing application's header and footer insertion tools. As an alternative, or if you have multiple headers and footers, you can use `start:body` and `end body` tags to distinguish the header and footer regions from the body of your report.

### Inserting Placeholders in the Header and Footer

At the time of this writing, Microsoft Word does not support form fields in the header and footer. You must therefore insert the placeholder syntax directly into the template (basic RTF method), or use the `start body/end body` syntax described in the next section.

### Multiple or Complex Headers and Footers

If your template requires multiple headers and footers, create them by using BI Publisher tags to define the body area of your report. You may also want to use this method if your header and footer contain complex objects that you wish to place in form fields. When you define the body area, the elements occurring before the beginning of the body area will compose the header. The elements occurring after the body area will compose the footer.

Use the following tags to enclose the body area of your report:

```
<?start:body?>
```

```
<?end body?>
```

Use the tags either directly in the template, or in form fields.

The Payables Invoice Register contains a simple header and footer and therefore does not require the start body/end body tags. However, if you wanted to add another header to the template, define the body area as follows:

1. Insert `<?start:body?>` before the Suppliers group tag:  
`<?for-each:G_VENDOR_NAME?>`
2. Insert `<?end body?>` after the Suppliers group closing tag: `<?end for-each?>`

The following figure shows the Payables Invoice Register with the start body/end body tags inserted:

**ORACLE**  
E-Business Suite

**Payables Invoice Register**  
Page 1 of 1  
25<sup>th</sup> July 2003

`<?start:body?>`

Group: Suppliers    Sort by Supplier

Supplier: **Supplier 1**

Invoice Num	Invoice Date	GL Date	Curr	Entered Amt	Accounted Amt
Group:Invoices 1234566	1-Jan-2004	1-Jan-2004	USD	1000.00	1000.00
				End:Invoices	
Total for Supplier: Supplier1				<b>1000.00</b>	<b>1000.00</b>

End:Suppliers  
`<?end body?>`

Company Confidential

## Different First Page and Different Odd and Even Page Support

If your report requires a different header and footer on the first page of your report; or, if your report requires different headers and footers for odd and even pages, you can define this behavior using Microsoft Word's Page Setup dialog.

**Note:** This feature is supported for PDF and RTF output only.

1. Select **Page Setup** from the **File** menu.
2. In the **Page Setup** dialog, select the **Layout** tab.
3. In the **Headers and footers** region of the dialog, select the appropriate check box:  
Different odd and even  
Different first page
4. Insert your headers and footers into your template as desired.

At runtime your generated report will exhibit the defined header and footer behavior.

## Inserting Images and Charts

### Images

BI Publisher supports several methods for including images in your published document:

#### Direct Insertion

Insert the jpg, gif, or png image directly in your template.

#### URL Reference

URL Reference

1. Insert a dummy image in your template.
2. In Microsoft Word's **Format Picture** dialog box select the **Web** tab. Enter the following syntax in the **Alternative text** region to reference the image URL:

```
url: {'http://image location'}
```

For example, enter:

```
url: {'http://www.oracle.com/images/ora_log.gif'}
```

#### Element Reference from XML File

1. Insert a dummy image in your template.
2. In Microsoft Word's **Format Picture** dialog box select the **Web** tab. Enter the following syntax in the **Alternative text** region to reference the image URL:

```
url: {IMAGE_LOCATION}
```

where IMAGE\_LOCATION is an element from your XML file that holds the full URL to the image.

You can also build a URL based on multiple elements at runtime. Just use the `concat` function to build the URL string. For example:

```
url: {concat (SERVER, '/', IMAGE_DIR, '/', IMAGE_FILE) }
```

where SERVER, IMAGE\_DIR, and IMAGE\_FILE are element names from your XML file that hold the values to construct the URL.

This method can also be used with the OA\_MEDIA reference as follows:

```
url: {concat ('${OA_MEDIA}', '/', IMAGE_FILE) }
```

## Rendering an Image Retrieved from BLOB Data

If results XML contains image data that had been stored as a BLOB in the database, use the following syntax in a form field inserted in your template where you want the image to render at runtime:

```
<fo:instream-foreign-object content-type="image/jpg">
<xsl:value-of select="IMAGE_ELEMENT"/>
</fo:instream-foreign-object>
```

where

*image/jpg* is the MIME type of the image (other options might be: *image/gif* and *image/png*)

and

*IMAGE\_ELEMENT* is the element name of the BLOB in your XML data.

Note that you can specify *height* and *width* attributes for the image to set its size in the published report. BI Publisher will scale the image to fit the box size that you define. For example, to set the size of the example above to three inches by four inches, enter the following:

```
<fo:instream-foreign-object content-type="image/jpg" height="3 in"
width="4 in">
<xsl:value-of select="IMAGE_ELEMENT"/>
</fo:instream-foreign-object>
```

Specify in pixels as follows:

```
<fo:instream-foreign-object content-type="image/jpg" height="300 px"
width="4 px">
...
```

or in centimeters:

```
<fo:instream-foreign-object content-type="image/jpg" height="3 cm"
width="4 cm">
...
```

or as a percentage of the original dimensions:

```
<fo:instream-foreign-object content-type="image/jpg" height="300%"
width="300%">
...
```

## Chart Support

**Note:** See also Inserting a Chart, page 5-20 in the chapter, "Creating RTF Templates Using the Template Builder for Word."

The following summarizes the steps to add a chart to your template. These steps will be discussed in detail in the example that follows:

1. Insert a dummy image in your template to define the size and position of your chart.

2. Add the definition for the chart to the Alternative text box of the dummy image. The chart definition requires XSL commands.
3. At runtime BI Publisher calls the charting engine to render the image that is then inserted into the final output document.

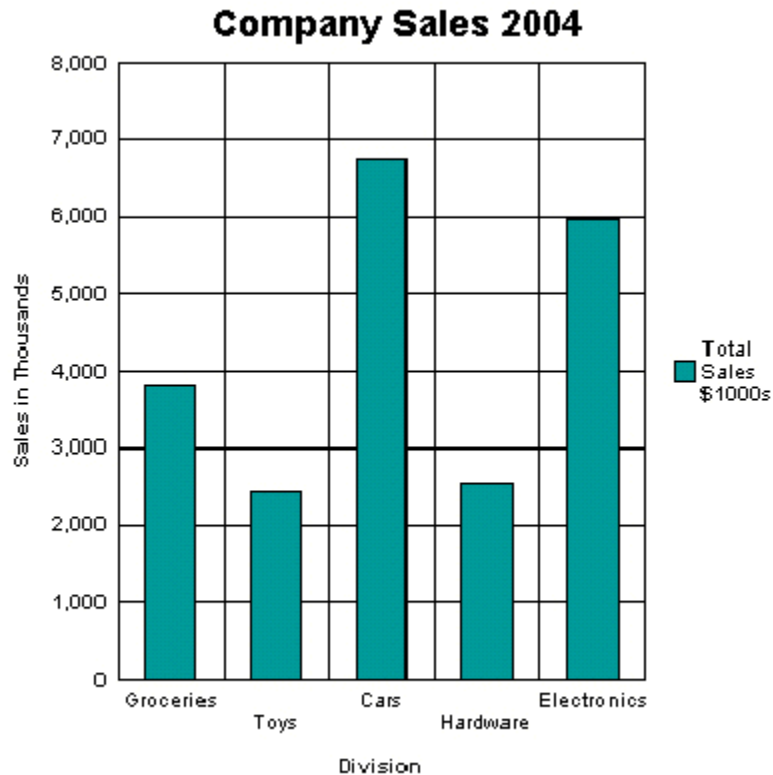
Note that RTF output is limited to raster images. PDF and HTML output support raster and vector images.

## Adding a Sample Chart

Following is a piece of XML data showing total sales by company division.

```
<sales year=2004>
  <division>
    <name>Groceries</name>
    <totalsales>3810</totalsales>
    <costofsales>2100</costofsales>
  </division>
  <division>
    <name>Toys</name>
    <totalsales>2432</totalsales>
    <costofsales>1200</costofsales>
  </division>
  <division>
    <name>Cars</name>
    <totalsales>6753</totalsales>
    <costofsales>4100</costofsales>
  </division>
  <division>
    <name>Hardware</name>
    <totalsales>2543</totalsales>
    <costofsales>1400</costofsales>
  </division>
  <division>
    <name>Electronics</name>
    <totalsales>5965</totalsales>
    <costofsales>3560</costofsales>
  </division>
</sales>
```

This example will show how to insert a chart into your template to display it as a vertical bar chart as shown in the following figure:



Note the following attributes of this chart:

- The style is a vertical bar chart.
- The chart displays a background grid.
- The components are colored.
- Sales totals are shown as Y-axis labels.
- Divisions are shown as X-axis labels.
- The chart is titled.
- The chart displays a legend.

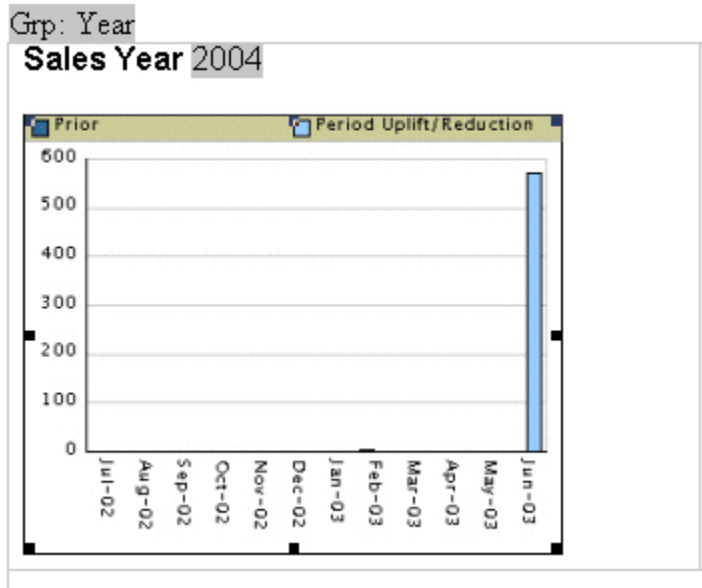
Each of these properties can be customized to suit individual report requirements.

#### **Inserting the Dummy Image**

The first step is to add a dummy image to the template in the position you want the chart to appear. The image size will define how big the chart image will be in the final document.

**Important:** You must insert the dummy image as a "Picture" and not any other kind of object.

The following figure shows an example of a dummy image:



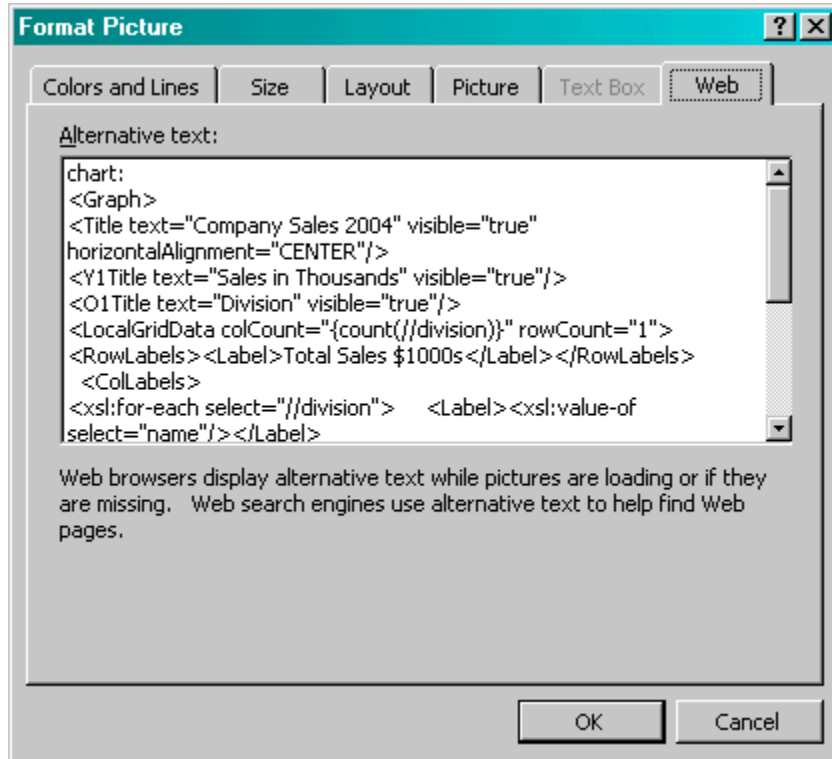
The image can be embedded inside a for-each loop like any other form field if you want the chart to be repeated in the output based on the repeating data. In this example, the chart is defined within the sales year group so that a chart will be generated for each year of data present in the XML file.

Right-click the image to open the **Format Picture** palette and select the **Web** tab. Use the **Alternative text** entry box to enter the code to define the chart characteristics and data definition for the chart.

#### **Adding Code to the Alternative Text Box**

The following graphic shows an example of the BI Publisher code in the **Format Picture Alternative text** box:





The content of the **Alternative text** represents the chart that will be rendered in the final document. For this chart, the text is as follows:

```

chart:
<Graph graphType = "BAR_VERT_CLUST">
  <Title text="Company Sales 2004" visible="true"
horizontalAlignment="CENTER"/>
  <Y1Title text="Sales in Thousands" visible="true"/>
  <O1Title text="Division" visible="true"/>
  <LocalGridData colCount="{count(//division)}" rowCount="1">
    <RowLabels>
      <Label>Total Sales $1000s</Label>
    </RowLabels>
    <ColLabels>
      <xsl:for-each select="//division">
        <Label>
          <xsl:value-of select="name"/>
        </Label>
      </xsl:for-each>
    </ColLabels>
    <DataValues>
      <RowData>
        <xsl:for-each select="//division">
          <Cell>
            <xsl:value-of select="totalsales"/>
          </Cell>
        </xsl:for-each>
      </RowData>
    </DataValues>
  </LocalGridData>
</Graph>

```

The first element of your chart text must be the `chart:` element to inform the RTF parser that the following code describes a chart object.

Next is the opening `<Graph>` tag. Note that the whole of the code resides within the tags of the `<Graph>` element. This element has an attribute to define the chart type: `graphType`. If this attribute is not declared, the default chart is a vertical bar chart. BI Beans supports many different chart types. Several more types are presented in this section. For a complete listing, see the BI Beans graph DTD documentation.

The following code section defines the chart type and attributes:

```
<Title text="Company Sales 2004" visible="true"
horizontalAlignment="CENTER"/>
  <YlTitle text="Sales in Thousands" visible="true"/>
  <OlTitle text="Division" visible="true"/>
```

All of these values can be declared or you can substitute values from the XML data at runtime. For example, you can retrieve the chart title from an XML tag by using the following syntax:

```
<Title text="{CHARTTITLE}" visible="true" horizontalAlignment="CENTER"/>
```

where "CHARTTITLE" is the XML tag name that contains the chart title. Note that the tag name is enclosed in curly braces.

The next section defines the column and row labels:

```
<LocalGridData colCount="{count(//division)}" rowCount="1">
  <RowLabels>
    <Label>Total Sales $1000s</Label>
  </RowLabels>
  <ColLabels>
    <xsl:for-each select="//division">
      <Label>
        <xsl:value-of select="name"/>
      </Label>
    </xsl:for-each>
  </ColLabels>
```

The `LocalGridData` element has two attributes: `colCount` and `rowCount`. These define the number of columns and rows that will be shown at runtime. In this example, a count function calculates the number of columns to render:

```
colCount="{count(//division)}"
```

The `rowCount` has been hard-coded to 1. This value defines the number of sets of data to be charted. In this case it is 1.

Next the code defines the row and column labels. These can be declared, or a value from the XML data can be substituted at runtime. The row label will be used in the chart legend (that is, "Total Sales \$1000s").

The column labels for this example are derived from the data: Groceries, Toys, Cars, and so on. This is done using a for-each loop:

```

<ColLabels>
  <xsl:for-each select="//division">
    <Label>
      <xsl:value-of select="name"/>
    </Label>
  </xsl:for-each>
</ColLabels>

```

This code loops through the <division> group and inserts the value of the <name> element into the <Label> tag. At runtime, this will generate the following XML:

```

<ColLabels>
  <Label>Groceries</Label>
  <Label>Toys</Label>
  <Label>Cars</Label>
  <Label>Hardware</Label>
  <Label>Electronics</Label>
</ColLabels>

```

The next section defines the actual data values to chart:

```

<DataValues>
  <RowData>
    <xsl:for-each select="//division">
      <Cell>
        <xsl:value-of select="totalsales"/>
      </Cell>
    </xsl:for-each>
  </RowData>
</DataValues>

```

Similar to the labels section, the code loops through the data to build the XML that is passed to the BI Beans rendering engine. This will generate the following XML:

```

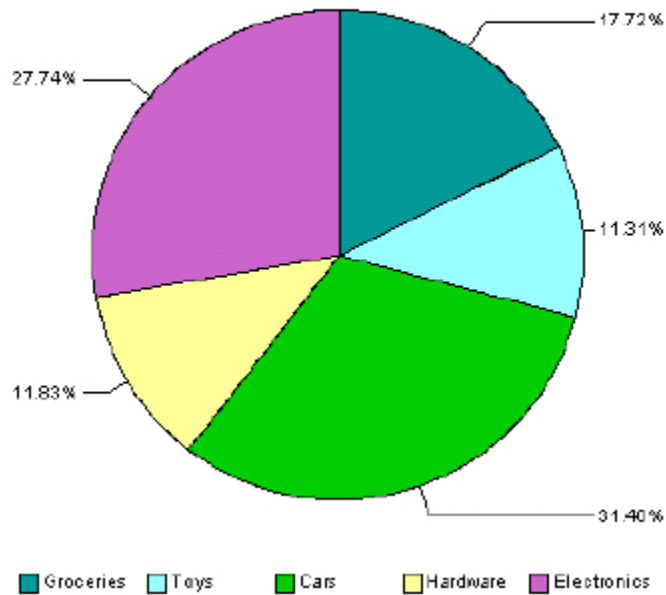
<DataValues>
  <RowData>
    <Cell>3810</Cell>
    <Cell>2432</Cell>
    <Cell>6753</Cell>
    <Cell>2543</Cell>
    <Cell>5965</Cell>
  </RowData>
</DataValues>

```

## Additional Chart Samples

You can also display this data in a pie chart as shown in the following figure:

## Company Sales 2004

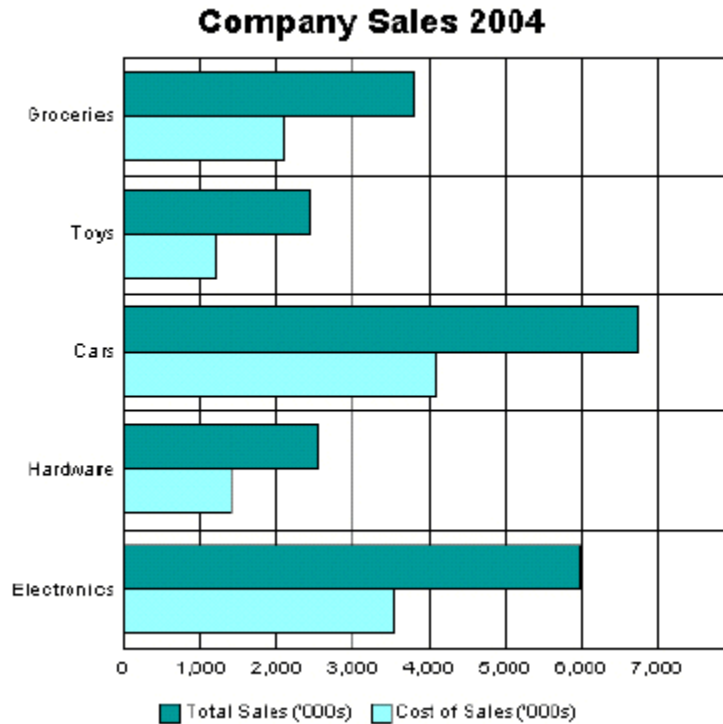


The following is the code added to the template to render this chart at runtime:

```
chart:
<Graph graphType="PIE">
  <Title text="Company Sales 2004" visible="true"
    horizontalAlignment="CENTER"/>
  <LocalGridData rowCount="{count(//division)}" colCount="1">
    <RowLabels>
      <xsl:for-each select="//division">
        <Label>
          <xsl:value-of select="name"/>
        </Label>
      </xsl:for-each>
    </RowLabels>
    <DataValues>
      <xsl:for-each select="//division">
        <RowData>
          <Cell>
            <xsl:value-of select="totalsales"/>
          </Cell>
        </RowData>
      </xsl:for-each>
    </DataValues>
  </LocalGridData>
</Graph>
```

### Horizontal Bar Chart Sample

The following example shows total sales and cost of sales charted in a horizontal bar format. This example also adds the data from the cost of sales element (`<costofsales>`) to the chart:



The following code defines this chart in the template:

```

chart:
<Graph graphType = "BAR_HORIZ_CLUST">
  <Title text="Company Sales 2004" visible="true"
horizontalAlignment="CENTER"/>
  <LocalGridData colCount="{count(//division)}" rowCount="2">
  <RowLabels>
    <Label>Total Sales ('000s)</Label>
    <Label>Cost of Sales ('000s)</Label>
  </RowLabels>
  <ColLabels>
    <xsl:for-each select="//division">
      <Label><xsl:value-of select="name"/></Label>
    </xsl:for-each>
  </ColLabels>
  <DataValues>
  <RowData>
    <xsl:for-each select="//division">
      <Cell><xsl:value-of select="totalsales"/></Cell>
    </xsl:for-each>
  </RowData>
  <RowData>
    <xsl:for-each select="//division">
      <Cell><xsl:value-of select="costofsales"/></Cell>
    </xsl:for-each>
  </RowData>
  </DataValues>
  </LocalGridData>
</Graph>

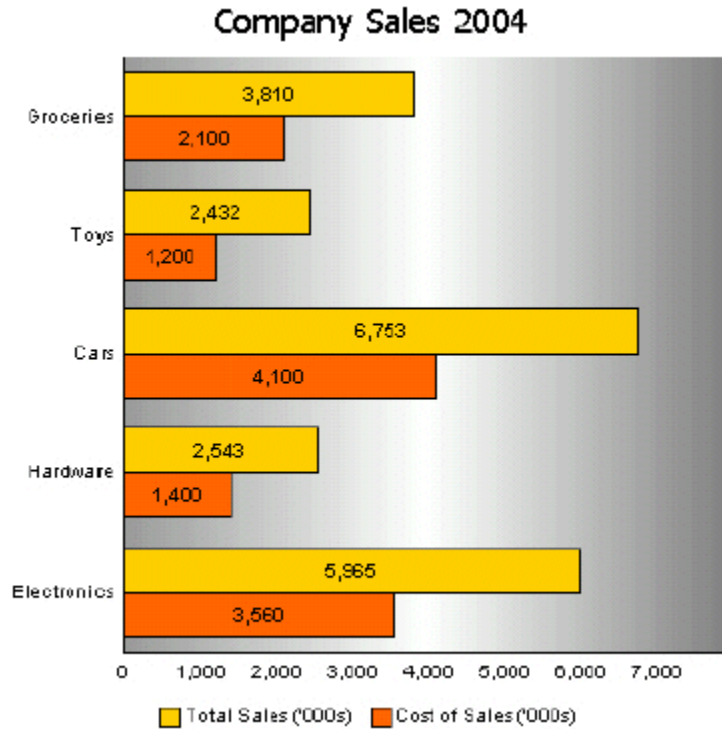
```

To accommodate the second set of data, the `rowCount` attribute for the

LocalGridData element is set to 2. Also note the DataValues section defines two sets of data: one for Total Sales and one for Cost of Sales.

## Changing the Appearance of Your Chart

There are many attributes available from the BI Beans graph DTD that you can manipulate to change the look and feel of your chart. For example, the previous chart can be changed to remove the grid, place a graduated background, and change the bar colors and fonts as shown in the following figure:



The code to support this is as follows:

```

chart:
<Graph graphType = "BAR_HORIZ_CLUST">
<SeriesItems>
  <Series id="0" color="#ffcc00"/>
  <Series id="1" color="#ff6600"/>
</SeriesItems>
<O1MajorTick visible="false"/>
<X1MajorTick visible="false"/>
<Y1MajorTick visible="false"/>
<Y2MajorTick visible="false"/>
<MarkerText visible="true" markerTextPlace="MTP_CENTER"/>
<PlotArea borderTransparent="true">
  <SFX fillType="FT_GRADIENT" gradientDirection="GD_LEFT"
    gradientNumPins="300">
    <GradientPinStyle pinIndex="1" position="1"
      gradientPinLeftColor="#999999"
      gradientPinRightColor="#cc6600"/>
  </SFX>
</PlotArea>
<Title text="Company Sales 2004" visible="true">
  <GraphFont name="Tahoma" bold="false"/>
</Title>
. . .
</Graph>

```

The colors for the bars are defined in the `SeriesItems` section. The colors are defined in hexadecimal format as follows:

```

<SeriesItems>
  <Series id="0" color="#ffcc00"/>
  <Series id="1" color="#ff6600"/>
</SeriesItems>

```

The following code hides the chart grid:

```

<O1MajorTick visible="false"/>
  <X1MajorTick visible="false"/>
  <Y1MajorTick visible="false"/>
  <Y2MajorTick visible="false"/>

```

The `MarkerText` tag places the data values on the chart bars:

```

<MarkerText visible="true" markerTextPlace="MTP_CENTER"/>

```

The `PlotArea` section defines the background. The `SFX` element establishes the gradient and the `borderTransparent` attribute hides the plot border:

```

<PlotArea borderTransparent="true">
  <SFX fillType="FT_GRADIENT" gradientDirection="GD_LEFT"
    gradientNumPins="300">
    <GradientPinStyle pinIndex="1" position="1"
      gradientPinLeftColor="#999999"
      gradientPinRightColor="#cc6600"/>
  </SFX>
</PlotArea>

```

The `Title text` tag has also been updated to specify a new font type and size:

```

<Title text="Company Sales 2004" visible="true">
  <GraphFont name="Tahoma" bold="false"/>
</Title>

```

## Drawing, Shape, and Clip Art Support

BI Publisher supports Microsoft Word drawing, shape, and clip art features. You can add these objects to your template and they will be rendered in your final PDF output or HTML output (not supported for other output types).

The following AutoShape categories are supported:

- Lines - straight, arrowed, connectors, curve, free form, and scribble
- Connectors - straight connectors only are supported. Curved connectors can be achieved by using a curved line and specifying the end styles to the line.
- Basic Shapes - all shapes are supported.
- Block arrows - all arrows are supported.
- Flowchart - all flowchart objects are supported.
- Stars and Banners - all objects are supported.
- Callouts - the "line" callouts are not supported.
- Clip Art - add images to your templates using the Microsoft Clip Art libraries

### Freehand Drawing

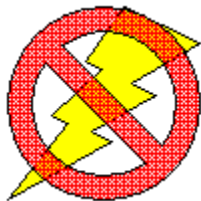
Use the freehand drawing tool in Microsoft Word to create drawings in your template to be rendered in the final PDF output.

### Hyperlinks

You can add hyperlinks to your shapes. See Hyperlinks, page 4-55.

### Layering

You can layer shapes on top of each other and use the transparency setting in Microsoft Word to allow shapes on lower layers to show through. The following graphic shows an example of layered shapes:





### 3-D Effects

BI Publisher does not currently support the 3-D option for shapes.

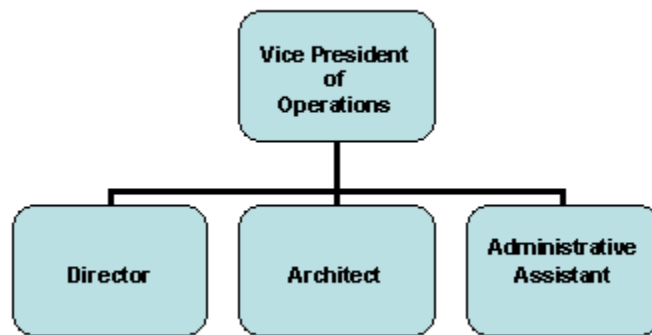
### Microsoft Equation

Use the equation editor to generate equations in your output. The following figure shows an example of an equation:

$$\sigma = \sqrt{\frac{1}{N} \sum_{i=1}^N \left( x_i - \bar{x} \right)^2}$$

### Organization Chart

Use the organization chart functionality in your templates and the chart will be rendered in the output. The following image shows an example of an organization chart:



### WordArt

You can use Microsoft Word's WordArt functionality in your templates. The following graphic shows a WordArt example:



**Note:** Some Microsoft WordArt uses a bitmap operation that currently cannot be converted to SVG. To use the unsupported WordArt in your template, you can take a screenshot of the WordArt then save it as an image (gif, jpeg, or png) and replace the WordArt with the image.

## Data Driven Shape Support

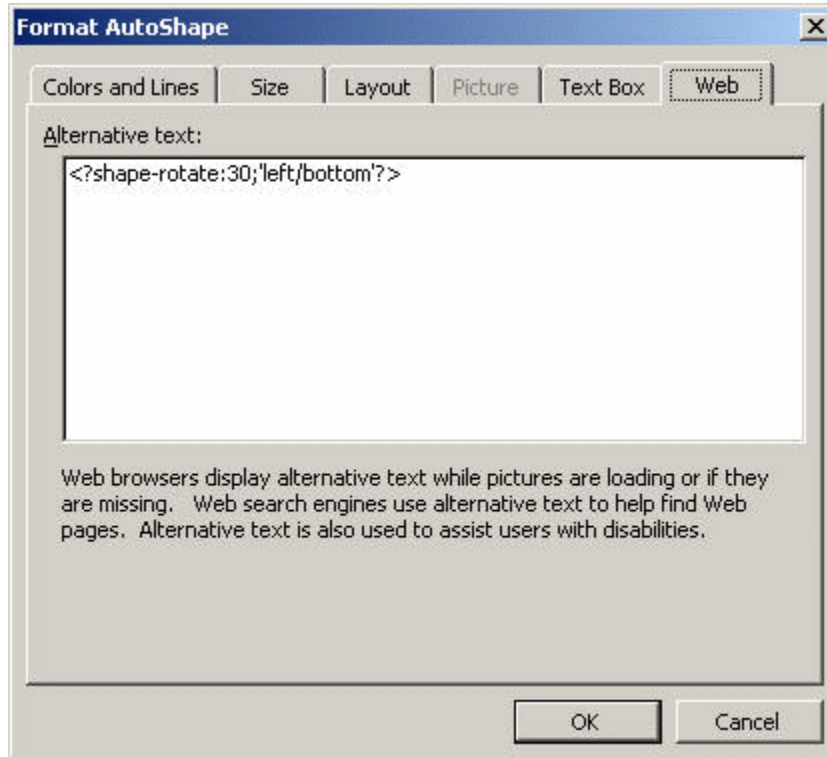
In addition to supporting the static shapes and features in your templates, BI Publisher supports the manipulation of shapes based on incoming data or parameters, as well. The following manipulations are supported:

- Replicate
- Move
- Change size
- Add text
- Skew
- Rotate

These manipulations not only apply to single shapes, but you can use the group feature in Microsoft Word to combine shapes together and manipulate them as a group.

### Placement of Commands

Enter manipulation commands for a shape in the Web tab of the shape's properties dialog as shown in the following example figure:



## Replicate a Shape

You can replicate a shape based on incoming XML data in the same way you replicate data elements in a for-each loop. To do this, use a `for-each@shape` command in conjunction with a `shape-offset` declaration. For example, to replicate a shape down the page, use the following syntax:

```
<?for-each@shape:SHAPE_GROUP?>
  <?shape-offset-y:(position()-1)*100?>
<?end for-each?>
```

where

`for-each@shape` opens the for-each loop for the shape context

`SHAPE_GROUP` is the name of the repeating element from the XML file. For each occurrence of the element `SHAPE_GROUP` a new shape will be created.

`shape-offset-y: -` is the command to offset the shape along the y-axis.

`(position()-1)*100` - sets the offset in pixels per occurrence. The XSL position command returns the record counter in the group (that is 1,2,3,4); one is subtracted from that number and the result is multiplied by 100. Therefore for the first occurrence the offset would be 0:  $(1-1) * 100$ . The offset for the second occurrence would be 100 pixels:  $(2-1) * 100$ . And for each subsequent occurrence the offset would be another 100 pixels down the page.

### Add Text to a Shape

You can add text to a shape dynamically either from the incoming XML data or from a parameter value. In the property dialog enter the following syntax:

```
<?shape-text:SHAPETEXT?>
```

where SHAPETEXT is the element name in the XML data. At runtime the text will be inserted into the shape.

### Add Text Along a Path

You can add text along a line or curve from incoming XML data or a parameter. After drawing the line, in the property dialog enter:

```
<?shape-text-along-path:SHAPETEXT?>
```

where SHAPETEXT is the element from the XML data. At runtime the value of the element SHAPETEXT will be inserted above and along the line.

### Moving a Shape

You can move a shape or transpose it along both the x and y-axes based on the XML data. For example to move a shape 200 pixels along the y-axis and 300 along the x-axis, enter the following commands in the property dialog of the shape:

```
<?shape-offset-x:300?>  
<?shape-offset-y:200?>
```

### Rotating a Shape

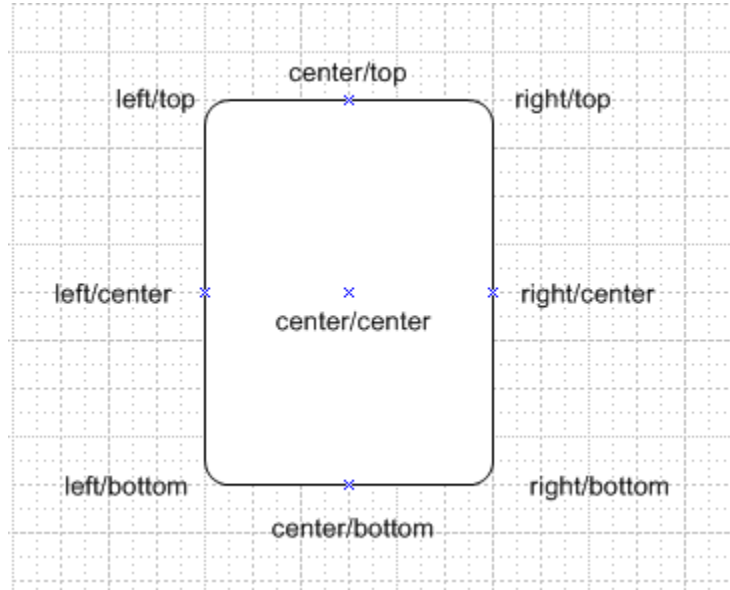
To rotate a shape about a specified axis based on the incoming data, use the following command:

```
<?shape-rotate:ANGLE;'POSITION'?>
```

where

ANGLE is the number of degrees to rotate the shape. If the angle is positive, the rotation is clockwise; if negative, the rotation is counterclockwise.

POSITION is the point about which to carry out the rotation, for example, 'left/top'. Valid values are combinations of left, right, or center with center, top, or bottom. The default is left/top. The following figure shows these valid values:



To rotate this rectangle shape about the bottom right corner, enter the following syntax:

```
<?shape-rotate:60,'right/bottom'??>
```

You can also specify an x,y coordinate within the shape itself about which to rotate.

### Skewing a Shape

You can skew a shape along its x or y axis using the following commands:

```
<?shape-skew-x:ANGLE;' POSITION'??>
<?shape-skew-y:ANGLE;' POSITION'??>
```

where

ANGLE is the number of degrees to skew the shape. If the angle is positive, the skew is to the right.

POSITION is the point about which to carry out the rotation, for example, 'left/top'. Valid values are combinations of left, right, or center with center, top, or bottom. See the figure under Rotating a Shape, page 4-34. The default is 'left/top'.

For example, to skew a shape by 30 degrees about the bottom right hand corner, enter the following:

```
<?shape-skew-x:number(.)*30;'right/bottom'??>
```

### Changing the Size of a Shape

You can change the size of a shape using the appropriate commands either along a single axis or both axes. To change a shape's size along both axes, use:

```
<?shape-size:RATIO??>
```

where RATIO is the numeric ratio to increase or decrease the size of the shape. Therefore a value of 2 would generate a shape twice the height and width of the

original. A value of 0.5 would generate a shape half the size of the original.

To change a shape's size along the x or y axis, use:

```
<?shape-size-x:RATIO?>  
<?shape-size-y:RATIO?>
```

Changing only the x or y value has the effect of stretching or shrinking the shape along an axis. This can be data driven.

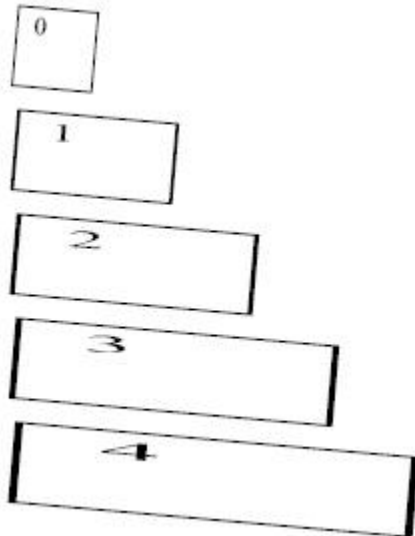
### Combining Commands

You can also combine these commands to carry out multiple transformations on a shape at one time. For example, you can replicate a shape and for each replication, rotate it by some angle and change the size at the same time.

The following example shows how to replicate a shape, move it 50 pixels down the page, rotate it by five degrees about the center, stretch it along the x-axis and add the number of the shape as text:

```
<for-each@shape:SHAPE_GROUP?>  
  <?shape-text:position()?>  
  <?shape-offset-y:position()*50?>  
  <?shape-rotate:5;'center/center'?>  
  <?shape-size-x:position()+1?>  
<end for-each?>
```

This would generate the output shown in the following figure:



### CD Ratings Example

This example demonstrates how to set up a template that will generate a star-rating based on data from an incoming XML file.

Assume the following incoming XML data:

```

<CATALOG>
  <CD>
    <TITLE>Empire Burlesque</TITLE>
    <ARTIST>Bob Dylan</ARTIST>
    <COUNTRY>USA</COUNTRY>
    <COMPANY>Columbia</COMPANY>
    <PRICE>10.90</PRICE>
    <YEAR>1985</YEAR>
    <USER_RATING>4</USER_RATING>
  </CD>
  <CD>
    <TITLE>Hide Your Heart</TITLE>
    <ARTIST>Bonnie Tylor</ARTIST>
    <COUNTRY>UK</COUNTRY>
    <COMPANY>CBS Records</COMPANY>
    <PRICE>9.90</PRICE>
    <YEAR>1988</YEAR>
    <USER_RATING>3</USER_RATING>
  </CD>
  <CD>
    <TITLE>Still got the blues</TITLE>
    <ARTIST>Gary More</ARTIST>
    <COUNTRY>UK</COUNTRY>
    <COMPANY>Virgin Records</COMPANY>
    <PRICE>10.20</PRICE>
    <YEAR>1990</YEAR>
    <USER_RATING>5</USER_RATING>
  </CD>
  <CD>
    <TITLE>This is US</TITLE>
    <ARTIST>Gary Lee</ARTIST>
    <COUNTRY>UK</COUNTRY>
    <COMPANY>Virgin Records</COMPANY>
    <PRICE>12.20</PRICE>
    <YEAR>1990</YEAR>
    <USER_RATING>2</USER_RATING>
  </CD>
</CATALOG>

```

Notice there is a USER\_RATING element for each CD. Using this data element and the shape manipulation commands, we can create a visual representation of the ratings so that the reader can compare them at a glance.

A template to achieve this is shown in the following figure:

Title	Artist	Rating
F TITLE	ARTIST	E 

The values for the fields are shown in the following table:

Field	Form Field Entry
F	<?for-each:CD?>
TITLE	<?TITLE?>
ARTIST	<?ARTIST?>
E	<?end for-each?>
(star shape)	Web Tab Entry: <pre> &lt;?for-each@shape:xdoxslt:foreach_number(\$_XDOCTX,1 ,USER_RATING,1)?&gt; &lt;?shape-offset-x:(position()-1)*25?&gt; &lt;?end for-each?&gt; </pre>

The form fields hold the simple element values. The only difference with this template is the value for the star shape. The replication command is placed in the Web tab of the Format AutoShape dialog.

In the for-each@shape command we are using a command to create a "for...next loop" construct. We specify 1 as the starting number; the value of USER\_RATING as the final number; and 1 as the step value. As the template loops through the CDs, we create an inner loop to repeat a star shape for every USER\_RATING value (that is, a value of 4 will generate 4 stars). The output from this template and the XML sample is shown in the following graphic:

Title	Artist	Rating
Empire Burlesque	Bob Dylan	★★★★
Hide Your Heart	Bonnie Tylor	★★★
Still got the blues	Gary More	★★★★★
This is US	Gary Lee	★★

### Grouped Shape Example

This example shows how to combine shapes into a group and have them react to the incoming data both individually and as a group. Assume the following XML data:

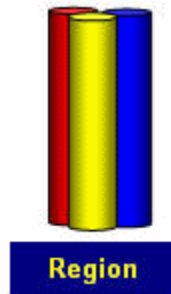


```

<SALES>
  <SALE>
    <REGION>Americas</REGION>
    <SOFTWARE>1200</SOFTWARE>
    <HARDWARE>850</HARDWARE>
    <SERVICES>2000</SERVICES>
  </SALE>
  <SALE>
    <REGION>EMEA</REGION>
    <SOFTWARE>1000</SOFTWARE>
    <HARDWARE>800</HARDWARE>
    <SERVICES>1100</SERVICES>
  </SALE>
  <SALE>
    <REGION>APAC</REGION>
    <SOFTWARE>900</SOFTWARE>
    <HARDWARE>1200</HARDWARE>
    <SERVICES>1500</SERVICES>
  </SALE>
</SALES>

```

You can create a visual representation of this data so that users can very quickly understand the sales data across all regions. Do this by first creating the composite shape in Microsoft Word that you wish to manipulate. The following figure shows a composite shape made up of four components:



The shape consists of three cylinders: red, yellow, and blue. These will represent the data elements software, hardware, and services. The combined object also contains a rectangle that is enabled to receive text from the incoming data.

The following commands are entered into the Web tab:

Red cylinder: <?shape-size-y:SOFTWARE div 1000;'left/bottom'??>

Yellow cylinder: <?shape-size-y:HARDWARE div 1000;'left/bottom'??>

Blue cylinder: <?shape-size-y:SERVICES div 1000;'left/bottom'??>

The shape-size command is used to stretch or shrink the cylinder based on the values of the elements SOFTWARE, HARDWARE, and SERVICES. The value is divided by 1000 to set the stretch or shrink factor. For example, if the value is 2000, divide that by 1000 to get a factor of 2. The shape will generate as twice its current height.

The text-enabled rectangle contains the following command in its Web tab:

<?shape-text:REGION??>

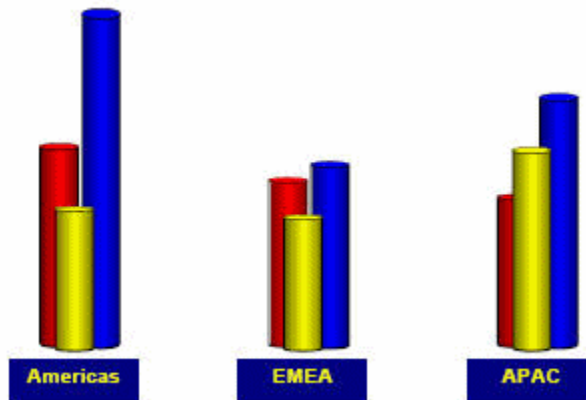
At runtime the value of the REGION element will appear in the rectangle.

All of these shapes were then grouped together and in the Web tab for the grouped object, the following syntax is added:

```
<?for-each@shape:SALE?>  
<?shape-offset-x:(position()-1)*110?>  
<?end for-each?>
```

In this set of commands, the `for-each@shape` loops over the SALE group. The `shape-offset` command moves the next shape in the loop to the right by a specific number of pixels. The expression `(position()-1)` sets the position of the object. The `position()` function returns a record counter while in the loop, so for the first shape, the offset would be  $1-1*110$ , or 0, which would place the first rendering of the object in the position defined in the template. Subsequent occurrences would be rendered at a 110 pixel offset along the x-axis (to the right).

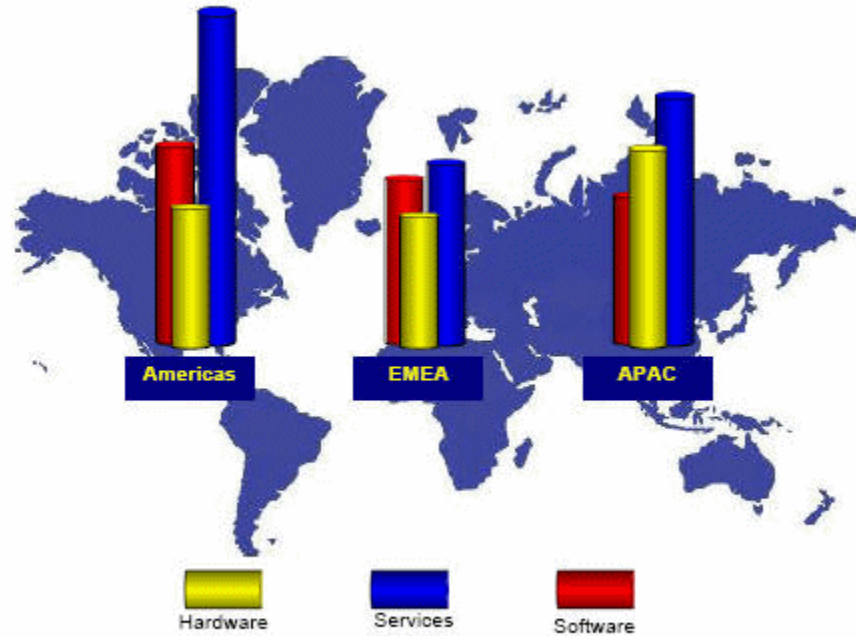
At runtime three sets of shapes will be rendered across the page as shown in the following figure:



To make an even more visually representative report, these shapes can be superimposed onto a world map. Just use the "Order" dialog in Microsoft Word to layer the map behind the grouped shapes.

**Microsoft Word 2000 Users:** After you add the background map and overlay the shape group, use the Grouping dialog to make the entire composition one group.

**Microsoft Word 2002/3 Users:** These versions of Word have an option under Tools > Options, General tab to "Automatically generate drawing canvas when inserting autosapes". Using this option removes the need to do the final grouping of the map and shapes. We can now generate a visually appealing output for our report as seen in the following figure:



## Supported Native Formatting Features

In addition to the features already listed, BI Publisher supports the following features of Microsoft Word.

### General Features

- Large blocks of text
- Page breaks

(Not supported for HTML output) To insert a page break, insert a Ctrl-Enter keystroke just before the closing tag of a group. For example if you want the template to start a new page for every Supplier in the Payables Invoice Register:

1. Place the cursor just before the Supplier group's closing `<?end for-each?>` tag.
2. Press Ctrl-Enter to insert a page break.

At runtime each Supplier will start on a new page.

Using this Microsoft Word native feature will cause a single blank page to print at the end of your report output. To avoid this single blank page, use BI Publisher's page break alias. See Special Features: Page Breaks, page 4-49.

- Page numbering

Insert page numbers into your final report by using the page numbering methods of your word processing application. For example, if you are using Microsoft Word:

1. From the **Insert** menu, select **Page Numbers...**
2. Select the **Position**, **Alignment**, and **Format** as desired.

At runtime the page numbers will be displayed as selected.

Note that page numbering is not supported for HTML output and has limited support in RTF output. After the RTF report is generated, press F9 to reset the page numbers.

- Hidden text

You can format text as "hidden" in Microsoft Word and the hidden text will be maintained in RTF output reports.

## Alignment

Use your word processor's alignment features to align text, graphics, objects, and tables.

**Note:** Bidirectional languages are handled automatically using your word processing application's left/right alignment controls.

## Tables

Supported table features include:

- Nested Tables
- Cell Alignment

You can align any object in your template using your word processing application's alignment tools. This alignment will be reflected in the final report output.

- Row spanning and column spanning

You can span both columns and rows in your template as follows:

1. Select the cells you wish to merge.
2. From the **Table** menu, select **Merge Cells**.
3. Align the data within the merged cell as you would normally.

At runtime the cells will appear merged.

- Table Autoformatting  
BI Publisher recognizes the table autoformats available in Microsoft Word.
  1. Select the table you wish to format.
  2. From the **Table** menu, select **Autoformat**.
  3. Select the desired table format.

At runtime, the table will be formatted using your selection.

- Cell patterns and colors  
You can highlight cells or rows of a table with a pattern or color.
  1. Select the cell(s) or table.
  2. From the **Table** menu, select **Table Properties**.
  3. From the **Table** tab, select the **Borders and Shading...** button.
  4. Add borders and shading as desired.

- Repeating table headers

**Note:** This feature is not supported for RTF output.

If your data is displayed in a table, and you expect the table to extend across multiple pages, you can define the header rows that you want to repeat at the start of each page.

1. Select the row(s) you wish to repeat on each page.
  2. From the **Table** menu, select **Heading Rows Repeat**.
- Prevent rows from breaking across pages.  
If you want to ensure that data within a row of a table is kept together on a page, you can set this as an option using Microsoft Word's **Table Properties**.
    1. Select the row(s) that you want to ensure do not break across a page.
    2. From the **Table** menu, select **Table Properties**.
    3. From the **Row** tab, deselect the check box "Allow row to break across pages".

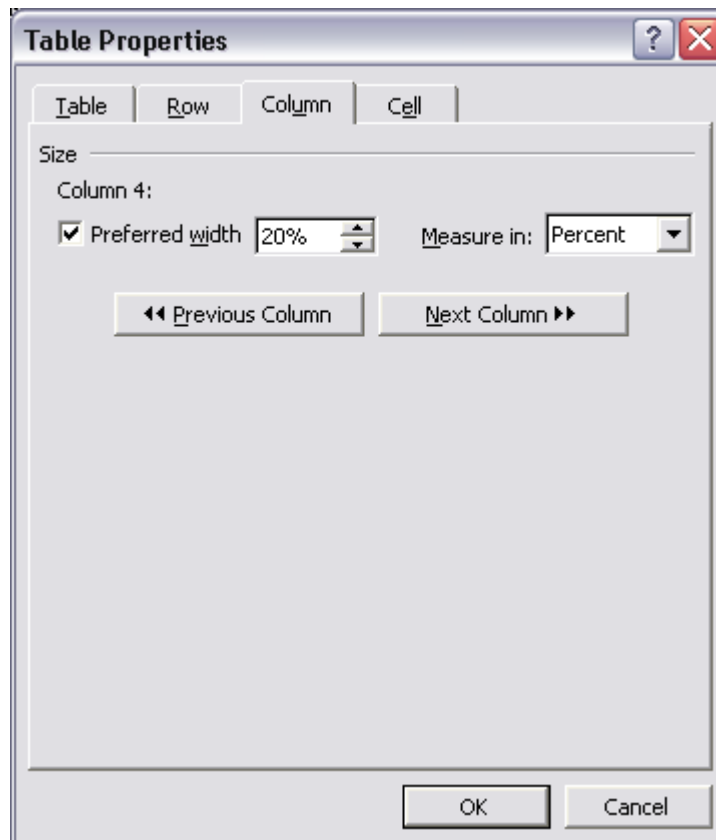
- Fixed-width columns

To set the widths of your table columns:

1. Select a column and then select **Table >Table Properties**.
2. In the **Table Properties** dialog, select the **Column** tab.
3. Enable the **Preferred width** checkbox and then enter the width as a **Percent** or in **Inches**.
4. Select the **Next Column** button to set the width of the next column.

Note that the total width of the columns must add up to the total width of the table.

The following figure shows the **Table Properties** dialog:



- Text truncation

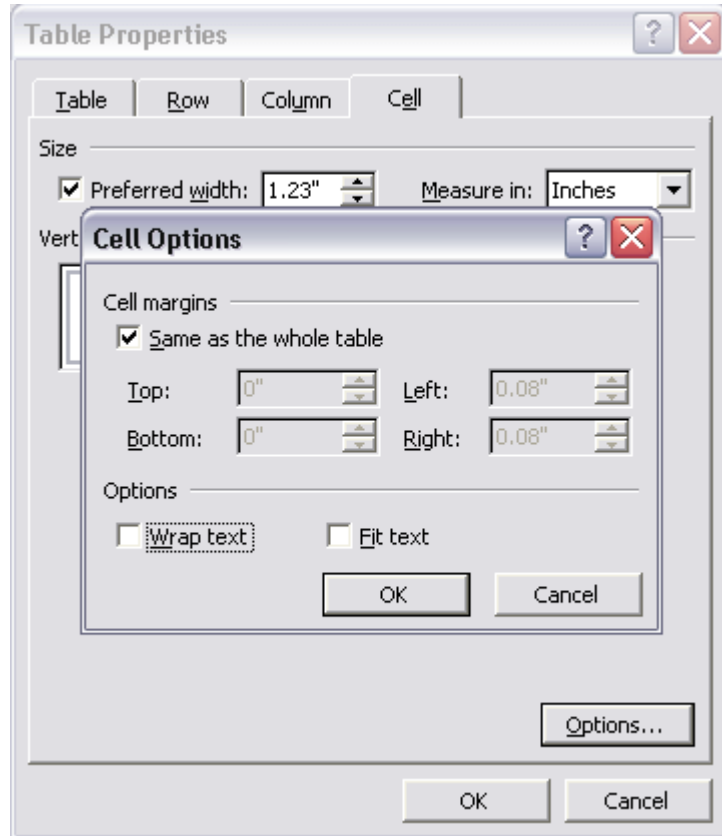
By default, if the text within a table cell will not fit within the cell, the text will be wrapped. To truncate the text instead, use the table properties dialog.

Note that table text truncation is supported for PDF and PPT outputs only.

1. Place your cursor in the cell in which you want the text truncated.
2. Right-click your mouse and select **Table Properties...** from the menu, or navigate to **Table >Table Properties...**

3. From the **Table Properties** dialog, select the **Cell** tab, then select **Options...**
4. Deselect the **Wrap Text** check box.

The following figure shows the Cell Options dialog.



An example of truncation is shown in the following graphic:

Wrap Text checked	Wrap Text unchecked
The quick brown fox jumped over the lazy river.	The quick brown fox

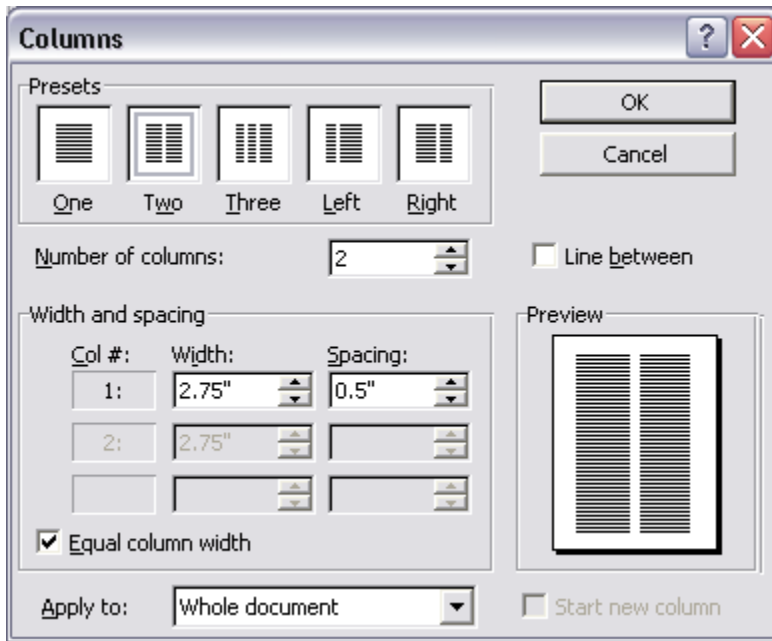
## Date Fields

Insert dates using the date feature of your word processing application. Note that this date will correspond to the publishing date, not the request run date.

## Multicolumn Page Support

BI Publisher supports Microsoft Word's Columns function to enable you to publish your output in multiple columns on a page. (Note that this is not supported for HTML output.)

Select **Format > Columns** to display the **Columns** dialog box to define the number of columns for your template. The following graphic shows the Columns dialog:



### Multicolumn Page Example: Labels

To generate address labels in a two-column format:

1. Divide your page into two columns using the Columns command.
2. Define the repeatable group in the first column. Note that you define the repeatable group only in the first column, as shown in the following figure:





F

Name	CUSTOMER_NAME
Number	CUSTOMER_NUMBER
City	CITY
State	STATE
Zip Code	Z I P _ C O D E

E

**Tip:** To prevent the address block from breaking across pages or columns, embed the label block inside a single-celled table. Then specify in the Table Properties that the row should not break across pages. See Prevent rows from breaking across pages, page 4-43.

This template will produce the following multicolumn output:

<table border="1"> <tr><td>Name</td><td>Nuts and Bolts Ltd</td></tr> <tr><td>Number</td><td>1220</td></tr> <tr><td>City</td><td>Espoo</td></tr> <tr><td>State</td><td>FI</td></tr> <tr><td>Zip Code</td><td>     </td></tr> </table>	Name	Nuts and Bolts Ltd	Number	1220	City	Espoo	State	FI	Zip Code		<table border="1"> <tr><td>Name</td><td>Big Co</td></tr> <tr><td>Number</td><td>1221</td></tr> <tr><td>City</td><td>Helsinki</td></tr> <tr><td>State</td><td>FI</td></tr> <tr><td>Zip Code</td><td>     </td></tr> </table>	Name	Big Co	Number	1221	City	Helsinki	State	FI	Zip Code	
Name	Nuts and Bolts Ltd																				
Number	1220																				
City	Espoo																				
State	FI																				
Zip Code																					
Name	Big Co																				
Number	1221																				
City	Helsinki																				
State	FI																				
Zip Code																					
<table border="1"> <tr><td>Name</td><td>My Company</td></tr> <tr><td>Number</td><td>1220</td></tr> <tr><td>City</td><td>Espoo</td></tr> <tr><td>State</td><td>FI</td></tr> <tr><td>Zip Code</td><td>     </td></tr> </table>	Name	My Company	Number	1220	City	Espoo	State	FI	Zip Code		<table border="1"> <tr><td>Name</td><td>Small Co</td></tr> <tr><td>Number</td><td>1221</td></tr> <tr><td>City</td><td>Helsinki</td></tr> <tr><td>State</td><td>FI</td></tr> <tr><td>Zip Code</td><td>     </td></tr> </table>	Name	Small Co	Number	1221	City	Helsinki	State	FI	Zip Code	
Name	My Company																				
Number	1220																				
City	Espoo																				
State	FI																				
Zip Code																					
Name	Small Co																				
Number	1221																				
City	Helsinki																				
State	FI																				
Zip Code																					

## Background and Watermark Support

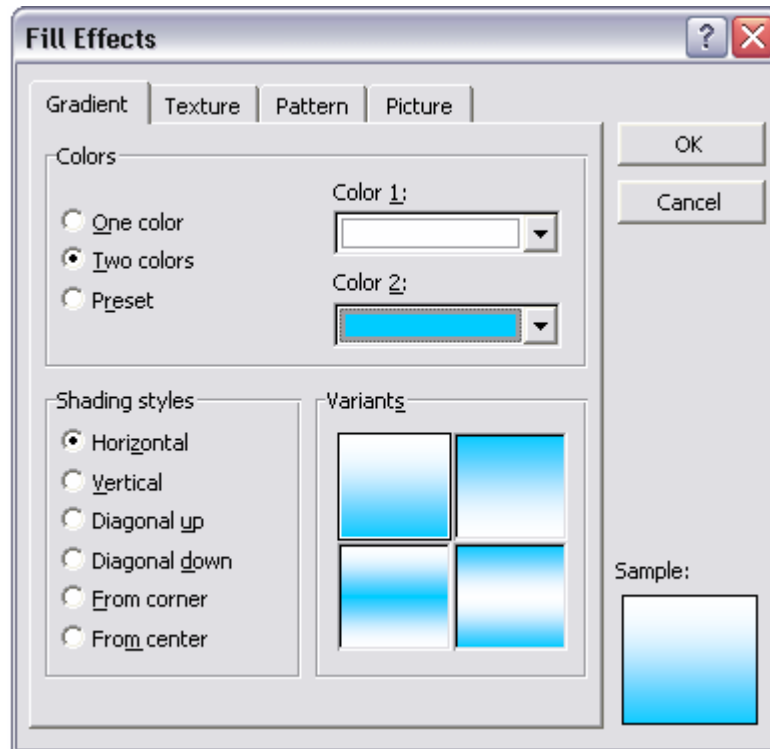
BI Publisher supports the "Background" feature in Microsoft Word. You can specify a single, graduated color or an image background for your template to be displayed in the PDF output. Note that this feature is supported for PDF output and PPT output only.

To add a background to your template, use the Format > Background menu option.

### Add a Background Using Microsoft Word 2000

From the Background pop up menu, you can:

- Select a single color background from the color palette
- Select Fill Effects to open the Fill Effects dialog. The Fill Effects dialog is shown in the following figure:



From this dialog select one of the following supported options:

- Gradient - this can be either one or two colors
- Texture - choose one of the textures provided, or load your own
- Pattern - select a pattern and background/foreground colors
- Picture - load a picture to use as a background image

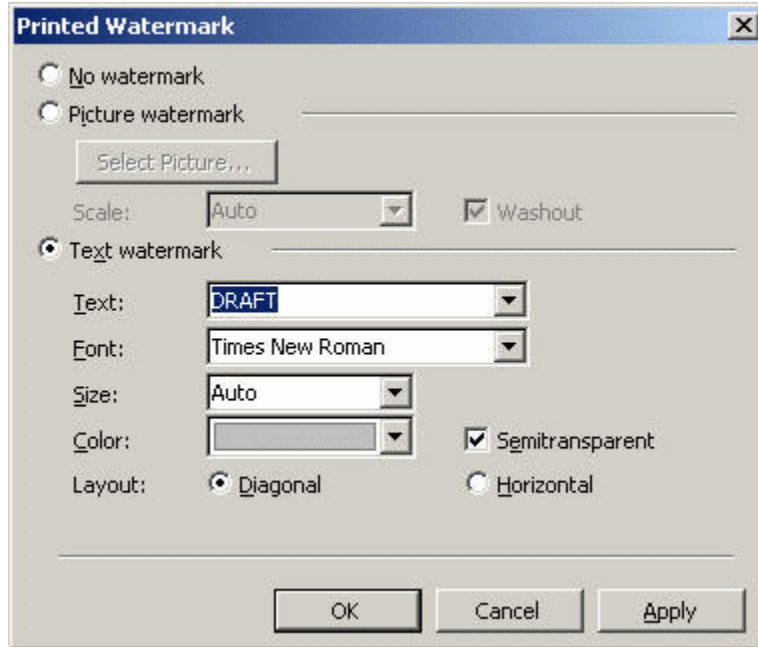
#### **Add a Text or Image Watermark Using Microsoft Word 2002 or later**

These versions of Microsoft Word allow you to add either a text or image watermark.

Use the Format > Background > Printed Watermark dialog to select either:

- Picture Watermark - load an image and define how it should be scaled on the document
- Text Watermark - use the predefined text options or enter your own, then specify the font, size and how the text should be rendered.

The following figure shows the Printed Watermark dialog completed to display a text watermark:



## Template Features

### Page Breaks

**Note:** Page breaks are supported for PDF, RTF, and PPT output. Page breaks are not supported for HTML output.

To create a page break after the occurrence of a specific element use the "split-by-page-break" alias. This will cause the report output to insert a hard page break between every instance of a specific element.

To insert a page break between each occurrence of a group, insert the "split-by-page-break" form field within the group immediately before the `<?end for-each?>` tag that closes the group. In the Help Text of this form field enter the syntax:

```
<?split-by-page-break:??>
```

#### Example

For the following XML, assume you want to create a page break for each new supplier:

```

<SUPPLIER>
  <NAME>My Supplier</NAME>
  <INVOICES>
    <INVOICE>
      <INVNUM>10001-1</INVNUM>
      <INVDATE>1-Jan-2005</INVDATE>
      <INVAMT>100</INVOICEAMT>
    </INVOICE>
    <INVOICE>
      <INVNUM>10001-2</INVNUM>
      <INVDATE>10-Jan-2005</INVDATE>
      <INVAMT>200</INVOICEAMT>
    </INVOICE>
  </INVOICES>
</SUPPLIER>
<SUPPLIER>
  <NAME>My Second Supplier</NAME>
  <INVOICES>
    <INVOICE>
      <INVNUM>10001-1</INVNUM>
      <INVDATE>11-Jan-2005</INVDATE>
      <INVAMT>150</INVOICEAMT>
    </INVOICE>
  ...

```

In the template sample shown in the following figure, the field called PageBreak contains the split-by-page-break syntax:

FE

Supplier: Supplier 1

Invoice Number	Invoice Date	Amount	Running Total
FE10001-1	1-Jan-2005	100.00	100.00EFE

PageBreak EFE

Place the PageBreak field with the `<?split-by-page-break:?>` syntax immediately before the `<?end for-each?>` field. The PageBreak field sits inside the end of the SUPPLIER loop. This will ensure a page break is inserted before the occurrence of each new supplier. This method avoids the ejection of an extra page at the end of the group when using the native Microsoft Word page break after the group.

## Initial Page Number

**Note:** Initial page number is supported for PDF and PPT output. It is not supported for HTML and RTF output.

Some reports require that the initial page number be set at a specified number. For example, monthly reports may be required to continue numbering from month to month. BI Publisher allows you to set the page number in the template to support this requirement.

Use the following syntax in your template to set the initial page number:

```
<?initial-page-number:pagenumber?>
```

where *pagenumber* is the XML element or parameter that holds the numeric value.

BI Publisher also supports continuing the page number from a previous section. The default behavior of a new section in a document is to reset the page numbering. However, if your report requires that the page numbering continue into the next section, use the following command:

```
<?initial-page-number:'auto'?>
```

This command will allow the continuation of the page numbering from the previous section.

### **Example 1 - Set page number from XML data element**

If your XML data contains an element to carry the initial page number, for example:

```
<REPORT>
  <PAGESTART>200<\PAGESTART>
  . . . .
</REPORT>
```

Enter the following in your template:

```
<?initial-page-number:PAGESTART?>
```

Your initial page number will be the value of the PAGESTART element, which in this case is 200.

### **Example 2 - Set page number by passing a parameter value**

If you define a parameter called PAGESTART, you can pass the initial value by calling the parameter.

Enter the following in your template:

```
<?initial-page-number:$PAGESTART?>
```

**Note:** You must first declare the parameter in your template. See *Defining Parameters in Your Template*, page 4-93.

## **Last Page Only Content**

**Note:** This feature is supported for PDF and PPT output only.

BI Publisher supports the Microsoft Word functionality to specify a different page layout for the first page, odd pages, and even pages. To implement these options, simply select **Page Setup** from the **File** menu, then select the **Layout** tab. BI Publisher will recognize the settings you make in this dialog.

However, Microsoft Word does not provide settings for a different last page only. This is useful for documents such as checks, invoices, or purchase orders on which you may want the content such as the check or the summary in a specific place only on the last page.

BI Publisher provides this ability. To utilize this feature, you must:

1. Create a section break in your template to ensure the content of the final page is separated from the rest of the report.
2. Insert the following syntax on the final page:

```
<?start@last-page:body?>
<?end body?>
```

Any content on the page that occurs above or below these two tags will appear only on the last page of the report. Also, note that because this command explicitly specifies the content of the final page, any desired headers or footers previously defined for the report must be reinserted on the last page.

### Example

This example uses the last page only feature for a report that generates an invoice listing with a summary to appear at the bottom of the last page.

Assume the following XML:

```
<?xml version="1.0" encoding="WINDOWS-1252"?>
<INVOICELIST>
  <VENDOR>
    <VENDOR_NAME>Nuts and Bolts Limited</VENDOR_NAME>
    <ADDRESS>1 El Camino Real, Redwood City, CA 94065</ADDRESS>
    <INVOICE>
      <INV_TYPE>Standard</INV_TYPE>
      <INVOICE_NUM>981110</INVOICE_NUM>
      <INVOICE_DATE>10-NOV-04</INVOICE_DATE>
      <INVOICE_CURRENCY_CODE>EUR</INVOICE_CURRENCY_CODE>
      <ENT_AMT>122</ENT_AMT>
      <ACCTD_AMT>122</ACCTD_AMT>
      <VAT_CODE>VAT22%</VAT_CODE>
    </INVOICE>
    <INVOICE>
      <INV_TYPE>Standard</INV_TYPE>
      <INVOICE_NUM>100000</INVOICE_NUM>
      <INVOICE_DATE>28-MAY-04</INVOICE_DATE>
      <INVOICE_CURRENCY_CODE>FIM</INVOICE_CURRENCY_CODE>
      <ENT_AMT>122</ENT_AMT>
      <ACCTD_AMT>20.33</ACCTD_AMT>
      <VAT_CODE>VAT22%</VAT_CODE>
    </INVOICE>
  </VENDOR>
  <VENDOR>
    ...
  <INVOICE>
    ...
  </INVOICE>
</VENDOR>
<SUMMARY>
  <SUM_ENT_AMT>61435</SUM_ENT_AMT>
  <SUM_ACCTD_AMT>58264.68</SUM_ACCTD_AMT>
  <TAX_CODE>EU22%</TAX_CODE>
</SUMMARY>
</INVOICELIST>
```

The report should show each VENDOR and their INVOICE data with a SUMMARY section that appears only on the last page, placed at the bottom of the page. The

template for this is shown in the following figure:

**Template Page One**

F  
Vendor: VENDOR\_NAME  
Address: ADDRESS

Invoice Type	Invoice Num	Invoice Date	Invoice Currency	Entered Amount	Accounted Amount
F Invoice	120000	01-Jan-2006	USD	100	100 E

E

<<insert section break>

Insert a Microsoft Word section break (type: next page) on the first page of the template. For the final page, insert new line characters to position the summary table at the bottom of the page. The summary table is shown in the following figure:

**Last Page Only Layout**

Last Page Placeholder

**Tax Summary**

Tax Code	Entered Amount	Accounted Amount
F VAT 18.5	100	100 E

In this example:

- The F and E components contain the for-each grouping statements.
- The grayed report fields are placeholders for the XML elements.
- The "Last Page Placeholder" field contains the syntax:

```
<?start@last-page:body?><?end body?>
```

to declare the last page layout. Any content above or below this statement will appear on the last page only. The content above the statement is regarded as the

header and the content below the statement is regarded as the footer.

If your reports contains headers and footers that you want to carry over onto the last page, you must reinsert them on the last page. For more information about headers and footers see Defining Headers and Footers, page 4-16.

You must insert a section break (type: next page) into the document to specify the last page layout. This example is available in the samples folder of the Oracle BI Publisher Template Builder for Word installation.

Because the default behavior of a new section in a document is to reset the page numbering the page number on the last page will be reset. To continue the page numbering from the previous section use the following command:

```
<?initial-page-number:'auto'?>
```

This command will allow the continuation of the page numbering from the previous section.

It is important to note that if the report is only one page in length, the first page layout will be used. If your report requires that a single page report should default to the last page layout (such as in a check printing implementation) then you can use the following alternate syntax for the "Last Page Placeholder" on the last page:

```
<?start@last-page-first:body?><?end body?>
```

Substituting this syntax will result in the last page layout for reports that are only one page long.

## End on Even or End on Odd Page

**Note:** This feature is supported for PDF and PDF output only. It is not supported for RTF and HTML output.

If your report has different odd and even page layouts, you may want to force your report to end specifically on an odd or even page. For example, you may include the terms and conditions of a purchase order in the footer of your report using the different odd/even footer functionality (see Different First Page and Different Odd and Even Page Support, page 4-17) and you want to ensure that the terms and conditions are printed on the final page.

Or, you may have binding requirements to have your report end on an even page, without specific layout.

### To end on an even page with layout:

Insert the following syntax in a form field in your template:

```
<?section:force-page-count;'end-on-even-layout'?>
```

### To end on an odd page layout:

```
<?section:force-page-count;'end-on-odd-layout'?>
```



If you do not have layout requirements for the final page, but would like a blank page ejected to force the page count to the preferred odd or even, use the following syntax:

```
<?section:force-page-count;'end-on-even'??>
```

or

```
<?section:force-page-count;'end-on-odd'??>
```

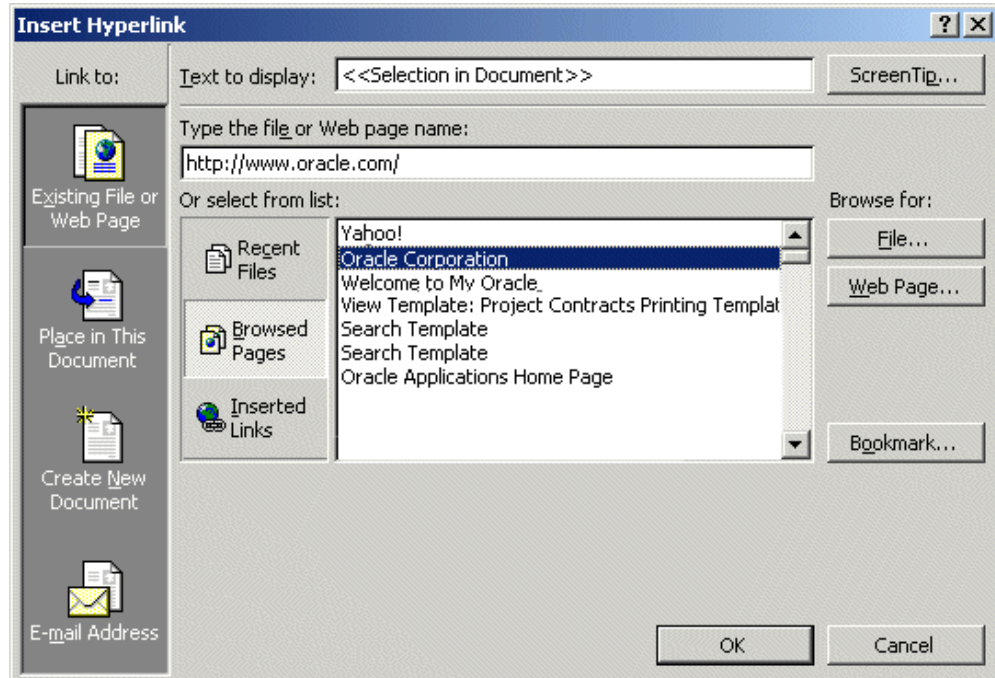
## Hyperlinks

**Note:** Hyperlinks are supported for PDF, RTF, HTML, PPT, and Excel output.

BI Publisher supports several different types of hyperlinks. The hyperlinks can be fixed or dynamic and can link to either internal or external destinations. Hyperlinks can also be added to shapes.

- To insert static hyperlinks to either text or a shape, use your word processing application's insert hyperlink feature:
  1. Select the text or shape.
  2. Use the right-mouse menu to select **Hyperlink**; or, select **Hyperlink** from the **Insert** menu.
  3. Enter the URL using any of the methods provided on the **Insert Hyperlink** dialog box.

The following screenshot shows the insertion of a static hyperlink using Microsoft Word's **Insert Hyperlink** dialog box.



- If your input XML data includes an element that contains a hyperlink or part of one, you can create dynamic hyperlinks at runtime. In the **Type the file or Web page name** field of the **Insert Hyperlink** dialog box, enter the following syntax:

{URL\_LINK}

where URL\_LINK is the incoming data element name.

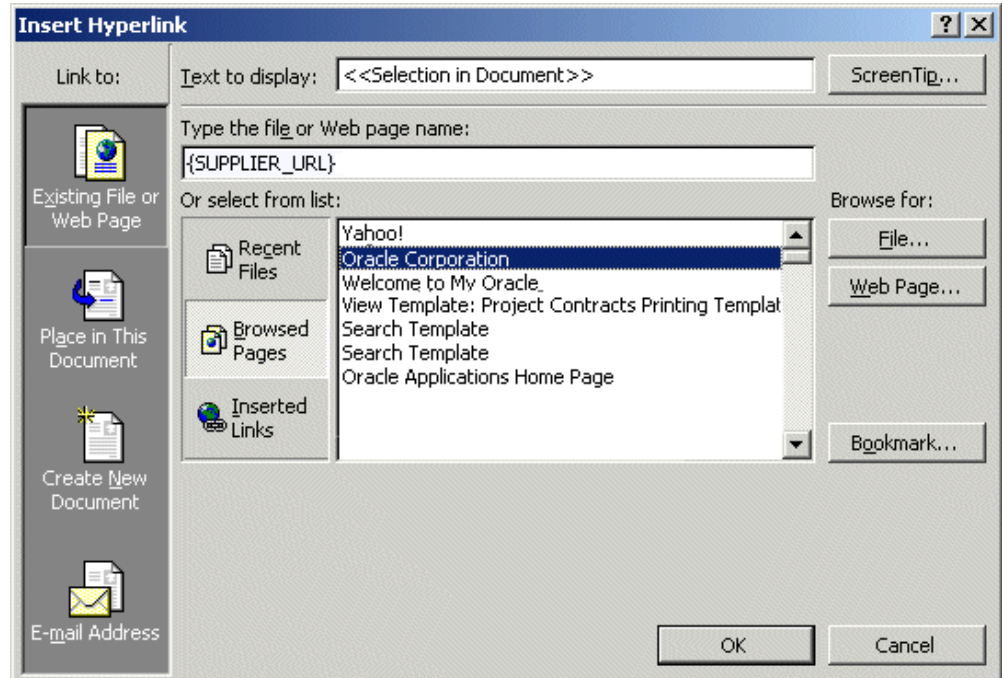
If you have a fixed URL that you want to add elements from your XML data file to construct the URL, enter the following syntax:

http://www.oracle.com?product={PRODUCT\_NAME}

where PRODUCT\_NAME is the incoming data element name.

In both these cases, at runtime the dynamic URL will be constructed.

The following figure shows the insertion of a dynamic hyperlink using Microsoft Word's **Insert Hyperlink** dialog box. The data element SUPPLIER\_URL from the incoming XML file will contain the hyperlink that will be inserted into the report at runtime.



- You can also pass parameters at runtime to construct a dynamic URL.

Enter the parameter and element names surrounded by braces to build up the URL as follows:

```
{$SERVER_URL}{$REPORT}/cstid={CUSTOMER_ID}
```

where `SERVER_URL` and `REPORT` are parameters passed to the template at runtime (note the \$ sign) and `CUSTOMER_ID` is an XML data element. This link may render as:

```
http://myserver.domain:8888/CustomReport/cstid=1234
```

To add the target attribute to a URL, add the following to the URL string:

```
??target=_target_value
```

For example:

```
http://www.oracle.com??target=_top
```

Values for the target attribute are:

- `_top`
- `_blank`
- `_self`
- `_parent`

- *framename*

You can pass in the value of target dynamically, using the following syntax:

```
http://www.oracle.com/index.html??target={myTarget}
```

where *myTarget* is the name of the parameter that holds the value.

### Inserting Internal Links

Insert internal links into your template using Microsoft Word's Bookmark feature.

1. Position your cursor in the desired destination in your document.
2. Select **Insert >Bookmark...**
3. In the **Bookmark** dialog, enter a name for this bookmark, and select **Add**.
4. Select the text or shape in your document that you want to link back to the Bookmark target.
5. Use the right-mouse menu to select **Hyperlink**; or select **Hyperlink** from the **Insert** menu.
6. On the **Insert Hyperlink** dialog, select **Bookmark**.
7. Choose the bookmark you created from the list.

At runtime, the link will be maintained in your generated report.

### Table of Contents

**Note:** Table of contents feature is supported for PDF and PPT output. RTF support is limited: After report generation, the user must press F9 to reset the page numbers.

BI Publisher supports the table of contents generation feature of the RTF specification. Follow your word processing application's procedures for inserting a table of contents.

BI Publisher also provides the ability to create dynamic section headings in your document from the XML data. You can then incorporate these into a table of contents.

To create dynamic headings:

1. Enter a placeholder for the heading in the body of the document, and format it as a "Heading", using your word processing application's style feature. You cannot use form fields for this functionality.

For example, you want your report to display a heading for each company reported. The XML data element tag name is `<COMPANY_NAME>`. In your

template, enter `<?COMPANY_NAME?>` where you want the heading to appear. Now format the text as a Heading.

2. Create a table of contents using your word processing application's table of contents feature.

At runtime the TOC placeholders and heading text will be substituted.

## Generating Bookmarks in PDF Output

If you have defined a table of contents in your RTF template, you can use your table of contents definition to generate links in the Bookmarks tab in the navigation pane of your output PDF. The bookmarks can be either static or dynamically generated.

**Important:** Note that bookmark support in RTF templates is limited to a single-point bookmark. This is to allow link (Goto) functionality within the document. Arrays in bookmarks are not supported.

For information on creating the table of contents, see Table of Contents, page 4-58.

- To create links for a static table of contents:

Enter the syntax:

```
<?copy-to-bookmark:??>
```

directly above your table of contents and

```
<?end copy-to-bookmark:??>
```

directly below the table of contents.

- To create links for a dynamic table of contents:

Enter the syntax:

```
<?convert-to-bookmark:??>
```

directly above the table of contents and

```
<?end convert-to-bookmark:??>
```

directly below the table of contents.

To control the initial state of the bookmark when the PDF file is opened, use the following command:

```
<?collapse-bookmark:state;level?>
```

where

state can have the following values:

- hide - collapses the table of contents entries

- show - expands the table of contents entries

and

level sets the table of contents collapse level. For example: "1" collapses the first level of entries in the table of contents; "2" collapses the and first and second level entries.

Use this command with `<?copy-to-bookmark:?>` and `<?convert-to-bookmark:?>` as shown in the following examples:

- To create a static table of contents that hides level 1 and level 2 of the table of contents entries, enter the following:

```
<?copy-to-bookmark:?>
<?collapse-bookmark:hide;2?>
```

directly above your table of contents and

```
<?end copy-to-bookmark:?>
```

directly below the table of contents.

- To create links for a dynamic table of contents that shows levels 1 and 2 of the table of contents expanded, enter the following:

```
<?convert-to-bookmark:>
<?collapse-bookmark:show;2?>
```

directly above the table of contents and

```
<?end convert-to-bookmark:?>
```

directly below the table of contents.

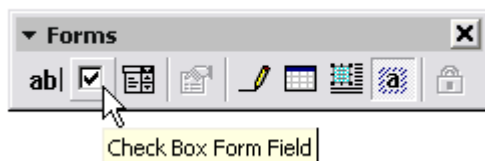
## Check Boxes

**Note:** Check boxes are supported in PDF output only.

You can include a check box in your template that you can define to display as checked or unchecked based on a value from the incoming data.

To define a check box in your template:

1. Position the cursor in your template where you want the check box to display, and select the Check Box Form Field from the Forms tool bar (shown in the following figure).

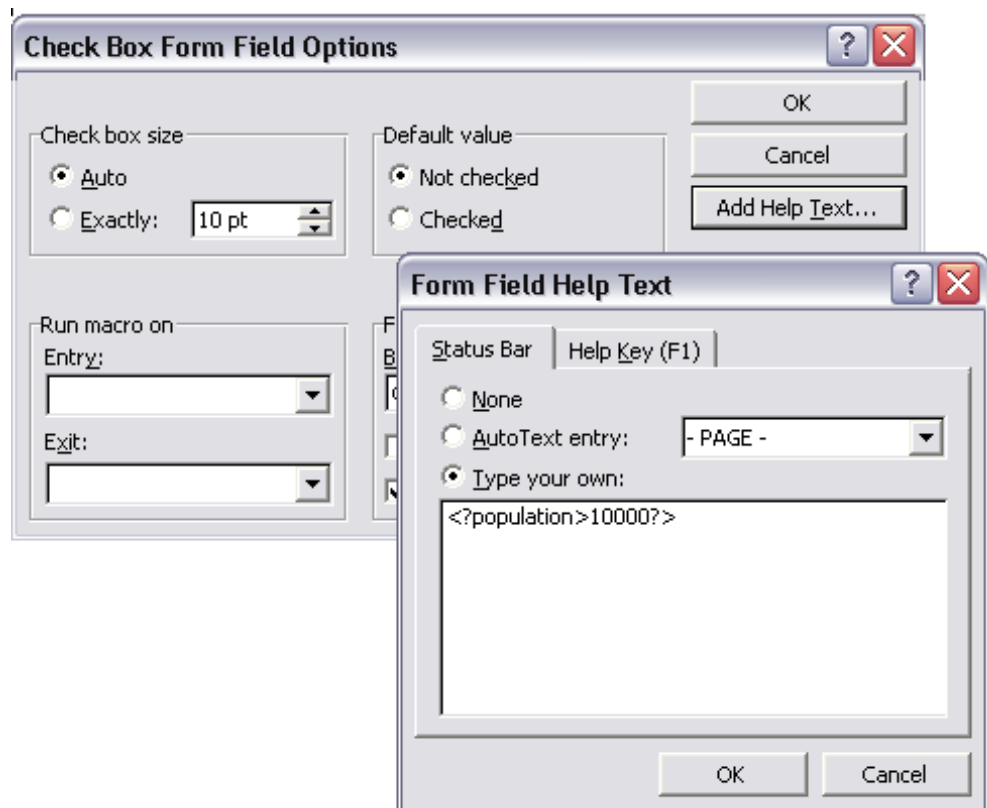


2. Right-click the field to open the **Check Box Form Field Options** dialog.
3. Specify the **Default value** as either Checked or Not Checked.
4. In the Form Field Help Text dialog, enter the criteria for how the box should behave. This must be a boolean expression (that is, one that returns a true or false result).

For example, suppose your XML data contains an element called <population>. You want the check box to appear checked if the value of <population> is greater than 10,000. Enter the following in the help text field:

```
<?population>10000?>
```

This is displayed in the following figure:



Note that you do not have to construct an "if" statement. The expression is treated as an "if" statement.

See the next section for a sample template using a check box.

## Drop Down Lists

BI Publisher allows you to use the drop-down form field to create a cross-reference in your template from your XML data to some other value that you define in the

drop-down form field.

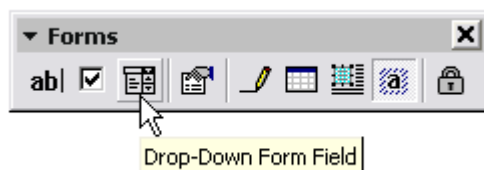
For example, suppose you have the following XML:

```
<countries>
  <country>
    <name>Chad</name>
    <population>7360000</population>
    <continentIndex>5</continentIndex>
  </country>
  <country>
    <name>China</name>
    <population>1265530000</population>
    <continentIndex>1</continentIndex>
  </country>
  <country>
    <name>Chile</name>
    <population>14677000</population>
    <continentIndex>3</continentIndex>
  </country>
  . . .
</countries>
```

Notice that each `<country>` entry has a `<continentindex>` entry, which is a numeric value to represent the continent. Using the drop-down form field, you can create an index in your template that will cross-reference the `<continentindex>` value to the actual continent name. You can then display the name in your published report.

To create the index for the continent example:

1. Position the cursor in your template where you want the value from the drop-down list to display, and select the Drop-Down Form Field from the Forms tool bar (shown in the following figure).



2. Right-click the field to display the **Drop-Down Form Field Options** dialog.
3. Add each value to the **Drop-down item** field and the click **Add** to add it to the **Items in drop-down list** group. The values will be indexed starting from one for the first, and so on. For example, the list of continents will be stored as follows:



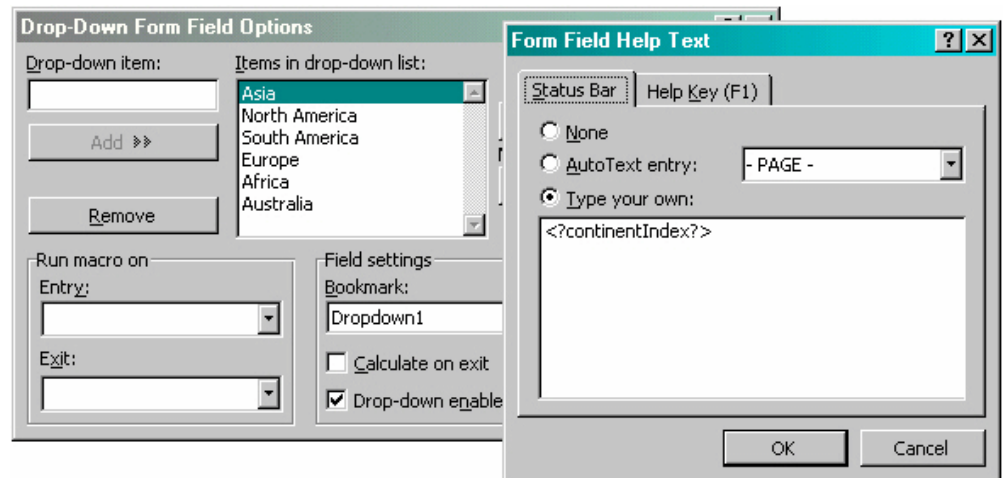
Index	Value
1	Asia
2	North America
3	South America
4	Europe
5	Africa
6	Australia

- Now use the Help Text box to enter the XML element name that will hold the index for the drop-down field values.

For this example, enter

```
<?continentIndex?>
```

The following figure shows the **Drop-Down Form Field Options** dialogs for this example:



Using the check box and drop-down list features, you can create a report to display population data with check boxes to demonstrate figures that reach a certain limit. An example is shown in the following figure:

Country	Population	more than 10M?	Continent
Chad	7,360,000	<input type="checkbox"/>	Africa
China	1,265,530,000	<input checked="" type="checkbox"/>	Asia
Chile	14,677,000	<input checked="" type="checkbox"/>	South America
Sweden	8,887,000	<input type="checkbox"/>	Europe
United States	270,312,000	<input checked="" type="checkbox"/>	North America
New Zealand	3,625,000	<input type="checkbox"/>	Australia

The template to create this report is shown in the next figure:

Country	Population	more than 10M?	Continent
FE China	1,000,000	<input type="checkbox"/>	Asia EFE

where the fields have the following values:

Field	Form Field Entry	Description
FE	<?for-each:country?>	Begins the <code>country</code> repeating group.
China	<?name?>	Placeholder for the <code>name</code> element.
1,000,000	<?population?>	Placeholder for the <code>population</code> element.
(check box)	<?population>1000000?>	Establishes the condition for the check box. If the value for the <code>population</code> element is greater than 1,000,000, the check box will display as checked.
Asia	<?continentIndex?>	The drop-down form field for the <code>continentIndex</code> element. See the preceding description for its contents. At runtime, the value of the XML element is replaced with the value it is cross-referenced to in the drop-down form field.
EFE	<?end for-each?>	Ends the <code>country</code> group.

## Conditional Formatting

**Note:** See also Inserting a Conditional Region, page 5-28 and Inserting a Conditional Format, page 5-29 in the chapter "Creating RTF Templates Using the Template Builder for Word."

Conditional formatting occurs when a formatting element appears only when a certain

condition is met. BI Publisher supports the usage of simple "if" statements, as well as more complex "choose" expressions.

The conditional formatting that you specify can be XSL or XSL:FO code, or you can specify actual RTF objects such as a table or data. For example, you can specify that if reported numbers reach a certain threshold, they will display shaded in red. Or, you can use this feature to hide table columns or rows depending on the incoming XML data.

## If Statements

Use an if statement to define a simple condition; for example, if a data field is a specific value.

1. Insert the following syntax to designate the beginning of the conditional area.

```
<?if:condition?>
```

2. Insert the following syntax at the end of the conditional area: `<?end if?>`.

For example, to set up the Payables Invoice Register to display invoices only when the Supplier name is "Company A", insert the syntax `<?if:VENDOR_NAME='COMPANY A'?>` before the Supplier field on the template.

Enter the `<?end if?>` tag after the invoices table.

This example is displayed in the figure below. Note that you can insert the syntax in form fields, or directly into the template.

The screenshot shows a template structure for a Suppliers group. It starts with a header 'Group: Suppliers' followed by a conditional statement `<?if:VENDOR_NAME='Company A'?>`. Below this, the text 'Supplier: Supplier 1' is displayed. A table with two columns, 'Invoice Num' and 'Invoice Date', is shown. The first row of the table contains the data 'Group:Invoices 1234566' and '1-Jan-2004'. Below the table, there is a blank line, followed by the closing conditional statement `<?end if?>`, and finally the footer 'End:Suppliers'.

Invoice Num	Invoice Date
Group:Invoices 1234566	1-Jan-2004

## If Statements in Boilerplate Text

Assume you want to incorporate an "if" statement into the following free-form text:

The program was (not) successful.

You only want the "not" to display if the value of an XML tag called <SUCCESS> equals "N".

To achieve this requirement, you must use the BI Publisher context command to place the if statement into the inline sequence rather than into the block (the default placement).

**Note:** For more information on context commands, see *Using Context Commands*, page 4-129.

For example, if you construct the code as follows:

```
The program was <?if:SUCCESS='N'?>not<?end if?> successful.
```

The following undesirable result will occur:

```
The program was  
not  
successful.
```

because BI Publisher applies the instructions to the block by default. To specify that the if statement should be inserted into the inline sequence, enter the following:

```
The program was <?if@inlines:SUCCESS='N'?>not<?end if?>  
successful.
```

This construction will result in the following display:

```
The program was successful.
```

If SUCCESS does not equal 'N';

or

```
The program was not successful.
```

If SUCCESS equals 'N'.

## If-then-Else Statements

BI Publisher supports the common programming construct "if-then-else". This is extremely useful when you need to test a condition and conditionally show a result. For example:

```
IF X=0 THEN  
  Y=2  
ELSE  
  Y=3  
END IF
```

You can also nest these statements as follows:

```

IF X=0 THEN
  Y=2
ELSE
  IF X=1 THEN
    Y=10
  ELSE Y=100
END IF

```

Use the following syntax to construct an if-then-else statement in your RTF template:

```
<?xdoxfx:if element_condition then result1 else result2 end if?>
```

For example, the following statement tests the AMOUNT element value. If the value is greater than 1000, show the word "Higher"; if it is less than 1000, show the word "Lower"; if it is equal to 1000, show "Equal":

```

<?xdoxfx:if AMOUNT > 1000 then 'Higher'
  else
    if AMOUNT < 1000 then 'Lower'
      else
        'Equal'
      end if?
end if?>

```

## Choose Statements

Use the `choose`, `when`, and `otherwise` elements to express multiple conditional tests. If certain conditions are met in the incoming XML data then specific sections of the template will be rendered. This is a very powerful feature of the RTF template. In regular XSL programming, if a condition is met in the `choose` command then further XSL code is executed. In the template, however, you can actually use visual widgets in the conditional flow (in the following example, a table).

Use the following syntax for these elements:

```

<?choose:??>
<?when:expression?>
<?otherwise?>

```

### "Choose" Conditional Formatting Example

This example shows a `choose` expression in which the display of a row of data depends on the value of the fields `EXEMPT_FLAG` and `POSTED_FLAG`. When the `EXEMPT_FLAG` equals "^", the row of data will render light gray. When `POSTED_FLAG` equals "\*" the row of data will render shaded dark gray. Otherwise, the row of data will render with no shading.

In the following figure, the form field default text is displayed. The form field help text entries are shown in the table following the example.

<b>Column Legend:</b>	<table border="1"> <tr> <td style="background-color: #cccccc;"></td> <td>'Not Posted'</td> </tr> <tr> <td style="background-color: #808080;"></td> <td>'Reduces Available Exemption Limit'</td> </tr> </table>		'Not Posted'		'Reduces Available Exemption Limit'
	'Not Posted'				
	'Reduces Available Exemption Limit'				

	Tax Code	Taxable Recoverable	Taxable Non-Recoverable	Recoverable Tax	Tax Non-Recoverable	Total
	<pre> &lt;Grp:VAT   &lt;Choose     &lt;When EXEMPT_FLAG='^'       VAT 15%   1000   1000   1000   1000   1000     End When&gt;     &lt;When POSTED_FLAG='*'       VAT 15%   1000   1000   1000   1000   1000     End When&gt;     Otherwise       VAT 15%   1000   1000   1000   1000   1000]     End Otherwise]   End Choose&gt; End VAT&gt; </pre>					

Default Text Entry in Example Form Field	Help Text Entry in Form Field
<Grp:VAT	<pre>&lt;?for-each:G_VAT?&gt;</pre> <p>starts the G_VAT group</p>
<Choose	<pre>&lt;?choose:??&gt;</pre> <p>opens the choose statement</p>
<When EXEMPT_FLAG='^'	<pre>&lt;?when: EXEMPT_FLAG='^'?&gt;</pre> <p>tests the EXEMPT_FLAG element, if true, use the first table shown</p>
End When>	<pre>&lt;?end when?&gt;</pre> <p>ends the EXEMPT_FLAG test</p>
<When POSTED_FLAG='*'	<pre>&lt;?when: POSTED_FLAG='*'?&gt;</pre> <p>tests the POSTED_FLAG element, if true, use the table following</p>
End When>	<pre>&lt;?end when?&gt;</pre> <p>ends the POSTED_FLAG test</p>

Default Text Entry in Example Form Field	Help Text Entry in Form Field
Otherwise	<?otherwise:?>  If none of above are true then use the following table
End Otherwise>	<?end otherwise?>  ends the otherwise statement
End Choose>	<?end choose?>  ends the choose statement
End Vat>	<?end for-each?>  ends the G_VAT group

## Column Formatting

You can conditionally show and hide columns of data in your document output. The following example demonstrates how to set up a table so that a column is only displayed based on the value of an element attribute.

This example will show a report of a price list, represented by the following XML:

```
<items type="PUBLIC"> <! - can be marked 'PRIVATE' - >
  <item>
    <name>Plasma TV</name>
    <quantity>10</quantity>
    <price>4000</price>
  </item>
  <item>
    <name>DVD Player</name>
    <quantity>3</quantity>
    <price>300</price>
  </item>
  <item>
    <name>VCR</name>
    <quantity>20</quantity>
    <price>200</price>
  </item>
  <item>
    <name>Receiver</name>
    <quantity>22</quantity>
    <price>350</price>
  </item>
</items>
```

Notice the `type` attribute associated with the `items` element. In this XML it is marked as "PUBLIC" meaning the list is a public list rather than a "PRIVATE" list. For the "public" version of the list we do not want to show the quantity column in the output,

but we want to develop only one template for both versions based on the list type.

The following figure is a simple template that will conditionally show or hide the quantity column:

<b>Name</b>	<b>IF</b> Quantity <b>end-if</b>	<b>Price</b>
grp.ItemPlasma TV	<b>IF</b> 20 <b>end-if</b>	1,000.00end grp

The following table shows the entries made in the template for the example:

Default Text	Form Field Entry	Description
grp:Item	<?for-each:item?>	Holds the opening for-each loop for the <code>item</code> element.
Plasma TV	<?name?>	The placeholder for the <code>name</code> element from the XML file.
IF	<?if@column:/items/@type="PRIVATE"?>	The opening of the if statement to test for the attribute value "PRIVATE" in the column header. Note that this syntax uses an XPath expression to navigate back to the "items" level of the XML to test the attribute. For more information about using XPath in your templates, see XPath Overview, page 4-132.
Quantity	N/A	Boilerplate heading
end-if	<?end if?>	Ends the if statement.
IF	<?if@column:/items/@type="PRIVATE"?>	The opening of the if statement to test for the attribute value "PRIVATE" in the column data.
20	<?quantity?>	The placeholder for the quantity element.
end-if	<?end if?>	Ends the if statement.
1,000.00	<?price?>	The placeholder for the <code>price</code> element.



Default Text	Form Field Entry	Description
end grp	<?end for-each?>	Closing tag of the for-each loop.

The conditional column syntax is the "if" statement syntax with the addition of the @column clause. It is the @column clause that instructs BI Publisher to hide or show the column based on the outcome of the if statement.

If you did not include the @column the data would not display in your report as a result of the if statement, but the column still would because you had drawn it in your template.

**Note:** The @column clause is an example of a context command. For more information, see Using Context Commands, page 4-129.

The example will render the output shown in the following figure:

Name	Price
Plasma TV	4,000.00
DVD Player	300.00
VCR	200.00
Receiver	350.00

If the same XML data contained the type attribute set to "PRIVATE" the following output would be rendered from the same template:

Name	Quantity	Price
Plasma TV	10	4,000.00
DVD Player	3	300.00
VCR	20	200.00
Receiver	22	350.00

## Row Formatting

BI Publisher allows you to specify formatting conditions as the row-level of a table. Examples of row-level formatting are:

- Highlighting a row when the data meets a certain threshold.
- Alternating background colors of rows to ease readability of reports.
- Showing only rows that meet a specific condition.

### Conditionally Displaying a Row

To display only rows that meet a certain condition, insert the `<?if:condition?>` `<?end if?>` tags at the beginning and end of the row, within the for-each tags for the group. This is demonstrated in the following sample template.

Industry	Year	Month	Sales
<code>for-each SALE if big INDUSTRY</code>	<code>YEAR</code>	<code>MONTH</code>	<code>SALES end if end SALE</code>

Note the following fields from the sample figure:

Default Text Entry	Form Field Help Text	Description
for-each SALE	<code>&lt;?for-each:SALE?&gt;</code>	Opens the for-each loop to repeat the data belonging to the SALE group.
if big	<code>&lt;?if:SALES&gt;5000?&gt;</code>	If statement to display the row only if the element SALES has a value greater than 5000.
INDUSTRY	<code>&lt;?INDUSTRY?&gt;</code>	Data field
YEAR	<code>&lt;?YEAR?&gt;</code>	Data field
MONTH	<code>&lt;?MONTH?&gt;</code>	Data field
SALES end if	<code>&lt;?end if?&gt;</code>	Closes the if statement.
end SALE	<code>&lt;?end for-each?&gt;</code>	Closes the SALE loop.

### Conditionally Highlighting a Row

This example demonstrates how to set a background color on every other row. The template to create this effect is shown in the following figure:

Industry	Year	Month	Sales
<code>for-each SALE format, INDUSTRY</code>	<code>YEAR</code>	<code>MONTH</code>	<code>SALES end SALE</code>

The following table shows values of the form fields in the template:

Default Text Entry	Form Field Help Text	Description
for-each SALE	<?for-each:SALE?>	Defines the opening of the for-each loop for the SALE group.
format;	<?if@row:position() mod 2=0?> <xsl:attribute name="background-color" xdofo:ctx="incontext">lightgray</xsl:attribute><?end if?>	For each alternate row, the background color attribute is set to gray for the row.
INDUSTRY	<?INDUSTRY?>	Data field
YEAR	<?YEAR?>	Data field
MONTH	<?MONTH?>	Data field
SALES	<?SALES?>	Data field
end SALE	<?end for-each?>	Closes the SALE for-each loop.

In the preceding example, note the "format;" field. It contains an if statement with a "row" context (@row). This sets the context of the if statement to apply to the current row. If the condition is true, then the <xsl:attribute> for the background color of the row will be set to light gray. This will result in the following output:

Industry	Year	Month	Sales
Oil	2000	Jan	100,000
Automotive	2000	Jan	200,000
Groceries	2000	Jan	50,000

**Note:** For more information about context commands, see Using Context Commands, page 4-129.

## Cell Highlighting

The following example demonstrates how to conditionally highlight a cell based on a value in the XML file.

For this example we will use the following XML:

```

<accounts>
  <account>
    <number>1-100-3333</number>
    <debit>100</debit>
    <credit>300</credit>
  </account>
  <account>
    <number>1-101-3533</number>
    <debit>220</debit>
    <credit>30</credit>
  </account>
  <account>
    <number>1-130-3343</number>
    <debit>240</debit>
    <credit>1100</credit>
  </account>
  <account>
    <number>1-153-3033</number>
    <debit>3000</debit>
    <credit>300</credit>
  </account>
</accounts>

```

The template lists the accounts and their credit and debit values. In the final report we want to highlight in red any cell whose value is greater than 1000. The template for this is shown in the following figure:

Account	Debit	Credit
FE:Account 1-232-4444	CH1 100.00	CH2 100.00 EFE

The field definitions for the template are shown in the following table:

Default Text Entry	Form Field Entry	Description
FE:Account	<?for-each:account?>	Opens the for each-loop for the element account.
1-232-4444	<?number?>	The placeholder for the number element from the XML file.
CH1	<?if:debit>1000?><xsl:attribute xdofo:ctx="block" name="background-color">red</xsl:attribute><?end if?>	This field holds the code to highlight the cell red if the debit amount is greater than 1000.
100.00	<?debit?>	The placeholder for the debit element.  IMPORTANT: The <?debit?> element must reside in its own field.

Default Text Entry	Form Field Entry	Description
CH2	<code>&lt;?if:credit&gt;1000?&gt;&lt;xsl:attribute xdofo:ctx="block" name="background-color"&gt;red&lt;/xsl:attribute&gt;&lt;?end if?&gt;</code>	This field holds the code to highlight the cell red if the credit amount is greater than 1000.
100.00	<code>&lt;?credit?&gt;</code>	The placeholder for the credit element.
EFE	<code>&lt;?end for-each?&gt;</code>	Closes the for-each loop.

The code to highlight the debit column as shown in the table is:

```
<?if:debit>1000?>
  <xsl:attribute
    xdofo:ctx="block" name="background-color">red
  </xsl:attribute>
<?end if?>
```

The "if" statement is testing if the debit value is greater than 1000. If it is, then the next lines are invoked. Notice that the example embeds native XSL code inside the "if" statement.

The "attribute" element allows you to modify properties in the XSL.

The xdofo:ctx component is an BI Publisher feature that allows you to adjust XSL attributes at any level in the template. In this case, the background color attribute is changed to red.

To change the color attribute, you can use either the standard HTML names (for example, red, white, green) or you can use the hexadecimal color definition (for example, #FFFFFF).

The output from this template is displayed in the following figure:

Account	Debit	Credit
1-100-3333	100.00	300.00
1-101-3533	220.00	30.00
1-130-3343	240.00	1100.00
1-153-3033	3000.00	300.00

## Page-Level Calculations

Note that all page-level calculations described in this section are supported for PDF and PPT outputs only.

## Displaying Page Totals

BI Publisher allows you to display calculated page totals in your report. Because the page is not created until publishing time, the totaling function must be executed by the formatting engine.

**Note:** Page totaling is performed in the PDF-formatting layer. Therefore this feature is not available for other outputs types: HTML, RTF, Excel.

**Note:** Note that this page totaling function will only work if your source XML has raw numeric values. The numbers must not be preformatted.

Because the page total field does not exist in the XML input data, you must define a variable to hold the value. When you define the variable, you associate it with the element from the XML file that is to be totaled for the page. Once you define total fields, you can also perform additional functions on the data in those fields.

To declare the variable that is to hold your page total, insert the following syntax immediately following the placeholder for the element that is to be totaled:

```
<?add-page-total:TotalFieldName;'element'??>
```

where

`TotalFieldName` is the name you assign to your total (to reference later) and `'element'` is the XML element field to be totaled.

You can add this syntax to as many fields as you want to total.

Then when you want to display the total field, enter the following syntax:

```
<?show-page-total:TotalFieldName;'Oracle-number-format'  
number-separators="{$_XDONFSEPARATORS}"??>
```

where

`TotalFieldName` is the name you assigned to give the page total field above and `Oracle-number-format` is the format you wish to use to for the display, using the Oracle format mask (for example: 'C9G999D00'). For the list of Oracle format mask symbols, see *Using the Oracle Format Mask*, page 4-116.

`number-separators="{$_XDONFSEPARATORS}"` is a required attribute to apply the grouping separator and decimal separator for the format mask you defined.

The following example shows how to set up page total fields in a template to display total credits and debits that have displayed on the page, and then calculate the net of the two fields.

This example uses the following XML:

```

<balance_sheet>
  <transaction>
    <debit>100</debit>
    <credit>90</credit>
  </transaction>
  <transaction>
    <debit>110</debit>
    <credit>80</credit>
  </transaction>
  ...
</balance_sheet>

```

The following figure shows the table to insert in the template to hold the values:

Debit	Credit
FE 100.00	90.00 Net EFE

The following table shows the form field entries made in the template for the example table:

Default Text Entry	Form Field Help Text Entry	Description
FE	<?for-each:transaction?>	This field defines the opening "for-each" loop for the transaction group.
100.00	<?debit?><?add-page-total:dt;'debit'?'>	This field is the placeholder for the debit element from the XML file. Because we want to total this field by page, the page total declaration syntax is added. The variable defined to hold the total for the debit element is dt.
90.00	<?credit?><?add-page-total:ct;'credit'?'>	This field is the placeholder for the credit element from the XML file. Because we want to total this field by page, the page total declaration syntax is added. The variable defined to hold the total for the credit element is ct.
Net	<add-page-total:net;'debit - credit'?'>	Creates a net page total by subtracting the credit values from the debit values.
EFE	<?end for-each?>	Closes the for-each loop.

Note that on the variable defined as "net" we are actually carrying out a calculation on the values of the credit and debit elements.

Now that you have declared the page total fields, you can insert a field in your template where you want the page totals to appear. Reference the calculated variables using the names you supplied (in the example, `ct` and `dt`). The syntax to display the page totals is as follows:

For example, to display the debit page total, enter the following:

```
<?show-page-total:dt;'C9G990D00';'(C9G990D00)'
number-separators="{$_XDONFSEPARATORS}"?>
```

Therefore to complete the example, place the following at the bottom of the template page, or in the footer:

```
Page Total Debit: <?show-page-total:dt;'C9G990D00';'(C9G990D00)'
number-separators="{$_XDONFSEPARATORS}"?>
```

```
Page Total Credit: <?show-page-total:ct;'C9G990D00';'(C9G990D00)'
number-separators="{$_XDONFSEPARATORS}"?>
```

```
Page Total Balance: <?show-page-total:net;'C9G990D00';'(C9G990D00)'
number-separators="{$_XDONFSEPARATORS}"?>
```

The output for this report is shown in the following graphic:

	Debit	Credit
	100.00	90.00
	110.00	80.00
	120.00	70.00
	130.00	60.00
	140.00	50.00
	150.00	40.00

Page 1

Page Total Debit:750.00  
Page Total Credit:390.00  
Page Total Balance:360.00

## Brought Forward/Carried Forward Totals

Many reports require that a page total be maintained throughout the report output and be displayed at the beginning and end of each page. These totals are known as "brought forward/carried forward" totals.

**Note:** The totaling for the brought forward and carried forward fields is performed in the PDF-formatting layer. Therefore this feature is not available for other outputs types: HTML, RTF, Excel.

An example is displayed in the following figure:



Page 1			Page 2 Brought Forward: 300			Page 3 Brought Forward: 600		
Inv	Date	Amount	Inv	Date	Amount	Inv	Date	Amount
1001	1-Jan-05	100	1004	1-Jan-05	100	1007	1-Jan-05	100
1002	1-Jan-05	100	1005	1-Jan-05	100	1008	1-Jan-05	100
1003	1-Jan-05	100	1006	1-Jan-05	100	1009	1-Jan-05	100
Carried Forward: 300			Carried Forward: 600					

At the end of the first page, the page total for the Amount element is displayed as the Carried Forward total. At the top of the second page, this value is displayed as the Brought Forward total from the previous page. At the bottom of the second page, the brought forward value plus the total for that page is calculated and displayed as the new Carried Forward value, and this continues throughout the report.

This functionality is an extension of the Page Totals, page 4-76 feature. The following example walks through the syntax and setup required to display the brought forward and carried forward totals in your published report.

Assume you have the following XML:

```
<?xml version="1.0" encoding="WINDOWS-1252"?>
<INVOICES>
  <INVOICE>
    <INVNUM>10001-1</INVNUM>
    <INVDATE>1-Jan-2005</INVDATE>
    <INVAMT>100</INVOICEAMT>
  </INVOICE>
  <INVOICE>
    <INVNUM>10001-2</INVNUM>
    <INVDATE>10-Jan-2005</INVDATE>
    <INVAMT>200</INVOICEAMT>
  </INVOICE>
  <INVOICE>
    <INVNUM>10001-1</INVNUM>
    <INVDATE>11-Jan-2005</INVDATE>
    <INVAMT>150</INVOICEAMT>
  </INVOICE>
  . . .
</INVOICES>
```

The following sample template creates the invoice table and declares a placeholder that will hold your page total:

Init PTs

Invoice	Date	Amount
FE 132342	10-May-2005	1,000.00 InvAmt EG

End PTs

The fields in the template have the following values:

Field	Form Field Help Text Entry	Description
Init PTs	<?init-page-total: InvAmt?>	Declares "InvAmt" as the placeholder that will hold the page total.
FE	<?for-each:INVOICE?>	Begins the INVOICE group.
10001-1	<?INVNUM?>	Placeholder for the Invoice Number tag.
1-Jan-2005	<?INVDATE?>	Placeholder for the Invoice Date tag.
100.00	<?INVAMT?>	Placeholder for the Invoice Amount tag.
InvAmt	<?add-page-total:InvAmt;INVAMT?>	Assigns the "InvAmt" page total object to the INVAMT element in the data.
EFE	<?end for-each?>	Closes the INVOICE group.
End PTs	<?end-page-total:InvAmt?>	Closes the "InvAmt" page total.

To display the brought forward total at the top of each page (except the first), use the following syntax:

```
<xdofo:inline-total
  display-condition="exceptfirst"
  name="InvAmt">
  Brought Forward:
<xdofo:show-brought-forward
  name="InvAmt"
  format="99G999G999D00" number-separators="{$_XDONFSEPARATORS}"/>/>
</xdofo:inline-total>
```

The following table describes the elements comprising the brought forward syntax:

Code Element	Description and Usage
<code>inline-total</code>	<p>This element has two properties:</p> <ul style="list-style-type: none"> <li>• <code>name</code> - name of the variable you declared for the field.</li> <li>• <code>display-condition</code> - sets the display condition. This is an optional property that takes one of the following values: <ul style="list-style-type: none"> <li>• <code>first</code> - the contents appear only on the first page</li> <li>• <code>last</code> - the contents appear only on the last page</li> <li>• <code>exceptfirst</code> - contents appear on all pages except first</li> <li>• <code>exceptlast</code> - contents appear on all pages except last</li> <li>• <code>everytime</code> - (default) contents appear on every page</li> </ul> </li> </ul> <p>In this example, <code>display-condition</code> is set to "exceptfirst" to prevent the value from appearing on the first page where the value would be zero.</p>
<code>Brought Forward:</code>	This string is optional and will display as the field name on the report.
<code>show-brought-forward</code>	<p>Shows the value on the page. It has the following two properties:</p> <ul style="list-style-type: none"> <li>• <code>name</code> - the name of the field to show. In this case, "InvAmt". This property is mandatory.</li> <li>• <code>format</code> - the Oracle number format to apply to the value at runtime. This property is optional, but if you want to supply a format mask, you must use the Oracle format mask. For more information, see <i>Using the Oracle Format Mask</i>, page 4-116 .</li> <li>• <code>number-separators="{\$_XDONFSEPARATORS}"</code> - this attribute is required to apply the grouping separator and number separator for the format mask you defined.</li> </ul>

Insert the brought forward object at the top of the template where you want the brought forward total to display. If you place it in the body of the template, you can insert the syntax in a form field.

If you want the brought forward total to display in the header, you must insert the full code string into the header because Microsoft Word does not support form fields in the header or footer regions. However, you can alternatively use the start body/end body syntax which allows you to define what the body area of the report will be. BI Publisher

will recognize any content above the defined body area as header content, and any content below as the footer. This allows you to use form fields. See *Multiple or Complex Headers and Footers*, page 4-16 for details.

Place the carried forward object at the bottom of your template where you want the total to display. The carried forward object for our example is as follows:

```
<xdofo:inline-total
  display-condition="exceptlast"
  name="InvAmt">
  Carried Forward:
<xdofo:show-carry-forward
  name="InvAmt"
  format="99G999G999D00" number-separators="{$_XDONFSEPARATORS}"/>
</xdofo:inline-total>
```

Note the following differences with the brought-forward object:

- The `display-condition` is set to `exceptlast` so that the carried forward total will display on every page except the last page.
- The display string is "Carried Forward".
- The `show-carry-forward` element is used to show the carried forward value. It has the same properties as `brought-carried-forward`, described above.

You are not limited to a single value in your template, you can create multiple brought forward/carried forward objects in your template pointing to various numeric elements in your data.

**Important:** Ensure not to include the commands `<?init-page-total:invAmnt?>` and `<?end-page-total:InvAmt?>` as shown in the preceding example. The `display-condition` logic computation depends on these commands to function correctly.

## Running Totals

### Example

The variable functionality (see *Using Variables*, page 4-92) can be used to add a running total to your invoice listing report. This example assumes the following XML structure:

```

<?xml version="1.0" encoding="WINDOWS-1252"?>
<INVOICES>
  <INVOICE>
    <INVNUM>10001-1</INVNUM>
    <INVDATE>1-Jan-2005</INVDATE>
    <INVAMT>100</INVOICEAMT>
  </INVOICE>
  <INVOICE>
    <INVNUM>10001-2</INVNUM>
    <INVDATE>10-Jan-2005</INVDATE>
    <INVAMT>200</INVOICEAMT>
  </INVOICE>
  <INVOICE>
    <INVNUM>10001-1</INVNUM>
    <INVDATE>11-Jan-2005</INVDATE>
    <INVAMT>150</INVOICEAMT>
  </INVOICE>
</INVOICES>

```

Using this XML, we want to create the report that contains running totals as shown in the following figure:

Invoice Number	Invoice Date	Amount	Running Total
1000-1	1-Jan-2005	100.00	100.00
1000-2	10-Jan-2005	200.00	300.00
1000-3	11-Jan-2005	150.00	450.00

To create the Running Total field, define a variable to track the total and initialize it to 0. The template is shown in the following figure:

RTotalVar

Invoice Number	Invoice Date	Amount	Running Total
FE10001-1	1-Jan-2005	100.00	100.00EFE

The values for the form fields in the template are shown in the following table:

Form Field	Syntax	Description
RtotalVar	<?xdoxslt:set_variable(\$_XDOCTX, 'RTotalVar', 0)?>	Declares the "RTotalVar" variable and initializes it to 0.
FE	<?for-each:INVOICE?>	Starts the Invoice group.
10001-1	<?INVNUM?>	Invoice Number tag
1-Jan-2005	<?INVDATE?>	Invoice Date tag

Form Field	Syntax	Description
100.00	<pre>&lt;?xdoxslt:set_variable(\$_XDOCTX, 'RTotalVar', xdoxslt:get_variable(\$_XDOCTX, 'RTotalVar') + INVAMT)?&gt; &lt;?xdoxslt:get_variable(\$_XDOCTX , 'RTotalVar')?&gt;</pre>	<p>Sets the value of RTotalVar to the current value plus the new Invoice Amount.</p> <p>Retrieves the RTotalVar value for display.</p>
EFE	<pre>&lt;?end for-each?&gt;</pre>	Ends the INVOICE group.

## Data Handling

### Sorting

You can sort a group by any element within the group. Insert the following syntax within the group tags:

```
<?sort:element name; order; data-type?>
```

where

`element name` is the name of the element you want the group sorted by

`order` is 'ascending' or 'descending'

`data-type` is the element data type. Valid values are: 'text' and 'number'.

If the order is not specified, by default, the sort order is ascending. If the data type is not specified, the type is assumed to be text.

For example, to sort a data set by an element named SALARY so that the highest salaries appear first, enter the following:

```
<?sort:SALARY;'descending';'number'?>
```

When you are sorting within a for-each group, enter the sort statement after the for-each statement. For example, to sort the Payables Invoice Register (shown at the beginning of this chapter) by Supplier (VENDOR\_NAME), enter the following:

```
<?for-each:G_VENDOR_NAME?><?sort:VENDOR_NAME?>
```

To sort a group by multiple fields, just enter additional sort statements in the appropriate order. For example, to sort by Supplier and then by Invoice Number, enter the following

```
<?sort:VENDOR_NAME?> <?sort:INVOICE_NUM;'ascending';'number'?>
```

## Checking for Nulls

Within your XML data there are three possible scenarios for the value of an element:

- The element is present in the XML data, and it has a value
- The element is present in the XML data, but it does not have a value
- The element is not present in the XML data, and therefore there is no value

In your report layout, you may want to specify a different behavior depending on the presence of the element and its value. The following examples show how to check for each of these conditions using an "if" statement. The syntax can also be used in other conditional formatting constructs.

- To define behavior when the element is present and the value is not null, use the following:

```
<?if:element_name!=' '>desired behavior <?end if?>
```

- To define behavior when the element is present, but is null, use the following:

```
<?if:element_name and element_name="">desired behavior <?end if?>
```

- To define behavior when the element is not present, use the following:

```
<?if:not(element_name)?>desired behavior <?end if?>
```

## Regrouping the XML Data

The RTF template supports the XSL 2.0 for-each-group standard that allows you to regroup XML data into hierarchies that are not present in the original data. With this feature, your template does not have to follow the hierarchy of the source XML file. You are therefore no longer limited by the structure of your data source.

### XML Sample

To demonstrate the for-each-group standard, the following XML data sample of a CD catalog listing will be regrouped in a template:

```

<CATALOG>
  <CD>
    <TITLE>Empire Burlesque</TITLE>
    <ARTIST>Bob Dylan</ARTIST>
    <COUNTRY>USA</COUNTRY>
    <COMPANY>Columbia</COMPANY>
    <PRICE>10.90</PRICE>
    <YEAR>1985</YEAR>
  </CD>
  <CD>
    <TITLE>Hide Your Heart</TITLE>
    <ARTIST>Bonnie Tylor</ARTIST>
    <COUNTRY>UK</COUNTRY>
    <COMPANY>CBS Records</COMPANY>
    <PRICE>9.90</PRICE>
    <YEAR>1988</YEAR>
  </CD>
  <CD>
    <TITLE>Still got the blues</TITLE>
    <ARTIST>Gary More</ARTIST>
    <COUNTRY>UK</COUNTRY>
    <COMPANY>Virgin Records</COMPANY>
    <PRICE>10.20</PRICE>
    <YEAR>1990</YEAR>
  </CD>
  <CD>
    <TITLE>This is US</TITLE>
    <ARTIST>Gary Lee</ARTIST>
    <COUNTRY>UK</COUNTRY>
    <COMPANY>Virgin Records</COMPANY>
    <PRICE>12.20</PRICE>
    <YEAR>1990</YEAR>
  </CD>

```

Using the regrouping syntax, you can create a report of this data that groups the CDs by country and then by year. You are not limited by the data structure presented.

## Regrouping Syntax

To regroup the data, use the following syntax:

```
<?for-each-group: BASE-GROUP; GROUPING-ELEMENT?>
```

For example, to regroup the CD listing by COUNTRY, enter the following in your template:

```
<?for-each-group: CD; COUNTRY?>
```

The elements that were at the same hierarchy level as COUNTRY are now children of COUNTRY. You can then refer to the elements of the group to display the values desired.

To establish nested groupings within the already defined group, use the following syntax:

```
<?for-each: current-group (); GROUPING-ELEMENT?>
```

For example, after declaring the CD grouping by COUNTRY, you can then further group by YEAR within COUNTRY as follows:

```
<?for-each: current-group (); YEAR?>
```



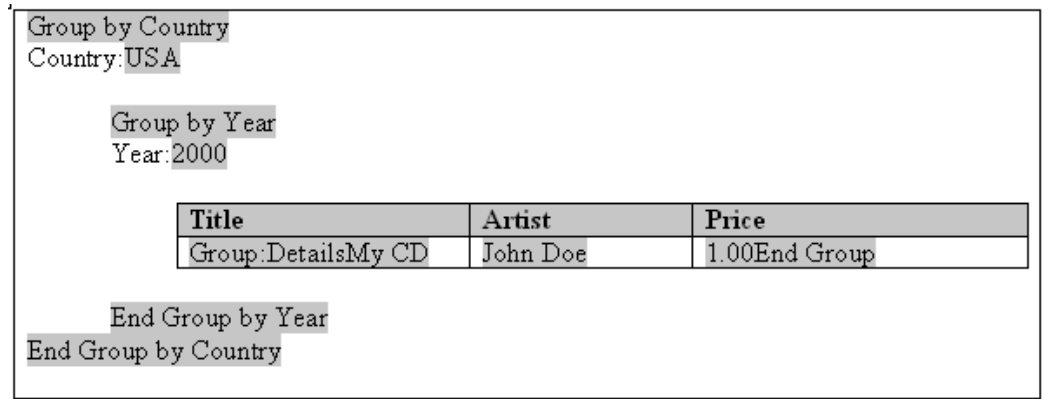
At runtime, BI Publisher will loop through the occurrences of the new groupings, displaying the fields that you defined in your template.

**Note:** This syntax is a simplification of the XSL for-each-group syntax. If you choose not to use the simplified syntax above, you can use the XSL syntax as shown below. The XSL syntax can only be used within a form field of the template.

```
<xsl:for-each-group
  select=expression
  group-by="string expression"
  group-adjacent="string expression"
  group-starting-with=pattern>
  <!--Content: (xsl:sort*, content-constructor) -->
</xsl:for-each-group>
```

**Template Example**

The following figure shows a template that displays the CDs by Country, then Year, and lists the details for each CD:



The following table shows the BI Publisher syntax entries made in the form fields of the preceding template:

Default Text Entry	Form Field Help Text Entry	Description
Group by Country	<?for-each-group:CD;COUNTRY?>	The <?for-each-group:CD;COUNTRY?> tag declares the new group. It regroups the existing CD group by the COUNTRY element.
USA	<?COUNTRY?>	Placeholder to display the data value of the COUNTRY tag.

Default Text Entry	Form Field Help Text Entry	Description
Group by Year	<code>&lt;?for-each-group:current-group ();YEAR?&gt;</code>	The <code>&lt;?for-each-group:current-group ();YEAR?&gt;</code> tag regroups the current group (that is, COUNTRY), by the YEAR element.
2000	<code>&lt;?YEAR?&gt;</code>	Placeholder to display the data value of the YEAR tag.
Group: Details	<code>&lt;?for-each:current-group () ?&gt;</code>	Once the data is grouped by COUNTRY and then by YEAR, the <code>&lt;?for-each:current-group () ?&gt;</code> command is used to loop through the elements of the current group (that is, YEAR) and render the data values (TITLE, ARTIST, and PRICE) in the table.
My CD	<code>&lt;?TITLE?&gt;</code>	Placeholder to display the data value of the TITLE tag.
John Doe	<code>&lt;?ARTIST?&gt;</code>	Placeholder to display the data value of the ARTIST tag.
1.00	<code>&lt;?PRICE?&gt;</code>	Placeholder to display the data value of the PRICE tag.
End Group	<code>&lt;?end for-each?&gt;</code>	Closes out the <code>&lt;?for-each:current-group () ?&gt;</code> tag.
End Group by Year	<code>&lt;?end for-each-group?&gt;</code>	Closes out the <code>&lt;?for-each-group:current-group ();YEAR?&gt;</code> tag.
End Group by Country	<code>&lt;?end for-each-group?&gt;</code>	Closes out the <code>&lt;?for-each-group:CD;COUNTRY?&gt;</code> tag.

This template produces the following output when merged with the XML file:

Country:USA		
Year:1985		
Title	Artist	Price
Empire Burlesque	Bob Dylan	10.90
Country:UK		
Year:1988		
Title	Artist	Price
Hide your heart	Bonnie Tylor	9.90
Year:1990		
Title	Artist	Price
Still got the blues	Gary More	10.20
This is US	Gary Lee	12.20

### Regrouping by an Expression

Regrouping by an expression allows you to apply a function or command to a data element, and then group the data by the returned result.

To use this feature, state the expression within the regrouping syntax as follows:

```
<?for-each:BASE-GROUP;GROUPING-EXPRESSION?>
```

#### Example

To demonstrate this feature, an XML data sample that simply contains average temperatures per month will be used as input to a template that calculates the number of months having an average temperature within a certain range.

The following XML sample is composed of <temp> groups. Each <temp> group contains a <month> element and a <degree> element, which contains the average temperature for that month:

```

<temps>
  <temp>
    <month>Jan</month>
    <degree>11</degree>
  </temp>
  <temp>
    <month>Feb</month>
    <degree>14</degree>
  </temp>
  <temp>
    <month>Mar</month>
    <degree>16</degree>
  </temp>
  <temp>
    <month>Apr</month>
    <degree>20</degree>
  </temp>
  <temp>
    <month>May</month>
    <degree>31</degree>
  </temp>
  <temp>
    <month>Jun</month>
    <degree>34</degree>
  </temp>
  <temp>
    <month>Jul</month>
    <degree>39</degree>
  </temp>
  <temp>
    <month>Aug</month>
    <degree>38</degree>
  </temp>
  <temp>
    <month>Sep</month>
    <degree>24</degree>
  </temp>
  <temp>
    <month>Oct</month>
    <degree>28</degree>
  </temp>
  <temp>
    <month>Nov</month>
    <degree>18</degree>
  </temp>
  <temp>
    <month>Dec</month>
    <degree>8</degree>
  </temp>
</temps>

```

You want to display this data in a format showing temperature ranges and a count of the months that have an average temperature to satisfy those ranges, as follows:

## Annual Temperature Summary

Range	Number of Months
0 F to 10 F	1 Month(s)
10 F to 20 F	4 Month(s)
20 F to 30 F	3 Month(s)
30 F to 40 F	4 Month(s)

Using the for-each-group command you can apply an expression to the `<degree>` element that will enable you to group the temperatures by increments of 10 degrees. You can then display a count of the members of each grouping, which will be the number of months having an average temperature that falls within each range.

The template to create the above report is shown in the following figure:

### Annual Temperature Summary

Range	Number of Months
Group by TmpRng	Months Month(s)End TmpRng
Range	

The following table shows the form field entries made in the template:

Default Text Entry	Form Field Help Text Entry
Group by TmpRng	<code>&lt;?for-each-group:temp;floor(degree div 10)?&gt;</code> <code>&lt;?sort:floor(degree div 10)?&gt;</code>
Range	<code>&lt;?concat(floor(degree div 10)*10,' F to ',floor(degree div 10)*10+10, 'F')?&gt;</code>
Months	<code>&lt;?count(current-group())?&gt;</code>
End TmpRng	<code>&lt;?end for-each-group?&gt;</code>

Note the following about the form field tags:

- The `<?for-each-group:temp;floor(degree div 10)?>` is the regrouping tag. It specifies that for the existing `<temp>` group, the elements are to be regrouped by the expression, `floor(degree div 10)`. The `floor` function is an XSL function that returns the highest integer that is not greater than the argument

(for example, 1.2 returns 1, 0.8 returns 0).

In this case, it returns the value of the <degree> element, which is then divided by 10. This will generate the following values from the XML data: 1, 1, 1, 2, 3, 3, 3, 3, 2, 2, 1, and 0.

These are sorted, so that when processed, the following four groups will be created: 0, 1, 2, and 3.

- The <?concat(floor(degree div 10)\*10,'F to ', floor(degree div 10)\*10+10, 'F'?)> displays the temperature ranges in the row header in increments of 10. The expression concatenates the value of the current group times 10 with the value of the current group times 10 plus 10.

Therefore, for the first group, 0, the row heading displays 0 to (0 +10), or "0 F to 10 F".

- The <?count(current-group())?> uses the count function to count the members of the current group (the number of temperatures that satisfy the range).
- The <?end for-each-group?> tag closes out the grouping.

## Using Variables

Updateable variables differ from standard XSL variables <xsl:variable> in that they are updateable during the template application to the XML data. This allows you to create many new features in your templates that require updateable variables.

The variables use a "set and get" approach for assigning, updating, and retrieving values.

Use the following syntax to declare/set a variable value:

```
<?xdoxslt:set_variable($_XDOCTX, 'variable name', value)?>
```

Use the following syntax to retrieve a variable value:

```
<?xdoxslt:get_variable($_XDOCTX, 'variable name')?>
```

You can use this method to perform calculations. For example:

```
<?xdoxslt:set_variable($_XDOCTX, 'x', xdoxslt:get_variable($_XDOCTX, 'x' + 1)?>
```

This sets the value of variable 'x' to its original value plus 1, much like using "x = x + 1".

The \$\_XDOCTX specifies the global document context for the variables. In a multi-threaded environment there may be many transformations occurring at the same time, therefore the variable must be assigned to a single transformation.

See the section on Running Totals, page 4-82 for an example of the usage of updateable variables.

## Defining Parameters

You can pass runtime parameter values into your template. These can then be referenced throughout the template to support many functions. For example, you can filter data in the template, use a value in a conditional formatting block, or pass property values (such as security settings) into the final document.

**Note:** For BI Publisher Enterprise users, all name-value parameter pairs are passed to the template. You must register the parameters that you wish to utilize in your template using the syntax described below.

### Using a parameter in a template

1. Declare the parameter in the template.

Use the following syntax to declare the parameter:

```
<?param@begin:parameter_name;parameter_value?>
```

where

*parameter\_name* is the name of the parameter

*parameter\_value* is the default value for the parameter (the *parameter\_value* is optional)

*param@begin:* is a required string to push the parameter declaration to the top of the template at runtime so that it can be referred to globally in the template.

The syntax must be declared in the Help Text field of a form field. The form field can be placed anywhere in the template.

2. Refer to the parameter in the template by prefixing the name with a "\$" character. For example, if you declare the parameter name to be "InvThresh", then reference the value using "\$InvThresh".
3. If you are not using BI Publisher Enterprise, but only the core libraries:

At runtime, pass the parameter to the BI Publisher engine programmatically.

Prior to calling the FOPProcessor API create a Properties class and assign a property to it for the parameter value as follows:

```
Properties prop = new Properties();  
prop.put("xslt.InvThresh", "1000");
```

### Example: Passing an invoice threshold parameter

This example illustrates how to declare a parameter in your template that will filter your data based on the value of the parameter.

The following XML sample lists invoice data:

```

<INVOICES>
  <INVOICE>
    <INVOICE_NUM>981110</INVOICE_NUM>
    <AMOUNT>1100</AMOUNT>
  </INVOICE>
  <INVOICE>
    <INVOICE_NUM>981111</INVOICE_NUM>
    <AMOUNT>250</AMOUNT>
  </INVOICE>
  <INVOICE>
    <INVOICE_NUM>981112</INVOICE_NUM>
    <AMOUNT>8343</AMOUNT>
  </INVOICE>
  . . .
</INVOICES>

```

The following figure displays a template that accepts a parameter value to limit the invoices displayed in the final document based on the parameter value.

### InvThresh Declaration

Invoice Number	Invoice Amount
FE IF 13222-2	\$100.00 EI EFE

Field	Form Field Help Text Entry	Description
InvThreshDeclaration	<?param@begin:InvThresh?>	Declares the parameter InvThresh.
FE	<?for-each:INVOICE?>	Begins the repeating group for the INVOICE element.
IF	<?if:AMOUNT>\$InvThresh?>	Tests the value of the AMOUNT element to determine if it is greater than the value of InvThresh.
13222-2	<?INVOICE_NUM?>	Placeholder for the INVOICE_NUM element.
\$100.00	<?AMOUNT?>	Placeholder for the AMOUNT element.
EI	<?end if?>	Closing tag for the if statement.
EFE	<?end for-each?>	Closing tag for the for-each loop.

In this template, only INVOICE elements with an AMOUNT greater than the InvThresh



parameter value will be displayed. If we pass in a parameter value of 1,000, the following output shown in the following figure will result:

Invoice Number	Invoice Amount
981110	1100
981112	8343

Notice the second invoice does not display because its amount was less than the parameter value.

## Setting Properties

BI Publisher properties that are available in the BI Publisher Configuration file can alternatively be embedded into the RTF template. The properties set in the template are resolved at runtime by the BI Publisher engine. You can either hard code the values in the template or embed the values in the incoming XML data. Embedding the properties in the template avoids the use of the configuration file.

**Note:** See BI Publisher Configuration File, *Oracle Fusion Middleware Administrator's Guide for Oracle Business Intelligence Publisher* for more information about the BI Publisher Configuration file and the available properties.

For example, if you use a nonstandard font in your template, rather than specify the font location in the configuration file, you can embed the font property inside the template. If you need to secure the generated PDF output, you can use the BI Publisher PDF security properties and obtain the password value from the incoming XML data.

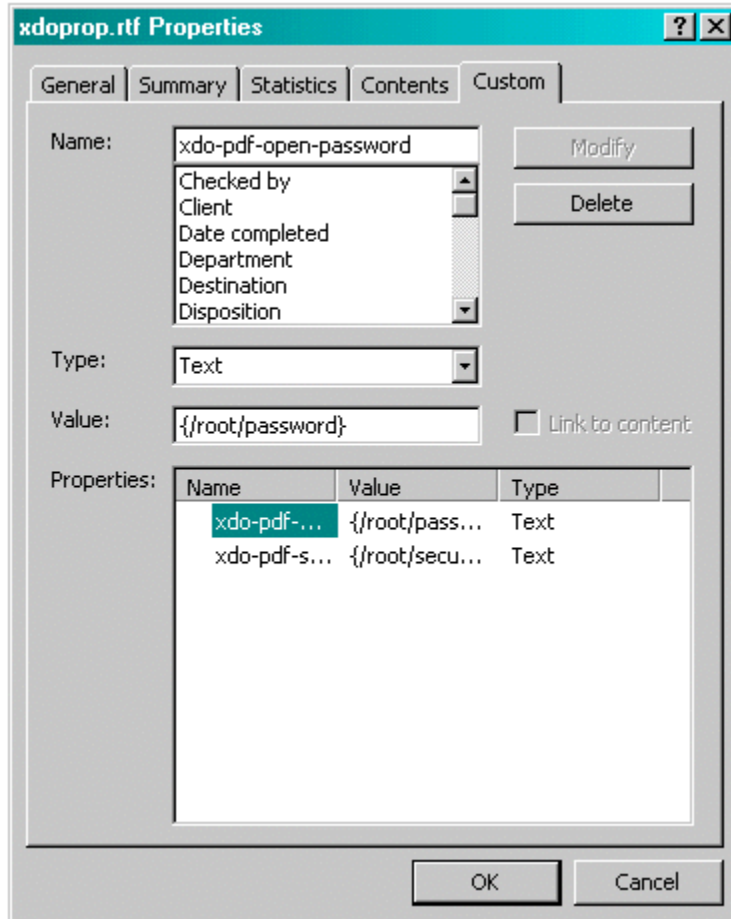
To add an BI Publisher property to a template, use the Microsoft Word **Properties** dialog (available from the **File** menu), and enter the following information:

**Name** - enter the BI Publisher property name prefixed with "xdo-"

**Type** - select "Text"

**Value** - enter the property value. To reference an element from the incoming XML data, enter the path to the XML element enclosed by curly braces. For example:  
{/root/password}

The following figure shows the Properties dialog:



### Embedding a Font Reference

For this example, suppose you want to use a font in the template called "XMLPScript". This font is not available on your server, therefore you must tell BI Publisher where to find the font at runtime. You tell BI Publisher where to find the font by setting the "font" property. Assume the font is located in "/tmp/fonts", then you would enter the following in the Properties dialog:

**Name:** xdo-font.XMLPScript.normal.normal

**Type:** Text

**Value:** truetype./tmp/fonts/XMLPScript.ttf

When the template is applied to the XML data on the server, BI Publisher will look for the font in the /tmp/fonts directory. Note that if the template is deployed in multiple locations, you must ensure that the path is valid for each location.

For more information about setting font properties, see *Font Definitions, Oracle Fusion Middleware Administrator's Guide for Oracle Business Intelligence Publisher*.

### Securing a PDF Output

For this example, suppose you want to use a password from the XML data to secure the PDF output document. The XML data is as follows:

```
<PO>
  <security>true</security>
  <password>welcome</password>
  <PO_DETAILS>
  ..
</PO>
```

In the Properties dialog set two properties: `pdf-security` to set the security feature as enabled or not, and `pdf-open-password` to set the password. Enter the following in the Properties dialog:

**Name:** xdo-pdf-security

**Type:** Text

**Value:** {/PO/security}

**Name:** xdo-pdf-open-password

**Type:** Text

**Value:** {/PO/password}

Storing the password in the XML data is not recommended if the XML will persist in the system for any length of time. To avoid this potential security risk, you can use a template parameter value that is generated and passed into the template at runtime.

For example, you could set up the following parameters:

- PDFSec - to pass the value for the xdo-pdf-security property
- PDFPWD - to pass the value for the password

You would then enter the following in the Properties dialog:

**Name:** xdo-pdf-security

**Type:** Text

**Value:** {\$PDFSec}

**Name:** xdo-pdf-open-password

**Type:** Text

**Value:** {\$PDFPWD}

For more information about template parameters, see *Defining Parameters in Your Template*, page 4-93.

## Advanced Report Layouts

### Batch Reports

It is a common requirement to print a batch of documents, such as invoices or purchase orders in a single PDF file. Because these documents are intended for different

customers, each document will require that the page numbering be reset and that page totals are specific to the document. If the header and footer display fields from the data (such as customer name) these will have to be reset as well.

BI Publisher supports this requirement through the use of a context command. This command allows you to define elements of your report to a specific section. When the section changes, these elements are reset.

The following example demonstrates how to reset the header and footer and page numbering within an output file:

The following XML sample is a report that contains multiple invoices:

```
...
<LIST_G_INVOICE>
  <G_INVOICE>
    <BILL_CUST_NAME>Vision, Inc. </BILL_CUST_NAME>
    <TRX_NUMBER>2345678</TRX_NUMBER>
    ...
  </G_INVOICE>
  <G_INVOICE>
    <BILL_CUST_NAME>Oracle, Inc. </BILL_CUST_NAME>
    <TRX_NUMBER>2345685</TRX_NUMBER>
    ...
  </G_INVOICE>
  ...
</LIST_G_INVOICE>
...
```

Each G\_INVOICE element contains an invoice for a potentially different customer. To instruct BI Publisher to start a new section for each occurrence of the G\_INVOICE element, add the @section command to the opening for-each statement for the group, using the following syntax:

```
<?for-each@section:group name?>
```

where *group\_name* is the name of the element for which you want to begin a new section.

For example, the for-each grouping statement for this example will be as follows:

```
<?for-each@section:G_INVOICE?>
```

The closing <?end for-each?> tag is not changed.

The following figure shows a sample template. Note that the G\_INVOICE group for-each declaration is still within the body of the report, even though the headers will be reset by the command.

```

Invoice# <?TRX_NUMBER?>

for-each G_INVOICE
INVOICE

BILL_CUST_NAME
BILL_ADDRESS1
BILL_ADDRESS2
BILL_CITY, BILL_STATE BILL_POSTAL_CODE

end G_INVOICE

```

The following table shows the values of the form fields from the example:

Default Text Entry	Form Field Help Text	Description
for-each G_INVOICE	<?for-each@section:G_INVOICE?>	Begins the G_INVOICE group, and defines the element as a Section. For each occurrence of G_INVOICE, a new section will be started.
<?TRX_NUMBER?>	N/A	Microsoft Word does not support form fields in the header, therefore the placeholder syntax for the TRX_NUMBER element is placed directly in the template.
end G_INVOICE	<?end for-each?>	Closes the G_INVOICE group.

Now for each new occurrence of the G\_INVOICE element, a new section will begin. The page numbers will restart, and if header or footer information is derived from the data, it will be reset as well.

### Handling No Data Found Conditions

When you use @section with the BI Publisher commands for-each or for-each-group (for example: <?for-each@section:ELEMENT\_NAME?>), and the input data file has no data, then an empty or invalid PDF output document may be generated for that for-each loop. To prevent this from happening, enter the following in your RTF template:

1. At the end of your RTF template, add a section break
2. On the last page (the new section page), add the command  
`<?if@section:not(ELEMENT_NAME)?>No Data Found<?end if?>`  
where ELEMENT\_NAME is the same data element that you are using in your for-each@section loop.  
  
Now if no data exists for ELEMENT\_NAME, a valid PDF will be generated with the text "No Data Found".

## Pivot Table Support

**Note:** Also see Inserting a Pivot Table, page 5-22 in the chapter "Creating RTF Templates Using the Template Builder for Word."

The columns of a pivot table are data dependent. At design-time you do not know how many columns will be reported, or what the appropriate column headings will be. Moreover, if the columns should break onto a second page, you need to be able to define the row label columns to repeat onto subsequent pages. The following example shows how to design a simple pivot table report that supports these features.

This example uses the following XML sample:

```

<ROWSET>
  <RESULTS>
    <INDUSTRY>Motor Vehicle Dealers</INDUSTRY>
    <YEAR>2005</YEAR>
    <QUARTER>Q1</QUARTER>
    <SALES>1000</SALES>
  </RESULTS>
  <RESULTS>
    <INDUSTRY>Motor Vehicle Dealers</INDUSTRY>
    <YEAR>2005</YEAR>
    <QUARTER>Q2</QUARTER>
    <SALES>2000</SALES>
  </RESULTS>
  <RESULTS>
    <INDUSTRY>Motor Vehicle Dealers</INDUSTRY>
    <YEAR>2004</YEAR>
    <QUARTER>Q1</QUARTER>
    <SALES>3000</SALES>
  </RESULTS>
  <RESULTS>
    <INDUSTRY>Motor Vehicle Dealers</INDUSTRY>
    <YEAR>2004</YEAR>
    <QUARTER>Q2</QUARTER>
    <SALES>3000</SALES>
  </RESULTS>
  <RESULTS>
    <INDUSTRY>Motor Vehicle Dealers</INDUSTRY>
    <YEAR>2003</YEAR>
    ...
  </RESULTS>
  <RESULTS>
    <INDUSTRY>Home Furnishings</INDUSTRY>
    ...
  </RESULTS>
  <RESULTS>
    <INDUSTRY>Electronics</INDUSTRY>
    ...
  </RESULTS>
  <RESULTS>
    <INDUSTRY>Food and Beverage</INDUSTRY>
    ...
  </RESULTS>
</ROWSET>

```

From this XML we will generate a report that shows each industry and totals the sales by year as shown in the following figure:

Industry	2005	2004	2003
Motor Vehicle Dealers	3000	6000	1200
Home Furnishings	3200	7770	3300
Electronics	9000	9000	4300
Food and Beverage	1200	900	5600

The template to generate this report is shown in the following figure. The form field entries are shown in the subsequent table.

<b>Industry</b> header column	for: YEAR end
for: INDUSTRY	for: sum(SALES) end end

The form fields in the template have the following values:

Default Text Entry	Form Field Help Text	Description
header column	<?horizontal-break-table:1?>	Defines the first column as a header that should repeat if the table breaks across pages. For more information about this syntax, see Defining Columns to Repeat Across Pages, page 4-104.
for:	<?for-each-group@column:RESULTS;YEAR?>	Uses the regrouping syntax (see Regrouping the XML Data, page 4-85) to group the data by YEAR; and the @column context command to create a table column for each group (YEAR). For more information about context commands, see Using the Context Commands, page 4-129.
YEAR	<?YEAR?>	Placeholder for the YEAR element.
end	<?end for-each-group?>	Closes the for-each-group loop.
for:	<?for-each-group:RESULTS;INDUSTRY?>	Begins the group to create a table row for each INDUSTRY.
INDUSTRY	<?INDUSTRY?>	Placeholder for the INDUSTRY element.
for:	<?for-each-group@cell:current-group();YEAR?>	Uses the regrouping syntax (see Regrouping the XML Data, page 4-85) to group the data by YEAR; and the @cell context command to create a table cell for each group (YEAR).
sum(Sales)	<?sum(current-group()//SALES)?>	Sums the sales for the current group (YEAR).
end	<?end for-each-group?>	Closes the for-each-group statement.
end	<?end for-each-group?>	Closes the for-each-group statement.



Note that only the first row uses the @column context to determine the number of columns for the table. All remaining rows need to use the @cell context to create the table cells for the column. (For more information about context commands, see Using the Context Commands, page 4-129.)

## Dynamic Data Columns

The ability to construct dynamic data columns is a very powerful feature of the RTF template. Using this feature you can design a template that will correctly render a table when the number of columns required by the data is variable.

For example, you are designing a template to display columns of test scores within specific ranges. However, you do not know how many ranges will have data to report. You can define a dynamic data column to split into the correct number of columns at runtime.

Use the following tags to accommodate the dynamic formatting required to render the data correctly:

- Dynamic Column Header

```
<?split-column-header:group element name?>
```

Use this tag to define which group to split for the column headers of a table.

- Dynamic Column `<?split-column-data:group element name?>`

Use this tag to define which group to split for the column data of a table.

- Dynamic Column Width

```
<?split-column-width:name?> or
```

```
<?split-column-width:@width?>
```

Use one of these tags to define the width of the column when the width is described in the XML data. The width can be described in two ways:

- An XML element stores the value of the width. In this case, use the syntax `<?split-column-width:name?>`, where *name* is the XML element tag name that contains the value for the width.
- If the element defined in the `split-column-header` tag, contains a width attribute, use the syntax `<?split-column-width:@width?>` to use the value of that attribute.
- Dynamic Column Width's unit value (in points) `<?split-column-width-unit:value?>`

Use this tag to define a multiplier for the column width. If your column widths are defined in character cells, then you will need to use the appropriate multiplier value to render the columns to the correct width in points. For example, if you are using

10 point courier font in your table, you would use a multiplier of 6, which is the approximate width of a character displayed in 10 point courier font. If the multiplier is not defined, the widths of the columns are calculated as a percentage of the total width of the table. This is illustrated in the following table:

<b>Width Definition</b>	<b>Column 1 (Width = 10)</b>	<b>Column 2 (Width = 12)</b>	<b>Column 3 (Width = 14)</b>
Multiplier not present -% width	10/10+12+14*100 28%	%Width = 33%	%Width =39%
Multiplier = 6 - width	60 pts	72 pts	84 pts

### Defining Columns to Repeat Across Pages

If your table columns expand horizontally across more than one page, you can define how many row heading columns you want to repeat on every page. Use the following syntax to specify the number of columns to repeat:

```
<?horizontal-break-table: number?>
```

where `number` is the number of columns (starting from the left) to repeat.

Note that this functionality is supported for PDF output only.

### Example of Dynamic Data Columns

A template is required to display test score ranges for school exams. Logically, you want the report to be arranged as shown in the following table:

<b>Test Score</b>	<b>Test Score Range 1</b>	<b>Test Score Range 2</b>	<b>Test Score Range 3</b>	<b>...Test Score Range n</b>
Test Category	# students in Range 1	# students in Range 2	# students in Range 3	# of students in Range n

but you do not know how many Test Score Ranges will be reported. The number of Test Score Range columns is dynamic, depending on the data.

The following XML data describes these test scores. The number of occurrences of the element `<TestScoreRange>` will determine how many columns are required. In this case there are five columns: 0-20, 21-40, 41-60, 61-80, and 81-100. For each column there is an amount element (`<NumOfStudents>`) and a column width attribute (`<TestScore width="15">`).

```

<?xml version="1.0" encoding="utf-8"?>
<TestScoreTable>
  <TestScores>
    <TestCategory>Mathematics</TestCategory>
    <TestScore width ="15">
      <TestScoreRange>0-20</TestScoreRange>
      <NumofStudents>30</NumofStudents>
    </TestScore>
    <TestScore width ="20">
      <TestScoreRange>21-40</TestScoreRange>
      <NumofStudents>45</NumofStudents>
    </TestScore>
    <TestScore width ="15">
      <TestScoreRange>41-60</TestScoreRange>
      <NumofStudents>50</NumofStudents>
    </TestScore>
    <TestScore width ="20">
      <TestScoreRange>61-80</TestScoreRange>
      <NumofStudents>102</NumofStudents>
    </TestScore>
    <TestScore width ="15">
      <TestScoreRange>81-100</TestScoreRange>
      <NumofStudents>22</NumofStudents>
    </TestScore>
  </TestScores>
</TestScoreTable>

```

Using the dynamic column tags in form fields, set up the table in two columns as shown in the following figure. The first column, "Test Score" is static. The second column, "Column Header and Splitting" is the dynamic column. At runtime this column will split according to the data, and the header for each column will be appropriately populated. The Default Text entry and Form Field Help entry for each field are listed in the table following the figure. (See Form Field Method, page 4-9 for more information on using form fields).

Test Score	Column Header and Splitting
Group: TestScores Test Category	Content and Splitting end: TestScores

Default Text Entry	Form Field Help Text Entry
Group:TestScores	<?for-each:TestScores?>
Test Category	<?TestCategory?>
Column Header and Splitting	<?split-column-header:TestScore?> <?split-column-width:@width?> <?TestScoreRange?>%
Content and Splitting	<?split-column-data:TestScore?> <?NumofStudents?>

Default Text Entry	Form Field Help Text Entry
end:TestScores	<?end for-each?>

- **Test Score** is the boilerplate column heading.
- Test Category is the placeholder for the <TestCategory> data element, that is, "Mathematics," which will also be the row heading.
- The second column is the one to be split dynamically. The width you specify will be divided by the number of columns of data. In this case, there are 5 data columns.
- The second column will contain the dynamic "range" data. The width of the column will be divided according to the split column width. Because this example does not contain the unit value tag (<?split-column-width-unit:value?>), the column will be split on a percentage basis. Wrapping of the data will occur if required.

**Note:** If the tag (<?split-column-width-unit:value?>) were present, then the columns would have a specific width in points. If the total column widths were wider than the allotted space on the page, then the table would break onto another page.

The "horizontal-break-table" tag could then be used to specify how many columns to repeat on the subsequent page. For example, a value of "1" would repeat the column "Test Score" on the subsequent page, with the continuation of the columns that did not fit on the first page.

The template will render the output shown in the following figure:

Test Score	0-20	21-40	41-60	61-80	81-100
Mathematics	30	45	50	102	22

## Number, Date, and Currency Formatting

### Number Formatting

BI Publisher supports two methods for specifying the number format:

- Oracle's format-number function (recommended)

- Microsoft Word's Native number format mask

**Note:** You can also use the native XSL format-number function to format numbers. See: Native XSL Number Formatting, page 4-138.

Use only one of these methods. If the number format mask is specified using both methods, the data will be formatted twice, causing unexpected behavior.

The group separator and the number separator will be set at runtime based on the template locale. If you are working in a locale other than en-US, or your templates will require translation, use the Oracle format masks.

### Data Source Requirements

To use the Oracle format mask or the Microsoft format mask, the numbers in your data source must be in a raw format, with no formatting applied (for example: 1000.00). If the number has been formatted for European countries (for example: 1.000,00) the format will not work.

**Note:** The BI Publisher parser requires the Java BigDecimal string representation. This consists of an optional sign ("-") followed by a sequence of zero or more decimal digits (the integer), optionally followed by a fraction, and optionally followed by an exponent. For example: -123456.3455e-3.

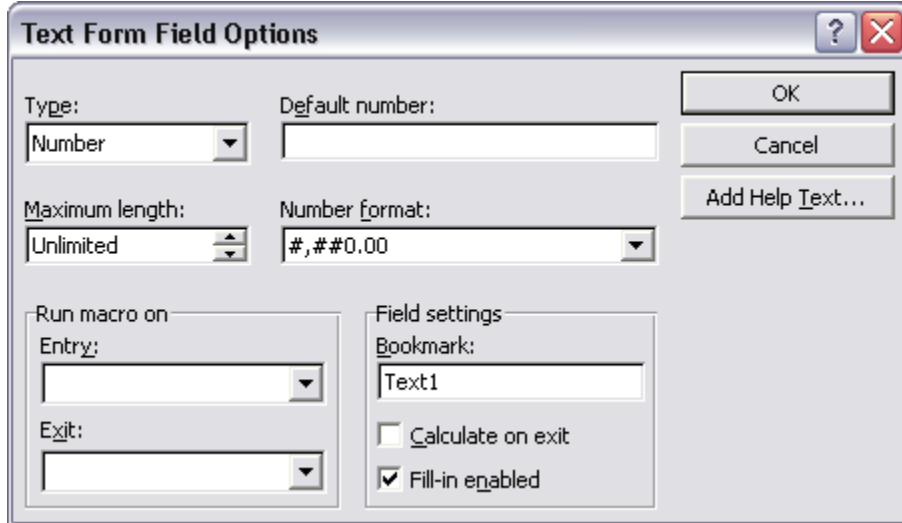
### Localization Considerations

If you are working in a locale other than en-US, or your templates will require translation, use the Oracle format masks. The Microsoft format masks can generate unexpected results in templates run in different locale settings.

Do not include "%" in the format mask because this will fix the location of the percent sign in the number display, while the desired position could be at the beginning or the end of a number, depending on the locale.

### Using the Microsoft Number Format Mask

To format numeric values, use Microsoft Word's field formatting features available from the Text Form Field Options dialog box. The following graphic displays an example:



To apply a number format to a form field:

1. Open the **Form Field Options** dialog box for the placeholder field.
2. Set the **Type** to Number.
3. Select the appropriate **Number format** from the list of options.

#### Supported Microsoft Format Mask Definitions

The following table lists the supported Microsoft format mask definitions:

Symbol	Location	Meaning
0	Number	Digit. Each explicitly set 0 will appear, if no other number occupies the position.  Example: Format mask: 00.0000 Data: 1.234 Display: 01.2340
#	Number	Digit. When set to #, only the incoming data is displayed.  Example: Format mask: ##.#### Data: 1.234 Display: 1.234

Symbol	Location	Meaning
.	Number	<p>Determines the position of the decimal separator. The decimal separator symbol used will be determined at runtime based on template locale.</p> <p>For example:</p> <p>Format mask: #,##0.00</p> <p>Data: 1234.56</p> <p>Display for English locale: 1,234.56</p> <p>Display for German locale: 1.234,56</p>
-	Number	Determines placement of minus sign for negative numbers.
,	Number	<p>Determines the placement of the grouping separator. The grouping separator symbol used will be determined at runtime based on template locale.</p> <p>For example:</p> <p>Format mask: #,##0.00</p> <p>Data: 1234.56</p> <p>Display for English locale: 1,234.56</p> <p>Display for German locale: 1.234,56</p>
E	Number	<p>Separates mantissa and exponent in a scientific notation.</p> <p>Example:</p> <p>0.###E+0 plus sign always shown for positive numbers</p> <p>0.###E-0 plus sign not shown for positive numbers</p>
;	Subpattern boundary	Separates positive and negative subpatterns. See Note below.
%	Prefix or Suffix	Multiply by 100 and show as percentage
'	Prefix or Suffix	Used to quote special characters in a prefix or suffix.

**Note:** Subpattern boundary: A pattern contains a positive and negative subpattern, for example, "#,##0.00;(##0.00)". Each subpattern has a prefix, numeric part, and suffix. The negative subpattern is optional. If

absent, the positive subpattern prefixed with the localized minus sign ("- in most locales) is used as the negative subpattern. That is, "0.00" alone is equivalent to "0.00;-0.00". If there is an explicit negative subpattern, it serves only to specify the negative prefix and suffix. The number of digits, minimal digits, and other characteristics are all the same as the positive pattern. That means that "#,##0.0#;(#)" produces precisely the same behavior as "#,##0.0#;(#,##0.0#)".

## Using the Oracle Format Mask

To apply the Oracle format mask to a form field:

1. Open the Form Field Options dialog box for the placeholder field.
2. Set the **Type** to "Regular text".
3. In the Form Field Help Text field, enter the mask definition according to the following example:

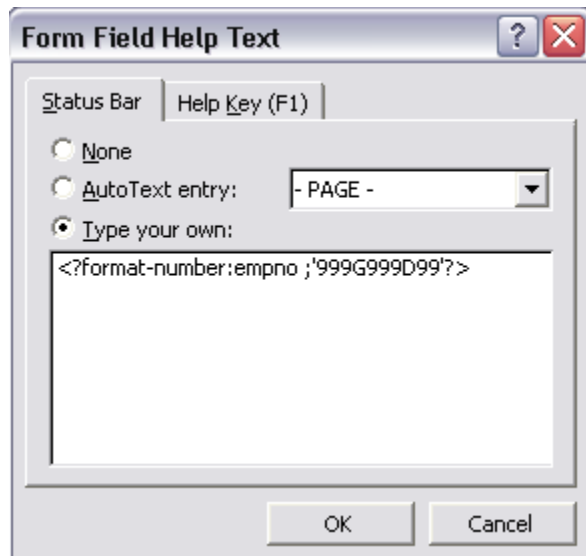
```
<?format-number:fieldname;'999G999D99'?>
```

where

*fieldname* is the XML tag name of the data element you are formatting and

*999G999D99* is the mask definition.

The following graphic shows an example Form Field Help Text dialog entry for the data element "empno":



The following table lists the supported Oracle number format mask symbols and their definitions:



<b>Symbol</b>	<b>Meaning</b>
0	<p>Digit. Each explicitly set 0 will appear, if no other number occupies the position.</p> <p>Example:</p> <p>Format mask: 00.0000</p> <p>Data: 1.234</p> <p>Display: 01.2340</p>
9	<p>Digit. Returns value with the specified number of digits with a leading space if positive or a leading minus if negative. Leading zeros are blank, except for a zero value, which returns a zero for the integer part of the fixed-point number.</p> <p>Example:</p> <p>Format mask: 99.9999</p> <p>Data: 1.234</p> <p>Display: 1.234</p>
C	Returns the ISO currency symbol in the specified position.
D	<p>Determines the placement of the decimal separator. The decimal separator symbol used will be determined at runtime based on template locale.</p> <p>For example:</p> <p>Format mask: 9G999D99</p> <p>Data: 1234.56</p> <p>Display for English locale: 1,234.56</p> <p>Display for German locale: 1.234,56</p>
EEEE	Returns a value in scientific notation.
G	<p>Determines the placement of the grouping (thousands) separator. The grouping separator symbol used will be determined at runtime based on template locale.</p> <p>For example:</p> <p>Format mask: 9G999D99</p> <p>Data: 1234.56</p> <p>Display for English locale: 1,234.56</p> <p>Display for German locale: 1.234,56</p>

Symbol	Meaning
L	Returns the local currency symbol in the specified position.
MI	Displays negative value with a trailing "-".
PR	Displays negative value enclosed by <>
PT	Displays negative value enclosed by ()
S (before number)	Displays positive value with a leading "+" and negative values with a leading "-"
S (after number)	Displays positive value with a trailing "+" and negative value with a trailing "-"

## Date Formatting

BI Publisher supports three methods for specifying the date format:

- Specify an explicit date format mask using Microsoft Word's native date format mask.
- Specify an explicit date format mask using Oracle's format-date function.
- Specify an abstract date format mask using Oracle's abstract date format masks. (Recommended for multilingual templates.)

Only one method should be used. If both the Oracle and MS format masks are specified, the data will be formatted twice causing unexpected behavior.

## Data Source Requirements

To use the Microsoft format mask or the Oracle format mask, the date from the XML data source must be in canonical format. This format is:

YYYY-MM-DDThh:mm:ss±HH:MM

where

- YYYY is the year
- MM is the month
- DD is the day
- T is the separator between the date and time component

- hh is the hour in 24-hour format
- mm is the minutes
- ss is the seconds
- ±HH:MM is the time zone offset from Universal Time (UTC), or Greenwich Mean Time

An example of this construction is:

2005-01-01T09:30:10-07:00

The data after the "T" is optional, therefore the following date: 2005-01-01 can be formatted using either date formatting option.

**Important:** Note that if the time component and time zone offset are not included in the XML source date, BI Publisher assumes it represents 12:00 AM UTC (that is, yyyy-mm-ddT00:00:00-00:00).

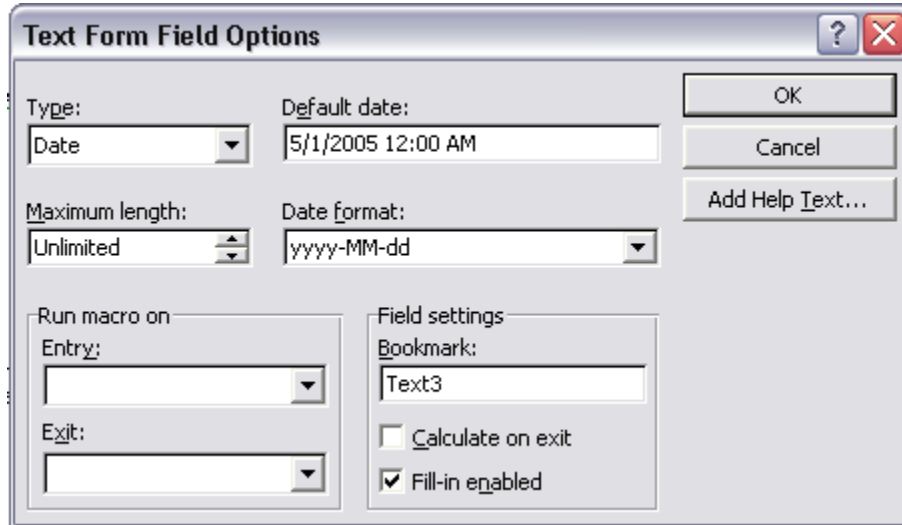
### Using the Microsoft Date Format Mask

To apply a date format to a form field:

1. Open the **Form Field Options** dialog box for the placeholder field.
2. Set the **Type** to Date, Current Date, or Current Time.
3. Select the appropriate **Date format** from the list of options.

If you do not specify the mask in the **Date format** field, the abstract format mask "MEDIUM" will be used as default. See Oracle Abstract Format Masks, page 4-119 for the description.

The following figure shows the Text Form Field Options dialog box with a date format applied:



The following table lists the supported Microsoft date format mask components:

Symbol	Meaning
d	The day of the month. Single-digit days will not have a leading zero.
dd	The day of the month. Single-digit days will have a leading zero.
ddd	The abbreviated name of the day of the week, as defined in <code>AbbreviatedDayNames</code> .
dddd	The full name of the day of the week, as defined in <code>DayNames</code> .
M	The numeric month. Single-digit months will not have a leading zero.
MM	The numeric month. Single-digit months will have a leading zero.
MMM	The abbreviated name of the month, as defined in <code>AbbreviatedMonthNames</code> .
MMMM	The full name of the month, as defined in <code>MonthNames</code> .
yy	The year without the century. If the year without the century is less than 10, the year is displayed with a leading zero.
yyyy	The year in four digits.
gg	The period or era. This pattern is ignored if the date to be formatted does not have an associated period or era string.

<b>Symbol</b>	<b>Meaning</b>
h	The hour in a 12-hour clock. Single-digit hours will not have a leading zero.
hh	The hour in a 12-hour clock. Single-digit hours will have a leading zero.
H	The hour in a 24-hour clock. Single-digit hours will not have a leading zero.
HH	The hour in a 24-hour clock. Single-digit hours will have a leading zero.
m	The minute. Single-digit minutes will not have a leading zero.
mm	The minute. Single-digit minutes will have a leading zero.
s	The second. Single-digit seconds will not have a leading zero.
ss	The second. Single-digit seconds will have a leading zero.
f	Displays seconds fractions represented in one digit.
ff	Displays seconds fractions represented in two digits.
fff	Displays seconds fractions represented in three digits.
ffff	Displays seconds fractions represented in four digits.
fffff	Displays seconds fractions represented in five digits.
ffffff	Displays seconds fractions represented in six digits.
fffffff	Displays seconds fractions represented in seven digits.
tt	The AM/PM designator defined in AMDesignator or PMDesignator, if any.
z	Displays the time zone offset for the system's current time zone in whole hours only. (This element can be used for formatting only)
zz	Displays the time zone offset for the system's current time zone in whole hours only. (This element can be used for formatting only)
zzz	Displays the time zone offset for the system's current time zone in hours and minutes.

Symbol	Meaning
:	The default time separator defined in TimeSeparator.
/	The default date separator defined in DateSeparator.
'	Quoted string. Displays the literal value of any string between two ' characters.
"	Quoted string. Displays the literal value of any string between two " characters.

### Using the Oracle Format Mask

To apply the Oracle format mask to a date field:

1. Open the **Form Field Options** dialog box for the placeholder field.
2. Set the **Type** to Regular Text.
3. Select the **Add Help Text...** button to open the **Form Field Help Text** dialog.
4. Insert the following syntax to specify the date format mask:

```
<?format-date:date_string;
'ABSTRACT_FORMAT_MASK'; 'TIMEZONE' ?>
```

or

```
<?format-date-and-calendar:date_string;
'ABSTRACT_FORMAT_MASK'; 'CALENDAR_NAME'; 'TIMEZONE' ?>
```

where time zone is optional. The detailed usage of format mask, calendar and time zone is described below.

If no format mask is specified, the abstract format mask "MEDIUM" will be used as default.

Example form field help text entry:

```
<?format-date:hiredate; 'YYYY-MM-DD' ?>
```

The following table lists the supported Oracle format mask components:

<b>Symbol</b>	<b>Meaning</b>
- / ' . ; :	Punctuation and quoted text are reproduced in the result.
"text"	
AD A.D.	AD indicator with or without periods.
AM A.M.	Meridian indicator with or without periods.
BC B.C.	BC indicator with or without periods.
CC	Century. For example, 2002 returns 21; 2000 returns 20.
DAY	Name of day, padded with blanks to length of 9 characters.
D	Day of week (1-7).
DD	Day of month (1-31).
DDD	Day of year (1-366).
DL	Returns a value in the long date format.
DS	Returns a value in the short date format.
DY	Abbreviated name of day.
E	Abbreviated era name.
EE	Full era name.

<b>Symbol</b>	<b>Meaning</b>
FF[1..9]	Fractional seconds. Use the numbers 1 to 9 after FF to specify the number of digits in the fractional second portion of the datetime value returned.  Example: 'HH:MI:SS.FF3'
HH	Hour of day (1-12).
HH12	Hour of day (1-12).
HH24	Hour of day (0-23).
MI	Minute (0-59).
MM	Month (01-12; JAN = 01).
MON	Abbreviated name of month.
MONTH	Name of month, padded with blanks to length of 9 characters.
PM	Meridian indicator with or without periods.
P.M.	
RR	Lets you store 20th century dates in the 21st century using only two digits.
RRRR	Round year. Accepts either 4-digit or 2-digit input. If 2-digit, provides the same return as RR. If you don't want this functionality, then simply enter the 4-digit year.
SS	Seconds (0-59).
TZD	Daylight savings information. The TZD value is an abbreviated time zone string with daylight savings information. It must correspond to the region specified in TZR.  Example: PST (for Pacific Standard Time) PDT (for Pacific Daylight Time)
TZH	Time zone hour. (See TZM format element.)



Symbol	Meaning
TZM	Time zone minute. (See TZH format element.)  Example:  'HH:MI:SS.FFTZH:TZM'
TZR	Time zone region information. The value must be one of the time zone regions supported in the database. Example: PST (Pacific Standard Time)
WW	Week of year (1-53) where week 1 starts on the first day of the year and continues to the seventh day of the year.
W	Week of month (1-5) where week 1 starts on the first day of the month and ends on the seventh.
X	Local radix character.
YYYY	4-digit year.
YY	Last 2, or 1 digit(s) of year.
Y	

### Default Format Mask

If you do not want to specify a format mask with either the MS method or the Oracle method, you can omit the mask definition and use the default format mask. The default format mask is the MEDIUM abstract format mask from Oracle. (See Oracle Abstract Format Masks, page 4-119 for the definition.)

To use the default option using the Microsoft method, set the **Type** to Date, but leave the **Date format** field blank in the **Text Form Field Options** dialog.

To use the default option using the Oracle method, do not supply a mask definition to the "format-date" function call, for example:

```
<?format-date:hiredate?>
```

### Oracle Abstract Format Masks

The abstract date format masks reflect the default implementations of date/time formatting in the I18N library. When you use one of these masks, the output generated will depend on the locale associated with the report.

Specify the abstract mask using the following syntax:

```
<?format-date:fieldname;'MASK'??>
```

where *fieldname* is the XML element tag and

MASK is the Oracle abstract format mask name

For example:

```
<?format-date:hiredate;'SHORT'??>  
<?format-date:hiredate;'LONG_TIME_TZ'??>  
<?format-date:xdoxslt:sysdate_as_xsdformat();'MEDIUM'??>
```

The following table lists the abstract format masks and the sample output that would be generated for US locale:

Mask	Output for US Locale
SHORT	2/31/99
MEDIUM	Dec 31, 1999
LONG	Friday, December 31, 1999
SHORT_TIME	12/31/99 6:15 PM
MEDIUM_TIME	Dec 31, 1999 6:15 PM
LONG_TIME	Friday, December 31, 1999 6:15 PM
SHORT_TIME_TZ	12/31/99 6:15 PM GMT
MEDIUM_TIME_TZ	Dec 31, 1999 6:15 PM GMT
LONG_TIME_TZ	Friday, December 31, 1999 6:15 PM GMT

## Displaying the System Date (sysdate) in Reports

To correctly display the sysdate, use the function `xdoxslt:sysdate_as_xsdformat()` with the `<?format-date:??>` command.

For example:

```
<?format-date:xdoxslt:sysdate_as_xsdformat();'MEDIUM'??>  
<?format-date:xdoxslt:sysdate_as_xsdformat();'LONG'??>  
<?format-date:xdoxslt:sysdate_as_xsdformat();'LONG_TIME_TZ'??>  
<?format-date-and-calendar:xdoxslt:sysdate_as_xsdformat();  
  'LONG_TIME';'ROC_OFFICIAL';??>
```

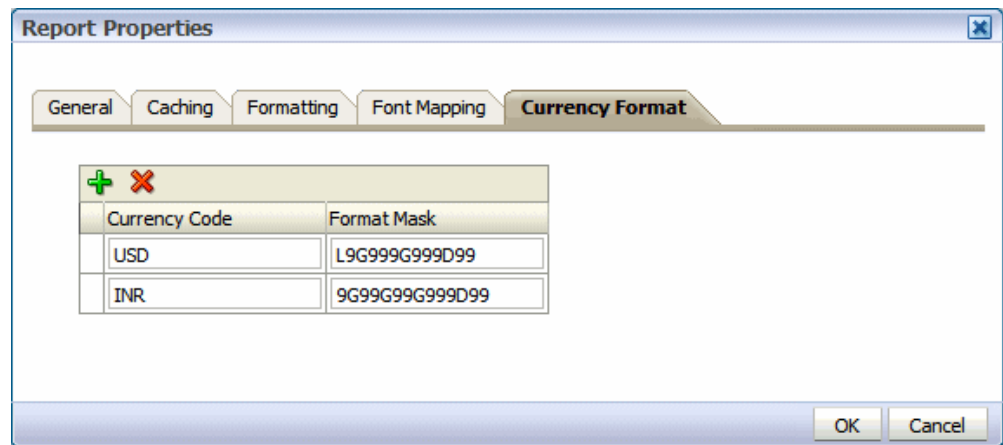
## Currency Formatting

BI Publisher enables you to define specific currency format masks to apply to your published data at runtime.

To utilize currency formatting, in your RTF template:

1. Set up your currency formats in BI Publisher's runtime configuration properties. The currency formats can be defined at the system level or at the report level.

When you set up the currency format property, you define the format to be used for a specified currency, using the International Standards Organization (ISO) currency code. A sample is shown in the following figure:



See *Configuring Currency Formats*, page 2-18 for more information.

2. Enter the `format-currency` command in your RTF template to apply the format to the field at runtime (usage described below).

### Applying a Currency Format to a Field

The parameters for the `format-currency` function are as follows:

```
<?format-currency:Amount_Field;CurrencyCode;displaySymbolOrNot?>
```

where

`Amount_Field` takes the tag name of the XML element that holds the amount value in your data.

`CurrencyCode` can either be set to a static value or it can be set dynamically. If the value will be static for the report, enter the ISO three-letter currency code in single-quotes, for example, 'USD'.

To set the value dynamically, enter the tag name of the XML element that holds the ISO currency code. Note that an element containing the currency code must be present in the data.

At runtime, the Amount\_Field will be formatted according to the format you set up for the currency code in the report properties.

displaySymbolOrNot takes one of the following values as shown in single quotes: 'true' or 'false'. When set to 'true', the currency symbol will be displayed in the report based on the value for CurrencyCode. If you do not wish the currency symbol to be displayed, you can either enter 'false' or simply do not specify the parameter.

#### Example: Displaying Multiple Currency Formats in a Report

The following example assumes you have set up the following currency formats in the report properties:

Currency Code	Format Mask
USD	9G999D99
INR	9G99G99G999D99

In this example, you do not need to set the currency code dynamically. You have the following elements in your XML data:

```
<TOTAL_SALES>
  <US_SALES>8596526459.56</US_SALES>
  <INDIA_SALES>60000000</INDIA_SALES>
</TOTAL_SALES>
```

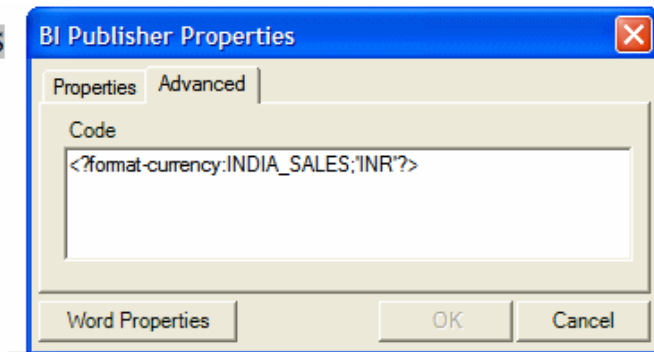
You want to display these two total fields in your template.

For US\_SALES, the syntax in the BI Publisher properties field will be:

```
<?format-currency:US_SALES;'USD'?>
```

The following figure shows the two fields in a template with the BI Publisher Properties dialog displaying the entry for INDIA\_SALES;

India Sales: INDIA\_SALES  
US Sales: US\_SALES



At runtime, the fields will display as follows:

India Sales: 6,00,00,000.00  
US Sales: 8,596,526,459.56

### Example: Displaying Multiple Currency Codes in a Single Report

The following simple XML sample contains an element containing the Amount (Trans\_amount) and an element containing the ISO currency code (Cur\_Code):

```
<ROW>  
<Trans_Amount>123</Trans_Amount>  
<Cur_Code>USD</Cur_Code>  
</ROW>  
<ROW>  
<Trans_Amount>-456</Trans_Amount>  
<Cur_Code>GBP</Cur_Code>  
</ROW>  
<ROW>  
<Trans_Amount>748</Trans_Amount>  
<Cur_Code>EUR</Cur_Code>  
</ROW>  
<ROW>  
<Trans_Amount>-987</Trans_Amount>  
<Cur_Code>JPY</Cur_Code>  
</ROW>
```

To display each of these amounts with the appropriate currency symbol, enter the following in your template for the field in which you want the amounts to display:

```
<?format-currency:Trans_Amount;Cur_Code;'true'?>
```

The following figure shows sample output that can be achieved:

USD Amount:	\$123.00
GBP Amount:	-£456.00
EUR Amount:	€ 748.00
JPY Amount:	-¥987

## Calendar and Timezone Support

### Calendar Specification

The term "calendar" refers to the calendar date displayed in the published report. The following types are supported:

- GREGORIAN
- ARABIC\_HIJRAH
- ENGLISH\_HIJRAH
- JAPANESE\_IMPERIAL
- THAI\_BUDDHA
- ROC\_OFFICIAL (Taiwan)

Use one of the following methods to set the calendar type:

- Call the format-date-and-calendar function and declare the calendar type.

For example:

```
<?format-date-and-calendar:hiredate;'LONG_TIME_TZ';'ROC_OFFICIAL';?>
```

The following graphic shows the output generated using this definition with locale set to zh-TW and time zone set to Asia/Taipei:

中華民國88年12月31日 星期五 下午 2:15 台北

- Set the calendar type using the profile option XDO: Calendar Type (XDO\_CALENDAR\_TYPE).

**Note:** The calendar type specified in the template will override the calendar type set in the profile option.

## Time Zone Specification

There are two ways to specify time zone information:

- Call the format-date or format-date-and-calendar function with the Oracle format.
- Set the user profile option Client Timezone (CLIENT\_TIMEZONE\_ID) in Oracle Applications.

If no time zone is specified, the report time zone is used.

In the template, the time zone must be specified as a Java time zone string, for example, America/Los Angeles. The following example shows the syntax to enter in the help text field of your template:

```
<?format-date:hiredate;'LONG_TIME_TZ';'Asia/Shanghai'?>
```

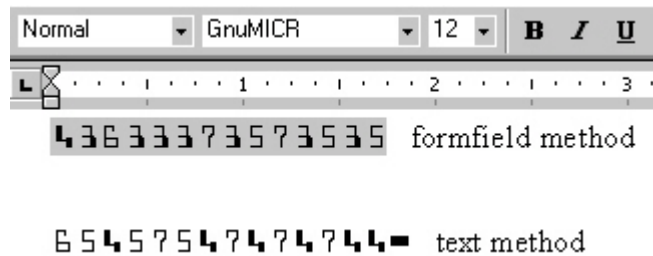
## Using External Fonts

BI Publisher enables you to use external fonts in your output that are not normally available on the server. To set up a new font for your report output, use the font to design your template on your client machine, then make it available on the server, and configure BI Publisher to access the font at runtime.

**Note:** External fonts are supported for PDF output only.

1. Use the font in your template.
  1. Copy the font to your `<WINDOWS_HOME>/fonts` directory.
  2. Open Microsoft Word and build your template.
  3. Insert the font in your template: Select the text or form field and then select the desired font from the font dialog box (Format > Font) or font drop down list.

The following graphic shows an example of the form field method and the text method:



2. Place the font on the BI Publisher server in the `ORACLE_HOME/common/fonts` directory.

**Note:** The predefined fonts are located in the Oracle Business Intelligence Oracle home, in: `ORACLE_HOME/common/fonts`. The font location is set by the `XDO_FONT_DIR` variable. If this variable is not set in your environment the fonts will be located in `$JAVA_HOME/jre/lib/fonts`.

3. Set the BI Publisher "font" property.

You can set the font property for the report in the BI Publisher Font Mappings page, or in the configuration file.

**To set the property in the Font Mappings page:**

To set the property in the Font Mappings page: Open the report in the report editor. Click **Properties**, then click **Font Mappings**. Enter the font and then select the font to which you want to map it. See *Configuring Report Properties*, page 2-11.

**To set the property in the configuration file:**

Update the BI Publisher configuration file "fonts" section with the font name and its location on the server. For example, the new entry for a TrueType font is structured as follows:

```
<font family="MyFontName" style="normal" weight="normal">  
  <truetype path="\user\fonts\MyFontName.ttf"/>  
</font>
```

See *BI Publisher Configuration File, Oracle Fusion Middleware Administrator's Guide for Oracle Business Intelligence Publisher* for more information.

Now you can run your report and BI Publisher will use the font in the output as designed. For PDF output, the advanced font handling features of BI Publisher embed the external font glyphs directly into the final document. The embedded font only contains the glyphs required for the document and not the complete font definition. Therefore the document is completely self-contained, eliminating the need to have external fonts installed on the printer.

## Using the Barcode Fonts Shipped with BI Publisher

The following barcodes are bundled with BI Publisher:

Font File	Supported Algorithm
128R00.TTF	code128a, code128b, and code128c
B39R00.TTF	code39, code39mod43
UPCR00.TTF	upca, upce

When you use one of these prepackaged fonts, BI Publisher will execute the preprocessing on your data prior to applying the barcode font to the data in the output document. For example, to calculate checksum values or start and end bits for the data before formatting them.

At design time it is not necessary that you apply the barcode font to the field in Microsoft Word. Instead, you can map the font that you apply to the field using BI Publisher's font mapping. At runtime, BI Publisher will apply the barcode font to any field using the base font you specified in the font mapping. Be sure to choose a font that is not used elsewhere in your template. For information on font mapping, see *Configuring Report Properties*, page 2-11.

If you want to use the font directly in Microsoft Word, then add the appropriate .TTF



file to your C:\WINDOWS\Fonts directory. To use the Template Builder Preview function, map the font in the Template Builder configuration file. See *Configuring Fonts for the BI Publisher Template Builder*, page 4-141.

To use these in your report output, perform the following:

1. Insert a field in your template where the barcode is to display in your report output.
2. In the form field, enter the following command:

```
<?format-barcode:data;'barcode_type'??>
```

where

data is the element from your XML data source to be encoded. For example:  
INVOICE\_NO

barcode\_type is one of the supported algorithms listed above.

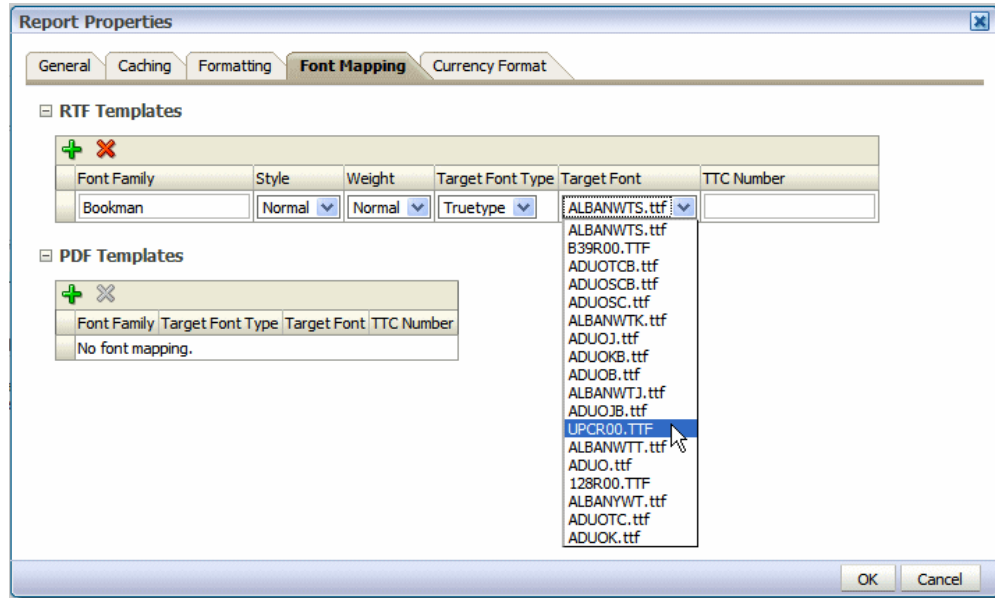
Examples:

```
<?format-barcode:INVOICE_NO;'code128a'??>
```

```
<?format-barcode:INVOICE_NO;'code39mod43'??>
```

```
<?format-barcode:INVOICE_NO;'upca'??>
```

3. In Microsoft Word, apply the font to the field. If you have not installed the barcode fonts on your client machine, then select a font that is not used elsewhere in your template, for example, Bookman.
4. Configure the font in the Font Mapping page. For more information about the Font Mapping page, see *Configuring Report Properties*, page 2-11. Following is a figure of the Font Mapping page:



Note the following:

- Microsoft Word may not render the barcode fonts properly even when they are installed on your client. To work around this issue, apply a different font to the field and map the font as described above.
- The upca algorithm accepts only UPC-A message string and encodes into UPC-A barcode.
- A string of 12 characters will be treated as UPC-A message with a check digit, 11 will be without a check digit.
- The upce algorithm accepts only UPC-E message strings and encodes into UPC-E barcode.
- A string of 8 characters will be treated as a UPC-E message with both a front and end guard bar; a string of 6 characters will be without guard bars.

## Custom Barcode Formatting

If you choose not to use one of the barcode fonts provided above, use this procedure to implement a custom barcode.

BI Publisher offers the ability to execute preprocessing on your data prior to applying a barcode font to the data in the output document. For example, you may need to calculate checksum values or start and end bits for the data before formatting them.

The solution requires that you register a barcode encoding class with BI Publisher that can then be instantiated at runtime to carry out the formatting in the template. This is

covered in *Advanced Barcode Font Formatting Class Implementation, Oracle Fusion Middleware Administrator's Guide for Oracle Business Intelligence Publisher*.

To enable the formatting feature in your template, you must use two commands in your template. The first command registers the barcode encoding class with BI Publisher. This must be declared somewhere in the template prior to the encoding command. The second is the encoding command to identify the data to be formatted.

## Register the Barcode Encoding Class

Use the following syntax in a form field in your template to register the barcode encoding class:

```
<?register-barcode-vendor:java_class_name;barcode_vendor_id?>
```

This command requires a Java class name (this will carry out the encoding) and a barcode vendor ID as defined by the class. This command must be placed in the template before the commands to encode the data in the template. For example:

```
<?register-barcode-vendor:'oracle.xdo.template.rtf.util.barcoder.BarcodeUtil';'XMLPBarVendor'?>
```

where

`oracle.xdo.template.rtf.util.barcoder.BarcodeUtil` is the Java class and `XMLPBarVendor` is the vendor ID that is defined by the class.

## Encode the Data

To format the data, use the following syntax in a form field in your template:

```
<?format-barcode:data;'barcode_type';'barcode_vendor_id'?>
```

where

`data` is the element from your XML data source to be encoded. For example:  
`LABEL_ID`

`barcode_type` is the method in the encoding Java class used to format the data (for example: `Code128a`).

`barcode_vendor_id` is the ID defined in the `register-barcode-vendor` field of the first command you used to register the encoding class.

For example:

```
<?format-barcode:LABEL_ID;'Code128a';'XMLPBarVendor'?>
```

At runtime, the `barcode_type` method is called to format the data value and the barcode font will then be applied to the data in the final output.

## Controlling the Placement of Instructions Using the Context Commands

The BI Publisher syntax is simplified XSL instructions. This syntax, along with any native XSL commands you may use in your template, is converted to XSL-FO at

runtime. The placement of these instructions within the converted stylesheet determines the behavior of your template.

BI Publisher's RTF processor places these instructions within the XSL-FO stylesheet according to the most common context. However, sometimes you need to define the context of the instructions differently to create a specific behavior. To support this requirement, BI Publisher provides a set of context commands that allow you to define the context (or placement) of the processing instructions. For example, using context commands, you can:

- Specify an if statement in a table to refer to a cell, a row, a column or the whole table.
- Specify a for-each loop to repeat either the current data or the complete section (to create new headers and footers and restart the page numbering)
- Define a variable in the current loop or at the beginning of the document.

You can specify a context for both processing commands using the BI Publisher syntax and those using native XSL.

- To specify a context for a processing command using the simplified BI Publisher syntax, simply add `@context` to the syntax instruction. For example:
  - `<?for-each@section:INVOICE?>` - specifies that the group INVOICE should begin a new section for each occurrence. By adding the section context, you can reset the header and footer and page numbering.  
  
If you do not wish to restart the page numbering, add the command:  
`<?initial-page-number: 'auto' ?>` after the `@section` command to continue the page numbering across sections.
  - `<?if@column:VAT?>` - specifies that the if statement should apply to the VAT column only.
- To specify a context for an XSL command, add the `xdofo:ctx="context"` attribute to your tags to specify the context for the insertion of the instructions. The value of the context determines where your code is placed.

For example:

```
<xsl:for-each xdofo:ctx="section" select ="INVOICE">
<xsl:attribute xdofo:ctx="inblock"
name="background-color">red</xsl:attribute>
```

BI Publisher supports the following context types:

Context	Description
section	<p>The statement affects the whole section including the header and footer. For example, a <code>for-each@section</code> context command creates a new section for each occurrence - with restarted page numbering and header and footer.</p> <p>Note that you can retain continuous page numbering across sections by using the <code>&lt;?initial-page-number: 'auto'?&gt;</code> command.</p> <p>See <i>Batch Reports</i>, page 4-97 for an example of this usage.</p>
column	<p>The statement will affect the whole column of a table. This context is typically used to show and hide table columns depending on the data.</p> <p>See <i>Column Formatting</i>, page 4-69 for an example.</p>
cell	<p>The statement will affect the cell of a table. This is often used together with <code>@column</code> in pivot tables to create a dynamic number of columns.</p> <p>See <i>Pivot Support</i>, page 4-100 for an example.</p>
block	<p>The statement will affect multiple complete <code>fo:blocks</code> (RTF paragraphs). This context is typically used for <code>if</code> and <code>for-each</code> statements. It can also be used to apply formatting to a paragraph or a table cell.</p> <p>See <i>Cell Highlighting</i>, page 4-73 for an example.</p>
inline	<p>The context will become the single statement inside an <code>fo:inline</code> block. This context is used for variables.</p>
incontext	<p>The statement is inserted immediately after the surrounding statement. This is the default for <code>&lt;?sort?&gt;</code> statements that need to follow the surrounding <code>for-each</code> as the first element.</p>
inblock	<p>The statement becomes a single statement inside an <code>fo:block</code> (RTF paragraph). This is typically not useful for control statements (such as <code>if</code> and <code>for-each</code>) but is useful for statements that generate text, such as <code>call-template</code>.</p>
inlines	<p>The statement will affect multiple complete inline sections. An inline section is text that uses the same formatting, such as a group of words rendered as bold.</p> <p>See <i>If Statements in Boilerplate Text</i>, page 4-65.</p>
begin	<p>The statement will be placed at the beginning of the XSL stylesheet. This is required for global variables. See <i>Defining Parameters</i>, page 4-93.</p>
end	<p>The statement will be placed at the end of the XSL stylesheet.</p>

The following table shows the default context for the BI Publisher commands:

Command	Context
apply-template	inline
attribute	inline
call-template	inblock
choose	block
for-each	block
if	block
import	begin
param	begin
sort	incontext
template	end
value-of	inline
variable	end

## Using XPath Commands

XPath is an industry standard developed by the World Wide Web Consortium (W3C). It is the method used to navigate through an XML document. XPath is a set of syntax rules for addressing the individual pieces of an XML document. You may not know it, but you have already used XPath; RTF templates use XPath to navigate through the XML data at runtime.

This section contains a brief introduction to XPath principles. For more information, see the W3C Web site: <http://www.w3.org/TR/xpath>

XPath follows the Document Object Model (DOM), which interprets an XML document as a tree of nodes. A node can be one of seven types:

- root

- element
- attribute
- text
- namespace
- processing instruction
- comment

Many of these elements are shown in the following sample XML, which contains a catalog of CDs:

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- My CD Listing -->
<CATALOG>
  <CD cattype=Folk>
    <TITLE>Empire Burlesque</TITLE>
    <ARTIST>Bob Dylan</ARTIST>
    <COUNTRY>USA</COUNTRY>
    <PRICE>10.90</PRICE>
    <YEAR>1985</YEAR>
  </CD>
  <CD cattype=Rock>
    <TITLE>Hide Your Heart</TITLE>
    <ARTIST>Bonnie Tylor</ARTIST>
    <COUNTRY>UK</COUNTRY>
    <PRICE>9.90</PRICE>
    <YEAR>1988</YEAR>
  </CD>
</CATALOG>
```

The root node in this example is CATALOG. CD is an element, and it has an attribute cattype. The sample contains the comment My CD Listing. Text is contained within the XML document elements.

## Locating Data

Locate information in an XML document using location-path expressions.

A node is the most common search element you will encounter. Nodes in the example CATALOG XML include CD, TITLE, and ARTIST. Use a path expression to locate nodes within an XML document. For example, the following path returns all CD elements:

```
//CATALOG/CD
```

where

the double slash (//) indicates that all elements in the XML document that match the search criteria are to be returned, regardless of the level within the document.

the slash (/) separates the child nodes. All elements matching the pattern will be returned.

To retrieve the individual `TITLE` elements, use the following command:

```
/CATALOG/CD/TITLE
```

This example will return the following XML:

```
<CATALOG>
  <CD cattype=Folk>
    <TITLE>Empire Burlesque</TITLE>
  </CD>
  <CD cattype=Rock>
    <TITLE>Hide Your Heart</TITLE>
  </CD>
</CATALOG>
```

Further limit your search by using square brackets. The brackets locate elements with certain child nodes or specified values. For example, the following expression locates all CDs recorded by Bob Dylan:

```
/CATALOG/CD[ARTIST="Bob Dylan"]
```

Or, if each CD element did not have an `PRICE` element, you could use the following expression to return only those CD elements that include a `PRICE` element:

```
/CATALOG/CD[PRICE]
```

Use the bracket notation to leverage the attribute value in your search. Use the `@` symbol to indicate an attribute. For example, the following expression locates all Rock CDs (all CDs with the `cattype` attribute value `Rock`):

```
//CD[@cattype="Rock"]
```

This returns the following data from the sample XML document:

```
<CD cattype=Rock>
  <TITLE>Hide Your Heart</TITLE>
  <ARTIST>Bonnie Tylor</ARTIST>
  <COUNTRY>UK</COUNTRY>
  <PRICE>9.90</PRICE>
  <YEAR>1988</YEAR>
</CD>
```

You can also use brackets to specify the item number to retrieve. For example, the first CD element is read from the XML document using the following XPath expression:

```
/CATALOG/CD[1]
```

The sample returns the first CD element:

```
<CD cattype=Folk>
  <TITLE>Empire Burlesque</TITLE>
  <ARTIST>Bob Dylan</ARTIST>
  <COUNTRY>USA</COUNTRY>
  <PRICE>10.90</PRICE>
  <YEAR>1985</YEAR>
</CD>
```

XPath also supports wildcards to retrieve every element contained within the specified node. For example, to retrieve all the CDs from the sample XML, use the following expression:

```
/CATALOG/*
```



You can combine statements with Boolean operators for more complex searches. The following expression retrieves all Folk and Rock CDs, thus all the elements from the sample:

```
//CD[@cattype="Folk"]|//CD[@cattype="Rock"]
```

The pipe (|) is equal to the logical OR operator. In addition, XPath recognizes the logical OR and AND, as well as the equality operators: <=, <, >, >=, ==, and !=. For example, we can find all CDs released in 1985 or later using the following expression:

```
/CATALOG/CD[YEAR >=1985]
```

## Starting Reference

The first character in an XPath expression determines the point at which it should start in the XML tree. Statements beginning with a forward slash (/) are considered absolute. No slash indicates a relative reference. An example of a relative reference is:

```
CD/*
```

This statement begins the search at the current reference point. That means if the example occurred within a group of statements the reference point left by the previous statement would be utilized.

As noted earlier, double forward slashes (//) retrieve every matching element regardless of location in the document, therefore the use of double forward slashes (//) should be used only when necessary to improve performance.

## Context and Parent

To select current and parent elements, XPath recognizes the dot notation commonly used to navigate directories. Use a single period (.) to select the current node and use double periods (..) to return the parent of the current node. For example, to retrieve all child nodes of the parent of the current node, use:

```
../*
```

Therefore, to access all CDs from the sample XML, use the following expression:

```
/CATALOG/CD/..
```

You could also access all the CD titles released in 1988 using the following:

```
/CATALOG/CD/TITLE[../YEAR=1988]
```

The .. is used to navigate up the tree of elements to find the YEAR element at the same level as the TITLE, where it is then tested for a match against "1988". You could also use // in this case, but if the element YEAR is used elsewhere in the XML document, you may get erroneous results.

XPath is an extremely powerful standard when combined with RTF templates allowing you to use conditional formatting and filtering in your template.

## Namespace Support

If your XML data contains namespaces, you must declare them in the template prior to referencing the namespace in a placeholder. Declare the namespace in the template using either the basic RTF method or in a form field. Enter the following syntax:

```
<?namespace:namespace name= namespace url?>
```

For example:

```
<?namespace:fsg=http://www.oracle.com/fsg/2002-30-20/?>
```

Once declared, you can use the namespace in the placeholder markup, for example:

```
<?fsg:ReportName?>
```

## Using XSL Elements

You can use any XSL element in your template by inserting the XSL syntax into a form field.

If you are using the basic RTF method, you cannot insert XSL syntax directly into your template. BI Publisher has extended the following XSL elements for use in RTF templates.

To use these in a basic-method RTF template, you must use the BI Publisher Tag form of the XSL element. If you are using form fields, use either option.

### Apply a Template Rule

Use this element to apply a template rule to the current element's child nodes.

**XSL Syntax:** `<xsl:apply-templates select="name">`

**BI Publisher Tag:** `<?apply:name?>`

This function applies to `<xsl:template-match="n">` where *n* is the element name.

### Copy the Current Node

Use this element to create a copy of the current node.

**XSL Syntax:** `<xsl:copy-of select="name">`

**BI Publisher Tag:** `<?copy-of:name?>`

### Call Template

Use this element to call a named template to be inserted into or applied to the current template. For example, use this feature to render a table multiple times.

**XSL Syntax:** `<xsl:call-template name="name">`

**BI Publisher Tag:** `<?call-template:name?>`

## Template Declaration

Use this element to apply a set of rules when a specified node is matched.

**XSL Syntax:** `<xsl:template name="name">`

**BI Publisher Tag:** `<?template:name?>`

## Variable Declaration

Use this element to declare a local or global variable.

**XSL Syntax:** `<xsl:variable name="name">`

**BI Publisher Tag:** `<?variable:name?>`

**Example:**

```
<xsl:variable name="color" select="'red'"/>
```

Assigns the value "red" to the "color" variable. The variable can then be referenced in the template.

## Import Stylesheet

Use this element to import the contents of one style sheet into another.

**Note:** An imported style sheet has lower precedence than the importing style sheet.

**XSL Syntax:** `<xsl:import href="url">`

**BI Publisher Tag:** `<?import:url?>`

## Define the Root Element of the Stylesheet

This and the `<xsl:stylesheet>` element are completely synonymous elements. Both are used to define the root element of the style sheet.

**Note:** An included style sheet has the same precedence as the including style sheet.

**XSL Syntax:** `<xsl:stylesheet xmlns:x="url">`

**BI Publisher Tag:** `<?namespace:x=url?>`

**Note:** The namespace must be declared in the template. See Namespace

## Native XSL Number Formatting

The native XSL `format-number` function takes the basic format:

```
format-number (number, format, [decimalformat])
```

---

Parameter	Description
number	Required. Specifies the number to be formatted.
format	Required. Specifies the format pattern. Use the following characters to specify the pattern: <ul style="list-style-type: none"><li>• # (Denotes a digit. Example: ####)</li><li>• 0 (Denotes leading and following zeros. Example: 0000.00)</li><li>• . (The position of the decimal point Example: ###.##)</li><li>• , (The group separator for thousands. Example: ##,###.##)</li><li>• % (Displays the number as a percentage. Example: ##%)</li><li>• ; (Pattern separator. The first pattern will be used for positive numbers and the second for negative numbers)</li></ul>
decimalformat	Optional. For more information on the decimal format please consult any basic XSLT manual.

---

## Using FO Elements

You can use the native FO syntax inside the Microsoft Word form fields.

For more information on XSL-FO see the W3C Website at <http://www.w3.org/2002/08/XSLFOsummary.html>

The full list of FO elements supported by BI Publisher can be found in the Appendix: Supported XSL-FO Elements, page D-1.

## Guidelines for Designing RTF Templates for Microsoft PowerPoint Output

BI Publisher can generate your RTF template as PowerPoint output enabling you to get

report data into your key business presentations. Currently, the PowerPoint document generated is a simple export of the formatted data and charts to PowerPoint.

## Limitations

Following are limitations when working with PowerPoint as an output:

- When designing tables for your PowerPoint slide, you must define the table border type as a single line (double border, dash, and other types are not supported).
- Hyperlinks are not supported.
- Shapes are not supported.
- Text position may be slightly incorrect if you use right-align.
- Paper size must be the same on all pages of the RTF template. You cannot have mixed paper sizes in the same document.
- Bidirectional languages are not supported.
- Text position may be slightly incorrect for Chinese, Japanese, and Korean fonts when using bold or italic effects on characters. This is because Microsoft uses bold or italic emulation when there is no bold or italic font.
- All Unicode languages, except bidirectional languages, are supported.
- BI Publisher's font fallback mechanism is not supported for PowerPoint templates. If you wish to use a font that is not installed, ensure that you have configured it with BI Publisher.

## Usage Guidelines

Following are guidelines to help you when designing an RTF template intended for PowerPoint output:

- PowerPoint output will preserve the page orientation (portrait or landscape) defined in the RTF template. Most presentations are oriented in landscape so this is the recommended orientation of your RTF template.
- A page break in your RTF template will generate a new slide.
- The background color of the slides are always generated as white. If you prefer a different background color, you must change the color after the PowerPoint file is generated.
- When highlighting characters in the page header or footer, when the font is not

predefined in xdo.cfg, be sure to specify the font for the whole tag "<?XXXXXX?>", including the "<?" and "?>" in the template.

## About Charts in PowerPoint Output

BI Publisher supports native PowerPoint charts for certain chart types rendered in PowerPoint2007 output. When the chart is inserted as a native chart, you can modify it in PowerPoint. If the chart is not inserted as a native chart, then BI Publisher inserts a PNG image that cannot be updated.

The following chart types can be rendered as native PowerPoint charts in PowerPoint2007 output:

- Pie
- Ring
- Line
- Area
- Radar
- Bubble
- Pareto
- Combination
- Stock

Any chart type that is not native to PowerPoint (for example, gauge or funnel) will be converted to a bar chart.

By default, native chart insertion is enabled. To disable native chart insertion for a report, set the report property **Enable PPTX native chart support** to false. See Setting Report Processing and Output Document Properties, page 10-1 for more information.

Note that when **Enable PPTX native chart support** is set to false, all charts will be rendered as images in your PowerPoint2007 output. Therefore, set this option to false only when your report includes the non-native chart types.

## Configuring Fonts for the BI Publisher Server

Support for PowerPoint output does not include the font fallback mechanism that is used for other types of output in BI Publisher. If you are using a nonstandard font in your template, you must configure the BI Publisher server for each font used in the RTF template for generating PowerPoint output. You will need to copy these fonts to your BI Publisher Server and define the Font Mappings for RTF templates. This can be done for

the entire system or for individual reports. See Defining Font Mappings, page 10-19 for more details.

## Configuring Fonts for the BI Publisher Template Builder

When using the BI Publisher Template Builder to design your report, to correctly preview PPT output that uses non-English or non-standard fonts, you will need to define the fonts in the BI Publisher configuration file. This configuration file is called `xdo.cfg` and is typically found in:

C:\Oracle\BI Publisher\BI Publisher Desktop\Template Builder for Word\config\

Note that if you have not used this file yet you may find the file "`xdo example.cfg`" instead. This file must be saved with an encoding of UTF-8 and provide a full and absolute path for each font defined. Otherwise, you will encounter issues such as characters overlays and wrapping that does not work.

1. Navigate to C:\Oracle\BI Publisher\BI Publisher Desktop\Template Builder for Word\config\
2. Open the `xdo.cfg` file and update the font mappings. For information on updating font mappings directly in the `xdo.cfg` file, see *Font Definitions, Oracle Fusion Middleware Administrator's and Developer's Guide for Business Intelligence Publisher*.
3. Save the `xdo.cfg` in UTF-8 format.

The following figure shows a sample `xdo.cfg` file:

```
<config version="1.0.0" xmlns="http://xmlns.oracle.com/oxp/config/">
  <!-- Properties -->
  <properties>
  </properties>

  <!-- Font setting -->
  <font>
    <font family="宋体" style="" weight="">
      <truetype
        path="C:\Program Files\Oracle\BI Publisher\BI Publisher Desktop\Template Builder for Word\fonts\simsun.ttc"/>
      </font>
    <font family="Times New Roman" style="" weight="">
      <truetype
        path="C:\Program Files\Oracle\BI Publisher\BI Publisher Desktop\Template Builder for Word\fonts\times.ttf"/>
      </font>
    </font>
  </font>
</config>
```





---

## Creating RTF Templates Using the Template Builder for Word

This chapter covers the following topics:

- Introduction
- Getting Started
- Accessing Data for Building Your Template
- Inserting Components to the Template
- Previewing a Template Template Builder preview options
- Template Editing Tools
- Uploading a Template to the BI Publisher Server
- Using the Template Builder Translation Tools
- Setting Options for the Template Builder
- Setting Up a Configuration File
- BI Publisher Menu Reference

### Introduction

The Template Builder is an add-in to Microsoft Word that simplifies the development of RTF templates. While the Template Builder is not required to create RTF templates, it provides many functions that will increase your productivity.

The Template Builder is tightly integrated with Microsoft Word and enables you to perform the following functions:

- Insert data fields
- Insert tables

- Insert forms
- Insert charts
- Preview your template with sample XML data
- Browse and update the content of form fields
- Extract boilerplate text into an XLIFF translation file and test translations

Note that the Template Builder automates insertion of the most frequently used components of an RTF template. RTF templates also support much more complex formatting and processing. For the full description of RTF template features, see *Creating RTF Templates*, page 4-2.

## Before You Get Started:

Your Template Builder installation provides samples and demo files to help you get started. The demos can be accessed from your Windows Start menu as follows:

Start > Programs > Oracle BI Publisher Desktop > Demos

The demos can also be accessed from the BI Publisher\BI Publisher Desktop\demos folder where you installed BI Publisher Desktop (for example: C:\Program Files\Oracle\BI Publisher\BI Publisher Desktop\demos).

The following demos are provided:

- Report Demo - demonstrates building a report layout using many key features of the Template Builder, including: connecting to the BI Publisher server, loading data for a report, inserting tables and charts, and defining conditional formatting.
- Invoice Demo - demonstrates how to take a prepared layout and use the Template Builder to insert the required fields to fill the template with data at runtime.
- Localization Demo - demonstrates the localization capabilities of the Template Builder and shows you how to extract an XLIFF file from the base RTF template. The XLIFF file can then be translated and the translations previewed in the Template Builder.

The sample files are located in the BI Publisher\BI Publisher Desktop\Samples folder. The Samples folder contains three subfolders:

- eText templates
- PDF templates
- RTF templates

The eText and PDF template samples can be used as references to create these types of

templates. The Template Builder is only available for the RTF templates. The RTF templates folder contains eight subfolders to provide samples of different types of reports. Refer to the TrainingGuide.html located in the RTF templates folder for additional information on what is contained in each sample.

## Prerequisites and Limitations

### Prerequisites:

- Your report data model has been created and runs successfully. For information on creating Data Models, see *Creating Data Models, Oracle Fusion Middleware Data Modeling Guide for Oracle Business Intelligence Publisher*.
- Supported versions of Microsoft Word and Microsoft Windows are installed on your client.

**Note:** See System Requirements and Certification, page xiv for the most up-to-date information on supported hardware and software.

- The BI Publisher Template Builder has been downloaded and installed on your client.

The Template Builder can be downloaded from the **Get Started** region of the **Home** page.

### Limitations

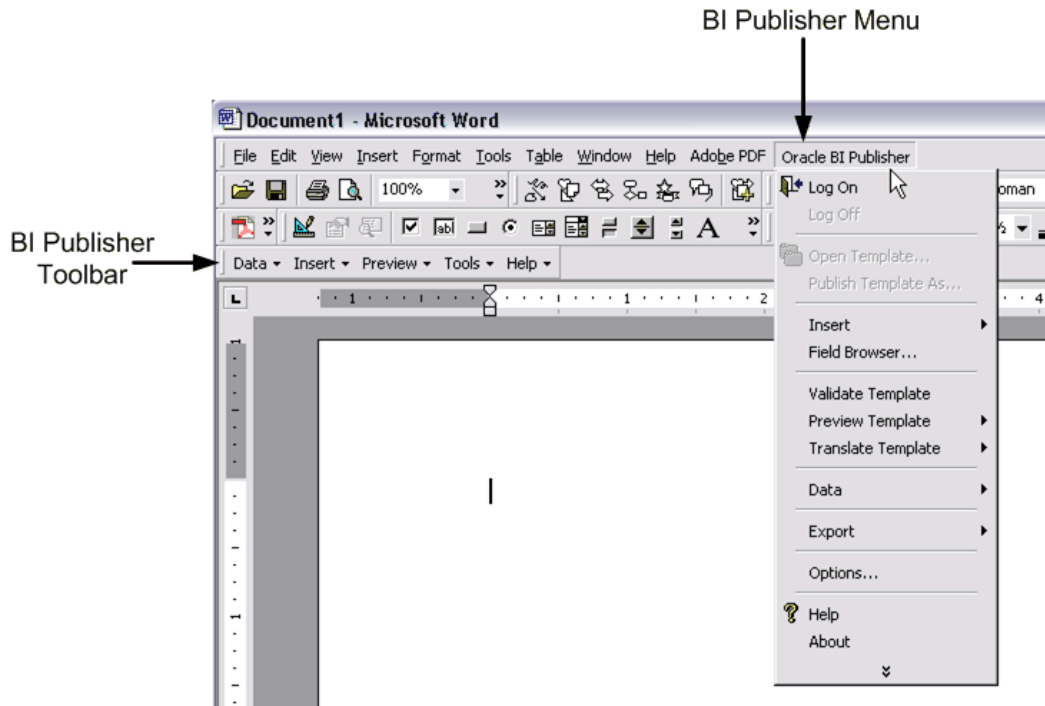
- The Template Builder does not support bidirectional display of text in the user interface.

## Getting Started

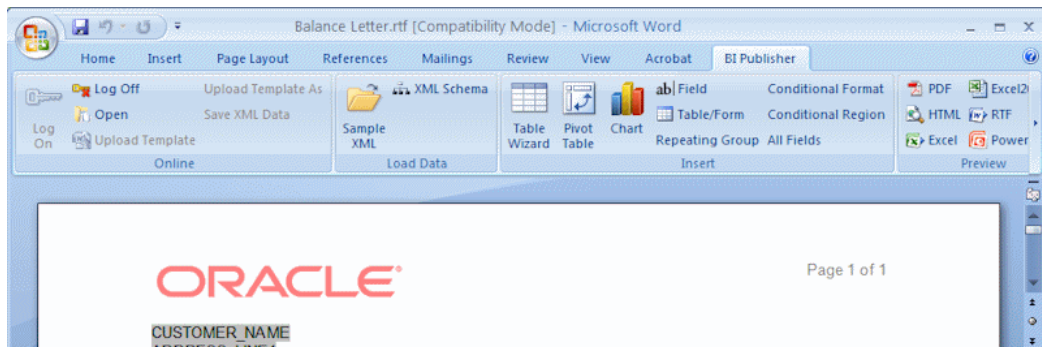
### Features of the Oracle BI Publisher Template Builder for Word

When you open Microsoft Word after installing the Template Builder you will notice the Oracle BI Publisher menu.

For versions of Microsoft Word prior to 2007 the menu and toolbar will appear as shown in the following figure:



For Microsoft Word 2007 users, the BI Publisher commands display in the ribbon format:



Use the menu (or toolbar) to perform the following:

- Insert data fields into your RTF templates
- Insert tables, forms, charts, and pivot tables
- Preview your template in multiple outputs
- Browse and update the content of form fields
- Validate your template

- Perform calculations on fields within the template
- Connect to the Oracle BI Publisher catalog to retrieve data to build your template
- Upload your template to the Oracle BI Publisher server
- Extract boilerplate text into an XLIFF translation file and test translations

## Building and Uploading Your Template

You can build and upload your template via a direct connection with the BI Publisher server, or you can build and upload your template in disconnected mode.

### Working in Connected Mode

1. Open Microsoft Word.
2. From the Oracle BI Publisher menu, select **Log On**.
3. Enter your BI Publisher credentials and the URL for the BI Publisher server, for example: <http://www.example.com:7001/xmlpserver>. (Contact your system administrator if you do not know the URL.)
4. The Open Template dialog presents the same folder structure as the BI Publisher catalog. Select the report or data model for which you want to build a template.
5. **If you selected a data model:**

Click **Create Report** to create a report for this data model in the BI Publisher catalog. This is the report you will upload your template to.

Enter a Report Name and select the folder in which to save it.

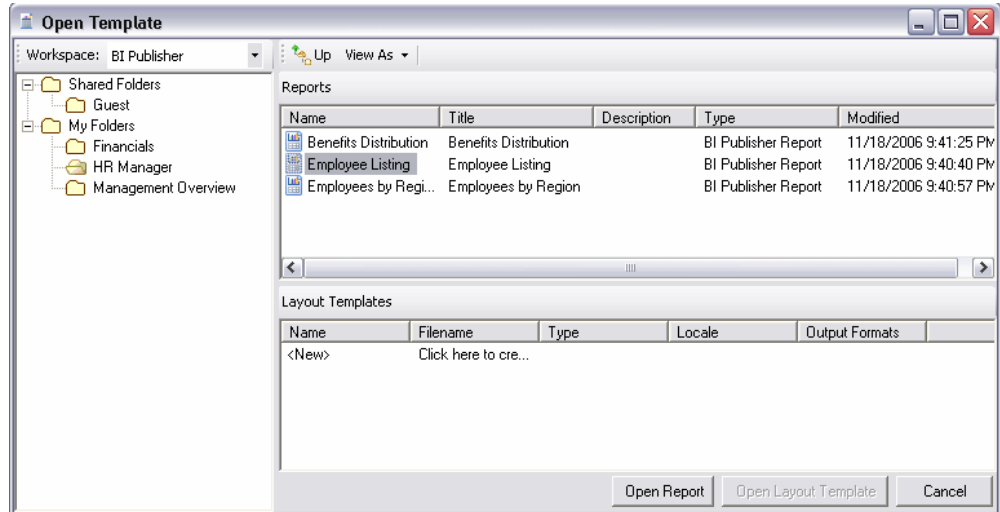
Click Save.

The sample data from the data model is loaded to the Template Builder.

#### **If you selected a report:**

Click **Open Report** to load the data to the Template Builder; or double-click <New> in the Layout Templates pane.

Note that any existing templates will be listed in the **Layout Templates** pane.



6. Follow the guidelines in this chapter to insert data fields and design your template using features such as tables, charts, and graphics. Use Microsoft Word to apply formatting to fonts and other objects in your template.

For more advanced template options, use the guidelines in *Creating RTF Templates*, page 4-2.

7. To upload your template file to the BI Publisher server and add it to your report definition, select **Upload Template As** from the Oracle BI Publisher menu.

If you have not saved your template, you will be prompted to save it in Rich Text Format.

8. Enter a name and select a locale in the **Upload as New** dialog. Note that this is the name that appears under **Layouts** in the Report Editor. This is also the layout name that will be displayed when a user runs this report.

9. Configure properties for this layout.

Navigate to the BI Publisher report editor to configure properties for this layout, such as output formats. See *Configuring Layouts*, page 2-7 for more information.

### Working in Disconnected Mode

To work in disconnected mode, you must have a sample data file available in your local work environment:

1. Save a sample data file to your local machine. See *Accessing Data for Building Your Template*, page 5-7.
2. Open Microsoft Word with the Template Builder installed.

3. On the Oracle BI Publisher menu in the **Load Data** group select **Sample XML**. Locate your sample data file in your local directory and click **Open**.

**Note:** The Template Builder also supports using XML Schema to design an RTF template. However, because the schema contains no data, the preview of your report will also contain no data.

4. Follow the guidelines in this chapter to insert data fields and design your template using features such as tables, charts, graphics, and other layout components. Use Microsoft Word to apply formatting to fonts and other objects in your template.

For more advanced template options, use the guidelines in *Creating RTF Template*, page 4-2s.

5. Upload your layout template file.

In the BI Publisher catalog, open your report in the Report Editor. Click **Add New Layout**.

Complete the fields in the dialog and then select **Upload**. The template will now appear as a layout for the report.

6. Configure properties for this layout.

See *Configuring Layouts*, page 2-7 for more information.

## Accessing Data for Building Your Template

The data model defines the XML format that will be merged with the RTF template. The Template Builder requires sample data to build your template. You must load sample data to use most of the template builder functionality.

If you are not connected to BI Publisher, then use the "Loading XML Data from a Local File" procedure. If you are connected, use the "Loading Data from the BI Publisher Catalog" procedure.

### Loading XML Data from a Local File

One method of loading data to the Template Builder is to save a sample of your report data to a local directory.

For information on saving sample data from your report data model, see the topic *Testing Data Models and Generating Sample Data*, *Oracle Fusion Middleware Data Modeling Guide for Oracle Business Intelligence Publisher*.

If you do not have access to the report data model, but you can access the report, you can alternatively save sample data from the report viewer. To save data from the report viewer:

1. In the BI Publisher catalog, navigate to the report.
2. Click **Open** to run the report in the report viewer.
3. Click the **Actions** icon, then click **Export**, then click **Data**. You will be prompted to save the XML file.
4. Save the file to a local directory.
5. Use the **Load Sample XML** feature below to load the saved XML file to the Template Builder.

The **Load Data** group from the Oracle BI Publisher menu enables you to select and load the saved XML file to the Template Builder.

- **Sample XML** - This function allows you to load a sample XML file that contains all fields that you want to insert into your template as a data source. If you are not connected to the BI Publisher server, use this method to load your data.
- **XML Schema** - This function allows you to load an XML Schema file (.xsd) that contains the fields available in your report XML data. The XML schema has the advantage of being complete (a sample XML file may not contain all the fields from the data source). For the preview, the Template Builder can generate dummy sample data for an XML Schema. However, the preview works better if you also upload real sample data.

## Loading Data from the BI Publisher Catalog

You can connect directly to the BI Publisher Server to load your BI Publisher report data to the Template Builder to use as sample data for designing layouts. You can also download an existing template to modify it.

To connect to BI Publisher and load a data source:

1. Log on to the BI Publisher Server: From the Oracle BI Publisher menu, select **Log On**. For more information on logging in to the BI Publisher server, see *Working in Connected Mode*, page 5-5.
2. After you are logged on, you can select **Open**. The Open Template dialog launches.
3. Navigate to the folder that contains the report or data model for which you want to create a template.

When you select a report, you can either select from the **Layout Templates** to open an existing template, select **Open Report** to load just the XML sample data to create a new layout, or double-click <New> to load the data to the Template Builder to build a new layout.



When you select a data model, you will be prompted to create a report in the catalog.

## Inserting Components to the Template

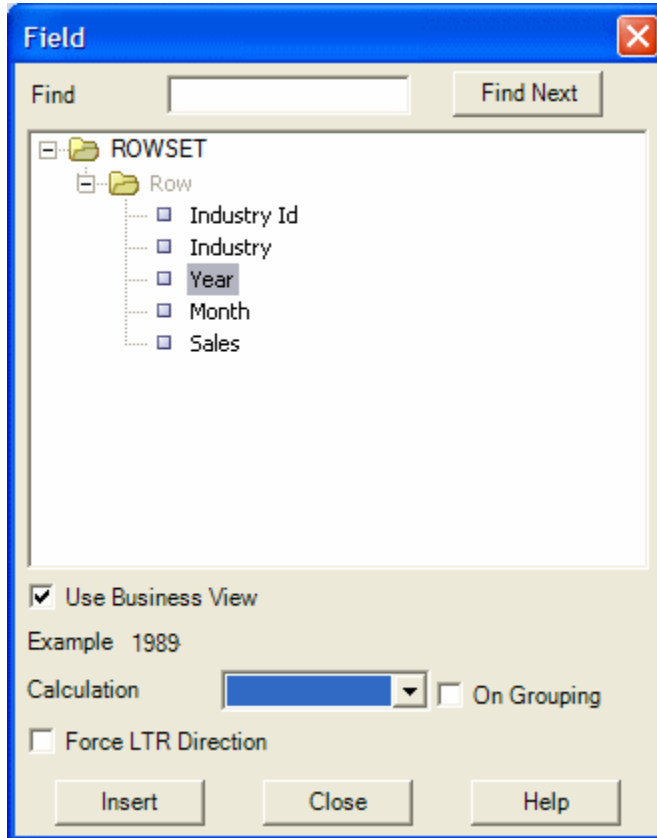
This section includes the following topics:

- Inserting a Field, page 5-9
- Inserting a Table Using the Table Wizard, page 5-12
- Inserting a Table or Form Using the Table/Form Dialog, page 5-16
- Inserting a Chart, page 5-20
- Inserting a Pivot Table, page 5-22
- Inserting a Repeating Group, page 5-25
- Inserting a Conditional Region, page 5-28
- Inserting a Conditional Format, page 5-29

### Inserting a Field

This dialog enables you to select data elements from the data source and insert them into the template.

In the Insert group select Field to open the Field dialog. The dialog shows the structure of your loaded data source in a tree view, as shown in the following figure:



Select a field that represents a single data field (a leaf node of the tree) and select Insert (you can also insert the field by dragging and dropping it into your document, or by double-clicking the field). A text form field with hidden BI Publisher commands is inserted at the cursor position in the template. You may either select and insert additional data fields or close the dialog by clicking the Close button.

### About the Insert Field Dialog

The Insert Field dialog fields are:

#### Find

For an XML document with a large and complicated structure, use the find functionality to find a specific field. Enter a partial string of the field name you are searching into the **Find** field and click **Find Next**. The next occurrence of a data element that includes your search expression will be selected. Click the **Find Next** button again to see the next occurrence.

#### Example

When you select a field name in the tree view, an example value for this field is shown.

## Force LTR (Left-to-Right) Direction

This check box is only needed if you are using the template in a language that prints the characters from right to left, such as Arabic or Hebrew. Use this feature to force left-to-right printing for fields such as phone numbers, addresses, postal codes, or bank account numbers.

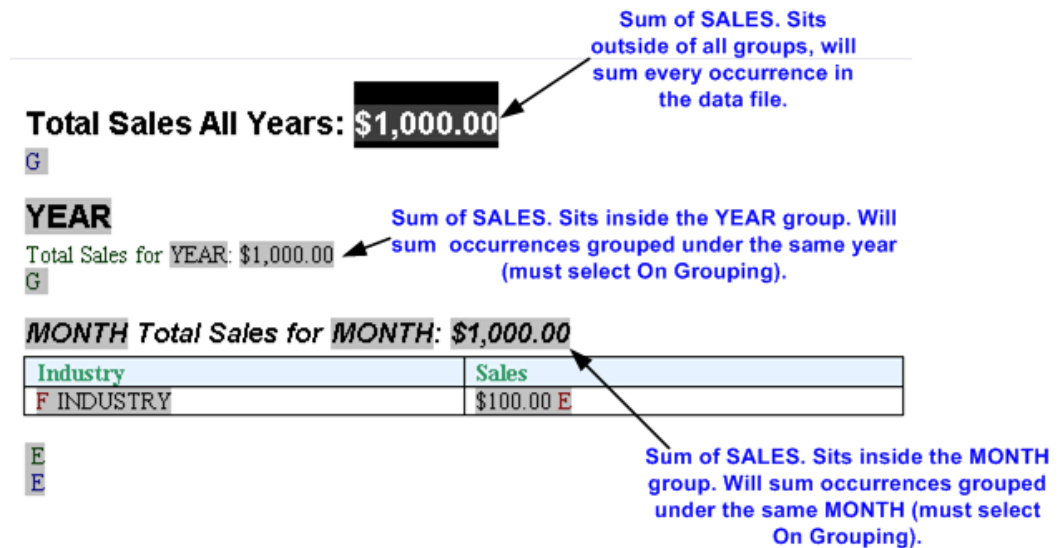
### Calculation

This feature enables you to perform aggregation functions on data fields, such as sum, average, count, minimum, and maximum.

For example, if you select sum for a data field, the field will show the sum of all occurring values for this data field, depending on the grouping.

It is important to understand the grouping context (marked by G and E form fields) to know exactly which fields are accumulated. If you insert a data field with an accumulation function into a repeating section (marked by G and E processing instruction form fields), you must select **On Grouping** to accumulate the data for the occurrences within the group. If you do not want the accumulation to be restricted to the group, you must place the accumulation field outside the group.

The following figure shows an example:



Also note that the data field must be a valid XSL number for the accumulation functions to work. Formatted numbers cannot be processed by BI Publisher (for example a number using a thousands separator: 10,000,000.00 cannot be processed).

For more information on groups in your template using the Template Builder, see Inserting a Repeating Group, page 5-25. Also see Defining Groups, page 4-12 in the chapter: Creating RTF Templates.

## Inserting a Table Using the Table Wizard

The **Insert Table Wizard** enables you to create standard reports. On the **Insert** menu select **Table Wizard**.

### Step 1: Select Report Format

Start by selecting the basic report format. Choose from **Table**, **Form**, or **Free Form**. The following example shows how the each selection will appear in the report. The following figure shows examples of each:

Table	<table border="1"><thead><tr><th>Last Name</th><th>First Name</th><th>Phone</th></tr></thead><tbody><tr><td>Doe</td><td>John</td><td>(123) 456-7890</td></tr></tbody></table>	Last Name	First Name	Phone	Doe	John	(123) 456-7890
Last Name	First Name	Phone					
Doe	John	(123) 456-7890					
Form	<table border="1"><tbody><tr><td>Last Name</td><td>John</td></tr><tr><td>First</td><td>Doe</td></tr><tr><td>Phone</td><td>(123) 456-7890</td></tr></tbody></table>	Last Name	John	First	Doe	Phone	(123) 456-7890
Last Name	John						
First	Doe						
Phone	(123) 456-7890						
Freeform	John Doe (123) 456-7890						

### Step 2: Select Table Data

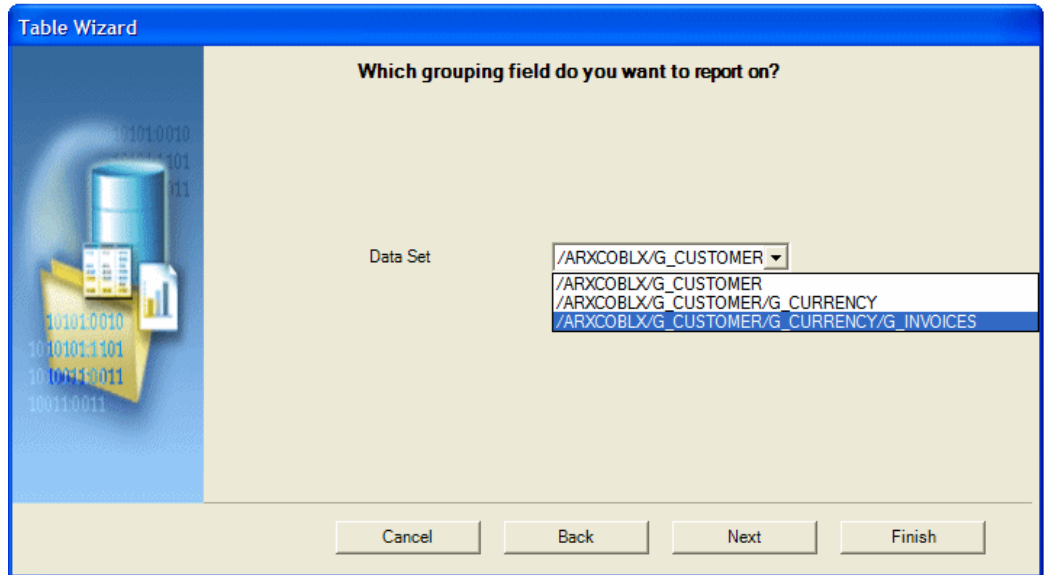
An XML document can include multiple grouped data sets. For example, a purchase order XML document may contain header level information, lines, shipments and contacts.

In this step, select the data group that contains the data required for your table.

For example, in the Balance Letter sample RTF template (found in the Template Builder installed files under Oracle\BI Publisher\BI Publisher Desktop\samples\RTF Templates), the sample XML file contains three data groups as follows:

- ARXCOBLX/G\_CUSTOMER
- ARXCOBLX/G\_CUSTOMER/G\_CURRENCY
- ARXCOBLX/G\_CUSTOMER/G\_CURRENCY/G\_INVOICES

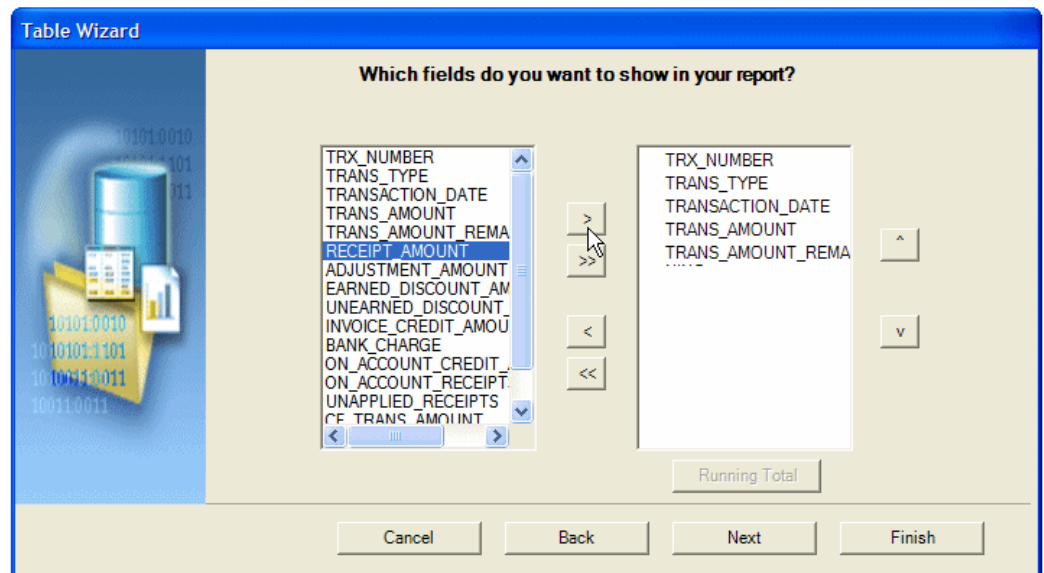
The Table Wizard presents a list of the available data groups in your XML data file. Select the group that contains the data fields for your table.



To build a table to list the invoices contained in the data, select:  
 ARXCOBLX/G\_CUSTOMER/G\_CURRENCY/G\_INVOICES  
 as your data set.

### Step 3: Select Data Fields

The Table Wizard presents the data fields from your selected data set.



Use the shuttle buttons to select the data fields to show in your table. Use the up and down arrows to reorder the fields after selecting them.

## Step 4: Group the Table

This step enables you to regroup the data by a particular field. This is optional.

For example, if you are building a table of invoices, you may want to group all invoices of a particular type or date to be grouped together in the report.

There are two options for grouping: Group Left or Group Above. Group Left will create a nested table. The Group By field will display to the left in the outer table. Group Above will create a new table for each new value of your group by field, displaying the value of the group by field as a table title.

Examples follow:

**Group Left** groups the group by element occurrences together as shown:

Date	Invoice	Amount	Amount Remaining
02-DEC-07	234	53.35	53.35
04-DEC-07	10020402	146776.07	146776.07
	10020403	172482.45	172482.45
	10020404	147740.25	147740.25
	10020405	71577.42	71577.42
	10020406	89344.81	89344.81
	10020407	223563.03	223563.03
	10020408	176353.55	176353.55
05-DEC-07	10020487	112902.54	112902.54
06-DEC-07	502444	19125	19125
	502445	12375	12375

**Group Above** shows the result as a table with a header:

## Grouping Field 1

Header 1	Header 2	Header 3	Header 4	Header 5
Data 1	Data 2	Data 3	Date 4	Data 5

Example of Group Above:

### 02-DEC-03

Invoice	Amount	Amount Remaining
234	53.35	53.35

### 04-DEC-03

Invoice	Amount	Amount Remaining
10020402	146776.07	146776.07
10020403	172482.45	172482.45
10020404	147740.25	147740.25
10020405	71577.42	71577.42
10020406	89344.81	89344.81
10020407	223563.03	223563.03
10020408	176353.55	176353.55

When you select an element to group by, BI Publisher sorts the data by the grouping element. If the data is already sorted by the grouping element, select the **Data already sorted** check box. This will improve performance.

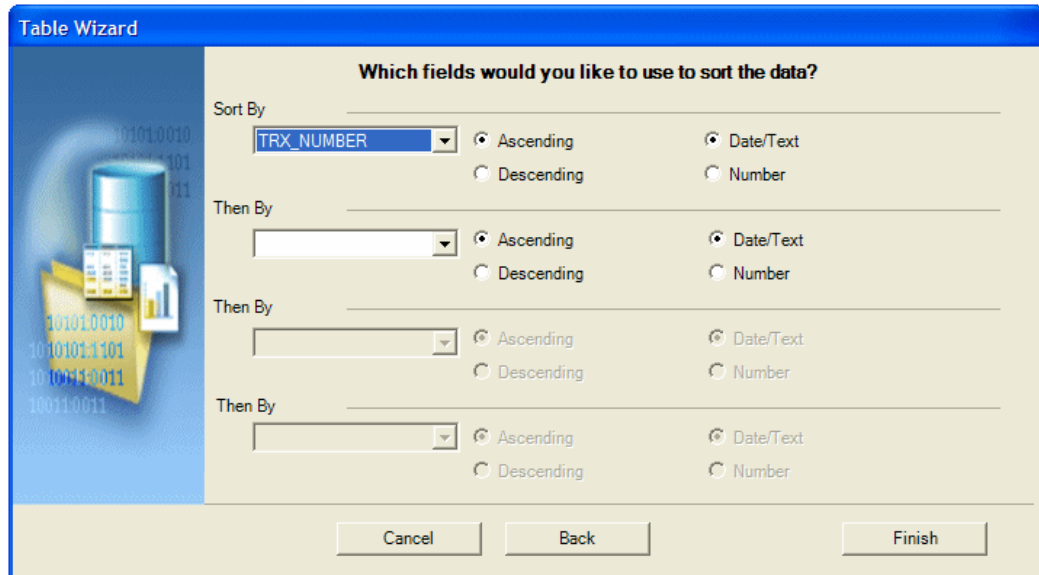
### Insert a Break for the Group

Use the **Break** option to insert either a **Page** break or **Section** break after each occurrence of this group. Note that a **Section** break can only be created on the top-level group. The subsequent grouping options only display the **Page** break option.

Note that a page break will start the next group on a new page; a section break will start the next group on a new page, reset page numbering, reset headers and footers, and reset any running calculations for each occurrence of the group.

### Step 5: Sort the Table

You can sort the data in the table by up to four different fields. Select a field and then define the sorting order (ascending or descending), and select the correct data type for the field. For example, if text is selected, "12" will come before "2" (alphanumerical order). If number is selected, "2" will come before "12".



### Step 6: Click Finish

Click **Finish** to create the table and insert it to your Microsoft Word document.

### Step 7: Customize the Table Using Microsoft Word Functionality

Customize the table by changing fonts, colors, column sizing, borders, shading, and so on, using Microsoft Word formatting commands.

## Inserting a Table or Form Using the Insert Table/Form Dialog

The Insert Table/Form dialog is the most flexible tool of the template builder. It allows you to perform the following tasks:

- Create a simple or nested table with a variable number of rows.
- Associate a group of data elements, such as a complete invoice or a purchase order line, with a form in the document that will be repeated for each occurrence of the data element.
- Select and define a layout for all the data fields in the template.
- Group or re-group the data.

The Insert Table/Form dialog shows you two tree view panes. The left pane shows the data source structure, while the right pane shows the elements that will be copied to the template when you click the Insert button.



## Selecting Data Fields

First select the data fields to insert in the template and then define how to format them. Drag an XML element from the left Data Source pane to the right Template pane. If the XML element has children, you will see a pop-up menu with the following options:

- Drop Single Node
- Drop All Nodes
- Cancel

Select **Drop Single Node** if you want to move only the selected node or **Drop All Nodes** if you want to move the node and all its children.

If you drag an additional data field from the left Data Source pane to the right Template pane, it is either inserted at the same level (Same Level) or below the node (Child) where you release the node. The Insert Position box defines where the node is inserted.

**Note:** If you use the left mouse button for drag and drop, the node and all children are copied. However, if you use the right mouse button for dragging, a dialog pops up when you release the mouse button. The dialog gives you the option to copy either only the selected node or the selected node and all children.

## Defining the Layout

When you select an element in the right Template pane, you will see its properties as well as a preview of how the node will be rendered. There are two kinds of nodes:

- Data Fields
- Data Groups

Data Field nodes (leaf nodes) do not have any child nodes. They represent simple attributes such as the total amount for an invoice or the subtotal for a purchase order line.

Data Group nodes (parent nodes) are nodes that do have child nodes. Typically, they do not represent data attributes, but groups of data – such as an invoice, a purchase order, a purchase order line or a shipment.

## Data Field Properties

If a Data Field node is selected, its properties are shown in the Properties pane. You have the following options to describe how the Template Builder should show the field:

- **Calculation**

You can select one of the aggregation functions for the data fields. These functions (besides count) only have an effect when there is more than one of the data fields in the context where you use the function.

- **Force LTR (Left-to-Right) Direction**

This option is only needed if you are using the template in a language that displays characters from right to left, such as Arabic or Hebrew. Use this option to force left-to-right printing for fields such as phone numbers, addresses, postal codes, or bank account numbers.

## Data Group Properties

The order in which the data elements are shown reflects the order of the columns in the table. If you want to reorder the columns, change the Insert Position box from Child to Same Level. Then drag the elements into the correct order.

If a Data Group node is selected, its properties are shown in the Properties pane. You have the following options to describe how the Template Builder should render the group:

- **Style**

To display the data as a horizontal table with a header, select Table. To display the fields below each other with labels in a table, use Form. If you want to insert the fields into a free-form text section that should be repeated for this element select Free Form.

- **Grouping**

Grouping is an advanced operation that allows you to group the data by a specific element in the data. For example, you may want to group all your invoices by customer. You can select a child element of the selected element as a grouping criterion. For more information see Grouping, page 5-19.

- **Show Grouping Value**

This property will only be shown if you have selected a node created by the Grouping functionality. By default, the field you have selected to group the data by will be displayed in the report. If you do not want the grouping data field displayed, select No.

- **Sort By**

Select an element by which the data groups are sorted.

- **Sort Order**

If you have selected an element for Sort By you can select if the data should be sorted either ascending or descending.

- **Sort Data Type**

If you have selected an element for Sort By the data is by default sorted as text. That means that 12 will be shown after 111. If the data is numeric, select Number as the sort data type.

- **Break**

This property allows you to insert a page break or a section break between every data group. If you select New Page per Element, then a page break will be inserted between each element after the first occurrence.

**Tip:** If you wish to insert a page break before the first occurrence of an element, use Microsoft Word's page break command.

If you select New Section per Element, a section break will be created for each data group. A section break has the following effects: it inserts page break, it resets the page numbers and new data can be displayed in the header and footer. You will typically use this option if you want to print multiple documents (for example invoices or purchase orders) to a single PDF file.

## Inserting Tables and Forms

Once you have dragged all data fields over and defined the layout, select the Insert button to place the tables and forms at the cursor position in your document.

## Grouping

You can group any Data Group node, by any of its child Data Field Nodes. For example if you have sales data for multiple quarters, you may want to show the sales data organized by quarter. In this case you would group the sales data rows by the quarter element.

Assume the following structure:

```
Sales Transaction
  Quarter
  Customer
  Amount
```

To group the child nodes of a node (Sales Transaction), you select one of the child nodes (Quarter) as the grouping property of the parent node (Sales Transaction). The Template Builder will make this node (e.g. quarter) the parent of the other child nodes (Customer and Amount).

The new structure will look like:

```
Sales Transaction
  Quarter
    Customer
    Amount
```

The grouping criterion (Quarter) now behaves like any other Data Group Node with

children. That means that you can define the layout of its children using the Create As Table, Style, Label, Grouping, and Show Grouping Value properties.

### Understanding Fields Inserted to the Template

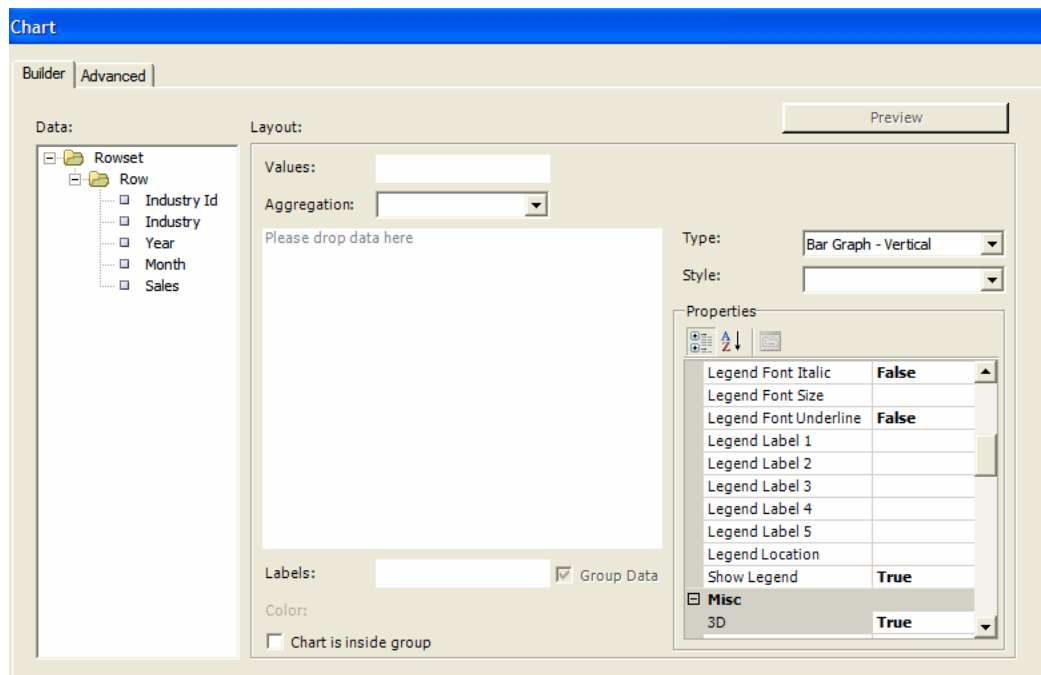
The Insert Table/Form Dialog creates two kinds of form fields:

- Form fields representing data elements
- Form fields with processing instructions for repeating table rows or document sections

Form fields representing data elements are replaced with the data when the template is processed. Form fields indicating repeating sections are shown as **for-each** and **end for-each** in the document. (Note: If you have selected the Abbreviated form field display option, the for-each and end for-each form fields will display as F and E.) The section of the document encapsulated by these two elements is repeated, if the associated data element is repeated in the data.

### Inserting a Chart

Use the Chart dialog to insert a chart into a template.



### Chart Type

BI Publisher supports a large variety of chart types. Expand the **Type** list to select the chart type for this template.

## Values

Drag and drop the data value you want to measure to the Values field (for example, SALES). You can select multiple Value elements (measures).

Note that the Values field will change depending on the Chart Type you select:

- Combination Graph- enables three fields for the Value selections.
- Scatter Graph - a scatter graph compares pairs of values. Drag and drop the X and Y data elements to compare.
- Bubble Graph - a bubble graph compares sets of three values. Similar to the scatter chart, the third value is displayed as the size of the bubble.
- Stock Graph - drag and drop the elements that represent the Open, High, Low, Close, and Volume values for the stock graph.

## Aggregation

You can choose to aggregate the Values data as a sum, a count, or an average.

## Labels

Drag and drop the data element for which you want to see the Value charted (for example, Year). Select Group Data to group the occurrences of the label element before rendering it in the chart. For example, if you are charting Sales by Year, selecting Group Data will accumulate the values for Year, so that only one occurrence of each year will appear in the chart. If you do not select Group Data, then the value for every occurrence of Year in the data will be plotted separately.

## Color

If you wish to add a series element to the chart, drag and drop the element to display as a series. Each value will display as a new color in the graph.

## Chart is Inside Group

Select this box if your chart is inside a grouping and you want the chart to display data only for the occurrences of the data elements within the group.

## Style

Select a color scheme and style for your chart.

## Properties

The properties region enables you to change value and label display names, select color, font, and other display options for your chart. The properties list will change depending

on your chart selection.

## Preview

Click Preview to display the chart with the sample data.

## Group Data

By default the data will be grouped by the Value element and aggregated by sum.

If you deselect the Group Data check box, each occurrence of the value element will be charted and aggregation functions will not be available.

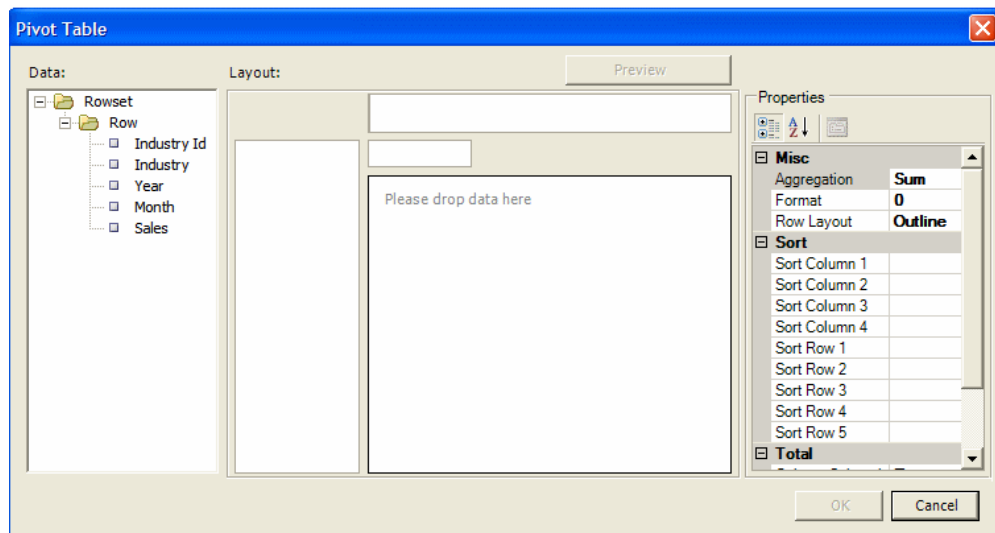
## Editing an Inserted Chart

To edit a chart that you have already inserted into your template, right-click the chart and select BI Publisher Chart from the menu. This will invoke the chart dialog to enable you to edit your chart.

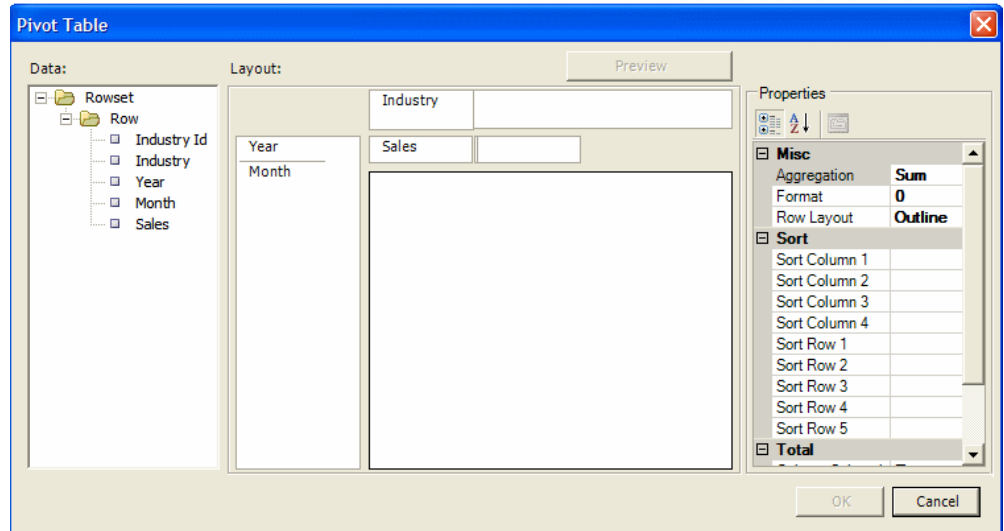
## Inserting a Pivot Table

To insert a pivot table

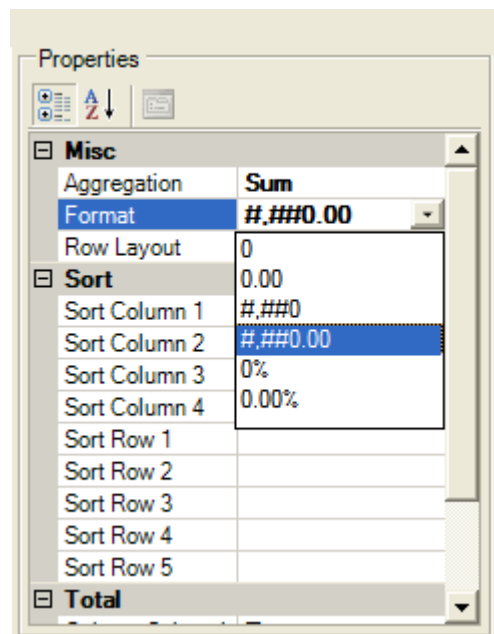
1. On the BI Publisher menu on the **Insert** group, click **Pivot Table**. The **Pivot Table** dialog presents your data in the left pane with empty **Layout** panes on the right for you to drag and drop data elements. The following figure shows the Pivot Table dialog:



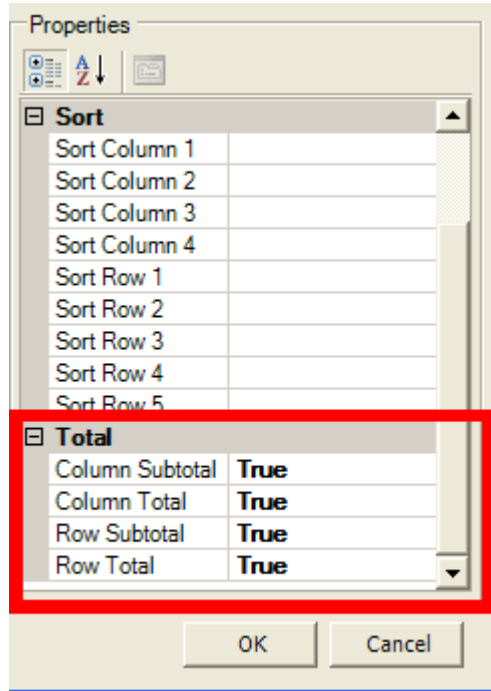
2. Drag and drop the elements from the Data pane to the Layout pane to build your pivot table structure. The following example Layout shows Sales by Industry accumulated by Year and by Month:



3. Use the Properties pane to select Aggregation. You can choose Sum, Count, or Average. Then choose a number **Format**.



4. By default subtotals for rows and columns are displayed. You can choose not to display the subtotals by setting the properties to False.



5. Click **Preview** to see how the pivot table will appear before you insert it into your template. Click **OK** to insert the pivot table into your template. The pivot table will appear as follows in your template:

CH		G	Total
		INDUSTRYE	
G YEAR		G 999.00E	999.00
B	G MONTH	G 999.00E	999.00E
		G 999.00E	999.00

At runtime, this pivot table will generate as follows:

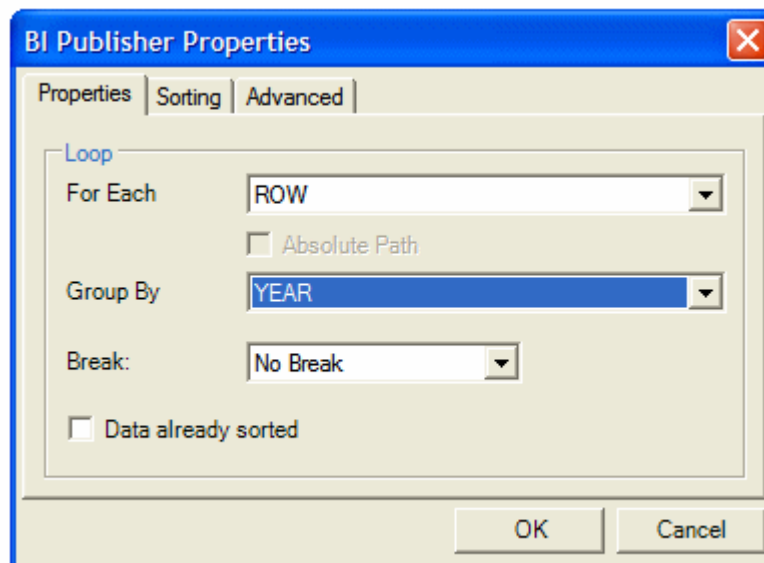


		Building material & garden eq. & supplies dealers	Clothing & clothing accessories stores	Electronics & appliance stores	Food services & drinking places	Food & beverage stores	Furniture & home furnishings stores
2006		48,531.00	31,528.00	15,472.00	61,967.00	81,303.00	17,073.00
	February	23,146.00	15,611.00	7,710.00	30,803.00	40,449.00	8,418.00
	March	25,385.00	15,917.00	7,762.00	31,164.00	40,854.00	8,655.00
2007		79,625.00	48,991.00	24,726.00	99,088.00	127,991.00	26,468.00
	February	26,369.00	16,497.00	8,320.00	33,239.00	42,723.00	8,859.00
	January	26,480.00	16,316.00	8,114.00	32,852.00	42,553.00	8,805.00
	March	26,776.00	16,178.00	8,292.00	32,997.00	42,715.00	8,804.00
		128,156.00	80,519.00	40,198.00	161,055.00	209,294.00	43,541.00

## Inserting a Repeating Group

To insert a repeating group:

1. Select the section of the template that contains the elements you want repeated.
2. On the Oracle BI Publisher menu, in the Insert group, click Repeating Group.
3. Enter the appropriate fields in the BI Publisher Properties dialog:



### For Each

Select the element that for each occurrence, you want the loop to repeat. When you select the For Each data field you are telling BI Publisher that for each occurrence of the selected field in the data you want the elements and processing instructions contained within the loop to be repeated.

For example, assume your data contains invoice data for customers and you want

to create a table with each customer's invoices. In this case, for each customer number you want the table to repeat. You would therefore select the customer number in the For Each field to create a new loop (or group) for each customer.

Note the following about creating repeating groups:

- For loops and groupings not inside another group (that is, outer groups or loops) you must select the repeating XML element to be used. For example if the data set is flat, the only repeatable element is /DATA/ROWSET/ROW. In cases with multiple data sources or hierarchical XML you can choose the data set.
- If you are creating nested groups (inserting a loop or group inside of another loop in the template), the For Each field is not updateable because it is already defined by the preexisting outer loop. The For Each field will display "Group Item" to inform you that an outer group is already defined.

### **Absolute Path**

Select this check box to use the Absolute Path to the element in the XML structure. This is important if your data contains the same element name grouped under different parent elements.

### **Group By**

Select a field from the list by which you want to group the data. If you just want to create a simple loop, do not select a group by element. Selecting a group by element will actually regroup the data into a new hierarchy based on the group by element.

### **Break**

Use this option to create either a Page break or Section break if you wish to insert a break after each occurrence of this group.

Note that a Section break can only be created on outer groups that surround the whole document. If the selected field is not an outer group, the Section break option will not be available.

Note also that when you insert a section break, the page numbering is reset, headers and footers are reset, and any running calculations will be reset for each occurrence of the group.

4. To sort the grouped data, select the Sorting tab. You can select up to four sort-by fields. For each sort by field, select the following:

**Sort order** - select Ascending or Descending.

**Data Type** - Select Number or Date/Text. It is important that you select the correct data type to achieve the expected sort order.

Note that if you are sorting by four criteria and your XML data element names are long, you may exceed the character length limitation (393 characters) of the Microsoft Word form field.

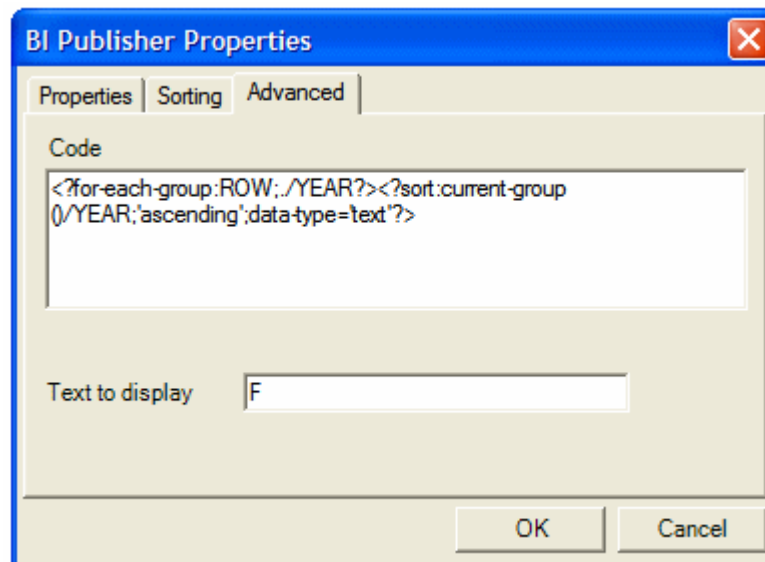
5. The Advanced tab enables you to edit the code directly and to enter Text to Display for the field.

The Code region displays the code and processing instructions that the Template Builder has inserted for the field. You can edit this if you wish to change the processing instructions for this field.

The Text to Display field shows how this field will display in your template. You can choose to enter descriptive text to enable you to understand each field better when reading the template, or you can enter abbreviated text entries that will be less intrusive to the look and feel of the template.

Note: The default display text can be set as Descriptive or Abbreviated using the Options tab.

The following figure shows the Advanced tab of the Repeating Group BI Publisher Properties dialog:



6. When you have completed the dialog options, click OK. This will insert the form fields in your template. By default, the beginning for-each form field will display the text "F" and will be inserted at the beginning of the selected template section. At the end of the selection, an "E" form field will be inserted to denote the end of the repeating group.

## Creating Grouping Fields Around an Existing Block

To create a group around an existing block of text or elements in a template:

1. Select the block of text. For example, a table row.

Note that if any preexisting BI Publisher tags are included in the block, you must be

sure to include the beginning and ending tags. For example, if your block contains any opening for-each, if, or for-each-group tags, you must include the end for-each, end-if, and end for-each-group tags in your selection.

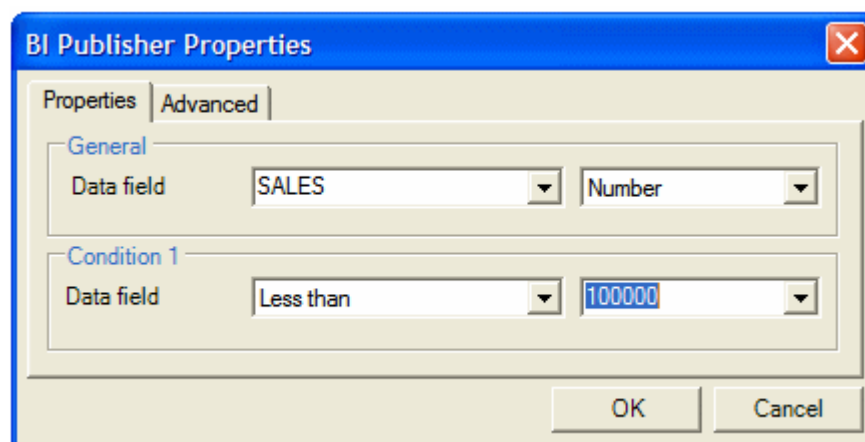
2. On the Oracle BI Publisher menu, on the Insert group, click Repeating Group.
3. In the BI Publisher Properties dialog, enter the fields to define the group as described in Inserting a Repeating Group, page 5-25.
4. Click OK to insert the grouping fields around the block. For example, if the block is a table row, the begin field will be inserted at the beginning of the first cell and the end field will put at the end of the last field.

## Inserting and Editing Conditional Regions

A conditional region is an area that is surrounded by a conditional statement. If the statement tests true, the area is displayed in the report; if the condition tests false, the area is suppressed from the report.

For example, your data contains sales information. Your report contains a table that displays sales by industry. You want this table in your report to display information for industries with sales amounts lower than 100,000. Using the insert conditional region functionality, you can select the region that contains the sales table and insert the condition that the sales element must be less than 100,000.

1. Select the region that you want to apply the condition to. For example, if you want to display a table only for a certain condition, select the region that contains the table. Note that the region must be inside a loop.
2. On the Oracle BI Publisher menu, on the Insert group, click Conditional Region. The following figure shows the BI Publisher dialog for a Conditional Region:



3. Enter the following fields:

**Data Field** — Select the field to test for the condition. Select the data type of the field: Number or Date/Text.

**(Condition 1) Data field** — Select the comparison operator.

Select the value to meet the condition. Note that you may enter an integer, enter text, or select another data element to define a comparison based on the incoming values.

4. Click OK. The form fields containing the conditional logic are inserted around the region. The beginning form field will display the text "C" and the form field closing the region will display the text "EC".

To edit the conditional region, double-click the inserted form field to launch the dialog for editing; or, right-click the form field and select BI Publisher, then Properties.

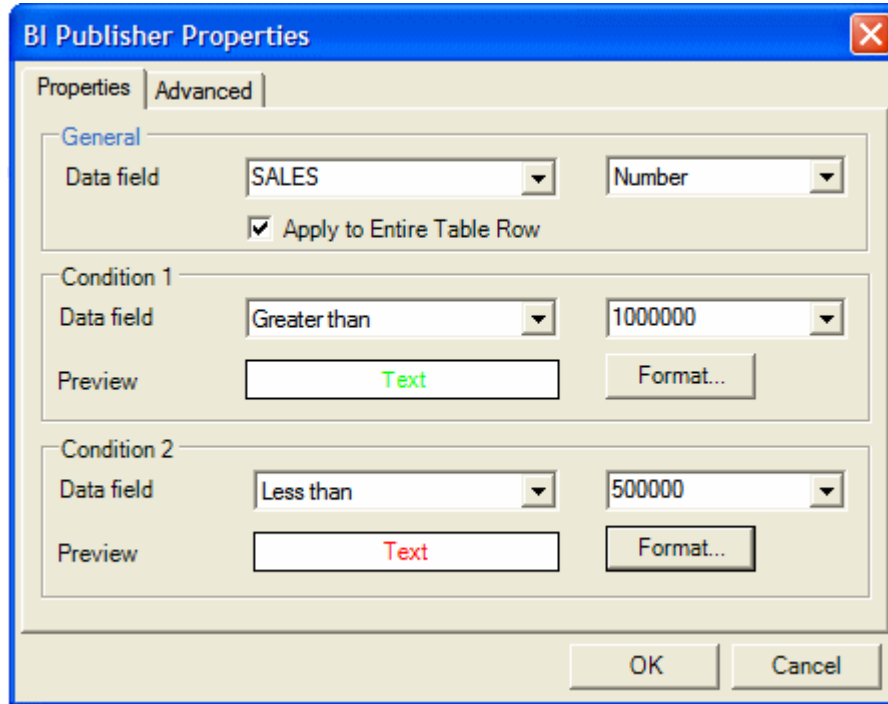
## Inserting Conditional Formatting

Using the Conditional Format feature you can insert simple conditional formats to apply to table rows or cells. The dialog provides several common options that you can select and the Template Builder inserts the code automatically. The Conditional Format dialog supports two conditions per field.

**Important:** The Conditional Format dialog cannot be used inside of pivot tables. You must insert the conditional formatting logic directly to the appropriate form fields.

To insert a conditional format:

1. Place the cursor in the table cell of the data element for which you want to define the condition.
2. On the Oracle BI Publisher menu, on the Insert group, click Conditional Format. The following figure shows the BI Publisher Properties dialog for a Conditional Format:



3. Enter the following in the Conditional Format dialog

**Data Field** — Select the element to test for the condition and the data type of the element (Number or Date/Text).

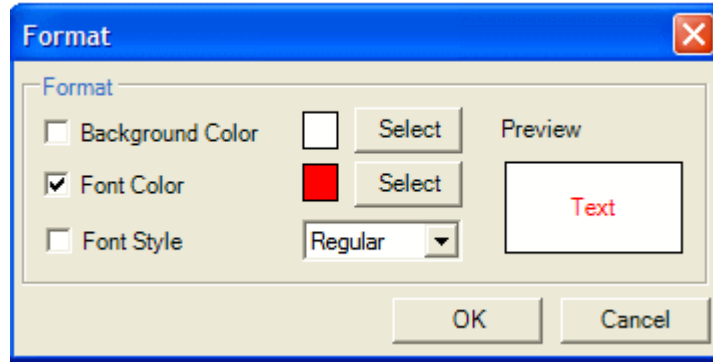
**Apply to Entire Table Row** — If you want the format applied to the entire table row, not just the cell of the selected element, select this box.

**Condition 1) Data field** — Select the comparison operator.

Select the value to meet the condition. Note that you may enter an integer, enter text, or select another data element to define a comparison based on the incoming values.

4. Click **Format** to define the format you want to apply when the condition is met. Options are background color, font color, and font style (regular, bold, italic, bold italic). Select the box and format of each option you want to apply. After you select the format, the Preview region will display the format chosen.

The following figure shows the Format dialog:



5. Define a second condition if desired.
6. Select OK. The conditional format field will be inserted as a form field with the display text "C".

To edit the conditional format, double-click the inserted form field to launch the dialog for editing; or, right-click the form field and select BI Publisher, then Properties.

## Previewing a Template

The Preview menu group enables you to preview your RTF template with sample XML data.

**Note:** If you have not already done so, you must load sample data to the Template Builder to preview the report. See Accessing Data for Building Your Template, page 5-7.

From the Preview group select the output format. If you have not yet saved your template as an RTF file, you will be prompted to do so.

- **PDF**

You must have Adobe Acrobat Reader version 5.0 or higher installed to preview documents in PDF format.

- **HTML**

Launches your default browser to display the report.

- **EXCEL**

To use this option, you must have Microsoft Excel 2003 or later. If you have Excel 2003 this option generates the document in MHTML and opens the document in Excel. If you have Excel 2007, this option generates the document in .xlsx, the default Office Excel 2007 XML-based file format.

- **EXCEL 2000**  
Generates HTML and launches Microsoft Excel to render it. Embedded images such as charts and logos are not supported in this output type. If you do not have Microsoft Excel 2003 or later, use this option.
- **RTF**  
Generates the report in Rich Text Format.
- **PowerPoint**  
Requires Microsoft PowerPoint 2003 or 2007.

## Template Editing Tools

This section describes additional tools provided with the Template Builder to help you validate and edit your template. This section includes:

- Editing and Viewing Field Properties, page 5-32
- Validating a Template, page 5-34
- Using the Field Browser , page 5-35
- Checking Accessibility, page 5-36

## Editing and Viewing Field Properties

Once you have inserted a data field (see Inserting a Field, page 5-9) you can view or edit the field properties in the BI Publisher Properties dialog.

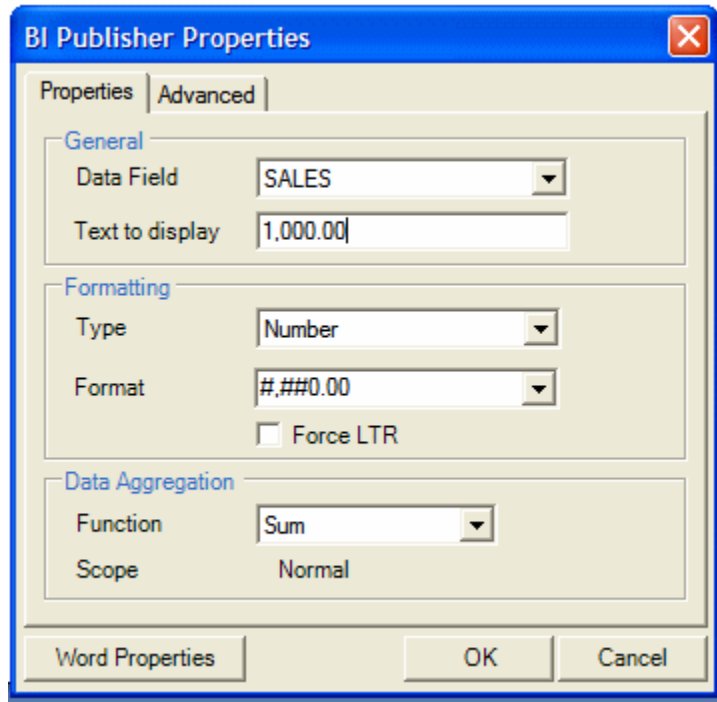
To invoke the BI Publisher Properties dialog, perform one of the following:

- Double-click the field
- Right-click the field, from the menu select BI Publisher, then Properties

The following figure shows the BI Publisher Properties dialog: note the Properties tab, the Advanced tab, and the Word Properties button:

**Note:** Some fields may only display the Advanced tab.





## About the Properties Tab

You can set the following properties for a data field:

**Data Field** — Select the data field from the list of available fields from the loaded data source.

**Text to Display** — Enter the display text for the form field in the template. This text will be replaced at runtime by the value in the data.

**Type** — Select the type of data. Options are Regular Text, Number, Date, Current Date, Current Time. The selection in this field will determine the format options.

**Format** — For any data type except Regular Text, you can select from several number or date display formatting masks or enter your own.

**Force LTR** — (Force Left-to-Right) Use this check box when you are publishing the template in a language that prints the characters from right to left, such as Arabic or Hebrew. Use this option to force left-to-right printing for fields such as phone numbers, addresses, postal codes, or bank account numbers.

**Function** — This feature enables you to perform aggregation functions (Sum, Average, Count, Minimum, Maximum) on data fields. For example, if you select sum for a data field, the field will show the sum of all occurring values for this data field depending on the scope (see below). See also Inserting a Field, page 5-9 for information on aggregation functions.

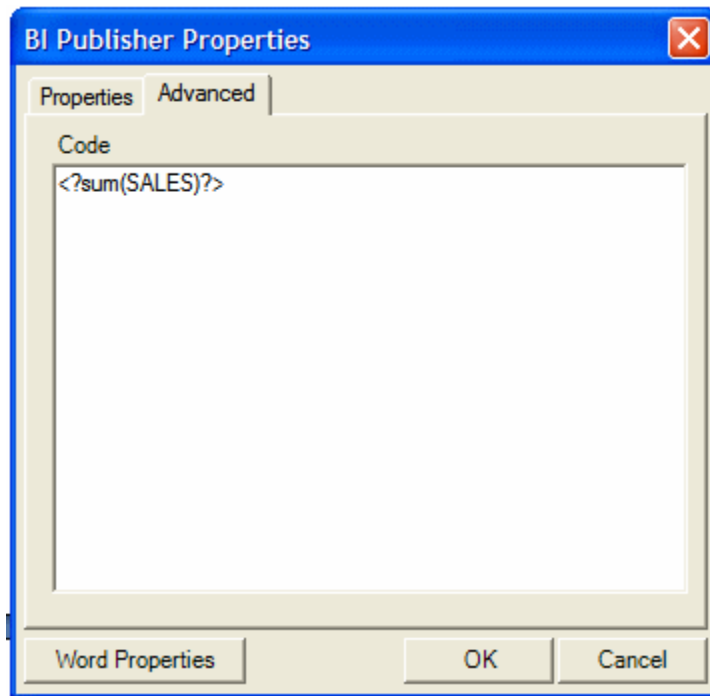
**Scope** (informational only) — This field has two possible values:

- **Group Item** — Indicates that the data field is inside a group. If you choose to perform a function on the field, only the occurrences of the field within the current group will be included in the aggregation.
- **Normal** — Indicates that the field is not inside a group. Aggregation functions will be performed on all occurrences of the field in the data.

### About the Advanced Tab

The Advanced tab displays the underlying code. If the code pattern within the form field is not recognized (for example, because you added commands manually to the field), the BI Publisher Properties dialog will display this tab only.

Use this tab to edit or add code to the form field manually. Select OK to update the template. The following figure shows the Advanced tab:



### About the Word Properties Button

The Word Properties button opens the Microsoft Word Text Form Field Options dialog. You can also use this dialog to set the data type and number format. The underlying code used by BI Publisher is also available by clicking the **Add Help Text** button.

### Validating a Template

The Template Builder provides a validation tool to check the template for incorrect use of BI Publisher commands and unsupported elements in the RTF file.

To validate your template:

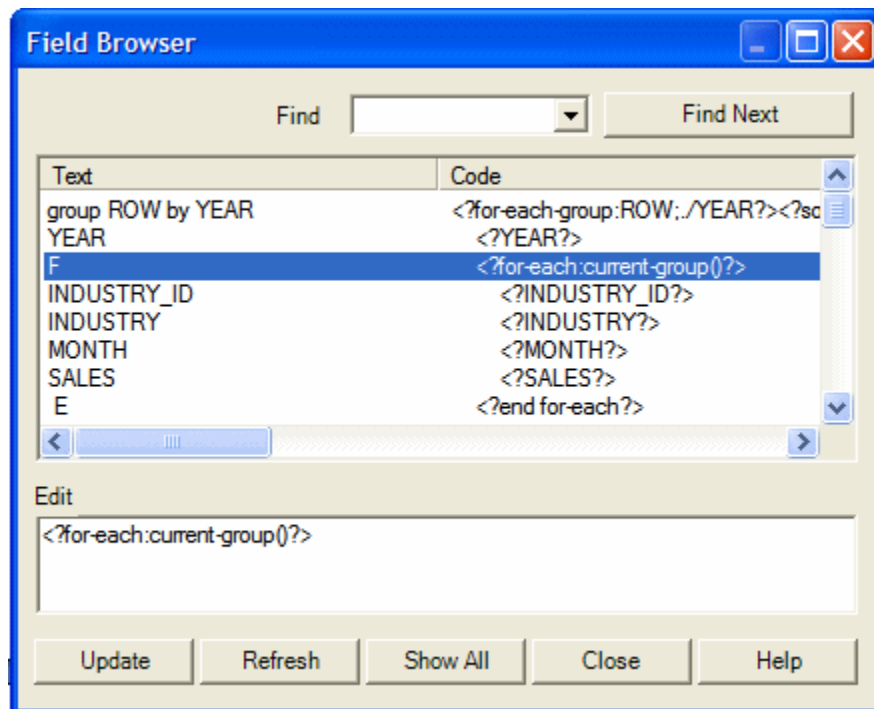
On the BI Publisher menu, on the Tools group, click Validate Template.

If there are no validation errors, No Error found will be returned. If an error is found, the error will be displayed. You can use the Field Browser to help locate the error.

## Using the Field Browser

The field browser dialog provides a fast way to review and update the BI Publisher instructions hidden in the Microsoft Word form fields. This dialog is particularly useful to understand and modify existing templates.

On the Tools group click Field Browser.



The Field Browser dialog shows a table with the display text of the form field in the Text column and the underlying code instructions in the second Code column. When you select a specific row in the dialog, the matching form field will be selected in the Microsoft Word document.

If you select some part of the text before opening the Field Browser, the dialog only shows the fields in your selection. If no text is selected, the field browser will show all fields in the document.

The options are described in the following table:

Option	Description
<b>Edit</b>	<p>You can update processing instructions directly from the Field Browser dialog.</p> <p>Select a field in the Text table. The Edit box shows the processing instructions for the field. To change the instructions for the field modify the text in the Edit field and click Update.</p>
<b>Refresh</b>	<p>The Field Browser dialog is not aware of any form fields that you have added or modified while the dialog is open. Click Refresh to show any changes to the document since the Field Browser dialog has been opened.</p>
<b>Show All</b>	<p>If you opened the browser with a part of the document selected you will only see the form fields in the selected area. Click Show All to see all the form fields in the document.</p>
<b>Close</b>	<p>Click Close to close the field property browser. The Close button will not automatically update any changes in the edit field, therefore ensure that you select Update if you want to save edits.</p>

## Checking Accessibility

The Template Builder provides an accessibility checker to check the template for features to enhance the accessibility of the report for report consumers who may need assistive technologies to view the report.

To check for the presence of accessibility features: On the **BI Publisher** tab, in the **Tools** group, click **Check Accessibility**. The tool will generate a report indicating areas of a template that do not include the following accessibility features:

- document title
- alternative text for images
- table summary for data tables
- column header for data tables
- row header for data tables

In some cases the accessibility checker will be unable to determine if the accessibility feature is present and will generate a warning. The report designer should then verify that the accessibility features are present.

For information on how to add these features to your template, see *Designing*

Accessible Reports, page C-1.

## Uploading a Template to the BI Publisher Server

If you used the Open Template dialog to connect to BI Publisher, and load your data to the Template Builder, or if you downloaded an existing template from the BI Publisher catalog, you can upload the new or updated layout back to the report definition on the server. See *Working in Connected Mode*, page 5-5.

If you downloaded an existing template and wish to upload the modifications to the template, select **Upload Template** from the Oracle BI Publisher menu.

If this is a new template for the report definition, use the **Upload Template As** option to upload the layout to the report definition on the server. Also use this option to upload modifications to an existing template under a different name.

## Using the Template Builder Translation Tools

The Template Builder provides tools to help you create and test translations for your templates.

### About Translations

There are two options for adding translated templates to your BI Publisher report definition:

- Create a separate RTF template that is translated (a localized template)
- Generate an XLIFF file from the original template (at runtime the original template is applied for the layout and the XLIFF file is applied for the translation)

Use the first option if the translated template requires a different layout from the original template.

If you only require translation of the text strings of the template layout, use the XLIFF option.

For detailed information on translation concepts and support, see the section *Translating Reports and Catalog Objects*, page 15-1.

To use the Template Builder translation tools to create your templates for translations, see the following topics in this section:

- [Extracting Text to an XLIFF File for Translation](#)
- [Previewing a Translation](#)
- [Localizing a Template](#)

For a demo on BI Publisher's localization capabilities, see the *LocalizationDemo.exe*

demo provided with your Template Builder installation (located in the BI Publisher\BI Publisher Desktop\demos folder where you installed BI Publisher Desktop).

## Extracting Text to an XLIFF File for Translation

This menu item allows you to create a standard XLIFF translation file containing the boilerplate text from your template. XLIFF is a standard file format that is understood by many translation software packages. Since an XLIFF is an XML file, you can translate the text in a regular text editor.

A "translatable string" is any text in the template that is intended for display in the published report, such as table headers and field labels. Text supplied at runtime from the data is not translatable, nor is any text that you supply in the Microsoft Word form fields.

1. From the BI Publisher menu, select Tools, then Translate Template, then Extract Text.
2. You will be prompted to save the extract file as an XML file type. Enter a name for the extract file and save to the desired location.
3. If you wish to translate the template yourself, open the .xlf file using a text editor and enter the translated strings in the file. For a detailed description of working with the BI Publisher generated .xlf files, see the topic: Working with Translation Files, page 15-2.
4. When done, you can Preview the translation. Then upload the file to the BI Publisher report definition.

## Previewing the Template and Translation File

To preview your template with your translated XLIFF file applied:

1. From the BI Publisher, in the Tools group, click Translation, then Preview Translation.
2. You will be prompted to select your saved XLIFF file. Locate the file, and select Open.

The Template Builder will merge the sample data, the translation file, and the RTF template to generate a PDF for you to preview.

## Localizing a Template

Localizing a template means that you are creating a template to be used for a specific language.

Because BI Publisher enables you to extract the boilerplate text strings from your

template into an XLIFF file that can be translated and then applied at runtime, if your reports for additional languages only require the translation of these text strings, then you only need to supply translated XLIFF files to accompany your base template.

However, you would localize a template when the requirements for the report in the specific language go beyond the simple translation of the text in the layout.

To save a template as a localized template:

1. From the Oracle BI Publisher menu, in the Tools group, select Translations, then Localize Template. This will invoke a warning that Localizing your template will overwrite the template. Click OK.
2. You will be prompted to select the XLIFF translation file. Locate the appropriate file and click Open.
3. The translated XLIFF file will be applied to the template you currently have open in Microsoft Word.
4. Save the localized template.
5. Upload the template file to the appropriate report definition in the BI Publisher catalog. Select the appropriate locale in the upload dialog.

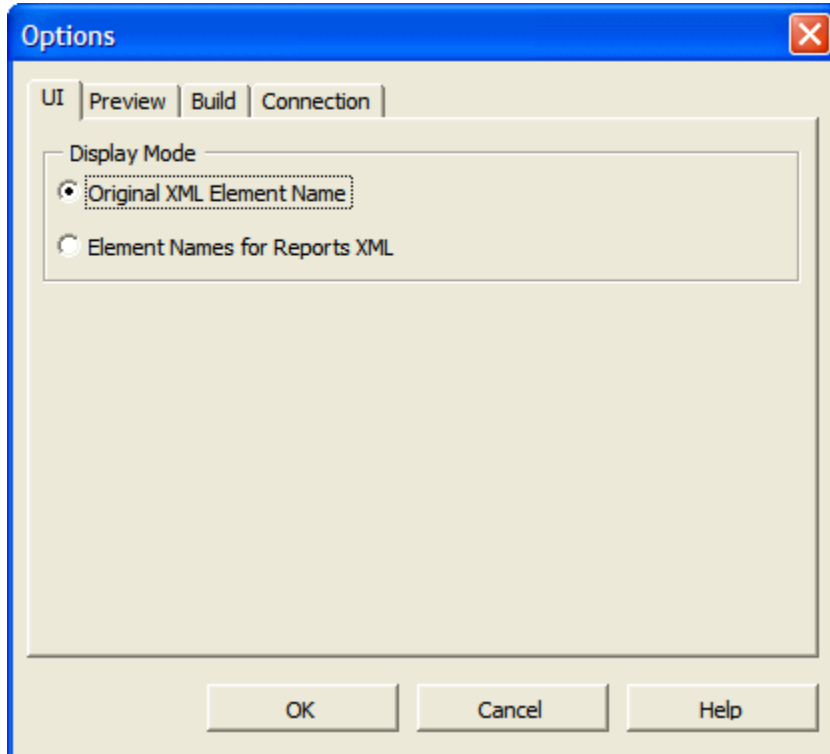
## Setting Options for the Template Builder

Access the **Options** dialog as follows: In the **Options** group, click **Options**.

The Options dialog contains four tabs: UI, Preview, Build, Connection.

### Setting UI Options

Use the UI Options tab to set options that influence the look and feel of the Template Builder:

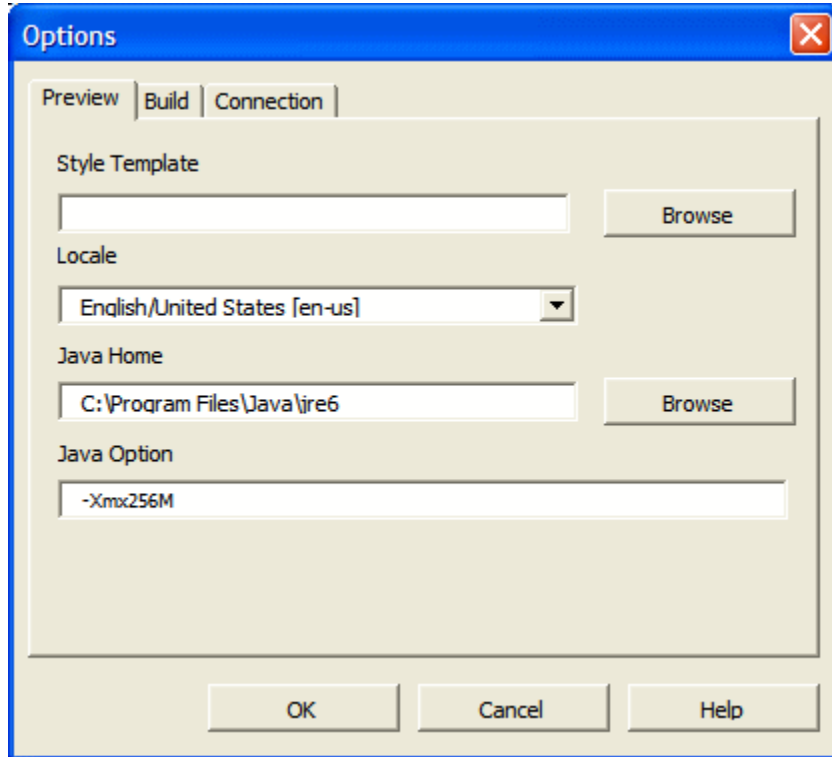


The tree view showing the data source can show either the correct XML tag names of the data source or they can show a slightly modified version that is easier to read. Select the option **Element Names for Report XML** to show the modified labels. These labels contain no < > characters, use "Title case" and use spaces (" ") instead of underscores ("\_").

## Setting Preview Options

The Preview Options tab allows you to specify options that influence the Preview functionality of the Template Builder.





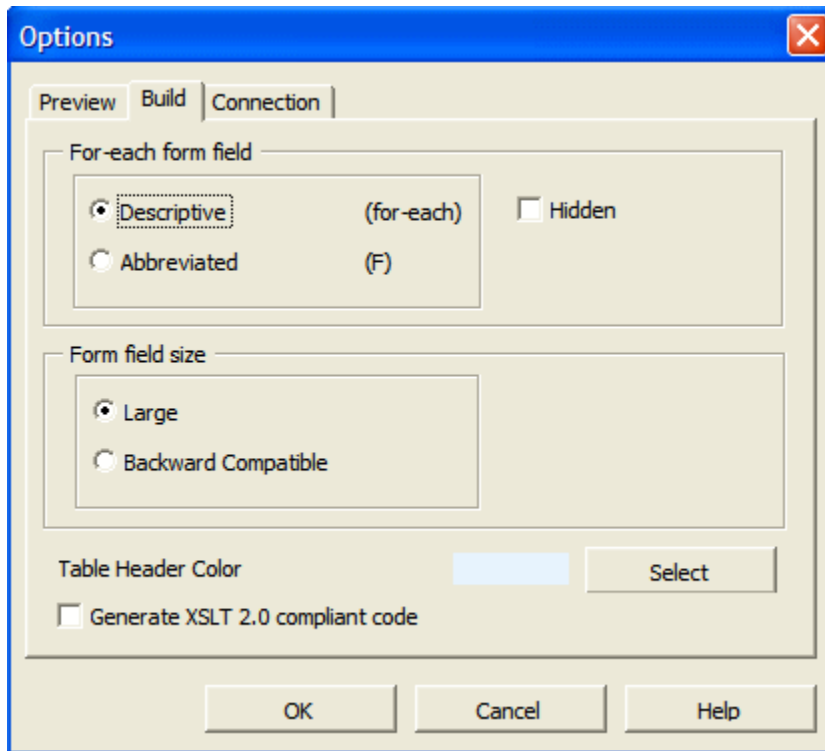
The following table describes the options available from the Preview tab:

Option	Description
<b>Style Template</b>	If you have a BI Publisher Style Template available locally you can specify it here. A style template is an RTF template that contains style information that can be applied to RTF layouts, similar to a style sheet. The style information in the style template is applied to RTF layouts at runtime to achieve a consistent look and feel across your enterprise reports. For more information, <i>Creating and Implementing Style Templates</i> , page 11-1.
<b>Locale</b>	You may choose the language and territory used for previewing your template. While this change will not automatically translate any files, it is important to set the correct locale for the preview to use the correct direction of the text (left-to-right or right-to-left), and to correctly set locale-specific date, number, and currency formats.
<b>Java Home</b>	The Preview (and export functionality) requires Java code. You can change the path to your JAVA HOME directory. If this option is not specified, the Template Builder assumes that the Java virtual machine (java.exe) is accessible in the PATH specified in your environment variables of Windows.

Option	Description
Java Option	Specify the memory to reserve for the Template Builder to process the template. The default value is -Xmx256M.

## Setting Build Options

Use the Build Options tab to specify options that influence how the Template Builder generates tables and forms.



The following table describes the options available from the Build tab:

Option	Description
<b>For-each form field</b>	<p>Choose how the Template Builder creates the form fields for processing instructions in the Insert Table/Form dialog.</p> <p>The <b>Descriptive</b> option (for example: <b>for-each Invoice</b>) renders a descriptive form field for the processing instructions. This option makes the layout template easier to understand. However, the longer fields may distract from the visual layout of the template. Note that the descriptive option does not apply to fields within table cells.</p> <p>The <b>Abbreviated</b> option (for example: <b>F</b>) provides a one letter abbreviation for each instruction.</p> <p>Select the <b>Hidden</b> box to generate the processing instruction form fields using Microsoft Word's hidden font effect. Hidden text is hidden in the Print Preview and you may display or hide the hidden text by changing the Hidden Text setting in the Display group of the Microsoft Word Options.</p>
<b>Form Field Size</b>	<p><b>Large</b> - inserts the BI Publisher code to a document variable. The document variable field can accommodate approximately 48 kilobytes of code line.</p> <p>It is important to note that this setting affects only fields that are created or edited while this option is set. The form fields created with the Large setting cannot be understood by Oracle BI Publisher 10g. If your template is intended for use with the 10g version of BI Publisher, use the Backward Compatibility setting.</p> <p><b>Backward Compatible</b> - in previous versions of the Template Builder the BI Publisher code was inserted to the Microsoft Word Form Field Help Text box. This limited the length of code that could be inserted for a single form field. By default, the Large option is used because it can accommodate much larger code strings. However, the Large option is not compatible with Oracle BI Publisher 10g.</p>
<b>Table Header Color</b>	<p>When you insert a table using the Table Wizard or the Insert Table/Form dialog the Template Builder applies the Table Header Color specified here to the table header background. Customize the default color for your templates.</p>
<b>Generate XSLT 2.0 compliant code</b>	<p>BI Publisher uses the XSLT processor provided by Oracle XDK 11.1.0.7.0, which supports the W3C XSL Transformations 1.0 recommendation. The processor also implements the current working drafts of the XSLT and XPath 2.0 standards. For more information about Oracle XDK see <i>Oracle XML Developer's Kit Programmer's Guide 11g</i>.</p> <p>By default, BI Publisher is compatible with XSLT 1.0. If you wish to use XSLT and XPath 2.0 features in your template enable this option. This configuration is performed at the template level. The template-level setting will override the server setting.</p>

## Setting Connection Options

Options on this tab are reserved for a future release.

## Setting Up a Configuration File

The Template Builder can be used with a BI Publisher configuration file.

The configuration file must be named `xdoconfig.xml` and must be stored in the config directory (example path: `C:\Program Files\Oracle\BI Publisher Desktop\Template Builder for Word\config`) under the BI Publisher directory.

Alternatively, you can use the file name `xdo.cfg`, which is used by the BI Publisher server. The configuration file allows you to:

- Define additional fonts such as Windings to test your templates
- Use security settings for PDF files

Refer to the *Oracle Fusion Middleware Administrator's and Developer's Guide for Oracle Business Intelligence Publisher* for the syntax of the configuration file.

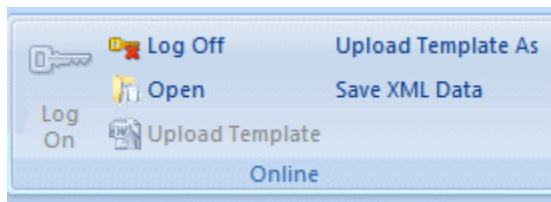
## BI Publisher Menu Reference

When you install the Template Builder the next time you open Microsoft Word, you will see the Oracle BI Publisher menu.

**Note:** If you are using Microsoft Word 2007 you may need to modify your Add-In settings: Click the Office Button, then click Word Options, then click Add-Ins.

## About the Online Group

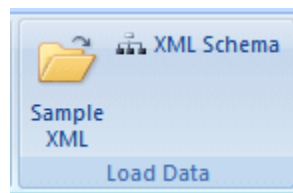
The Online group of commands enable you to initiate interaction with the BI Publisher application. For more information about working with the online commands, see: Working in Connected Mode, page 5-5.



Command	Description
<b>Log on</b>	<p>Enables you to log in to BI Publisher. Enter your username and password. Select or enter the URL for the BI Publisher Report Server (see your Administrator if you do not know the URL). When you log on, the <b>Open Template</b> dialog is displayed.</p> <p><b>Note:</b> You must log in directly to the BI Publisher server. For example:  <a href="http://www.example.com:7001/xmlpserver">http://www.example.com:7001/xmlpserver</a>.</p>
<b>Open</b>	<p>After you log on, this command becomes available to enable you to open a report in the BI Publisher catalog.</p>
<b>Upload Template</b>	<p>If you used the <b>Open Template</b> dialog to download a template from the BI Publisher catalog, use this option to upload the updated layout back to the report definition in the catalog.</p>
<b>Upload Template As</b>	<p>If you used the <b>Open Template</b> dialog to download a template or to open a report from the catalog, use this option to upload the layout to the report definition in the catalog. Also use this option to upload modifications to an existing template under a different name.</p>
<b>Save XML Data</b>	<p>If you are working in connected mode, use this command to save the data to a local directory if you will also need access to the data in disconnected mode.</p>

## About the Load Data Group

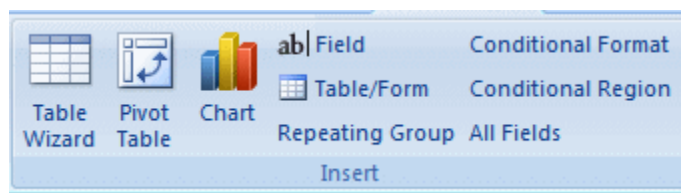
The data group of commands enables you to load a saved sample data file or sample schema to the Template Builder. You must load data to use most of the Template Builder functionality. See *Accessing Data for Building Your Template*, page 5-7 for more options for loading data to the Template Builder.



Command	Description
Sample XML	This command enables you to load a previously saved sample XML file from your report data source. If you are not connected to the BI Publisher server, use this method to load your data.
XML Schema	This command enables you to load an XML Schema file (.xsd) that contains the fields available in your report XML data. The XML schema has the advantage of being complete (a sample xml file may not contain all the fields from the data source). For the preview, the Template Builder can generate dummy sample data for an XML Schema. However, the preview works better if you also upload real sample data.

## About the Insert Group

Use the Insert group to insert the layout components to your template. For more information see [Inserting Components to the Template](#), page 5-9.

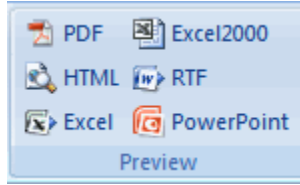


Command	Description
Table Wizard	This function provides a wizard that guides you through the creation of tables used in typical reports.
Pivot Table	The Pivot Table function enables you to drag and drop the data elements to a pivot table structure.
Chart	BI Publisher does not recognize native Microsoft Word charts. The Insert Chart function allows you to insert a chart that is understood by Oracle BI Publisher.

Command	Description
<b>Field</b>	<p>This function allows you to select fields from your data source and insert them into your template.</p> <p><b>Note:</b> As a beginner, you should use Insert Fields only for data fields that are unique – none repeating - in your document. See Inserting a Table, page 5-12 for additional information on how to insert repetitive fields.</p>
<b>Table/Form</b>	<p>Use this function to insert data fields to be organized as a simple or nested table or as a form that is repeated with different data. You may even organize all the data fields for the whole document before inserting them.</p>
<b>Repeating Group</b>	<p>Enables you to select or define a group of elements that you want repeated for each occurrence of an element in the data.</p>
<b>Conditional Format</b>	<p>Enables you to define simple conditional formats to apply to table rows or cells.</p>
<b>Conditional Region</b>	<p>Enables you to insert a conditional statement around a region of the template.</p>
<b>All Fields</b>	<p>This function inserts all fields found in the XML data into your document. It will also insert processing instructions into your document that will repeat a section – such as a table row – when the associated XML element is repeated.</p> <p><b>Note:</b> XML documents often contain a large number of fields in a deeply nested hierarchy. For example, an Oracle Purchasing purchase order contains purchase order lines, which contain shipments, which contain distributions. The purchase order line alone contains more than 150 data fields. In these cases, you should use the Insert Table/Form function to have more control over which fields are inserted.</p>

## About the Preview Group

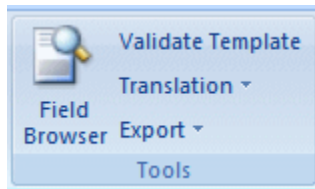
The preview group enables you to preview your RTF template with the sample XML data. . The preview menu offers you PDF, HTML, RTF, PowerPoint, Excel (MHTML format) and EXCEL2000 as output formats. When you select any of these output formats, the Template Builder will merge the data into your template and create the output document.



**Note:** You must have Adobe Acrobat Reader version 5.0 or higher installed to preview documents in PDF format.

## About the Tools Group

For more information about using the commands in the Tools group refer to Template Editing Tools, page 5-32 and Using the Template Builder Translation Tools, page 5-37.



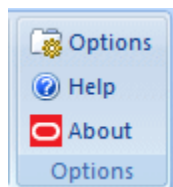
Command	Description
<b>Field Browser</b>	The field browser is a tool for advanced users who need to change the BI Publisher commands hidden in the form fields. It shows the commands behind each form field and allows you to change them. Use this tool to correct flawed RTF templates or to update multiple fields efficiently.
<b>Validate Template</b>	The validation function checks the template for incorrect use of BI Publisher commands and unsupported elements in the Word file.



Command	Description
Translation	<p data-bbox="597 306 1003 333">Includes the following subcommands:</p> <p data-bbox="597 363 1458 552"><b>Extract Text</b> - This command enables you to create a standard XLIFF translation file containing the boilerplate text from your template. XLIFF is a standard file format that is understood by many translation software packages. Because an XLIFF is an XML file, you can translate the text in a text editor. For more information on working with XLIFF files, see <i>Working with Translation Files</i>, page 15-2.</p> <p data-bbox="597 577 1458 667"><b>Preview Translation</b> - this command enables you to preview your template as a PDF file using a specified XLIFF translation file. This functionality enables you to test translation files.</p> <p data-bbox="597 693 1458 816"><b>Localize Template</b> - this command applies a translation file to an RTF template. This means that in your current RTF template all boilerplate text is translated. The main function of this feature is to create a language-specific version of a template.</p>
Export	<p data-bbox="597 863 1003 890">Includes the following subcommands:</p> <p data-bbox="597 919 1458 1010"><b>XSL-FO Stylesheet</b> - this function allows you convert the RTF template into an enhanced XSL-FO stylesheet. This function can be used to generate XSL-FO for debugging or further customization.</p> <p data-bbox="597 1035 1463 1125"><b>Formatted XML</b> - this function allows you to apply the XSL-FO stylesheet generated from the Word document to the sample data and save the intermediate FO format. This function is mainly for debugging.</p> <p data-bbox="597 1150 1192 1178"><b>PDF</b> This function converts the Word document to PDF.</p>

## About the Options Group

The options function allows you to define some preferences and options for using BI Publisher and access online help.



See *Setting Options for the Template Builder*, page 5-39.



---

# Creating Excel Templates

This chapter covers the following topics:

- Introduction
- Concepts
- Building a Simple Template
- Formatting Dates
- Defining BI Publisher Functions
- Preprocessing the Data Using an XSL Transformation (XSLT) File
- Using the Template Viewer to Debug a Template

## Introduction

An Excel template is a report layout that you design in Microsoft Excel for retrieving and formatting your 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 your data is output to Excel workbooks.

## Features of Excel Templates

With Excel templates you can:

- Define the structure for your 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 your data to manipulate it prior to rendering
- Use native Excel functionality

## 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 will show only on the first sheet.
- There is no 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

## 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 will need knowledge of XSL and XSLT.
- The report data model has been created.

## Supported Output

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

## Desktop Tools

BI Publisher provides a downloadable add-in to Excel that enables you to preview your template with sample data. This facilitates design by enabling you to test and edit your template without having to upload it to the BI Publisher catalog first.

The Template Builder for Excel is installed automatically when you install the Template Builder for Word. The tools can be downloaded from the Home page of Oracle Business Intelligence Publisher or Oracle Business Intelligence Enterprise Edition, as follows:

Under the **Get Started** region, click **Download BI Publisher Tools**.

## Sample Excel Templates

The Template Builder includes sample Excel templates.

To access the samples from a Windows desktop:

Click **Start**, then **Programs**, then **Oracle BI Publisher Desktop**, then **Samples**, then **Excel**.

This will launch the folder that contains the Excel sample templates.

## Concepts

Similar to RTF template design, Excel template design follows the paradigm of mapping fields from your 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.

### 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.

Your template content and layout must correspond to the content and hierarchy of the XML data file used as input to your report. Each group of repeating elements in your template must correspond to a parent-child relationship in the XML file. If your 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 Preprocessing the Data Using an XSL Transformation (XSLT) File, page 6-41 and Grouping Functions, page 6-39 for more information about these options.

### 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 your 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.

## 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.

## 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 will automatically adjust the region ranges for those named regions so that the formulas calculate correctly.

However, if you create nested groups in your template, 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.

## About the XDO\_METADATA Sheet

Each Excel template requires a sheet within the template workbook called "XDO\_METADATA". Use this sheet to identify your 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 your completed template to the BI Publisher catalog. This will prevent report consumers from seeing it in the final report output.

**Note:** For more information see Format of the XDO\_METADATA Sheet, page 6-11 and Defining BI Publisher Functions, page 6-18.

## Building a Simple Template

This section will demonstrate the concepts of Excel templates by walking through the steps to create a simple Excel template and testing it with the Excel Template Builder. This procedure will follow these steps:

1. Obtain sample XML data from your data model.

2. Open the BlankExcelTemplate.xls file and save as your template name.
3. Design the layout in Excel.
4. Assign the BI Publisher defined names.
5. Prepare the XDO\_METADATA sheet.
6. Test the template using the desktop Excel Template Builder.

## Step 1: Obtain sample XML data from your data model

You need sample data in order to know your field names and the hierarchical relationships to properly mark up the template. For information on saving sample data from your report data model, see the topic "Testing Data Models and Generating Sample Data" in the *Oracle Fusion Middleware Data Modeling Guide for Oracle Business Intelligence Publisher*.

If you do not have access to the report data model, but you can access the report, you can alternatively save sample data from the report viewer. To save data from the report viewer:

1. In the BI Publisher catalog, navigate to the report.
2. Click **Open** to run the report in the report viewer.
3. Click the **Actions** menu, then click **Export**, then click **Data**. You will be prompted to save the XML file.
4. Save the 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"?>
<! - Generated by Oracle BI Publisher 11.1.1.4.0 - >
<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>

```

## Step 2: Open the BlankExcelTemplate.xls file and save as your template name

**Note:** It is recommended that you install the Template Builder for Excel. For information on downloading the tool, see Desktop Tools, page 6-2.

The Template Builder installation includes a set of sample Excel templates, including a sample blank Excel template called BlankExcelTemplate.xls. This template file contains a blank Sheet1 and the XDO\_METADATA sheet. It is recommended that you either start with this provided template or copy the XDO\_METADATA sheet into your own Excel workbook.

**Tip:** If you are building a new template from an existing template, be sure to clear any existing defined names in the template sheet.

To open the BlankExcelTemplate.xls:

1. From a Windows desktop, click **Start**, then **Programs**, then **Oracle BI Publisher Desktop**, then **Samples**, then **Excel**.
2. In the Excel Templates sample folder, double-click BlankExcelTemplate.xls to open it.



3. Save the file as your selected name in the Microsoft Excel 97-2003 Workbook format (\*.xls).

### Step 3: Design the layout in Excel

In Excel, determine how you want to render the data and create a sample design, as shown in the following figure:

	A	B	C	D	E	F
1	<b>Employees by Department Report</b>					
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 will require a calculation.

### Step 4: 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 your data delimited by "?"
  - a unique name that you will 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 the XML tag name for the parent element in your XML data delimited by "?".
- a unique name that you will 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.

### Applying 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 will display 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 your data. For example: XDO\_?EMP\_NAME?
4. Press Enter.

The following figure shows the defined name for the Employee Name field entered in the Name box:

The screenshot shows an Excel worksheet with the following data:

Employee Name	Employee ID	Email	Telephone	Salary
Jennifer Whalen	200	JWHALEN	515-123-4444	10,000.00
				10,000.00

The Name Box in the formula bar is highlighted with a red box and contains the text "XDO\_?EMP\_NAME?".

5. Repeat for each of the following data fields: DEPARTMENT\_NAME, EMPLOYEE\_ID, EMAIL, PHONE\_NUMBER, and SALARY.

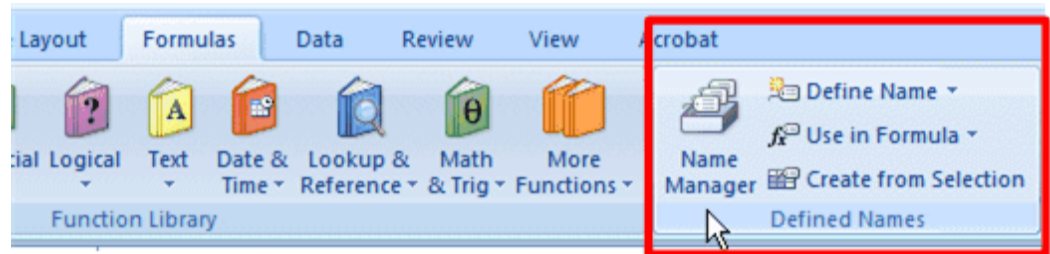
**Tip:** If you navigate out of the Name box without pressing Enter, the name you entered will not be maintained.

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

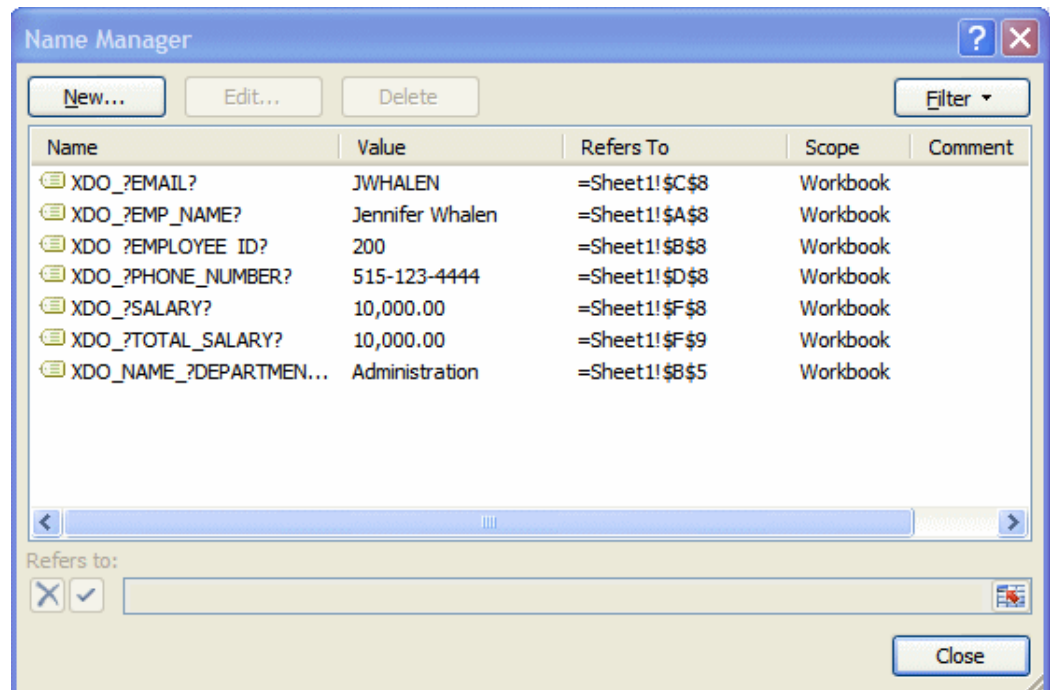
The name cannot be more than 255 characters in length.

- For the total salary field, a calculation will be mapped to that cell. For now, name that cell XDO\_?TOTAL\_SALARY?. The calculation will be 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:



After you have named all the cells for this example, the Name Manager dialog will appear as shown:



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

## 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) will display in the worksheet.

- For each occurrence of the <DEPT> element, the department name and the list of employees belonging to that department will display.

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

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

establishes the grouping desired for the report.

### 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 your data. For example: XDO\_GROUP\_?EMPS?
3. Press Enter.

The following figure shows the XDO\_GROUP\_ defined named 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.

	A	B	C	D	E
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					

To define the department group, include the department name cell and all the employee fields beneath it (A5-E9) as shown in the following figure:

:DO_GROUP_?DEPT? Department:					
	A	B	C	D	E
1	<b>Employees by Department Report</b>				
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					

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.

## Step 5: 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.

### Format of the XDO\_METADATA Sheet

The XDO\_METADATA sheet must have the format shown in the following figure:

	A	B	C
1	Version		
2	ARU-dbdv		
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			
12			

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 Defining BI Publisher Functions, page 6-18.

### Creating the XDO\_METADATA Sheet

If you copied the XDO\_METADATA sheet in Step 2, skip this section and proceed to Adding the Calculation for the XDO\_?TOTAL\_SALARY? Field, page 6-12; otherwise, set up the hidden sheet as follows:

1. Create a new sheet in your 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-dbdv
  - 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

### Adding the Calculation for the XDO\_?TOTAL\_SALARY? Field

Earlier in this procedure you assigned the defined named XDO\_?TOTAL\_SALARY? to the cell that is to display the total salaries listed in the SALARY column. In this step, you will add the calculation to the Data Constraints section of the XDO\_METADATA sheet and map the calculation to the XDO\_?TOTAL\_SALARY? field.

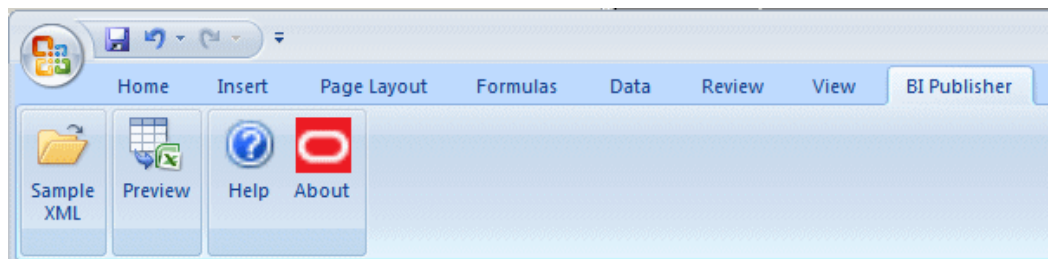
1. In the Data Constraints section, in Column A, enter the defined name of the cell:  
XDO\_?TOTAL\_SALARY?
2. In Column B enter the calculation as an XPATH function. To calculate the sum of the SALARY element for all employees in the group, enter the following:  
<?sum(./SALARY)?>

The completed XDO\_METADATA sheet is shown in the following figure:

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

## Step 6: Test the template

If you have installed the Template Builder for Excel, the BI Publisher tab will appear on the ribbon menu as shown in the following figure:



To preview your report using sample data:

1. Click **Sample XML**. You will be prompted to select the sample data file.
2. Click **Preview**.

The sample data will be applied to your template and the output document will be opened in a new workbook. The following figure shows the preview of the template with the sample data:

1	<b>Employees by Department Report</b>				
2					
3					
4					
5	Department:	Marketing			
6					
7	<b>Employee Name</b>	<b>Employee ID</b>	<b>Email</b>	<b>Telephone</b>	<b>Salary</b>
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	<b>Employee Name</b>	<b>Employee ID</b>	<b>Email</b>	<b>Telephone</b>	<b>Salary</b>
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	<b>Employee Name</b>	<b>Employee ID</b>	<b>Email</b>	<b>Telephone</b>	<b>Salary</b>
24	Susan Mavris	203	SMAVRIS	515.123.7777	6,500.00
25					6,500.00

## Formatting Dates

Excel cannot recognize canonical date format. If the date format in your 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 your date is to use the Excel REPLACE and SUBSTITUTE functions. This option will retain the full date and timestamp. If you only require the date portion in your data (YYY-MM-DD), another option is to use the DATEVALUE function. The following example shows how to use both options.

### Example

#### 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 the following figure:



	A	B	C	D	E	F
1	<b>Employees by Department Report</b>					
2						
3						
4						
5	<b>Department:</b>	Admin				
6						
7	<b>Employee Name</b>	<b>Employee ID</b>	<b>Email</b>	<b>Telephone</b>	<b>Hire Date</b>	<b>Salary</b>
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 as shown above, follow these steps:

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 the following figure:

	A	B	C	D	E	F
1	<b>Employees by Department Report</b>					
2						
3						
4						
5	<b>Department:</b>	Admin				
6						
7	<b>Employee Name</b>	<b>Employee ID</b>	<b>Email</b>	<b>Telephone</b>	<b>Hire Date</b>	<b>Salary</b>
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 will display 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 will now be column F as shown in the following figure:

	A	B	C	D	E	F	G
1	<b>Employees by Department Report</b>						
2							
3							
4							
5	<b>Department:</b>	Admin					
6							
7	<b>Employee Name</b>	<b>Employee ID</b>	<b>Email</b>	<b>Telephone</b>	<b>Hire Date</b>	<b>Hire Date</b>	<b>Salary</b>
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:

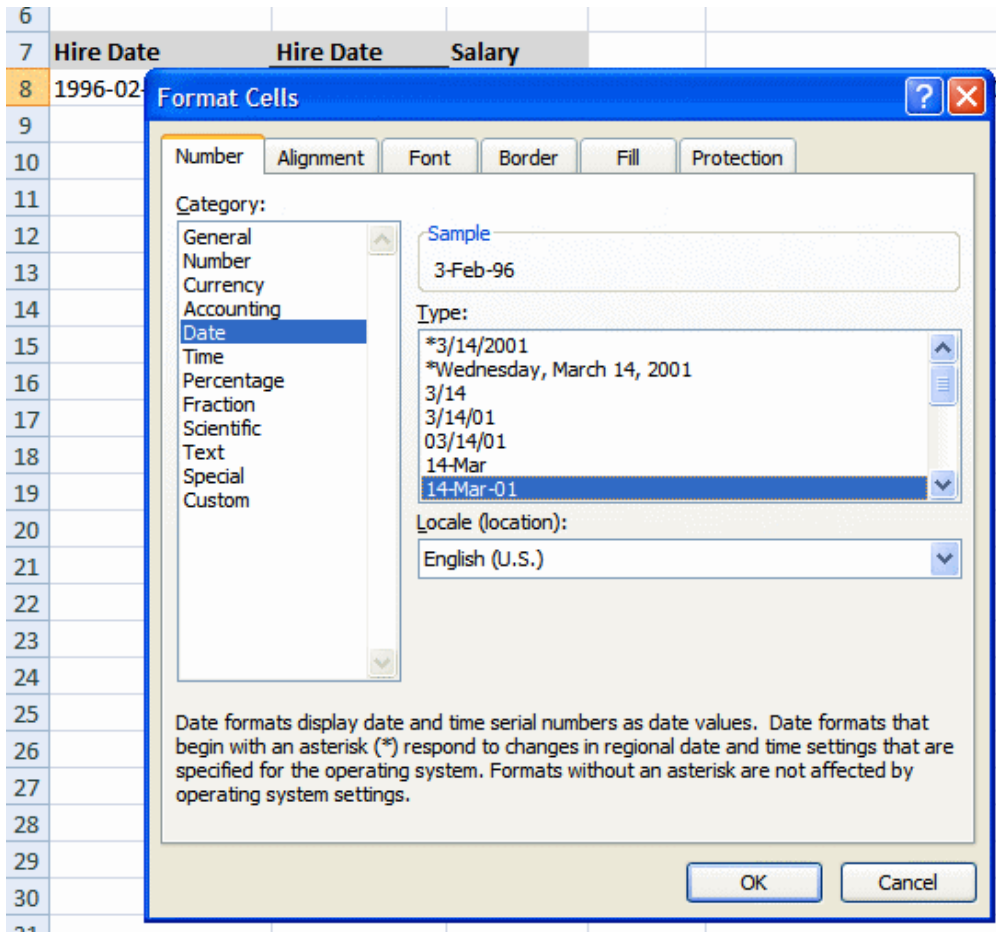
- To retain the full date and timestamp, enter:  
`==REPLACE (SUBSTITUTE (E8, "T", " " ), LEN (E8) - 6, 6, "")`
- To retain only the date portion (YYY-MM-DD), enter:  
`DATEVALUE (LEFT (E8, 10) )`

After you enter the function, it will populate the F8 cell as shown in the following figure:

	A	B	C	D	E	F	G
1	<b>Employees by Department Report</b>						
2							
3							
4							
5	<b>Department:</b>	Admin					
6							
7	<b>Employee Name</b>	<b>Employee ID</b>	<b>Email</b>	<b>Telephone</b>	<b>Hire Date</b>	<b>Hire Date</b>	<b>Salary</b>
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 the following figure:



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

6. Finally, hide the E column, so that report consumers will not see the canonical date that is converted.

	A	B	C	D	F	G	H
1	<b>Employees by Department Report</b>						
2							
3							
4							
5	<b>Department:</b> Admin						
6							
7	<b>Employee Name</b>	<b>Employee ID</b>	<b>Email</b>	<b>Telephone</b>	<b>Hire Date</b>	<b>Salary</b>	
8	John Thomas	100001	john@oracle.com	111-333-3333	3-Feb-96	10,000	
9						10,000	
10							
11							
12							
13							

## 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:

- Reporting Functions, page 6-18
- Formatting Functions That Rely on Specific Data Attribute Values, page 6-29
- Grouping Functions, page 6-39

## Reporting Functions

The following functions can be added to your template using the commands shown and a combination of BI Publisher syntax and XSL. A summary list of the commands is shown in the following table. See the corresponding section for details on usage.

Function	Commands
Split the report data into multiple sheets, page 6-19	XDO_SHEET_? with XDO_SHEET_NAME_?
Define a parameter, page 6-22	XDO_PARAM_?n?
Define a link, page 6-24	XDO_LINK_?link object name?
Import a subtemplate, page 6-26	XDO_SUBTEMPLATE_?n?
Reference Java extension libraries, page 6-28	XDO_EXT_?n?

## 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 will show only on the first sheet.

Use the this set of commands to define the logic to split the report data into multiple sheets:

- 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.

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

Column A Entry	Column B Entry	Column C Entry
XDO_SHEET_NAME_?	<p>&lt;?xsl_expression to name the sheet?&gt;</p> <p>Example:</p> <pre>&lt;?concat (./DEPARTMENT_NAME, '-', count (./EMP_NAME) ) ?&gt;</pre>	<p>(Optional)</p> <p>&lt;?original sheet name?&gt;</p> <p>Example: &lt;?Sheet3?&gt;</p>

XDO\_SHEET\_? must refer to an existing high-level node in your XML data. The example <?./DEPT?> will create a new sheet for each occurrence of <DEPT> in the data.

If your data is flat 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 Preprocessing the Data Using an XSL Transformation (XSLT) File, page 6-41.

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 your final sheet name. This example:

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

will name 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 will apply the naming to the first sheet in the workbook that contains XDO\_names. You would need to 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 would enter <?SHEET3?> in Column C.

### Example: Splitting the data into multiple sheets

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

- Create a new worksheet for each department
  - Name each worksheet the name of the department with the number of employees in the department, for example: Sales-21.
1. Enter the defined names for each cell of employee data and create the group for the repeating employee data:

	A	B	C	D	E	F	G	
4								
5	Department:	Administration						
6								
7	Employee Name	Employee ID	Email	Telephone	Hire Date	Salary		
8	Jennifer Whalen	200	JWHALEN	515-123-4444	3-Feb-96	10,000.00		
9						10,000.00		
10								
11								

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

- Enter the following in the Data Constraints section of the XDO\_METADATA sheet:

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

The entries are shown in the following figure:

	A	B
1	Version	
2	ARU-dbdv	
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_SHEET_?	<?./DEPT?>
12	XDO_SHEET_NAME_?	<?concat(./DEPARTMENT_NAME, '-', count(./EMP_NAME))?>
13	XDO_?TOTAL_SALARY?	<?sum(./SALARY)?>
14		
15		
16		

The following figure shows the generated report. Each department data now displays on its own sheet, which shows the naming convention specified:

	A	B	C	D	F	G	H
2							
3							
4							
5	<b>Department:</b>	Purchasing					
6							
7	<b>Employee Name</b>	<b>Employee ID</b>	<b>Email</b>	<b>Telephone</b>	<b>Hire Date</b>	<b>Salary</b>	
8	Shelli Baida	116	SBAIDA	515.127.4563	24-Dec-97	2,900	
9	Sigal Tobias	117	STOBIAS	515.127.4564	24-Jul-97	2,800	
10	Karen Colmenare	119	KCOLMENA	515.127.4566	10-Aug-99	2,500	
11	Alexander Khoo	115	AKHOO	515.127.4562	18-May-95	3,100	
12	Guy Himuro	118	GHIMURO	515.127.4565	15-Nov-98	2,600	
13	Den Raphaely	114	DRAPHEAL	515.127.4561	7-Dec-94	11,000	
14						24,900	
15							
16							

## 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 your data model.

To declare the parameter, use the following command:

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

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 follows:



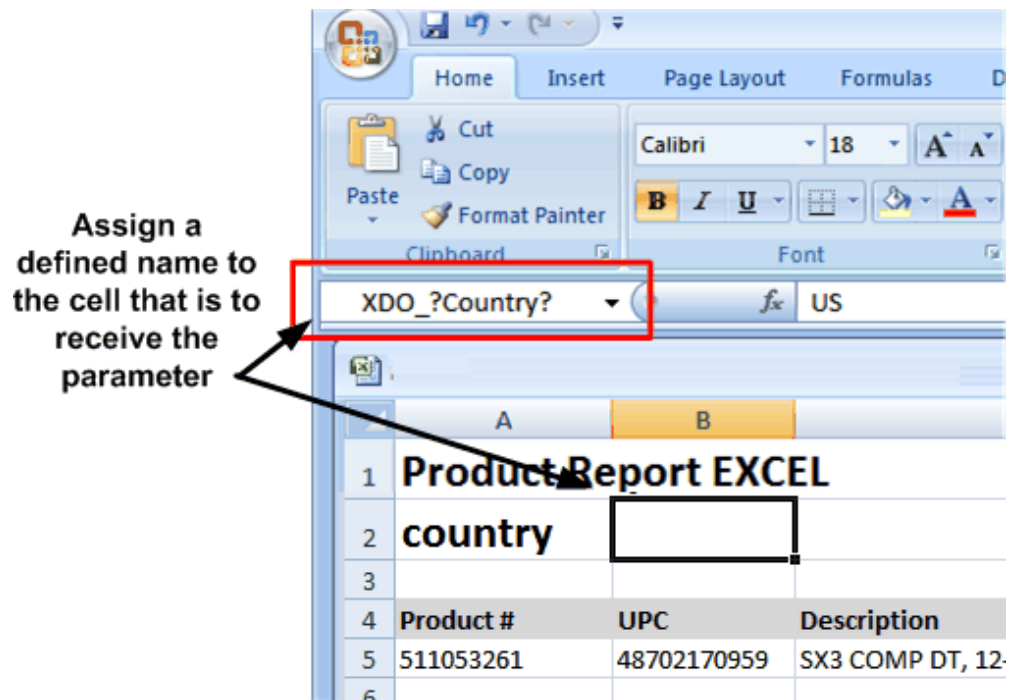
Column A Entry	Column B Entry
XDO_PARAM_?parameter_name?	<?\$parameter_name?>.
for example:	For example:
XDO_PARAM_?Country?	<?\$Country?>

You can also refer to the parameter in other logic or calculations in the XDO\_METADATA sheet using \$parameter\_name.

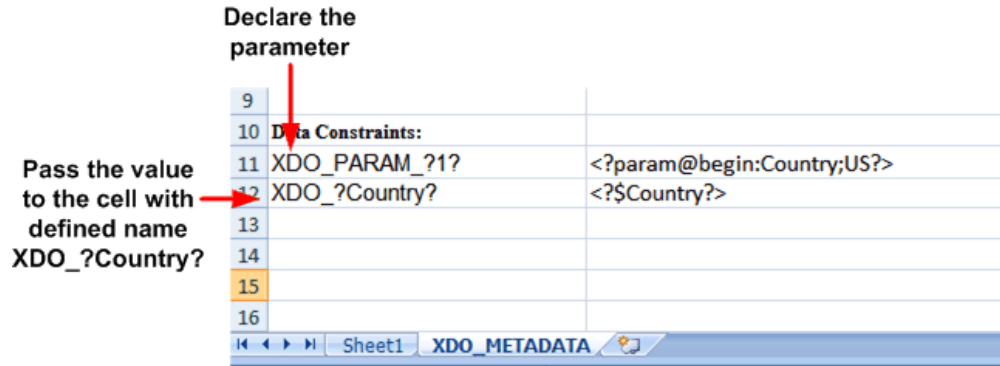
**Example: Defining and passing a parameter**

In this example, 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?



2. In the hidden sheet assign that cell the parameter value as follows:



## Defining a Link

Use the XDO\_LINK\_? command to define a hyperlink for any data cell.

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)"/>

### Example: Defining a Link

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). You can generate a report that creates a dynamic link to each invoice as follows:

1. In your template sheet, assign the cell that is to display the INVOICE\_NO the XDO defined name: XDO\_?INVOICE\_NO?.

	A	B	C	D
1				
2				
3				
4		<b>Customer</b>	<b>Priority</b>	<b>Invoice Number</b>
5		Smith	2	1111
6				
7				
8				

2. In the XDO\_METADATA sheet, enter the following:

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

The entries in Excel are shown in the following figure:

10	<b>Data Constraints:</b>	
11	XDO_LINK_?INVOICE_NO?	<pre>&lt;xsl:value-of select="concat('https://server.company.com/documents/invoice_print.show?c_rptno= ',./INVOICE_NO)"/&gt;</pre>
12		

The report output will display as shown in the following figure. The logic defined in the XDO\_METADATA sheet will be applied to create a hyperlink for each INVOICE\_NO entry:

Customer	Priority	Invoice Number
Aaron	3	<a href="#">5952174</a>
Abner	3	<a href="#">8280605</a>
Acheson	2	<a href="#">7604618</a>
Addison	3	<a href="#">9645172</a>
Aeschelle	3	<a href="#">9616238</a>
Agers	2	<a href="#">9685385</a>
Ahab	3	<a href="#">9644915</a>
Aiden	3	<a href="#">9663648</a>
Ajackal	3	<a href="#">9685016</a>
Akkers	3	<a href="#">9706403</a>
DZIMA	3	<a href="#">9645249</a>
CGUO	3	<a href="#">9706193</a>

### 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 Builder for Excel does not support preview for templates that import subtemplates.

To import the subtemplate, enter the following command:

Column A Entry	Column B Entry
XDO_SUBTEMPLATE_?n?	<code>&lt;xsl:import href="xdoxsl:///path to subtemplate.xsb?"/&gt;</code>
where <i>n</i> is a unique identifier.	For example:
For example:	<code>&lt;xsl:import href="xdoxsl:///Shared Folders/Financial Reports/SubTemplates/MySubTemplate.xsb?"/&gt;</code>
XDO_SUBTEMPLATE_?1?	
.	

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:

Column A Entry	Column B Entry
XDO_?cell object name?	<pre>&lt;xsl:call-template name="template_name"&gt; &lt;/xsl:call-template&gt;</pre>

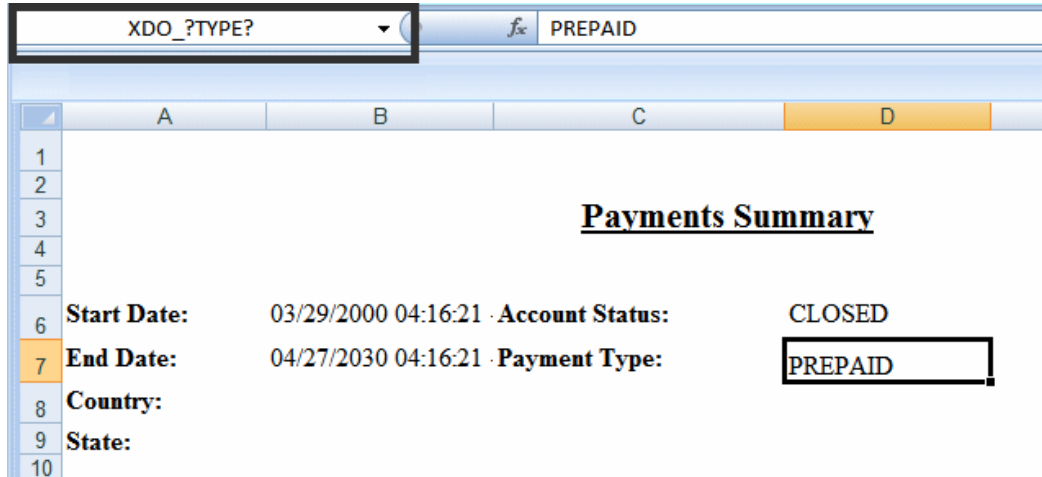
For more information on XSL subtemplates and creating the subtemplate object in the catalog, see *Designing XSL Subtemplates*, page 14-1.

### Example: Importing and Calling a Subtemplate

Assume you have the following subtemplate uploaded to the BI Publisher catalog as `PaymentsSummary-SubTemplate.xsb`. This subtemplate will evaluate the value of a parameter named `pPayType` and based on the value, 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 your Excel template, you have defined a field with the XDO Defined Name `XDO_?TYPE?`, which will be populated based on the string returned from code performed in the subtemplate, as shown in the following figure:



Enter the following in the Data Constraints region:

Column A Entry	Column B Entry
XDO_SUBTEMPLATE_?1?	<pre>&lt;xsl:import href="xdoxsl:///Shared Folders/Financial Reports/SubTemplates/PaymentsSummary-SubTe mplate.xsb?"/&gt;</pre>
XDO_?TYPE?	<pre>&lt;xsl:call-template name="BRM_PAY_TYPES"&gt; &lt;xsl:with-param name="pPayType" select="string('10000')"/&gt; &lt;/xsl:call-template&gt;</pre>

The XDO\_SUBTEMPLATE\_?1? function imports the subtemplate from the BI Publisher catalog.

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

## Referencing Java Extension Libraries

You can include the reference to a Java extension library in your template and then call methods from this library to perform processing in your template. Use this command to reference the Java extension libraries:

Column A Entry	Column B Entry
XDO_EXT_?n? where <i>n</i> is a unique identifier. Example: XDO_EXT?1?	<?namespace:xmlns:bipext="extension library"?>  Example:  <?namespace:xmlns:bipext="http://www.oracle.com/XSL/Transform/java/oracle.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 your 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. For example:

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 the following figure:

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

## 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 the following table. See the corresponding section for details on usage.

Function	Command
Define border and underline styles, page 6-30	XDO_STYLE_n_?cell object name?
Skip a row, page 6-36	XDO_SKIPROW_?cell object name?

### 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.

Column A Entry	Column B Entry	Column C Entry
XDO_STYLE_n_?cell_object_name?	<i>&lt;xsl evaluation that returns a supported value&gt;</i>	Style type
For example:	For example:	For example: BottomBorderStyle
XDO_STYLE_1_?TOTAL_SALARY?	<code>&lt;xsl:value-of select="//TOTAL_SALARY/@borderStyle"/&gt;</code>	

BI Publisher supports the following normal Excel style types and values:



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

You can also set a color using one of the following types:

<b>Style Type</b>	<b>Supported Value (Must be in returned by evaluation in Column B)</b>	<b>Supported Types (Enter in Column C)</b>
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 following values:

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 will be 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 will be used to apply the defined style based on logic defined in the template.

```

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

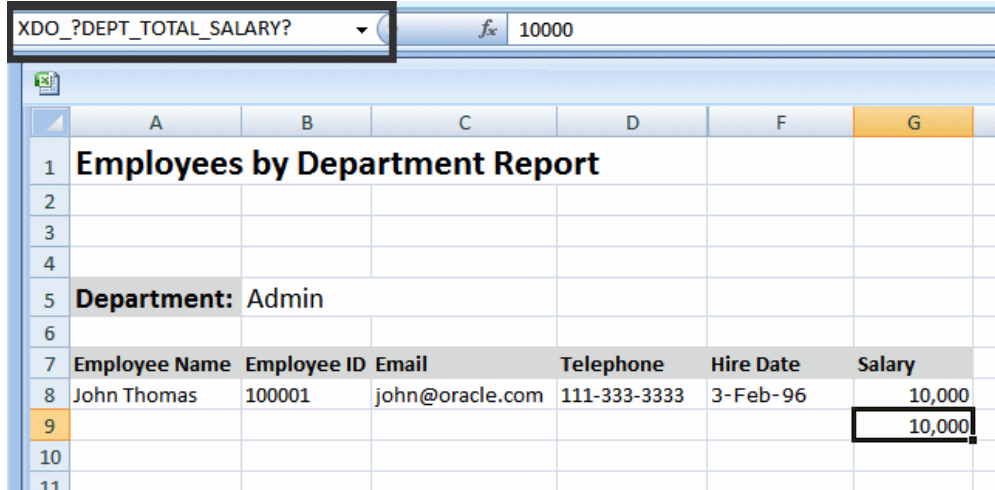
<EMPLOYEES>
  <G_DEPT>
    <DEPARTMENT_ID>10</DEPARTMENT_ID>
    <DEPARTMENT_NAME>Administration</DEPARTMENT_NAME>
    <LIST_G_EMP>
      <G_EMP>
        <EMPLOYEE_ID>200</EMPLOYEE_ID>
        <EMP_NAME>Jennifer Whalen</EMP_NAME>
        <EMAIL>JWHALEN</EMAIL>
        <PHONE_NUMBER>515.123.4444</PHONE_NUMBER>
        <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>

```

1. In the Excel template assign the defined named XDO\_?DEPT\_TOTAL\_SALARY? to the field that is to display the DEPT\_TOTAL\_SALARY from the data.



2. In the XDO\_METADATA sheet, enter the following:

- To define the top border style, enter:

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, enter:

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, enter:

Column A Entry	Column B Entry	Column C Entry
XDO_STYLE_3_?DEPT_T OTAL_SALARY?	<?.//DEPT_TOTAL_SA LARY/@underLineStyle le?>	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.

The following figure shows the three entries in the Data Constraints region:

10	<b>Data Constraints:</b>		
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			
15			

When you run the report, the style commands will be applied to the `XDO_?DEPT_TOTAL_SALARY?` cell, as shown in the following figure:

	A	B	C	D	F	G
1	<b>Employees by Department Report</b>					
2						
3						
4						
5	<b>Department:</b>	Administration				
6						
7	<b>Employee Name</b>	<b>Employee ID</b>	<b>Email</b>	<b>Telephone</b>	<b>Hire Date</b>	<b>Salary</b>
8	Jennifer Whalen	200	JWHALEN	515.123.4444	17-Sep-87	4,400
9						<u>4,400</u>
10	<b>Department:</b>	Marketing				
11						
12	<b>Employee Name</b>	<b>Employee ID</b>	<b>Email</b>	<b>Telephone</b>	<b>Hire Date</b>	<b>Salary</b>
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						<u>19,000</u>
16	<b>Department:</b>	Purchasing				
17						
18	<b>Employee Name</b>	<b>Employee ID</b>	<b>Email</b>	<b>Telephone</b>	<b>Hire Date</b>	<b>Salary</b>
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						<u>24,900</u>
26	<b>Department:</b>	Human Resources				

### 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".

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

### Example: Skipping a Row Based on Data Element Attribute

In this example, the Excel template will suppress 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 following in the Data Constraints section:

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

The output from this template is shown in the following figure. Note that the employee Michael Hartstein is not included in the report:



	A	B	C	D	F	G
1	<b>Employees by Department Report</b>					
2						
3						
4						
5	<b>Department:</b>	Marketing				
6						
7	<b>Employee Name</b>	<b>Employee ID</b>	<b>Email</b>	<b>Telephone</b>	<b>Hire Date</b>	<b>Salary</b>
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						<u>14,000</u>
11						

## Grouping Functions

Use these functions to create groupings of data in the template.

Function	Command
Group data, page 6-39	XDO_GROUP_?group element?
Regroup data, page 6-40	XDO_REGROUP_?

## Grouping the data

Use the XDO\_GROUP command to group flat data when your layout requires a specific data grouping, for example, to split the data across multiple sheets.

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

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.

## Regrouping the Data

The `XDO_REGROUP` regroups the data by declaring the structure using the defined names. It does not require the XSLT logic.

Column A Entry	Column B Entry
<code>XDO_REGROUP_?</code>	<code>XDO_REGROUP_?UniqueGroupID?levelName?groupByName?sortByName?sortByName?sortByName?</code>  where <ul style="list-style-type: none"> <li>• <code>UniqueGroupID</code> is the ID of the group. It can be the same as the <code>levelName</code> or you can assign it unique name.</li> <li>• <code>levelName</code> is the XML level tag name in the XML data file or <code>current-group()</code> in the context of the <code>XDO_</code> grouping structure.</li> <li>• <code>groupByName</code> is the field name that you want to use for the <code>GroupBy</code> operation for the current group. This name can be empty if the <code>XDO_REGROUP_?</code> command is used for the most inner group.</li> <li>• <code>sortByName</code> is the field name that you want to sort the group by. You can have multiple <code>sortBy</code> fields. If no <code>sortByName</code> is declared, the data from the XML file will not be sorted.</li> </ul>

The following shows an example of how to create three nested groupings:

Column A Entry	Column B Entry
<code>XDO_REGROUP_?</code>	<code>XDO_REGROUP_?PAYMENTSUMMARY_Q1?PAYMENTSUMMARY_Q1?PAY_TYPE_NAME?</code>

In the above definition, the most outer group is defined as `PAYMENTSUMMARY_Q1`, and it is grouped by `PAY_TYPE_NAME`

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

The above definition creates a second outer group. The group is assigned the name COUNTRY\_GRP and it is grouped by the element COUNTRY.

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

This definition creates the inner group STATEGRP and it includes a sortByName parameter: STATE.

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

For the best performance, you should design your 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 Builder for Excel 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 catalog, as you would a template:
  1. Navigate to the report in the catalog.
  2. Click **Edit**.

3. Click **Add New Layout**.
4. Click **Upload**.
5. Complete the fields in the Upload dialog and select "XSL Stylesheet (HTML/XML/Text)" as the template **Type**.
6. After upload, click **View a List**. Deselect **Active**, so that users will not see this template as an option when they view the report.

**Note:** For testing purposes, you may want to maintain the XSL template as active to enable you to view the intermediate data when the template is applied to the data. After testing is complete, set the template to inactive.

7. Save the report definition.

3. In your 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

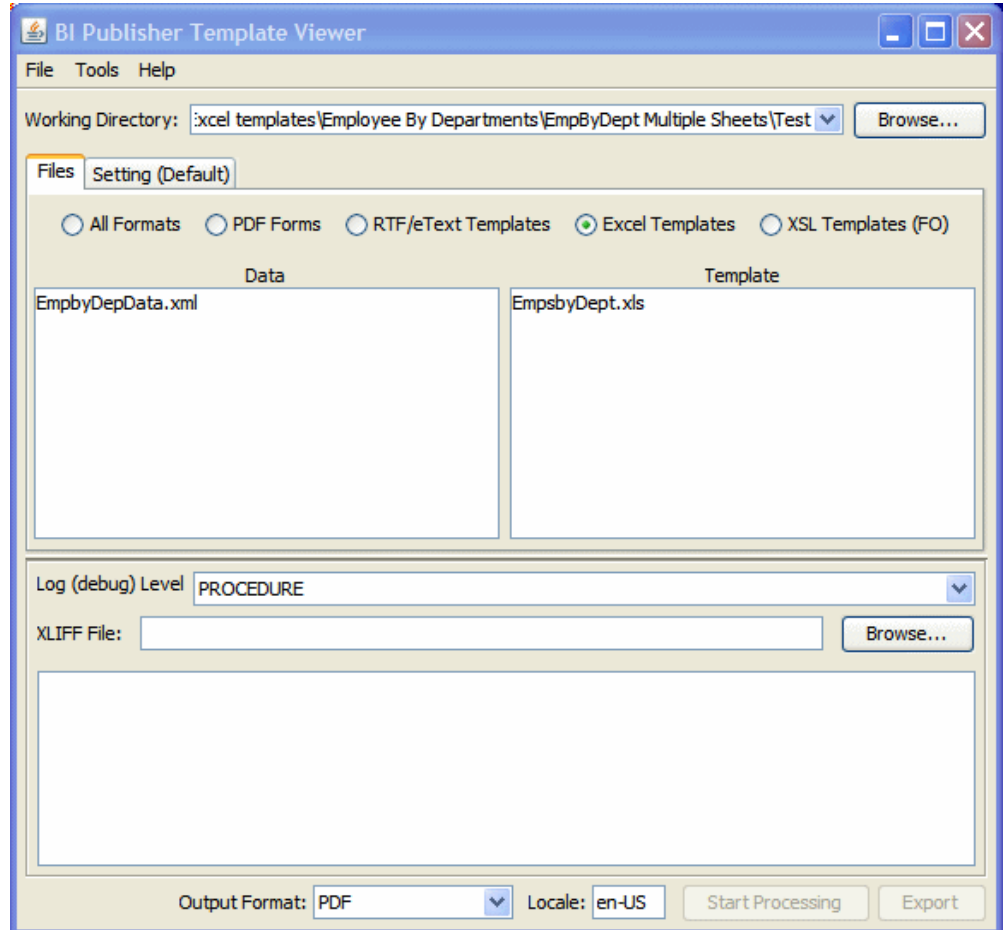
## Using the Template Viewer to Debug a Template

If your template preview is not generating the results expected, 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 this can be a very useful debugging tool.

The Template Viewer is installed when you install the Template Builder for Word; see Desktop Tools, page 6-2 for more information.

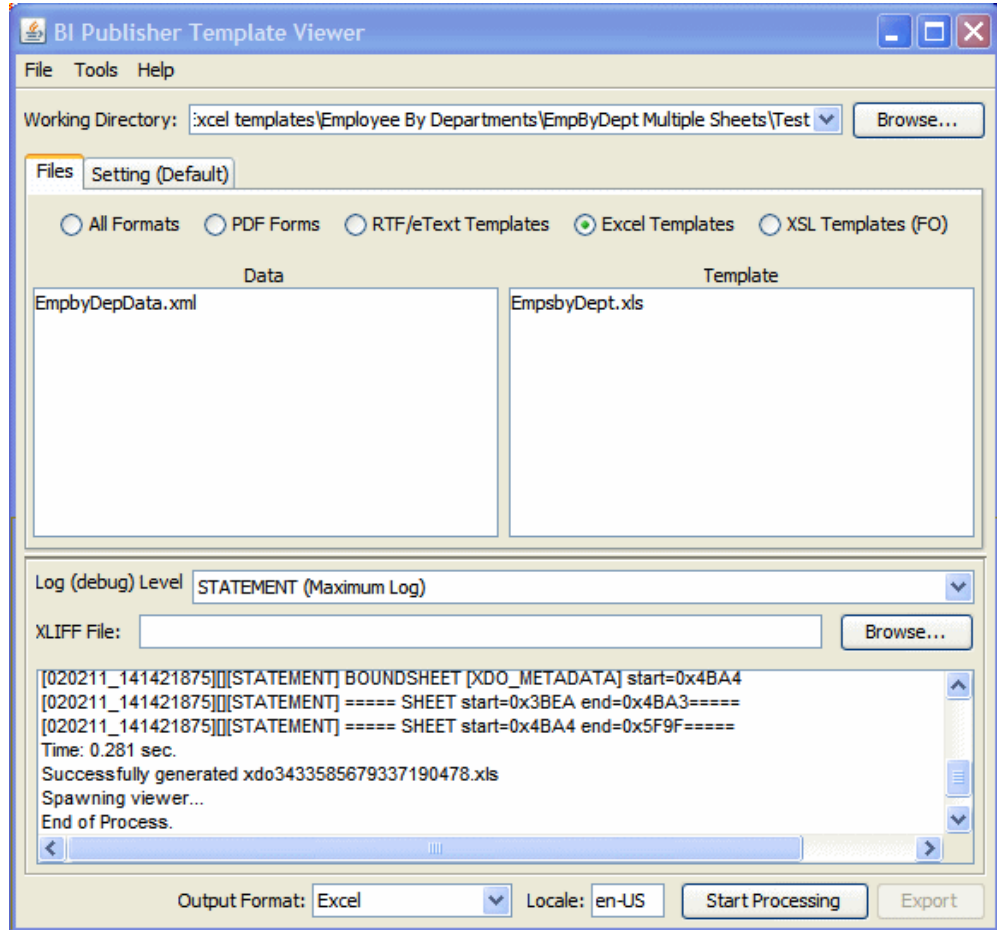
### To preview with the Template Viewer and view log messages:

1. Open the Template Viewer:  
From your Windows desktop, click **Start**, then **Programs**, then **Oracle BI Publisher Desktop**, then **Template Viewer**.
2. Click **Browse** to locate the folder that contains your 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 will display all .xml files and all .xls files present in the directory.



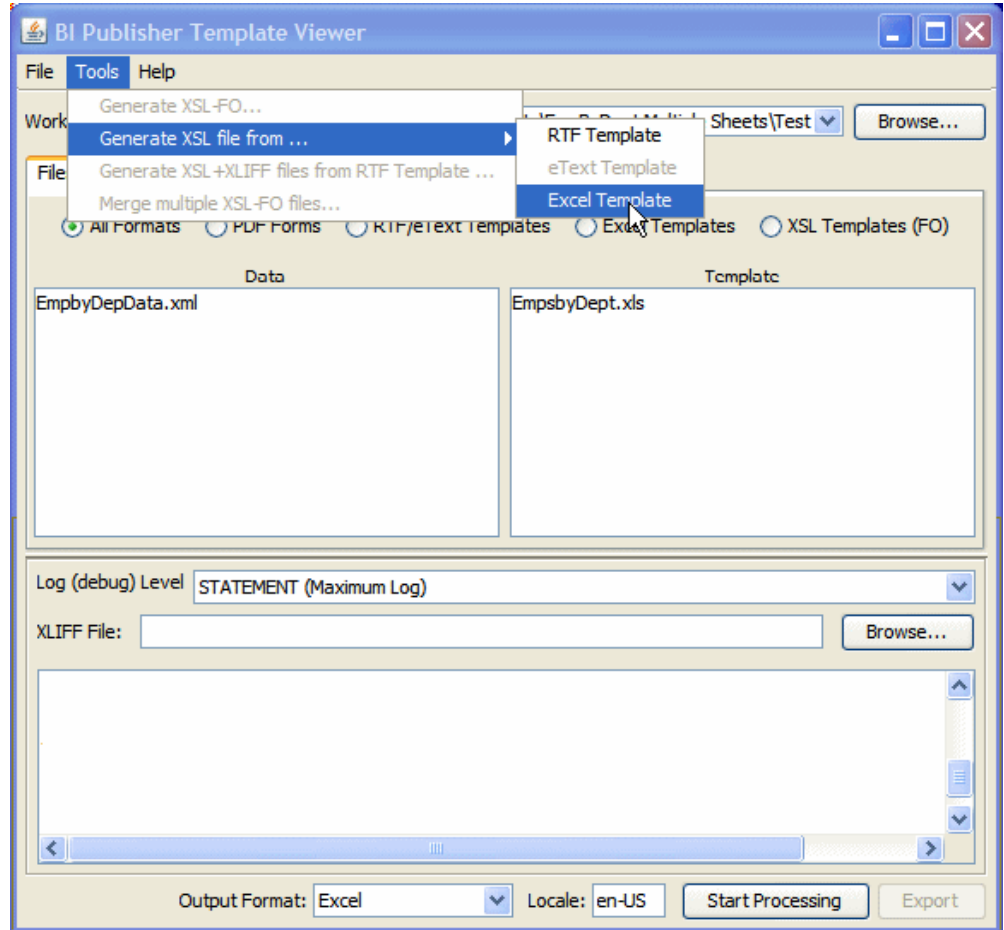
4. Click the appropriate data and template files to select them.
5. Select the log level.
6. From the **Output Format** list, select Excel.
7. Click **Start Processing**.

The Template Viewer will merge the selected data with the selected template and spawn the appropriate viewer. View any log messages in the message box as shown in the following figure:



**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**



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





---

# Creating PDF Templates

## Overview

To create a PDF template, take any existing PDF document and apply the BI Publisher markup. Because you can use a PDF from any source, you have multiple design options. For example:

- Design the template using any application that generates documents that can be converted to PDF
- Scan a paper document to use as a template
- Download a PDF document from a third-party Web site

**Note:** The steps required to create a template from a third-party PDF depend on whether form fields have been added to the document. For more information, see *Creating a template from a Predefined PDF Form*, page 7-17.

If you are designing the template, note that once you have converted to PDF, your template is treated like a set background. When you mark up the template, you draw fields on top of this background. To edit the template, you must edit your original document and then convert back to PDF.

For this reason, the PDF template is not recommended for documents that will require frequent updates. However, it is appropriate for forms that will have a fixed template, such as invoices or purchase orders.

## Requirements

To apply or edit form fields to a PDF document you must have Adobe Acrobat Professional. BI Publisher supports Adobe Acrobat 5.0 and later as a tool for updating your template. Please note, however, that regardless of version that you are using to

design your template, you must save your PDF file as Adobe Acrobat 5.0 (PDF specification version 1.4).

**Important:** If you are using a later version of Adobe Acrobat Professional, you must use the **Reduce File Size Option** (available from the **Document** menu or from the **File** menu depending on your version) to save your file as **Adobe Acrobat 5.0 compatible**.

For PDF conversion, BI Publisher supports any PDF conversion utility, such as Adobe Acrobat Distiller.

## Designing the Template

To design the template you can use any desktop application that generates documents that can be converted to PDF. Or, scan in an original paper document to use as the background for the template.

The following is the template for a sample purchase order. It was designed using Microsoft Word and converted to PDF using Adobe Acrobat Distiller.

**ORACLE**  
E-Business Suite

**Purchase Order**

PURCHASE ORDER NO.		REVISION	PAGE
--------------------	--	----------	------

SHIP TO:

BILL TO:

VENDOR:

CUSTOMER ACCOUNT NO.	VENDOR NO.	DATE OF ORDER/BUYER	REVISED DATE/BUYER
PAYMENT TERMS		SHIP VIA	JOB
FREIGHT TERMS		REQUESTOR/DELIVER TO	CONFIRM TO/TELEPHONE

ITEM	PART NUMBER/DESCRIPTION	DELIVERY	QUANTITY	UNIT	UNIT PRICE	EXTENSION	TAX
						<b>Total</b>	

Oracle E-Business Suite

AUTHORIZED SIGNATURE

The following is the XML data that will be used as input to this template:

```

<?xml version="1.0"?>
<POXPRPOP2>
  <G_HEADERS>
    <POH_PO_NUM>1190-1</POH_PO_NUM>
    <POH_REVISION_NUM>0</POH_REVISION_NUM>
    <POH_SHIP_ADDRESS_LINE1>3455 108th Avenue</POH_SHIP_ADDRESS_LINE1>
  <POH_SHIP_ADDRESS_LINE2></POH_SHIP_ADDRESS_LINE2>
  <POH_SHIP_ADDRESS_LINE3></POH_SHIP_ADDRESS_LINE3>
  <POH_SHIP_ADR_INFO>Seattle, WA 98101</POH_SHIP_ADR_INFO>
  <POH_SHIP_COUNTRY>United States</POH_SHIP_COUNTRY>
  <POH_VENDOR_NAME>Allied Manufacturing</POH_VENDOR_NAME>
  <POH_VENDOR_ADDRESS_LINE1>1145 Brokaw Road</POH_VENDOR_ADDRESS_LINE1>
  <POH_VENDOR_ADR_INFO>San Jose, CA 95034</POH_VENDOR_ADR_INFO>
  <POH_VENDOR_COUNTRY>United States</POH_VENDOR_COUNTRY>
  <POH_BILL_ADDRESS_LINE1>90 Fifth Avenue</POH_BILL_ADDRESS_LINE1>
  <POH_BILL_ADR_INFO>New York, NY 10022-3422</POH_BILL_ADR_INFO>
  <POH_BILL_COUNTRY>United States</POH_BILL_COUNTRY>
  <POH_BUYER>Smith, J</POH_BUYER>
  <POH_PAYMENT_TERMS>45 Net (terms date + 45)</POH_PAYMENT_TERMS>
  <POH_SHIP_VIA>UPS</POH_SHIP_VIA>
  <POH_FREIGHT_TERMS>Due</POH_FREIGHT_TERMS>
  <POH_CURRENCY_CODE>USD</POH_CURRENCY_CODE>
  <POH_CURRENCY_CONVERSION_RATE></POH_CURRENCY_CONVERSION_RATE>
  <LIST_G_LINES>
  <G_LINES>
    <POL_LINE_NUM>1</POL_LINE_NUM>
    <POL_VENDOR_PRODUCT_NUM></POL_VENDOR_PRODUCT_NUM>
    <POL_ITEM_DESCRIPTION>PCMCIA II Card Holder</POL_ITEM_DESCRIPTION>
    <POL_QUANTITY_TO_PRINT></POL_QUANTITY_TO_PRINT>
    <POL_UNIT_OF_MEASURE>Each</POL_UNIT_OF_MEASURE>
    <POL_PRICE_TO_PRINT>15</POL_PRICE_TO_PRINT>
    <C_FLEX_ITEM>CM16374</C_FLEX_ITEM>
    <C_FLEX_ITEM_DISP>CM16374</C_FLEX_ITEM_DISP>
    <PLL_QUANTITY_ORDERED>7500</PLL_QUANTITY_ORDERED>
    <C_AMOUNT_PLL>112500</C_AMOUNT_PLL>
    <C_AMOUNT_PLL_DISP>112,500.00 </C_AMOUNT_PLL_DISP>
  </G_LINES>
</LIST_G_LINES>
<C_AMT_POL_RELEASE_TOTAL_ROUND>312420</C_AMT_POL_RELEASE_TOTAL_ROUND>
</G_HEADERS>
</POXPRPOP2>

```

## Adding Markup to the Template

After you have converted your document to PDF, you define form fields that will display the data from the XML input file. These form fields are placeholders for the data.

The process of associating the XML data to the PDF template is the same as the process for the RTF template. See: Associating the XML data to the template, page 4-4.

When you draw the form fields in Adobe Acrobat, you are drawing them *on top* of the template that you designed. There is not a relationship between the design elements on your template and the form fields. You therefore must place the fields exactly where you want the data to display on the template.

## Creating a Placeholder

You can define a placeholder as text, a check box, or a radio button, depending on how you want the data presented.

**Note:** The steps for adding a form field depend on the version of Adobe Acrobat Professional that you are using. See the Adobe documentation for your version. If you are using Adobe Acrobat 9 Pro, from the **Forms** menu, select **Add or Edit Fields**.

### Naming the Placeholder

The name of the placeholder must match the XML source field name.

### Creating a Text Placeholder

The following describes how to create a text Form Field placeholder using Adobe Acrobat 9 Pro. If you are using a different version of Adobe Acrobat Professional, refer to the documentation for details.

1. From the **Forms** menu, select **Add or Edit Fields**.
2. From the **Add New Field** list, choose **Text Field**. The cursor becomes a crosshair.
3. Place the crosshair in the form where you want the field to reside and click. The **Field Name** dialog pops up.
4. Enter the name. The name of the text field must match the name of the XML element from your data that is to populate this field at runtime.
5. To set more properties, click **Show All Properties**  
Use the **Properties** dialog box to set other attributes for the placeholder. For example, enforce maximum character size, set field data type, data type validation, visibility, and formatting.
6. If the field is not placed exactly where desired, or is not the correct size, drag the field for exact placement and resize the field using the handles.
7. **IMPORTANT:** When you have added all your fields, you must make your template compatible with Adobe Acrobat 5.0. From the **Document** menu, select **Reduce File Size**. From the **Make Compatible with** list, choose **Adobe Acrobat 5.0 and later**.

### Supported Field Properties Options

BI Publisher supports the following options available from the **Field Properties** dialog box. For more information about these options, see the Adobe Acrobat documentation.

- **General**
  - Read Only
 

The setting of this check box in combination with a set of configuration properties control the read-only/updateable state of the field in the output PDF. See Setting Fields as Updateable or Read Only, page 7-16.
  - Required
  - Visible/Hidden
  - Orientation (in degrees)
- **Appearance**
  - Border Settings: color, background, width, and style
  - Text Settings: color, font, size
  - Border Style
- **Options tab**
  - Multi-line
  - Scrolling Text
- **Format tab** - Number category options only
- **Calculate tab** - all calculation functions

### Creating a Check Box

A check box is used to present options from which more than one can be selected. Each check box represents a different data element. You define the value that will cause the check box to display as "checked."

For example, a form contains a check box listing of automobile options such as Power Steering, Power Windows, and Sunroof. Each of these represents a different element from the XML file (for example <POWER\_STEERING>). If the XML file contains a value of "Y" for any of these fields, you want the check box to display as checked. All or none of these options may be selected.

The following describes how to create a check box field using Adobe Acrobat 9 Pro. If you are using a different version of Adobe Acrobat Professional, refer to the documentation for details.

1. From the **Forms** menu, select **Add or Edit Fields**.

2. From the **Add New Field** list, choose **Check Box**. The cursor becomes a crosshair.
3. Place the crosshair in the form where you want the field to reside and click. The **Field Name** dialog pops up.
4. Enter the name. The name of the check box field must match the name of the XML element from your data that is to determine its state (checked or unchecked).
5. Click **Show All Properties**
6. Click the **Options** tab.
7. Select the **Check Box Style** type from the list.
8. In the **Export Value** field enter the value that the XML data field should match to enable the "checked" state.  
For the example, enter "Y" for each check box field.
9. Set other **Properties** as desired.

#### **Creating a Radio Button Group**

A radio button group is used to display options from which only one can be selected.

For example, your XML data file contains a field called <SHIPMENT\_METHOD>. The possible values for this field are "Standard" or "Overnight". You represent this field in your form with two radio buttons, one labeled "Standard" and one labeled "Overnight". Define both radio button fields as placeholders for the <SHIPMENT\_METHOD> data field. For one field, define the "on" state when the value is "Standard". For the other, define the "on" state when the value is "Overnight".

The following describes how to create a radio button group using Adobe Acrobat 9 Pro. If you are using a different version of Adobe Acrobat Professional, refer to the documentation for details.

1. From the **Forms** menu, select **Add or Edit Fields**.
2. From the **Add New Field** list, choose **Radio Button**. The cursor becomes a crosshair.
3. Place the crosshair in the form where you want the radio button group to reside and click. The **Radio Group Name** dialog pops up.
4. Enter the name. The name of the radio group must match the name of the XML element from your data that is to determine its state (selected or unselected).
5. In the **Button Value** field enter the value that the XML data field should match to enable the "on" state.

For the example, enter "Standard" for the field labeled "Standard".

6. To enter another radio button to the group, click **Add another button to group**. The name of the radio group will default into the name field.
7. In the **Button Value** field enter the value that the XML data field should match to enable the "on" state for this button.  
For example, enter "Overnight" for the field labeled "Overnight".
8. If you wish to change any of the properties, click **Show All Properties**. To change the radio button style, click the **Options** tab.
9. Select **Radio Button** from the **Type** drop down list.
10. Set other **Properties** as desired.

## Defining Groups of Repeating Fields

In the PDF layout, you explicitly define the area on the page that will contain the repeating fields. For example, on the purchase order layout, the repeating fields should display in the block of space between the Item header row and the Total field.

### To define the area to contain the group of repeating fields:

1. Insert a Text Field at the beginning of the area that is to contain the group.
2. In the **Field Name** dialog, enter any unique name you choose. This field is not mapped.
3. In the **Tooltip** field of the **Text Field Properties** dialog, enter the following syntax:  
`<?rep_field="BODY_START"?>`
4. Define the end of the group area by inserting a Text Field at the end of the area that is to contain the group.
5. In the **Field Name** dialog, enter any unique name you choose. This field is not mapped. Note that the name you assign to this field must be different from the name you assigned to the "body start" field.
6. In the **Tooltip** field of the **Text Field Properties** dialog, enter the following syntax:  
`<?rep_field="BODY_END"?>`

### To define a group of repeating fields:

1. Insert a placeholder for the first element of the group.

**Note:** The placement of this field in relationship to the BODY\_START tag defines the distance between the repeating rows



for each occurrence. See Placement of Repeating Fields, page 7-15.

2. For each element in the group, enter the following syntax in the **Tooltip** field:

```
<?rep_field="T1_Gn"?>
```

where n is the row number of the item on the layout.

For example, the group in the sample report is laid out in three rows.

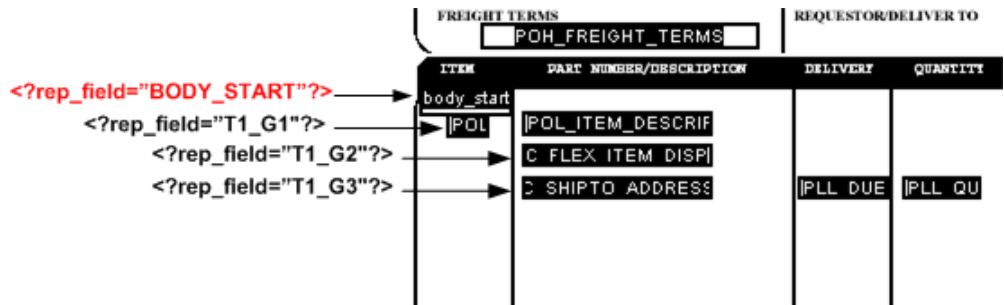
- For the fields belonging to the row that begins with "PO\_LINE\_NUM" enter  

```
<?rep_field="T1_G1"?>
```
- For the fields belonging to the row that begins with "C\_FLEX\_ITEM\_DISP"  
enter  

```
<?rep_field="T1_G2"?>
```
- For the fields belonging to the row that begins with "C\_SHIP\_TO\_ADDRESS"  
enter  

```
<?rep_field="T1_G3"?>
```

The following graphic shows the entries for the **Short Description/Tooltip** field:



3. (Optional) Align your fields. To ensure proper alignment of a row of fields, it is recommended that you use Adobe Acrobat's alignment feature.

## Adding Page Numbers

This section describes how to add the following page-features to your PDF layout:

- Page Numbers
- Page Breaks

## Adding Page Numbers

To add page numbers, define a field in the layout where you want the page number to appear and enter an initial value in that field as follows:

1. Decide the position on the layout where you want the page number to be displayed.
2. Create a placeholder field called @pagenum@ (see *Creating a Text Placeholder*, page 7-5).
3. Enter a starting value for the page number in the **Default** field (Text Field Properties > Options tab). If the XML data includes a value for this field, the start value assigned in the layout will be overridden. If no start value is assigned, it will default to 1.

## Adding Page Breaks

You can define a page break in your layout to occur after a repeatable field. To insert a page break after the occurrence of a specific field, add the following to the syntax in the **Tooltip** field of the **Text Field Properties** dialog:

```
page_break="yes"
```

For example:

```
<?rep_field="T1_G3", page_break="yes"?>
```

The following example demonstrates inserting a page break in a layout. The XML sample contains salaries of employees by department:

```

<?xml version="1.0"?>
<! - Generated by Oracle Reports version 6.0.8.22.0 - >
<ROOT>
  <LIST_G_DEPTNO>
    <G_DEPTNO>
      <DEPTNO>10</DEPTNO>
      <LIST_G_EMPNO>
        <G_EMPNO>
          <EMPNO>7782</EMPNO>
          <ENAME>CLARK</ENAME>
          <JOB>MANAGER</JOB>
          <SAL>2450</SAL>
        </G_EMPNO>
        <G_EMPNO>
          <EMPNO>7839</EMPNO>
          <ENAME>KING</ENAME>
          <JOB>PRESIDENT</JOB>
          <SAL>5000</SAL>
        </G_EMPNO>
        <G_EMPNO>
          <EMPNO>125</EMPNO>
          <ENAME>KANG</ENAME>
          <JOB>CLERK</JOB>
          <SAL>2000</SAL>
        </G_EMPNO>
        <G_EMPNO>
          <EMPNO>7934</EMPNO>
          <ENAME>MILLER</ENAME>
          <JOB>CLERK</JOB>
          <SAL>1300</SAL>
        </G_EMPNO>
        <G_EMPNO>
          <EMPNO>123</EMPNO>
          <ENAME>MARY</ENAME>
          <JOB>CLERK</JOB>
          <SAL>400</SAL>
        </G_EMPNO>
        <G_EMPNO>
          <EMPNO>124</EMPNO>
          <ENAME>TOM</ENAME>
          <JOB>CLERK</JOB>
          <SAL>3000</SAL>
        </G_EMPNO>
      </LIST_G_EMPNO>
      <SUMSALPERDEPTNO>9150</SUMSALPERDEPTNO>
    </G_DEPTNO>

    <G_DEPTNO>
      <DEPTNO>30</DEPTNO>
      <LIST_G_EMPNO>
        .
        .
        .

      </LIST_G_EMPNO>
      <SUMSALPERDEPTNO>9400</SUMSALPERDEPTNO>
    </G_DEPTNO>
  </LIST_G_DEPTNO>
  <SUMSALPERREPORT>29425</SUMSALPERREPORT>
</ROOT>

```

We want to report the salary information for each employee by department as shown in the following layout:

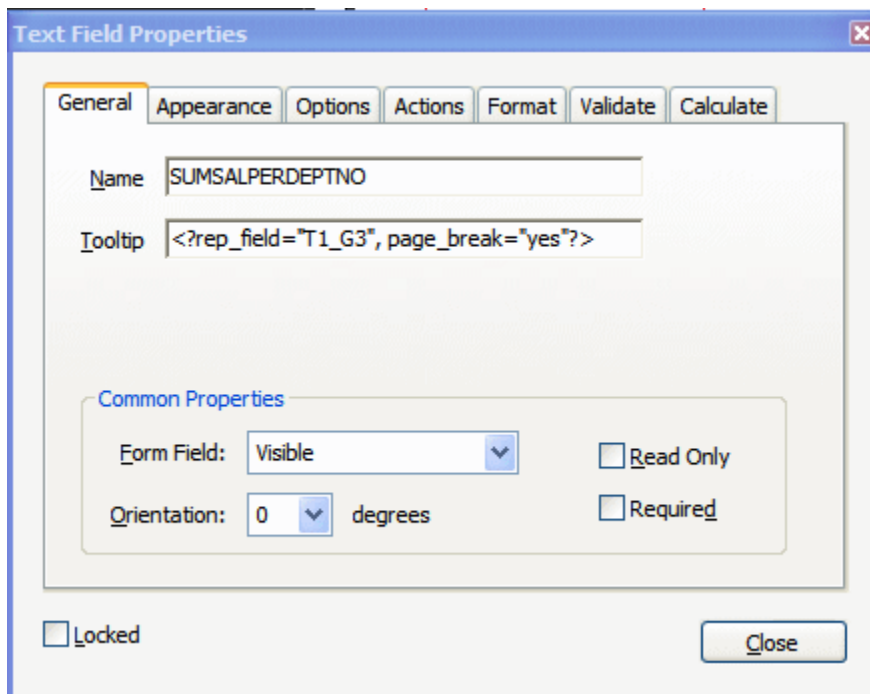
## Department Salary Summary

<code>body_start</code>	Dept No.	Emp No	Emp Name	Job	Salary
	<code>DEPTNO</code>	<code>EMPNO</code>	<code>ENAME</code>	<code>JOB</code>	<code>SAL</code>
					<code>SUMSALPERDEP</code>

To insert a page break after each department, insert the page break syntax in the Tooltip field for the SUMSALPERDEPTNO field as follows:

```
<?rep_field="T1_G3", page_break="yes"?>
```

The Field Properties dialog box for the field is shown in the following figure:



Note that in order for the break to occur, the field must be populated with data from the XML file.

The sample report with data is shown in the following figure:

## Department Salary Summary

Dept No.	Emp No	Emp Name	Job	Salary
10	7782	CLARK	MANAGER	2450
	7839	KING	PRESIDENT	5000
	125	KANG	CLERK	2000
	7934	MILLER	CLERK	1300
	123	MARY	CLERK	400
	124	TOM	CLERK	3000
				9150

## Department Salary Summary

Dept No.	Emp No	Emp Name	Job	Salary
20	7369	SMITH	CLERK	800
	7876	ADAMS	CLERK	1100
	7902	FORD	ANALYST	3000
	7788	SCOTT	ANALYST	3000
	7566	JONES	MANAGER	2975
				10875

The page breaks after each department.

## Performing Calculations

Adobe Acrobat provides a calculation function in the **Field Properties** dialog box. To

create a field to display a calculated total on your report:

1. Create a text field to display the calculated total. Give the field any **Name** you choose.
2. In the **Field Properties** dialog box, select the **Format** tab.
3. Select **Number** from the **Category** list.
4. Select the **Calculate** tab.
5. Select the radio button next to "Value is the <List of operations> of the following fields:"
6. Select **sum (+)** from the list.
7. Click the **Pick...** button and select the fields to be totaled.

## Completed PDF Layout Example

The following figure shows the completed PDF layout:

**ORACLE**  
E-Business Suite

**Purchase Order**

PURCHASE ORDER NO.	REVISES	PAGE
IPOH PO VL	IPOH	

**SHIP TO:**

IPOH SHIP ADDRESS
IPOH SHIP ADR INFO
IPOH SHIP COUNTRY

**BILL TO:**

IPOH BILL ADDRESS
IPOH BILL ADR INFO
IPOH BILL COUNTRY

**VENDOR:**

IPIH VENDOR NAME
IPOH VENDOR ADDRESS III
IPIH VENDOR ADR INFO
IPIH VENDOR COUNTRY

CUSTOMER ACCOUNT NO.	VENDOR NO.	DATE OF ORDER/BY	REVEAL DATE/BY
IPOH CUST	IPOH VE	IPOH CR IPOH ARI	
PAYMENT TERMS		SHIP VIA	FOB
POH PAYMENT TERMS		IPOH SHIP V	<input type="checkbox"/> POH FOB
FREIGHT TERMS		REQUEST/DELIVER TO	CONTACT TELEPHONE
POH FREIGHT TERMS			

ITEM	PART NUMBER/DESCRIPTION	DELIVERY	QUANTITY	UNIT	UNIT PRICE	EXTENSION	TAX
body_of	IPOH						
	IPOH ITEM DES						
	IC FLEX ITEM						
	IC SHIPTO ADR	IPII DI	IPII F			IC AMT	IPII
body							

<b>Total</b>		IC AMOUNT P
		AUTHORIZED SIGNATURE

Oracle E-Business Suite

## Runtime Behavior

### Placement of Repeating Fields

As already noted, the placement, spacing, and alignment of fields that you create on the layout are independent of the underlying form layout. At runtime, BI Publisher places each repeating row of data according to calculations performed on the placement of the rows of fields that you created, as follows:

**First occurrence:**

The first row of repeating fields will display exactly where you have placed them on the layout.

#### **Second occurrence, single row:**

To place the second occurrence of the group, BI Publisher calculates the distance between the BODY\_START tag and the first field of the first occurrence. The first field of the second occurrence of the group will be placed this calculated distance below the first occurrence.

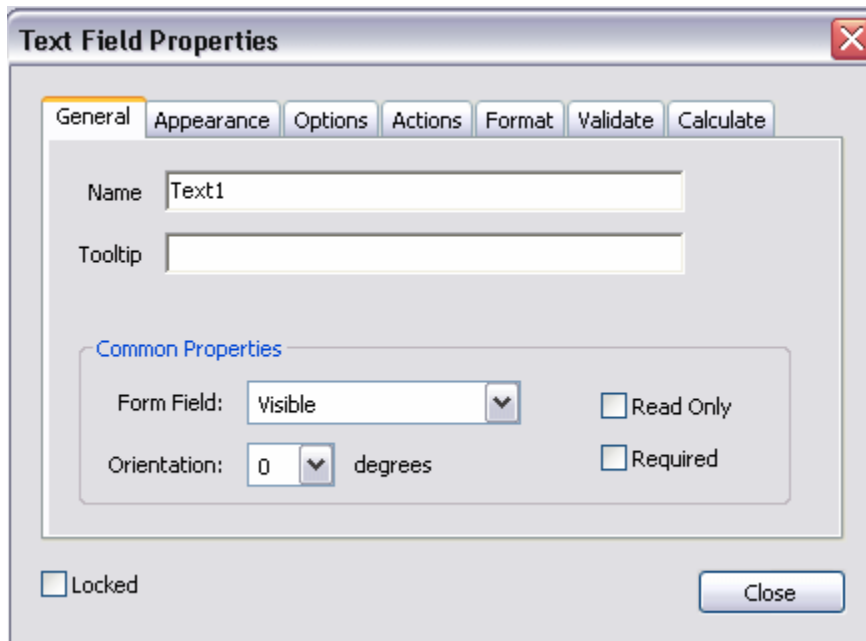
#### **Second occurrence, multiple rows:**

If the first group contains multiple rows, the second occurrence of the group will be placed the calculated distance below the last row of the first occurrence.

The distance between the rows within the group will be maintained as defined in the first occurrence.

## **Setting Fields as Updateable or Read Only**

When you define a field in the layout you have the option of selecting "Read Only" for the field, as shown in the following sample Text Field Properties dialog:



Regardless of what you choose at design time for the Read Only check box, the default behavior of the PDF processing engine is to set all fields to read-only for the output PDF. You can change this behavior using the following report properties (see Setting Report Properties, page 10-1):

- all-field-readonly



- all-fields-readonly-asis
- remove-pdf-fields

Note that in the first two options, you are setting a state for the field in the PDF output. The setting of individual fields can still be changed in the output using Adobe Acrobat Professional. Also note that because the fields are maintained, the data is still separate and can be extracted. In the third option, "remove-pdf-fields" the structure is flattened and no field/data separation is maintained.

**To make all fields updateable:**

Set the "all-field-readonly" property to "false". This sets the Read Only state to "false" for all fields regardless of the individual field settings at design time.

**To make all fields read only:**

This is the default behavior. No settings are required.

**To maintain the Read Only check box selection for each field:**

To maintain the setting of the Read Only check box on a field-by-field basis in the output PDF, set the property "all-fields-readonly-asis" to "true". This property will override the settings of "all-field-readonly".

**To remove all fields from the output PDF:**

Set the property "remove-pdf-fields" to "true".

## Overflow Data

When multiple pages are required to accommodate the occurrences of repeating rows of data, each page will display identically except for the defined repeating area, which will display the continuation of the repeating data. For example, if the item rows of the purchase order extend past the area defined on the layout, succeeding pages will display all data from the purchase order form with the continuation of the item rows.

## Creating a Layout from a Predefined PDF Form

There are many PDF forms available online that you may want to use as layouts for your report data. For example, government forms that your company is required to submit. You can use these downloaded PDF files as your report layouts, supplying the XML data at runtime to fill in the report fields.

Some of these forms already have form fields defined, some do not (see Determining If a PDF Has Form Fields Defined, page 7-18 if you are unsure). If the PDF form already has fields defined, you can use one of the following methods to match the form field names to the data field names:

- Use Adobe Acrobat Professional to rename the fields in the document to match the names of the elements in your XML data file. See Using a Predefined Form as a

Layout by Renaming the Form Fields, page 7-18.

- Use BI Publisher's Data Model Editor to rename the XML element names in your data file to match the field names in the PDF form. See Using the Structure View to Edit Your Data Structure, *Oracle Fusion Middleware Data Modeling Guide for Oracle Business Intelligence Publisher*.

If the form fields are not already defined in the downloaded PDF, you must create them. See Adding Markup to the Layout, page 7-4 for instructions on inserting the form field placeholders.

### Determining If a PDF Has Form Fields Defined

To determine if your PDF form has form fields defined:

1. Open your document in Adobe Acrobat Reader or Adobe Acrobat Professional.
2. Click **Highlight Fields**. Form fields that exist in the document will be highlighted.

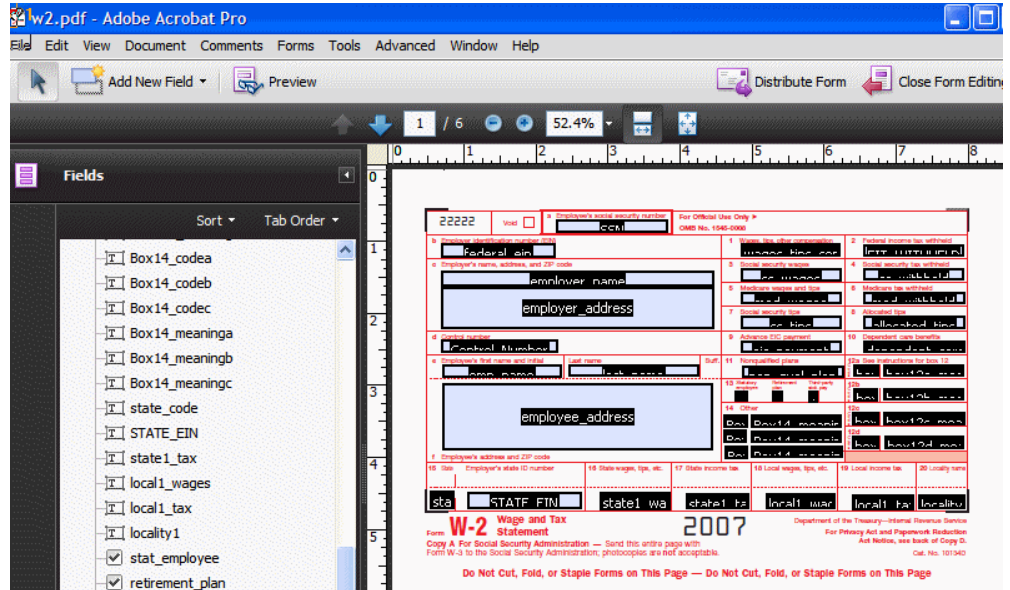
To get a list of the field names:

1. Open your document in Adobe Acrobat Professional.
2. From the **Form** menu, select **Add or Edit Fields**. The field names will display in the document as well as in the **Fields** pane.

### Using a Predefined PDF Form as a Layout by Renaming the Form Fields

1. Download or import the PDF file to your local system.
2. Open the file in Adobe Acrobat Professional.
3. From the **Form** menu, select **Add or Edit Fields**. This will highlight text fields that have already been defined.

The following figure shows a sample W-2 PDF form after selecting **Add or Edit Fields** to highlight the text fields.



To map the existing form fields to the data from your incoming XML file, rename the fields to match the element names in your XML file.

4. Open the form field **Text Field Properties** dialog by either double-clicking the field, or by selecting the field then selecting **Properties** from the right-mouse menu.
5. In the **Name** field, enter the element name from your input XML file.
6. Repeat for all fields that you want populated by your data file.
7. When all fields have been updated, click **Close Form Editing**.
8. Save your layout.
9. **IMPORTANT:** Make your layout compatible with Adobe Acrobat 5.0. From the **Document** menu, select **Reduce File Size**. From the **Make Compatible with** list, choose **Adobe Acrobat 5.0 and later**.

## Adding or Designating a Field for a Digital Signature

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

Implementing digital signature requires several tasks across the BI Publisher product. This topic describes how to add a new field or configure an existing field in your PDF template for the digital signature. For more information and a description of the other

required tasks and options, see *Implementing a Digital Signature, Oracle Fusion Middleware Administrator's Guide for Oracle Business Intelligence Publisher*.

## About Signature Field Options

For PDF templates you have the following options for designating a digital signature field for your output report:

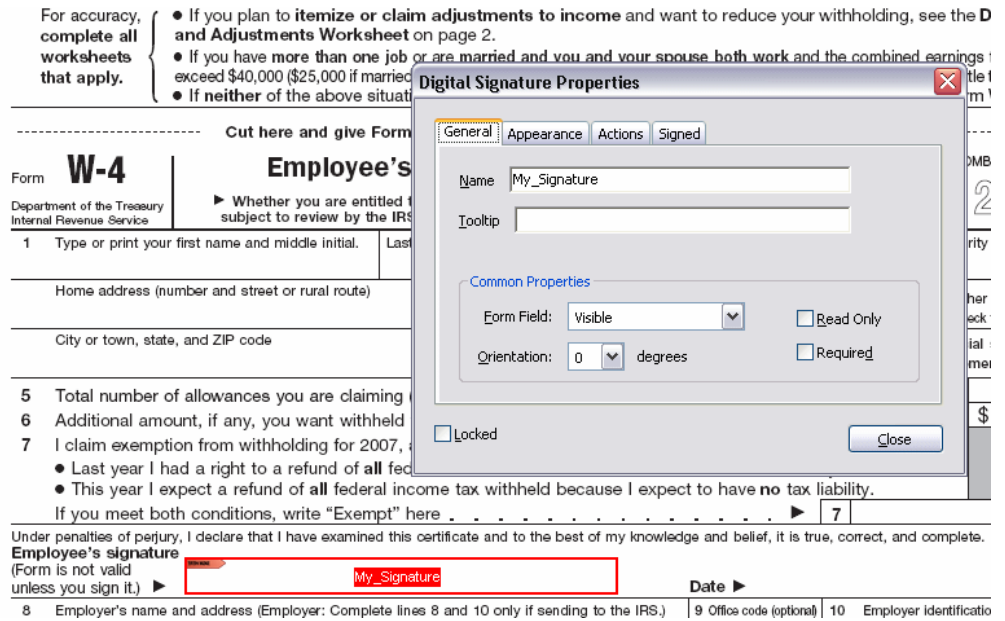
- Add a signature field to the PDF layout.  
Use this option if you want the digital signature to appear in a specific field and your PDF template does not already include a signature field. See *Adding a Signature Field*, page 7-20.
- Use an existing signature field in the PDF template.  
Use this option if your PDF template already includes a signature field that you want to use. To designate an existing field for the digital signature, define the field in the Runtime Configuration page. See *Configuring the Report to Insert the Digital Signature at Runtime*, page 7-21.
- Designate the position of the digital signature on the output report by setting x and y coordinates.  
Use this option if you prefer to designate the x and y coordinates for the placement of the digital signature, rather than use a signature field. You set the position using runtime properties. For information on setting these properties, see *PDF Digital Signature Properties*, page 10-7.

All three options require setting configuration properties for the report in the Report Properties page after you have uploaded the template.

## Adding a Signature Field

To add a signature field:

1. Open the template in Adobe Acrobat Professional.
2. From the **Form** menu, select **Add or Edit Fields**. Then click **Add New Field**. Choose **Digital Signature** from the list of fields.
3. Draw the signature field in the desired location on the layout. When you release the mouse button, a dialog will prompt you to enter a name for the field.
4. Enter a name for your signature field. The following figure shows an inserted digital signature field called "My\_Signature."



5. Save your template.
6. Proceed to Configuring the Report to Insert the Digital Signature at Runtime, page 7-21.

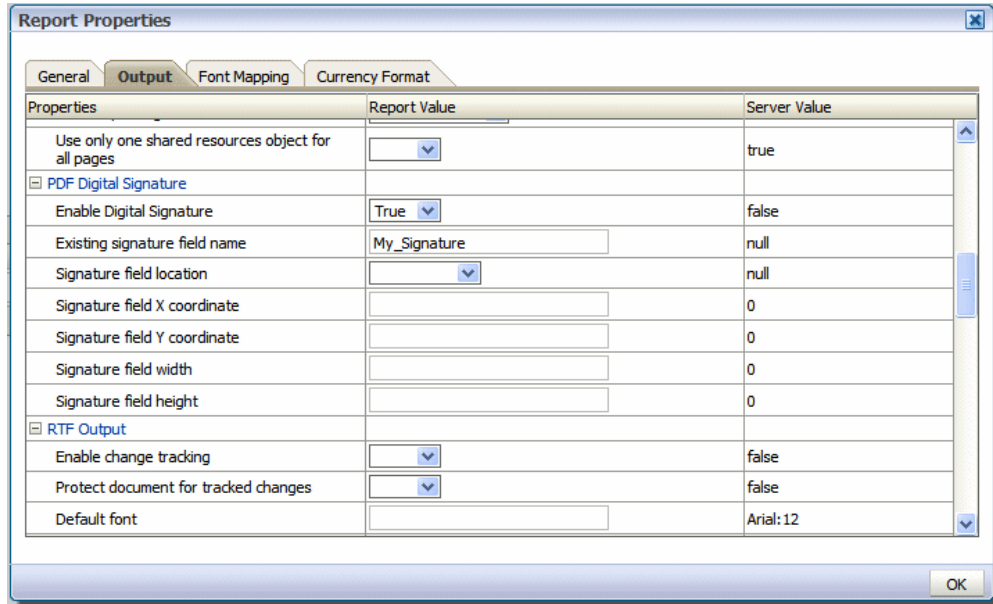
## Configuring the Report to Insert the Digital Signature at Runtime

After you have uploaded your PDF template to the report definition (see Adding a Layout, page 2-5) you must enable digital signature and specify the signature field in the Report Properties.

1. From the edit report page, click **Properties** and then click the **Formatting** tab.
2. Scroll to the **PDF Digital Signature** group of properties.
3. Set **Enable Digital Signature** to **True**.
4. For the property **Existing signature field name**, enter the field name from the PDF template.

No other properties are required for this method.

The following figure shows the "My\_Signature" field name entered into the properties field.



5. Click **OK**.

Note that the runtime properties that you have just set are at the report level and not the layout level. Therefore any layouts associated with the report will now include the digital signature as specified in the Report Properties. When an **Existing signature field name** is specified, the template must contain the field for the signature to be applied.

---

# Creating Flash Templates

This chapter covers the following topics:

- Introduction
- Building a Flash Template
- Uploading the Flash Template to the Report Definition
- Setting Properties for PDF Output
- For More Information

## Introduction

BI Publisher's support for Flash templates enables you to develop Adobe Flex templates that can be applied to BI Publisher reports to generate interactive Flash output documents.

**Note:** Adobe Flex is an open-source technology for building interactive cross-platform applications. Flex applications can be delivered using Adobe Flash Player. For more information see the Flex Web site at <http://www.flex.org>.

BI Publisher's integration with Flex enables you to build Flex templates, test them on your desktop, and deploy them to the BI Publisher server to generate Flash output. Users are then able to run the reports from the BI Publisher user interface or schedule them for delivery to report consumers.

This chapter will describe how to set up a Flex template with a BI Publisher "flat" data source (that is, there is no hierarchy in the XML data) and how to include simpler objects such as tables and charts. For more information about interactivity, connectivity between components and more advanced topics, refer to Adobe's Flex documentation.

## Prerequisites for Building and Viewing Flash Templates

Following are the prerequisites for building and viewing Flash templates:

- For viewing output:
  - To view the report output from the Flash Template, you must have Adobe Flash Player 9 installed on your computer. If viewing reports over the BI Publisher user interface, your Web browser must also support the Adobe Flash Player 9 plug-in.
- For building templates:
  - The FlexBuilder IDE from Adobe  
  
Oracle BI Publisher is currently certified with version 2.0.1. The tool can be downloaded and purchased from the Adobe Web site at <http://www.adobe.com/products/flex/>.  
  
Note that the charting functionality requires an additional license fee.
  - A report data model set up in BI Publisher that generates flat XML. For information on setting up your data model, see *Defining the Data Model, Oracle Fusion Middleware Data Modeling Guide for Oracle Business Intelligence Publisher*.

## Required Configuration Settings for Viewing PDF Output

Tightened security settings in the latest versions of Adobe Reader (9.3) disable multimedia content like Flash by default. Because of this change, when accessing PDF report output that contains embedded Flash objects, users might get an error message such as "Some features are disabled to avoid potential security risks."

To enable Flash content in PDF output modify the Flash configuration settings as follows:

1. In Adobe Acrobat, on the **Edit** menu, click **Preferences**.
2. In the **Preferences** dialog, from the **Categories** list, click **Multimedia Trust (legacy)**.
3. Select Display Permissions for: Other Documents.
4. Select "Permissions for Adobe Flash Player is set to Prompt".
5. Set "Change permission for selected multimedia player to" to "Always", then select all three check boxes below and click OK.



## Building a Flash Template

This section describes how to build a Flash template and includes the following topics:

- Adding the Data Source
- Creating the Layout
- Data Binding

### Adding the Data Source

To add the data source:

1. Generate a sample data file from your report data model as follows:

From the **Data Model Editor**, select the **Get XML Output** toolbar button. From the Report Viewer, select the number of rows to return and then click **Run**. From the **Actions** toolbar list, select **Export XML** and save the results as an XML file to a local directory.

This example is based on the following data:

```
<ROWSET>
<ROW>
<NAME>Neena Kochhar</NAME>
<FIRST_NAME>Neena</FIRST_NAME>
<LAST_NAME>Kochhar</LAST_NAME>
<SALARY>17000</SALARY>
<ANNUAL_SALARY>204000</ANNUAL_SALARY>
<FED_WITHHELD>57120</FED_WITHHELD>
<JOB_TITLE>Administration Vice President</JOB_TITLE>
<DEPARTMENT_NAME>Executive</DEPARTMENT_NAME>
<MANAGER>Steven King</MANAGER>
</ROW>
<ROW>
...
</ROWSET>
```

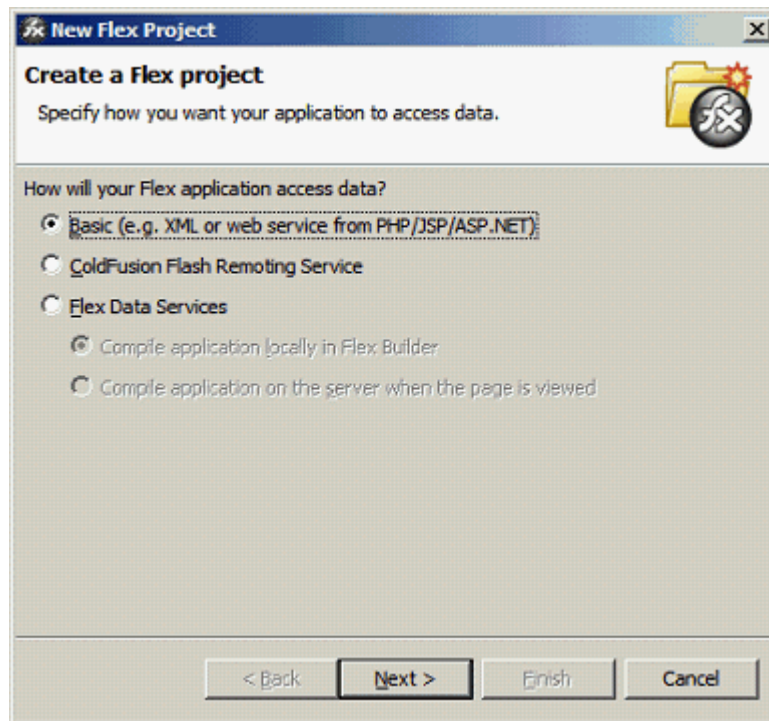
This data is generated from the following simple query-based report:

```

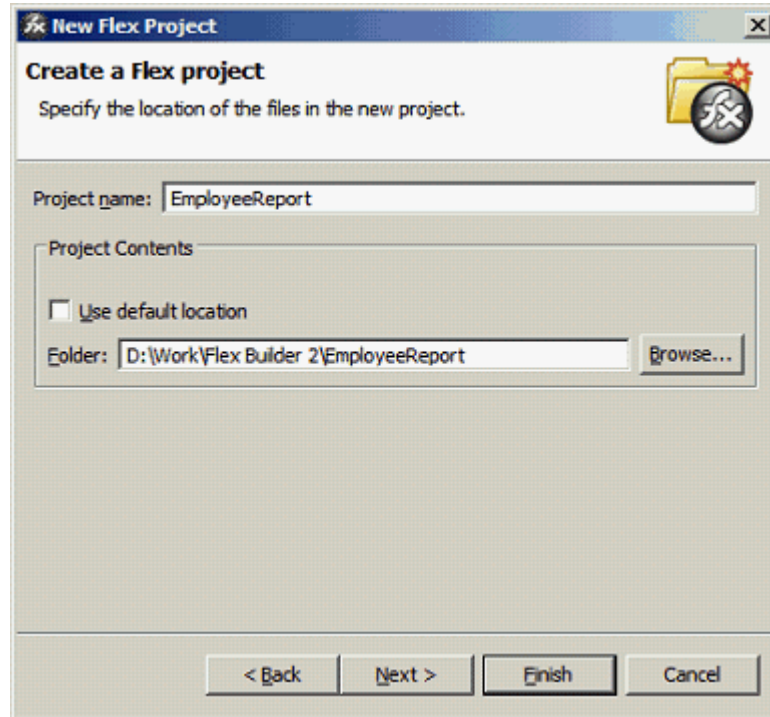
select
    e.first_name || ' ' || e.last_name name,
    e.first_name,
    e.last_name,
    e.salary,
    e.salary*12 ANNUAL_SALARY,
    e.salary*12*0.28 FED_WITHHELD,
    j.job_title,
    d.department_name,
    m.first_name || ' ' || m.last_name manager
from employees e,
     employees m,
     departments d,
     jobs j
where e.department_id = d.department_id
     and j.job_id = e.job_id
     and e.manager_id = m.employee_id

```

2. Open the Flex IDE and create a new Flex Project; select the "Basic" data access method, as shown in the following example.



In the next dialog, give the project a name as shown in the following example. The name you use here will be assigned to the template file name you are going to create.



Click **Finish**.

The IDE creates the Flex template definition file, which is an MXML file. An MXML file is an XML format. Following is a sample:

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml"
layout="absolute">
</mx:Application>
```

You can now update it manually or by using the visual builder.

3. Connect the XML you downloaded from your report data model:

To connect the data, use the XML data services that Flex supports and embed the sample data into the MXML file.

The sample MXML file with the connected data is shown. See the following section for a description of the file components.

```

<?xml version="1.0" encoding="utf-8"?>
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml"
layout="absolute">
  <mx:Script>
    <![CDATA[
      [Bindable]
      public var dataXML:XML =
<ROWSET>
<ROW>
<NAME>Neena Kochhar</NAME>
<FIRST_NAME>Neena</FIRST_NAME>
<LAST_NAME>Kochhar</LAST_NAME>
<SALARY>17000</SALARY>
<ANNUAL_SALARY>204000</ANNUAL_SALARY>
<FED_WITHHELD>57120</FED_WITHHELD>
<JOB_TITLE>Administration Vice President</JOB_TITLE>
<DEPARTMENT_NAME>Executive</DEPARTMENT_NAME>
<MANAGER>Steven King</MANAGER>
</ROW>
<ROW>
...
</ROWSET>;
    ]]>
  </mx:Script>
</mx:Application>

```

The XML portion should look familiar as the data you downloaded. The additional components to note are:

- `<mx:Script>` — This denotes the start of the template scripting code. There is also a closing `</mx:Script>` statement.
- `[Bindable]` — This denotes that the following variable is bindable to a layout component.
- `public var dataXML:XML` — This is the data variable declaration:
  - `public` — The value of the variable is available to the whole template.
  - `var` — Declares there is a variable in the report.
  - `dataXML` — The name of the variable. Note this is a compulsory name. You must use this name to use the template with BI Publisher.
  - `:XML` — Declares that the variable is an XML type.
- `;` — Notice the semicolon after the end of the XML data you provided.

At runtime the BI Publisher server will generate the runtime data from the report and inject it into the Flex template replacing the sample data held within the `dataXML` variable. This feature allows the Flex report to be distributed to users without needing to connect to the server.

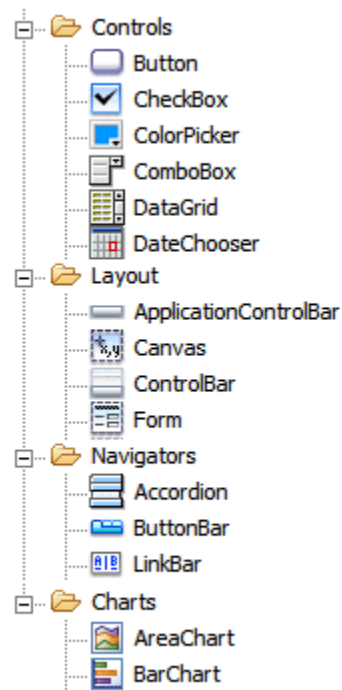
## Creating the Layout

The Flex IDE creates a default canvas for you to drop objects onto. You can modify the canvas as required to suit your report.

**Important:** If you intend to embed the Flash output in a PDF document, you must set the Width and Height of the template in the **Size** region of the **Layout** properties. Even if you wish to accept the default size, you must explicitly enter values in these fields.

Create the layout by adding report objects to the layout palette. This example uses the **Flex Design** tab to add the objects to the layout. Click the **Design** tab to see the available objects in the **Component Navigator** pane.

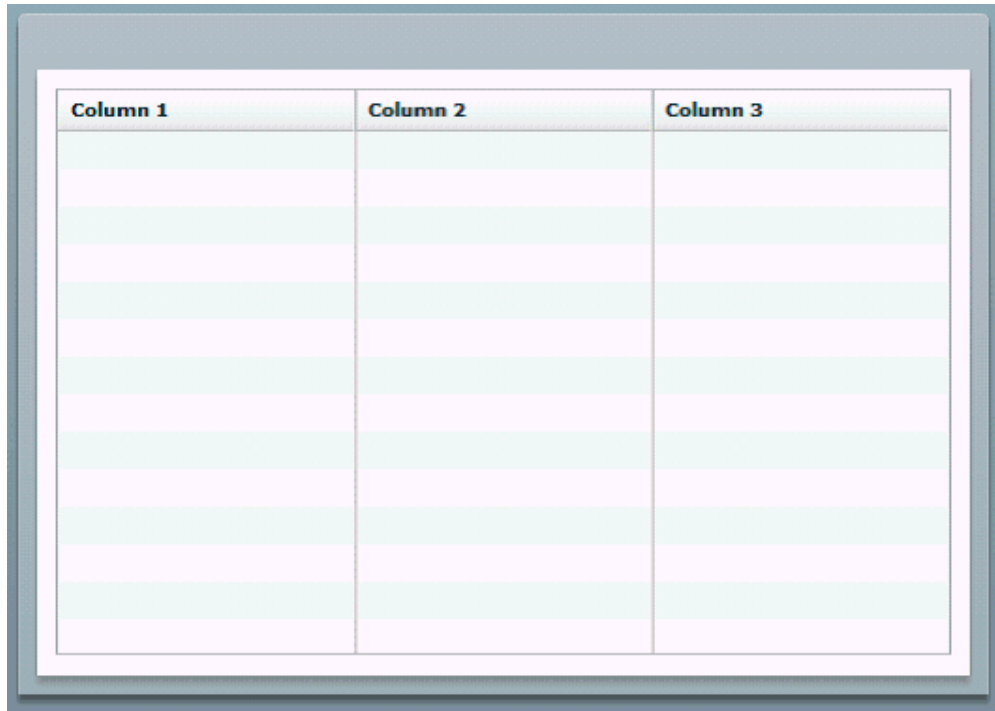
The following figure shows an example of the available objects in the Component Navigator pane:



These objects can be dragged and dropped to the design palette.

1. Start by dragging a **Panel** object from under the **Layout** node to the design palette. Notice as you drag the panel around the edge of the palette, the guidelines are displayed in blue. Use these guides to aid you in aligning objects.
2. Drop the panel onto the top left hand corner of the palette.

3. Now drag the bottom right edge of the panel across to the right hand side of the palette.
4. Then drag it down to about half the height of the palette. Alternatively, use the property palette on the right hand side to set the size of the panel.
5. Now select a **Datagrid** object. This is the object to render the data in a tabular format. Drop it onto the panel you created in Step 1. The Datagrid is now a child of the panel; you can resize it as needed. The end result is shown in the following figure:



Column 1	Column 2	Column 3

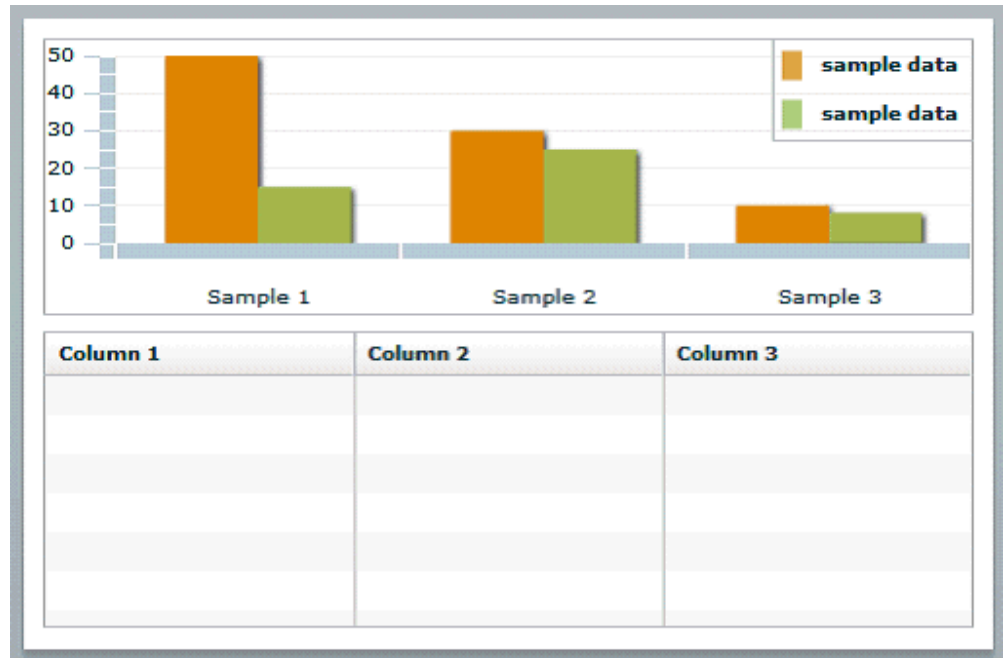
By default three columns are generated. In the next section, *Binding the Layout Objects to the Data Source*, page 8-9, you will override the default in the MXML code.

## Adding a Chart

If you have purchased the charting option you can add charts to your layout.

1. First make some room for the chart in your layout. Highlight the Datagrid object and pull the top edge down to about half the height of the hosting panel.
2. For this example, select and drag a Column Chart from the design palette and drop it onto the hosting panel. Use the guidelines to align it.
3. Once you drop it, notice that the default size overlaps the Datagrid and that the

chart legend is in the top left-hand corner. Resize the chart and move the legend to the right to look similar to the following figure:



This is a sample chart. You will bind it to the data in the next section.

## Binding the Layout Objects to the Data Source

Now that the layout is complete, bind the layout objects to the data source. Flex offers some help through the property palette of the objects to define the binding, but not enough to complete the task. Therefore you must update the MXML directly using the **Source** editor.

### Binding the DataGrid

To bind the DataGrid:

1. Start by highlighting the DataGrid in the design palette, and then click the **Source** tab to display the MXML source. You will see that the first line of the DataGrid code has been highlighted for you. This is a useful feature if you have built complex Flex templates and need to locate the code easily.

The DataGrid code is as follows:

```
<mx:DataGrid x="10" y="160" width="476" height="152">
  <mx:columns>
    <mx:DataGridColumn headerText="Column 1" dataField="col1"/>
    <mx:DataGridColumn headerText="Column 2" dataField="col2"/>
    <mx:DataGridColumn headerText="Column 3" dataField="col3"/>
  </mx:columns>
</mx:DataGrid>
```

Notice that the code defines the relative x,y position of the grid within its parent container and its width and height. The next element defines the columns with attributes for the header label and the data fields.

The goal is to achieve a table that looks like the following figure:

Employee	Title	Monthly Salary	Annual Salary
Neena Kochhar	Administration Vice	17000	204000
Lex De Haan	Administration Vice	17000	204000
Alexander Hunold	Programmer	9000	108000
Bruce Ernst	Programmer	6000	72000

2. Make the DataGrid aware of the data source by adding an attribute to the `<mx:DataGrid>` element as follows:

```
dataProvider="{dataXML.ROW}"
```

This attribute defines the data object to be used at runtime to populate the grid. Remember that in this example, the XML data variable was defined as "dataXML"; now use that definition followed by "ROW" (that is, dataXML.ROW). ROW is the repeating group in the data set. Note that the syntax requires the curly braces to let the Flex engine know it is a data source.

3. Bind the columns. In the basic structure provided, replace the values for `dataField` with the appropriate element name from your data source. Also replace `headerText` values with the desired column heading names. For example, for the first column, replace

```
<mx:DataGridColumn headerText="Column 1" dataField="col1"/>
```

with

```
<mx:DataGridColumn headerText="Employee" dataField="NAME" />
```

This defines the first column header name as "Employee" and binds the column data to the "NAME" element in the XML data source.

The completed DataGrid sample code follows:



```

<mx:DataGrid x="10" y="160" width="476" height="152"
dataProvider="{dataXML.ROW}">
  <mx:columns>
    <mx:DataGridColumn headerText="Employee" dataField="NAME" />
    <mx:DataGridColumn headerText="Title" dataField="JOB_TITLE"/>
    <mx:DataGridColumn headerText="Monthly Salary"
dataField="SALARY"/>
    <mx:DataGridColumn headerText="Annual Salary"
dataField="ANNUAL_SALARY"/>
  </mx:columns>
</mx:DataGrid>

```

4. You can now preview the template with your sample data. Select Run, then Run EmployeeReport. This will open a new browser window and render the table with your sample data.

## Binding the Chart

To bind the chart:

1. From the **Design** tab, highlight the chart. Next, switch back to the **Source** view to find the chart code:

```

<mx:ColumnChart x="10" y="10" id="columnchart1" width="476"
height="142">
  <mx:series>
    <mx:ColumnSeries displayName="Series 1" yField="" />
  </mx:series>
</mx:ColumnChart>
<mx:Legend dataProvider="{columnchart1}" x="383" y="10"/>

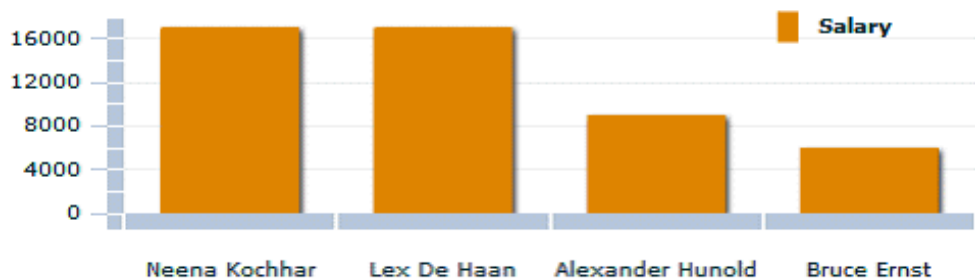
```

2. To bind the data source to the chart object, add the dataProvider attribute to the <mx:ColumnChart> element as follows:

```
dataProvider="{dataXML.ROW}"
```

3. Next add in the binding for the horizontal axis and the column series. Refer to the Flex help files for more details.

To create a chart showing salary by employee, similar to the following example:



make the following updates to the code:

- Add a <horizontalAxis> element to define the element from the data source

that will be used for the horizontal axis of the chart. Use the `categoryField` attribute to assign the data element value. In this example, the data element `NAME` is assigned.

- Modify the `<series>` group to bind the `SALARY` value to each employee `NAME` to create a bar for each employee.

Following is the sample code:

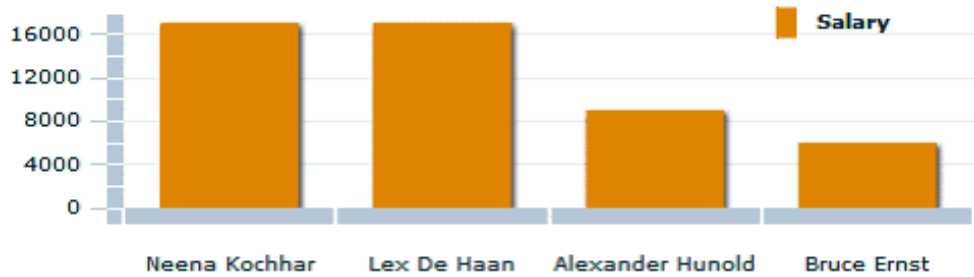
```
<mx:ColumnChart x="10" y="10" id="columnchart1" width="476"
height="142" dataProvider="{dataXML.ROW}">
  <mx:horizontalAxis>
    <mx:CategoryAxis categoryField="NAME" />
  </mx:horizontalAxis>
  <mx:series >
    <mx:ColumnSeries xField="NAME" yField="SALARY"
displayName="Salary"/>
  </mx:series>
</mx:ColumnChart>
<mx:Legend dataProvider="{columnchart1}" x="383" y="10"/>
```

Note in the preceding sample, the `<mx:horizontalAxis>` element has been added and the `categoryField` attribute has the `NAME` data element assigned. This element is required to render the chart.

The `<mx:series>` element has been updated binding the `SALARY` value to each employee `NAME` to create a bar for each employee.

You do not need to update the legend code. Notice the `id` attribute of the `<mx:ColumnChart>` element matches the `dataProvider` attribute value of the `<mx:Legend>` element.

4. You can now run the template using your sample data. You should get an output showing the chart above the tabulated data as shown in the following figure:



Employee	Title	Monthly Salary	Annual Salary
Neena Kochhar	Administration Vice	17000	204000
Lex De Haan	Administration Vice	17000	204000
Alexander Hunold	Programmer	9000	108000
Bruce Ernst	Programmer	6000	72000

## Uploading the Flash Template to the Report Definition

To upload the template to your report definition:

1. Navigate to your report in the catalog. Click **Edit** to launch the **Report Editor**.
2. Click **Add New Layout**.
3. Under **Upload or Generate New Layout**, click **Upload**.
4. In the **Upload Template File** dialog:
  - Enter a **Layout Name**.
  - Click **Browse** and navigate to the Flex project directory. Under this directory open the bin directory and select the EmployeeReport.swf file.
  - From the **Template Type** list, select Flash Template.
  - Select the **Locale** for this template.
5. Click **Upload** to add the Flash template to the available layouts for the report.

## Setting Properties for PDF Output

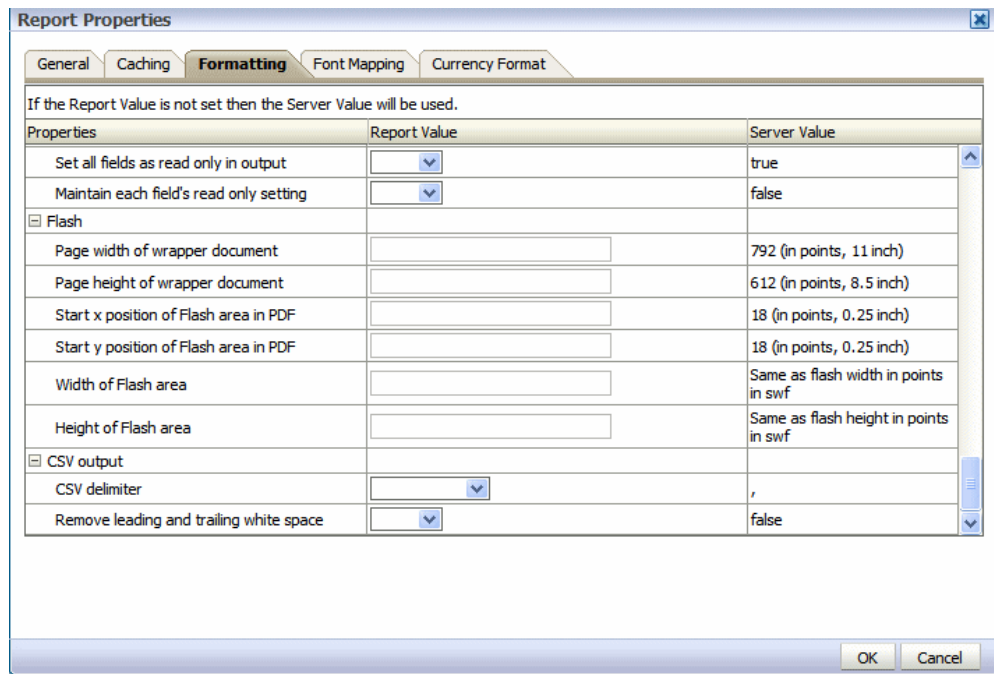
The Report Properties page includes a set of properties specific to rendering Flash

templates. These properties enable you to specify the size and placement of the Flash object when you select PDF as the output type.

**Important:** To produce PDF output you must specify the height and width of the template in the Flex Builder. See *Creating the Layout*, page 8-7.

To set properties for the PDF output:

1. Navigate to your report in the catalog. Click **Edit** to launch the **Report Editor**.
2. In the Report Editor, click **Properties** to open the **Report Properties** dialog.
3. Click the **Formatting** tab and scroll down to the set of properties under the **Flash** heading.



4. Enter values for the properties. Note that no properties are required. If you do not enter any values, the default values assume an 11 inch by 8.5 inch document (standard landscape), with a quarter inch inset from the upper left corner of the page as the insertion point of the Flash object. The default area in the document will be the size of the SWF object.
  - **Page width of wrapper document** – specify in points the width of the output PDF document. The default is 792, or 11 inches.
  - **Page height of wrapper document** – specify in points the height of the output

PDF document. The default is 612, or 8.5 inches.

- **Start x position of Flash area in PDF** – using the left edge of the document as the 0 axis point, specify in points the beginning horizontal position of the Flash object in the PDF document. The default is 18, or .25 inch.
- **Start y position of Flash area in PDF** – using the upper left corner of the document as the 0 axis point, specify in points the beginning vertical position of the Flash object in the PDF document. The default is 18, or .25 inch.
- **Width of Flash area** – enter in points the width of the area in the document for the Flash object to occupy. The default is the width of the SWF object.
- **Height of Flash area** – enter in points the height of the area in the document for the Flash object to occupy. The default is the height of the SWF object.

## For More Information

This chapter demonstrated how to build a simple Flex template, but Adobe Flex allows you to build far more complex interactive reports for your users. The animation, "wiring" together and formatting of layout objects can be achieved with Flex. You can also summarize and create calculated fields on the incoming data. Please reference the Flex documentation for these more advanced features.



---

# Creating eText Templates

This chapter covers the following topics:

- Introduction
- Structure of eText Templates
- Constructing the Data Tables
- Setup Command Tables
- Creating a Filler Block
- Expressions, Control Structures, and Functions
- Identifiers, Operators, and Literals

## Introduction

An eText template is an RTF-based template that is used to generate text output for Electronic Funds Transfer (EFT) and Electronic Data Interchange (EDI). At runtime, BI Publisher applies this template to an input XML data file to create an output text file that can be transmitted to a bank or other customer. Because the output is intended for electronic communication, the eText templates must follow very specific format instructions for exact placement of data.

**Note:** An EFT is an electronic transmission of financial data and payments to banks in a specific fixed-position format flat file (text).

EDI is similar to EFT except it is not only limited to the transmission of payment information to banks. It is often used as a method of exchanging business documents, such as purchase orders and invoices, between companies. EDI data is delimiter-based, and also transmitted as a flat file (text).

Files in these formats are transmitted as flat files, rather than printed on paper. The length of a record is often several hundred characters and therefore difficult to layout

on standard size paper.

To accommodate the record length, the EFT and EDI templates are designed using tables. Each record is represented by a table. Each row in a table corresponds to a field in a record. The columns of the table specify the position, length, and value of the field.

These formats can also require special handling of the data from the input XML file. This special handling can be on a global level (for example, character replacement and sequencing) or on a record level (for example, sorting). Commands to perform these functions are declared in command rows. Global level commands are declared in setup tables.

At runtime, BI Publisher constructs the output file according to the setup commands and layout specifications in the tables.

## Prerequisites

This section is intended for users who are familiar with EDI and EFT transactions. Preparers of eText templates will require both functional and technical knowledge; that is, functional expertise to understand bank and country specific payment format requirements and sufficient technical expertise to understand XML data structure and eText specific coding syntax commands, functions, and operations.

## Structure of eText Templates

There are two types of eText templates: fixed-position based (EFT templates) and delimiter-based (EDI templates). The templates are composed of a series of tables. The tables define layout and setup commands and data field definitions. The required data description columns for the two types of templates vary, but the commands and functions available are the same. A table can contain just commands, or it can contain commands and data fields.

The following graphic shows a sample from an EFT template to display the general structure of command and data rows:



**Format Setup:**

*Hint: Define formatting options...*

<TEMPLATE TYPE>	FIXED POSITION BASED
<OUTPUT CHARACTER SET>	iso-8859-1
<NEW RECORD CHARACTER>	Carriage Return

**Sequences:**

*Hint: Define sequence generators...*

<DEFINE SEQUENCE>	PaymentsSeq
<RESET AT LEVEL>	PayerInstrument
<INCREMENT BASIS>	LEVEL
<END DEFINE SEQUENCE >	PaymentsSeq

} Commands

**Format Data Records:**

<LEVEL>	<POSITION>	<LENGTH>	PayerInstrument	<FORMAT>	<PAD>	<DATA>	<COMMENTS>
<NEW RECORD>			FileHeaderRec				
1	1	1	Number			1	Record Type Code
2	2	2	Alpha	R, \ \ \		'01'	Priority Code
4	1	1	Alpha	R, \ \ \			Immediate Origin
5	9	9	Alpha	R, \ \ \		BankAccount/BankNumber	Immediate Origin
14	1	1	Alpha	R, \ \ \		'1'	Mutually Agreed
15	9	9	Alpha	R, \ \ \		Payer/TaxIdentifier	Immediate Origin
24	6	6	Date			SYSDATE	File Creation Date

} Data Elements or Functions

Commands that apply globally, or commands that define program elements for the template, are "setup" commands. These must be specified in the initial tables of the template. Examples of setup commands are Template Type and Character Set.

In the data tables you provide the source XML data element name and the specific placement and formatting definitions required by the receiving bank or entity. You can also define functions to be performed on the data and conditional statements.

The data tables must always start with a command row that defines the "Level." The Level associates the table to an element from the XML data file, and establishes the hierarchy. The data fields that are then defined in the table for the Level correspond to the child elements of the XML element.

The following figure illustrates the relationship between the XML data hierarchy and the template Level. The XML element "RequestHeader" is defined as the Level. The data elements defined in the table ("FileID" and "Encryption") are children of the RequestHeader element.

```

?xml version = "1.0" ?>
!DOCTYPE OutBoundPayments:BatchRequest SYSTEM "OutBoundPayments.dtd">
OutBoundPayments:BatchRequest>
  <PaymentsCommon:RequestHeader>
    <PaymentsCommon:Preamble>
      <PaymentsCommon:Version>1.0.11.5.7</PaymentsCommon:Version>
    </PaymentsCommon:Preamble>
    <PaymentsCommon:TrxnParties>
      <PaymentsCommon:RequesterParty>Oracle Germany</PaymentsCommon:RequesterParty>
      <PaymentsCommon:FinancialInstitution>Cit1 Bank</PaymentsCommon:FinancialInstitution>
    </PaymentsCommon:TrxnParties>
    <PaymentsCommon:Encryption>N</PaymentsCommon:Encryption>
    <PaymentsCommon:FileID>10180300001</PaymentsCommon:FileID>
    <PaymentsCommon:BaseRecordCount>94457</PaymentsCommon:BaseRecordCount>
  </PaymentsCommon:RequestHeader>
  <OutBoundPayments:BatchRequest>
    <OutBoundPayment>
      <OutBoundPayment>
      <OutBoundPayment>
      <OutBoundPayment>
      <OutBoundPayment>
      <OutBoundPayment>
      ...
  </OutBoundPayments:BatchRequest>

```

<LEVEL>		RequestHeader			
<POSITION>	<LENGTH>	<FORMAT>	<PAD>	<DATA>	
<NEW RECORD>		FileHeaderRec			
1	3	Alpha		FileID	
4	4	Number	L, '0'	FileID	
8	1	Alpha	R, ' '	Encryption	
9	6	Date, YYMMDD		SYSDATE	
15	6	Number	L, '0'	SEQUENCE NUMBER (AllRecordsSeq)	
21	129	Alpha	' '		

<LEVEL>		Batch			
<POSITION>	<LENGTH>	<FORMAT>	<PAD>	<DATA>	

The order of the tables in the template determines the print order of the records. At runtime the system loops through all the instances of the XML element corresponding to a table (Level) and prints the records belonging to the table. The system then moves on to the next table in the template. If tables are nested, the system will generate the nested records of the child tables before moving on to the next parent instance.

### Command Rows, Data Rows, and Data Column Header Rows

The following figure shows the placement of Command Rows, Data Rows, and Data Column Header Rows:

XDO file name:  
APXNACHA.rtf

Mapping of Payment Format:  
**US NACHA Payments EFT Format**

**Format Setup:**

*Hint: Define formatting options...*

<TEMPLATE TYPE>	FIXED POSITION BASED
<CHARACTER SET>	iso-8859-1
<NEW RECORD CHARACTER>	Carriage Return

**Sequences:**

*Hint: Define sequence generators...*

<DEFINE SEQUENCE>	PaymentsSeq
<RESET AT LEVEL>	PayerInstrument
<INCREMENT BASIS>	LEVEL
<END DEFINE SEQUENCE >	PaymentsSeq

**Format Data Records:**

<LEVEL>	<POSITION>	<LENGTH>	PayerInstrument	<FORMAT>	<PAD>	<DATA>	<COMMENT>
<NEW RECORD>			FileHeaderRec				
1	1	1	Number			1	Record Type
2	2	2	Alpha	R, \ \ \		01	Priority Code
4	1	1	Alpha	R, \ \ \			Immediate D blank
5	9	9	Alpha	R, \ \ \		BankAccount/BankNumber	Immediate D
14	1	1	Alpha	R, \ \ \		1	Mutually Agre
15	9	9	Alpha	R, \ \ \		Payer/TaxIdentifier	Immediate O
24	6	6	Date,			SYSDATE	File Creation

Command rows are used to specify commands in the template. Command rows always have two columns: command name and command parameter. Command rows do not have column headings. The commands control the overall setup and record structures of the template.

Blank rows can be inserted anywhere in a table to improve readability. Most often they are used in the setup table, between commands. Blank rows are ignored by BI Publisher when the template is parsed.

**Data Column Header Rows**

Data column headers specify the column headings for the data fields (such as Position, Length, Format, Padding, and Comments). A column header row usually follows the Level command in a table (or the sorting command, if one is used). The data column header row must come before any data rows in the table. Additional empty column header rows can be inserted at any position in a table to improve readability. The empty rows will be ignored at runtime.

The required data column header rows vary depending on the template type. See Structure of the Data Row, page 9-12.

**Data Rows**

Data rows provide the values for the data column header attributes for each data field. The content of the data rows varies depending on the template type. See Structure of

the Data Row, page 9-12.

## Constructing the Data Tables

The data tables contain a combination of command rows and data field rows. Each data table must begin with a Level command row that specifies its XML element. Each record must begin with a New Record command that specifies the start of a new record, and the end of a previous record (if any).

The required columns for the data fields vary depending on the Template Type.

### Command Rows

The command rows always have two columns: command name and command parameter. The supported commands are:

- Level
- New record
- Sort ascending
- Sort descending
- Display condition

The usage for each of these commands is described in the following sections.

#### Level Command

The level command associates a table with an XML element. The parameter for the level command is an XML element. The level will be printed once for each instance the XML element appears in the data input file.

The level commands define the hierarchy of the template. For example, Payment XML data extracts are hierarchical. A batch can have multiple child payments, and a payment can have multiple child invoices. This hierarchy is represented in XML as nested child elements within a parent element. By associating the tables with XML elements through the level command, the tables will also have the same hierarchical structure.

Similar to the closing tag of an XML element, the level command has a companion end-level command. The child tables must be defined between the level and end-level commands of the table defined for the parent element.

An XML element can be associated with only one level. All the records belonging to a level must reside in the table of that level or within a nested table belonging to that level. The end-level command will be specified at the end of the final table.

Following is a sample structure of an EFT file record layout:

- FileHeaderRecordA
  - BatchHeaderRecordA
  - BatchHeaderRecordB
  - PaymentRecordA
  - PaymentRecordB
  - InvoiceRecordA
  - Batch FooterRecordC
  - BatchFooterRecordD
  
- FileFooterRecordB

Following would be its table layout:

---

<LEVEL>	<b>RequestHeader</b>
<NEW RECORD>	FileHeaderRecordA
Data rows for the FileHeaderRecordA	

---

<LEVEL>	<b>Batch</b>
<NEW RECORD>	BatchHeaderRecordA
Data rows for the BatchHeaderRecordA	
<NEW RECORD>	BatchHeaderRecordB
Data rows for the BatchHeaderRecordB	

---

<LEVEL>	<b>Payment</b>
<NEW RECORD>	PaymentRecordA
Data rows for the PaymentRecordA	

---

---

<NEW RECORD> PaymentRecordB

Data rows for the PaymentRecordB

---

---

<LEVEL> **Invoice**

<NEW RECORD> InvoiceRecordA

Data rows for the InvoiceRecordA

<END LEVEL> **Invoice**

---

---

<END LEVEL> **Payment**

---

---

<LEVEL> **Batch**

<NEW RECORD> BatchFooterRecordC

Data rows for the BatchFooterRecordC

<NEW RECORD> BatchFooterRecordD

Data rows for the BatchFooterRecordD

<END LEVEL> **Batch**

---

---

<LEVEL> **RequestHeader**

<NEW RECORD> FileFooterRecordB

Data rows for the FileFooterRecordB

<END LEVEL> **RequestHeader**

---

Multiple records for the same level can exist in the same table. However, each table can

only have one level defined. In the example above, the BatchHeaderRecordA and BatchHeaderRecordB are both defined in the same table. However, note that the END LEVEL for the Payment must be defined in its own separate table after the child element Invoice. The Payment END LEVEL cannot reside in the same table as the Invoice Level.

Note that you do not have to use all the levels from the data extract in your template. For example, if an extract contains the levels: RequestHeader > Batch > Payment > Invoice, you can use just the batch and invoice levels. However, the hierarchy of the levels must be maintained.

The table hierarchy determines the order that the records are printed. For each parent XML element, the records of the corresponding parent table are printed in the order they appear in the table. The system loops through the instances of the child XML elements corresponding to the child tables and prints the child records according to their specified order. The system then prints the records of the enclosing (end-level) parent table, if any.

For example, given the EFT template structure above, assume the input data file contains the following:

- Batch1
  - Payment1
    - Invoice1
    - Invoice2
  - Payment2
    - Invoice1
- Batch2
  - Payment1
    - Invoice1
    - Invoice2
    - Invoice3

This will generate the following printed records:

<b>Record Order</b>	<b>Record Type</b>	<b>Description</b>
1	FileHeaderRecordA	One header record for the EFT file
2	BatchHeaderRecordA	For Batch1
3	BatchHeaderRecordB	For Batch1
4	PaymentRecordA	For Batch1, Payment1
5	PaymentRecordB	For Batch1, Payment1
6	InvoiceRecordA	For Batch1, Payment1, Invoice1
7	InvoiceRecordA	For Batch1, Payment1, Invoice2
8	PaymentRecordA	For Batch1, Payment2
9	PaymentrecordB	For Batch1, Payment2
10	InvoiceRecordA	For Batch1, Payment2, Invoice1
11	BatchFooterRecordC	For Batch1
12	BatchFooterRecordD	For Batch1
13	BatchHeaderRecordA	For Batch2
14	BatchHeaderRecordB	For Batch2
15	PaymentRecordA	For Batch2, Payment1
16	PaymentRecordB	For Batch2, Payment1
17	InvoiceRecordA	For Batch2, Payment1, Invoice1
18	InvoiceRecordA	For Batch2, Payment1, Invoice2



Record Order	Record Type	Description
19	InvoiceRecordA	For Batch2, Payment1, Invoice3
20	BatchFooterRecordC	For Batch2
21	BatchFooterRecordD	For Batch2
22	FileFooterRecordB	One footer record for the EFT file

### New Record Command

The new record command signifies the start of a record and the end of the previous one, if any. Every record in a template must start with the new record command. The record continues until the next new record command, or until the end of the table or the end of the level command.

A record is a construct for the organization of the elements belonging to a level. The record name is not associated with the XML input file.

A table can contain multiple records, and therefore multiple new record commands. All the records in a table are at the same hierarchy level. They will be printed in the order in which they are specified in the table.

The new record command can have a name as its parameter. This name becomes the name for the record. The record name is also referred to as the record type. The name can be used in the COUNT function for counting the generated instances of the record. See COUNT, page 9-32 function, for more information.

Consecutive new record commands (or empty records) are not allowed.

### Sort Ascending and Sort Descending Commands

Use the sort ascending and sort descending commands to sort the instances of a level. Enter the elements you wish to sort by in a comma-separated list. This is an optional command. When used, it must come right after the (first) level command and it applies to all records of the level, even if the records are specified in multiple tables.

### Display Condition Command

The display condition command specifies when the enclosed record or data field group should be displayed. The command parameter is a boolean expression. When it evaluates to true, the record or data field group is displayed. Otherwise the record or data field group is skipped.

The display condition command can be used with either a record or a group of data

fields. When used with a record, the display condition command must follow the new record command. When used with a group of data fields, the display condition command must follow a data field row. In this case, the display condition will apply to the rest of the fields through the end of the record.

Consecutive display condition commands are merged as AND conditions. The merged display conditions apply to the same enclosed record or data field group.

## Structure of the Data Rows

The output record data fields are represented in the template by table rows. In FIXED\_POSITION\_BASED templates, each row has the following attributes (or columns):

- Position
- Length
- Format
- Pad
- Data
- Comments

The first five columns are required and must appear in the order listed.

For DELIMITER\_BASED templates, each data row has the following attributes (columns):

- Maximum Length
- Format
- Data
- Tag
- Comments

The first three columns are required and must be declared in the order stated.

In both template types, the Comments column is optional and ignored by the system. You can insert additional information columns if you wish, as all columns after the required ones are ignored.

The usage rules for these columns are as follows:

## Position

Specifies the starting position of the field in the record. The unit is in number of characters. This column is only used with FIXED\_POSITION\_BASED templates.

## Length/Maximum Length

Specifies the length of the field. The unit is in number of characters. For FIXED\_POSITION\_BASED templates, all the fields are fixed length. If the data is less than the specified length, it is padded. If the data is longer, it is truncated. The truncation always occurs on the right.

For DELIMITER\_BASED templates, this value specifies the maximum length of the field. If the data exceeds the maximum length, it will be truncated. Data is not padded if it is less than the maximum length.

## Format Column

Specifies the data type and format setting. There are three accepted data types:

- Alpha
- Number
- Date

Refer to Field Level Key Words, page 9-37 for their usage.

### Number Data Type

Numeric data has three optional format settings: Integer, Decimal, or you can define a format mask. Specify the optional settings with the Number data type as follows:

- Number, Integer
- Number, Decimal
- Number, <format mask>

For example:

Number, ###,###.00

The Integer format uses only the whole number portion of a numeric value and discards the decimal. The Decimal format uses only the decimal portion of the numeric value and discards the integer portion.

The following table shows examples of how to set a format mask. When specifying the mask, # represents that a digit is to be displayed when present in the data; 0 represents that the digit placeholder is to be displayed whether data is present or not.

When specifying the format mask, the group separator must always be "," and the decimal separator must always be "." To alter these in the actual output, you must use the Setup Commands NUMBER THOUSANDS SEPARATOR and NUMBER DECIMAL

SEPARATOR. See Setup Command Tables, page 9-16 for details on these commands.

The following table shows sample Data, Format Specifier, and Output. The Output assumes the default group and decimal separators.

<b>Data</b>	<b>Format Specifier</b>	<b>Output</b>
123456789	###,###.00	123,456,789.00
123456789.2	###.00	123456789.20
1234.56789	###.000	1234.568
123456789.2	#	123456789
123456789.2	##.	123456789.2
123456789	##.	123456789

#### **Date Data Type**

The Date data type format setting must always be explicitly stated. The format setting follows the SQL date styles, such as MMDDYY.

#### **Mapping EDI Delimiter-Based Data Types to eText Data Types**

Some EDI (DELIMITER\_BASED) formats use more descriptive data types. These are mapped to the three template data types in the following table:

<b>ASC X12 Data Type</b>	<b>Format Template Data Type</b>
A - Alphabetic	Alpha
AN -Alphanumeric	Alpha
B - Binary	Number
CD - Composite data element	N/A
CH - Character	Alpha
DT - Date	Date
FS - Fixed-length string	Alpha

ASC X12 Data Type	Format Template Data Type
ID - Identifier	Alpha
IV - Incrementing Value	Number
Nn - Numeric	Number
PW - Password	Alpha
R - Decimal number	Numer
TM - Time	Date

Now assume you have specified the following setup commands:

NUMBER THOUSANDS SEPARATOR	.
NUMBER DECIMAL SEPARATOR	,

The following table shows the Data, Format Specifier, and Output for this case. Note that the Format Specifier requires the use of the default separators, regardless of the setup command entries.

Data	Format Specifier	Output
123456789	###,###.00	123.456.789,00
123456789.2	###.00	123456789,20
1234.56789	###.000	1234,568
123456789.2	#	123456789
123456789.2	###	123456789,2
123456789	###	123456789

## Pad

This applies to `FIXED_POSITION_BASED` templates only. Specify the padding side (L = left or R = right) and the character. Both numeric and alphanumeric fields can be padded. If this field is not specified, Numeric fields are left-padded with "0"; Alpha fields are right-padded with spaces.

Example usage:

- To pad a field on the left with a "0", enter the following in the Pad column field:  
L, '0'
- To pad a field on the right with a space, enter the following the Pad column field:  
R, ' '

## Data

Specifies the XML element from the data extract that is to populate the field. The data column can simply contain the XML tag name, or it can contain expressions and functions. For more information, see *Expressions, Control Structure, and Functions*, page 9-31.

## Tag

Acts as a comment column for `DELIMITER_BASED` templates. It specifies the reference tag in EDIFACT formats, and the reference IDs in ASC X12.

## Comments

Use this column to note any free form comments to the template. Usually this column is used to note the business requirement and usage of the data field.

# Setup Command Tables

## Setup Command Table

A template always begins with a table that specifies the setup commands. The setup commands define global attributes, such as template type and output character set and program elements, such as sequencing and concatenation.

The setup commands are:

- TEMPLATE TYPE
- OUTPUT CHARACTER SET
- OUTPUT LENGTH MODE

- NEW RECORD CHARACTER
- INVALID CHARACTERS
- REPLACE CHARACTERS
- NUMBER THOUSANDS SEPARATOR
- NUMBER DECIMAL SEPARATOR
- DEFINE LEVEL
- DEFINE SEQUENCE
- DEFINE CONCATENATION
- CASE CONVERSION

Some example setup tables are shown in the following figures:

XDO file name:  
XINT-01.rtf

*Mapping of Payment Format:*  
**International Payments EFT Format**

**Format Setup:**

*Hint: Define formatting options...*

<TEMPLATE TYPE>	FIXED_POSITION_BASED
<OUTPUT CHARACTER SET>	iso-8859-1
<NEW RECORD CHARACTER>	Carriage Return

<INVALID CHARACTERS>	¿
<REPLACE CHARACTERS>	
A	AO
E	EO
I	IO
O	OO
U	UO
<END REPLACE CHARACTERS>	

**Format Data Levels:**

*Hint: Define data levels that are needed in the format which do not exist in data extract...*

<DEFINE LEVEL>	PaymentsByPayDatePayee
<BASE LEVEL>	Payment
<GROUPING CRITERIA>	'PaymentDate, PayeeName'
<END DEFINE LEVEL>	PaymentsByPayDatePayee

<DEFINE LEVEL>	InvoicesByReportingCatAndAttrib
<BASE LEVEL>	Invoice



<GROUPING CRITERIA>	<pre> `InvoiceTrxnReportingCat`, (IF InvoiceTrxnReportingCat = 'V' THEN `InvoiceDEVTransitGoods' END IF), (IF InvoiceTrxnReportingCat = 'V' THEN `InvoiceDEVGoodsIndexNum' END IF), (IF InvoiceTrxnReportingCat = 'V' THEN `InvoiceDEVPassingTrade' END IF) </pre>
<END DEFINE LEVEL>	InvoicesByReportingCatAndAttrib

### Sequences :

*Hint: Define sequence generators...*

<DEFINE SEQUENCE>	AllRecordsSeq
<RESET AT LEVEL>	PERIODIC_SEQUENCE
<INCREMENT BASIS>	RECORD
<START AT>	BaseRecordCount + 1
<MAXIMUM>	999999
<END DEFINE SEQUENCE >	AllRecordsSeq

<DEFINE SEQUENCE>	PaymentsSeq
<RESET AT LEVEL>	Batch
<INCREMENT BASIS>	LEVEL
<END DEFINE SEQUENCE >	PaymentsSeq

### Concatenated Records :

*Hint: Define fields that are composed of concatenated records...*

<DEFINE CONCATENATION>	ConcatenatedInvoiceInfo
<BASE LEVEL>	Invoice
<ELEMENT>	InvoiceNum
<DELIMITER>	','
<END DEFINE CONCATENATION>	ConcatenatedInvoiceInfo

## TEMPLATE TYPE Command

This command specifies the type of template. There are two types: FIXED\_POSITION\_BASED and DELIMITER\_BASED.

Use the FIXED\_POSITION\_BASED templates for fixed-length record formats, such as EFTs. In these formats, all fields in a record are a fixed length. If data is shorter than the specified length, it will be padded. If longer, it will be truncated. The system specifies the default behavior for data padding and truncation. Examples of fixed position based formats are EFTs in Europe, and NACHA ACH file in the U.S.

In a DELIMITER\_BASED template, data is never padded and only truncated when it has reached a maximum field length. Empty fields are allowed (when the data is null). Designated delimiters are used to separate the data fields. If a field is empty, two delimiters will appear next to each other. Examples of delimited-based templates are EDI formats such as ASC X12 820 and UN EDIFACT formats - PAYMUL, DIRDEB, and CREMUL.

In EDI formats, a record is sometimes referred to as a segment. An EDI segment is

treated the same as a record. Start each segment with a new record command and give it a record name. You should have a data field specifying the segment name as part of the output data immediately following the new record command.

For DELIMITER\_BASED templates, you insert the appropriate data field delimiters in separate rows between the data fields. After every data field row, you insert a delimiter row. You can insert a placeholder for an empty field by defining two consecutive delimiter rows.

Empty fields are often used for syntax reasons: you must insert placeholders for empty fields so that the fields that follow can be properly identified.

There are different delimiters to signify data fields, composite data fields, and end of record. Some formats allow you to choose the delimiter characters. In all cases you should use the same delimiter consistently for the same purpose to avoid syntax errors.

In DELIMITER\_BASED templates, the <POSITION> and <PAD> columns do not apply. They are omitted from the data tables.

Some DELIMITER\_BASED templates have minimum and maximum length specifications. In those cases Oracle Payments validates the length.

## **DEFINE LEVEL Command**

Some formats require specific additional data levels that are not in the data extract. For example, some formats require that payments be grouped by payment date. Using the Define Level command, a payment date group can be defined and referenced as a level in the template, even though it is not in the input extract file.

When you use the Define Level command you declare a base level that exists in the extract. The Define Level command inserts a new level one level higher than the base level of the extract. The new level functions as a grouping of the instances of the base level.

The Define Level command is a setup command, therefore it must be defined in the setup table. It has three subcommands:

- **BASE LEVEL** command - defines the level (XML element) from the extract that the new level is based on. The Define Level command must always have one and only one base level subcommand.
- **GROUPING CRITERIA**- defines the XML extract elements that are used to group the instances of the base level to form the instances of the new level. The parameter of the grouping criteria command is a comma-separated list of elements that specify the grouping conditions.

The order of the elements determines the hierarchy of the grouping. The instances of the base level are first divided into groups according to the values of the first criterion, then each of these groups is subdivided into groups according to the second criterion, and so on. Each of the final subgroups will be considered as an instance of the new level.

- GROUP SORT ASCENDING or GROUP SORT DESCENDING - defines the sorting of the group. Insert the <GROUP SORT ASCENDING> or <GROUP SORT DESCENDING> command row anywhere between the <DEFINE LEVEL> and <END DEFINE LEVEL> commands. The parameter of the sort command is a comma-separated list of elements by which to sort the group.

For example, the following table shows five payments under a batch:

Payment Instance	PaymentDate (grouping criterion 1)	PayeeName (grouping criterion 2)
Payment1	PaymentDate1	PayeeName1
Payment2	PaymentDate2	PayeeName1
Payment3	PaymentDate1	PayeeName2
Payment4	PaymentDate1	PayeeName1
Payment5	PaymentDate1	PayeeName3

In the template, construct the setup table as follows to create a level called "PaymentsByPayDatePayee" from the base level "Payment" grouped according to Payment Date and Payee Name. Add the Group Sort Ascending command to sort each group by PaymentDate and PayeeName:

<DEFINE LEVEL>	PaymentsByPayDatePayee
<BASE LEVEL>	Payment
<GROUPING CRITERIA>	PaymentDate, PayeeName
<GROUP SORT ASCENDING>	PaymentDate, PayeeName
<END DEFINE LEVEL>	PaymentsByPayDatePayee

The five payments will generate the following four groups (instances) for the new level:

Payment Group Instance	Group Criteria	Payments in Group
Group1	PaymentDate1, PayeeName1	Payment1, Payment4

Payment Group Instance	Group Criteria	Payments in Group
Group2	PaymentDate1, PayeeName2	Payment3
Group3	PaymentDate1, PayeeName3	Payment5
Group4	PaymentDate2, PayeeName1	Payment2

The order of the new instances is the order that the records will print. When evaluating the multiple grouping criteria to form the instances of the new level, the criteria can be thought of as forming a hierarchy. The first criterion is at the top of the hierarchy, the last criterion is at the bottom of the hierarchy.

Generally there are two kinds of format-specific data grouping scenarios in EFT formats. Some formats print the group records only; others print the groups with the individual element records nested inside groups. Following are two examples for these scenarios based on the five payments and grouping conditions previously illustrated.

### Example

First Scenario: Group Records Only

EFT File Structure:

- BatchRec
  - PaymentGroupHeaderRec
  - PaymentGroupFooterRec

Record Sequence	Record Type	Description
1	BatchRec	
2	PaymentGroupHeaderRec	For group 1 (PaymentDate1, PayeeName1)
3	PaymentGroupFooterRec	For group 1 (PaymentDate1, PayeeName1)
4	PaymentGroupHeaderRec	For group 2 (PaymentDate1, PayeeName2)
5	PaymentGroupFooterRec	For group 2 (PaymentDate1, PayeeName2)
6	PaymentGroupHeaderRec	For group 3 (PaymentDate1, PayeeName3)
7	PaymentGroupFooterRec	For group 3 (PaymentDate1, PayeeName3)

Record Sequence	Record Type	Description
8	PaymentGroupHeaderRec	For group 4 (PaymentDate2, PayeeName1)
9	PaymentGroupFooterRec	For group 4 (PaymentDate2, PayeeName1)

### Example

Scenario 2: Group Records and Individual Records

EFT File Structure:

BatchRec

- PaymentGroupHeaderRec
  - PaymentRec
- PaymentGroupFooterRec

Generated output:

Record Sequence	Record Type	Description
1	BatchRec	
2	PaymentGroupHeaderRec	For group 1 (PaymentDate1, PayeeName1)
3	PaymentRec	For Payment1
4	PaymentRec	For Payment4
5	PaymentGroupFooterRec	For group 1 (PaymentDate1, PayeeName1)
6	PaymentGroupHeaderRec	For group 2 (PaymentDate1, PayeeName2)
7	PaymentRec	For Payment3
8	PaymentGroupFooterRec	For group 2 (PaymentDate1, PayeeName2)
9	PaymentGroupHeaderRec	For group 3 (PaymentDate1, PayeeName3)
10	PaymentRec	For Payment5

Record Sequence	Record Type	Description
11	PaymentGroupFooterRec	For group 3 (PaymentDate1, PayeeName3)
12	PaymentGroupHeaderRec	For group 4 (PaymentDate2, PayeeName1)
13	PaymentRec	For Payment2
14	PaymentGroupFooterRec	For group 4 (PaymentDate2, PayeeName1)

Once defined with the Define Level command, the new level can be used in the template in the same manner as a level occurring in the extract. However, the records of the new level can only reference the base level fields that are defined in its grouping criteria. They cannot reference other base level fields other than in summary functions.

For example, the PaymentGroupHeaderRec can reference the PaymentDate and PayeeName in its fields. It can also reference thePaymentAmount (a payment level field) in a SUM function. However, it cannot reference other payment level fields, such as PaymentDocName or PaymentDocNum.

The DEFINE LEVEL command must always have one and only one grouping criteria subcommand. The DEFINE LEVEL command has a companion END DEFINE LEVEL command. The subcommands must be specified between the DEFINE LEVEL and END DEFINE LEVEL commands. They can be declared in any order.

#### DEFINE SEQUENCE Command

The DEFINE SEQUENCE command define a sequence that can be used in conjunction with the SEQUENCE\_NUMBER function to index either the generated EFT records or the extract instances (the database records). The EFT records are the physical records defined in the template. The database records are the records from the extract. To avoid confusion, the term "record" will always refer to the EFT record. The database record will be referred to as an extract element instance or level.

The DEFINE SEQUENCE command has four subcommands: RESET AT LEVEL, INCREMENT BASIS, START AT, and MAXIMUM:

##### RESET AT LEVEL

The RESET AT LEVEL subcommand defines where the sequence resets its starting number. It is a mandatory subcommand. For example, to number the payments in a batch, define RESET AT LEVEL as Batch. To continue numbering across batches, define RESET AT LEVEL as RequestHeader.

In some cases the sequence is reset outside the template. For example, a periodic sequence may be defined to reset by date. In these cases, the PERIODIC\_SEQUENCE keyword is used for the RESET AT LEVEL. The system saves the last sequence number used for a payment file to the database. Outside events control resetting the sequence in

the database. For the next payment file run, the sequence number is extracted from the database for the start at number (see start at subcommand).

### **INCREMENT BASIS**

The INCREMENT BASIS subcommand specifies if the sequence should be incremented based on record or extract instances. The allowed parameters for this subcommand are RECORD and LEVEL.

Enter RECORD to increment the sequence for every record.

Enter LEVEL to increment the sequence for every new instance of a level.

Note that for levels with multiple records, if you use the level-based increment all the records in the level will have the same sequence number. The record-based increment will assign each record in the level a new sequence number.

For level-based increments, the sequence number can be used in the fields of one level only. For example, suppose an extract has a hierarchy of batch > payment > invoice and you define the INCREMENT BASIS by level sequence, with reset at the batch level. You can use the sequence in either the payment or invoice level fields, but not both. You cannot have sequential numbering across hierarchical levels.

However, this rule does not apply to increment basis by record sequences. Records can be sequenced across levels.

For both increment basis by level and by record sequences, the level of the sequence is implicit based on where the sequence is defined.

## **Define Concatenation Command**

Use the define concatenation command to concatenate child-level extract elements for use in parent-level fields. For example, use this command to concatenate invoice number and due date for all the invoices belonging to a payment for use in a payment-level field.

The define concatenation command has three subcommands: base level, element, and delimiter.

### **Base Level Subcommand**

The base level subcommand specifies the child level for the operation. For each parent-level instance, the concatenation operation loops through the child-level instances to generate the concatenated string.

### **Element Subcommand**

The element subcommand specifies the operation used to generate each element. An element is a child-level expression that will be concatenated together to generate the concatenation string.

### **Delimiter Subcommand**

The delimiter subcommand specifies the delimiter to separate the concatenated items in

the string.

### Using the SUBSTR Function

Use the SUBSTR function to break down concatenated strings into smaller strings that can be placed into different fields. For example, the following table shows five invoices in a payment:

Invoice	InvoiceNum
1	car_parts_inv0001
2	car_parts_inv0002
3	car_parts_inv0003
4	car_parts_inv0004
5	car_parts_inv0005

Using the following concatenation definition:

<DEFINE CONCATENATION>	ConcatenatedInvoiceInfo
<BASE LEVEL>	Invoice
<ELEMENT>	InvoiceNum
<DELIMITER>	','
<END DEFINE CONCATENATION>	ConcatenatedInvoiceInfo

You can reference ConcatenatedInvoiceInfo in a payment level field. The string will be:

```
car_parts_inv0001,car_parts_inv0002,car_parts_inv0003,car_parts_
inv0004,car_parts_inv0005
```

If you want to use only the first forty characters of the concatenated invoice information, use either TRUNCATE function or the SUBSTR function as follows:

```
TRUNCATE (ConcatenatedInvoiceInfo, 40)
```

```
SUBSTR (ConctenatedInvoiceInfo, 1, 40)
```

Either of these statements will result in:

```
car_parts_inv0001,car_parts_inv0002,car_
```





conversion. The character set conversion is performed on the XML extract directly, before the formatting. After the character set conversion, the invalid characters will be checked in terms of the output character set. If no invalid characters are found, the system will proceed to formatting.

### **Output Character Set and New Record Character Commands**

Use the new record character command to specify the character(s) to delimit the explicit and implicit record breaks at runtime. Each new record command represents an explicit record break. Each end of table represents an implicit record break. The parameter is a list of constant character names separated by commas.

Some formats contain no record breaks. The generated output is a single line of data. In this case, leave the new record character command parameter field empty.

If you do not define a "new record character" field in the template, the system will set "\n" as default new record character.

### **Output Length Mode**

Output Length Mode can be set to "character" or "byte". When OUTPUT LENGTH MODE is set to "character", the output record length for each field is based on character length. When OUTPUT LENGTH MODE is set to "byte", the output record length for each field is based on byte length.

If no OUTPUT LENGTH MODE is setting is provided, "character" will be used.

### **Number Thousands Separator and Number Decimal Separator**

The default thousands (or group) separator is a comma (",") and the default decimal separator is a period ("."). Use the Number Thousands Separator command and the Number Decimal Separator command to specify separators other than the defaults. For example, to define "." as the group separator and "," as the decimal separator, enter the following:

---

<NUMBER THOUSANDS SEPARATOR>	.
<NUMBER DECIMAL SEPARATOR>	,

---

Note that when you set "NUMBER DECIMAL SEPARATOR", you must also set "NUMBER THOUSANDS SEPARATOR". Ensure to set the appropriate format mask for the field to be displayed. For more information on formatting numbers, see Format Column, page 9-13.

### **CASE CONVERSION**

Use CASE CONVERSION to convert strings from lowercase to uppercase for fields with format type ALPHA. This command is used with FIXED\_POSITION\_BASED templates.







control structure can be nested.

The IN predicate is supported in the IF-THEN-ELSE control structure. For example:

```
IF PaymentAmount/Currency/Code IN ('USD', 'EUR', 'AON', 'AZM') THEN
    PayeeAccount/FundsCaptureOrder/OrderAmount/Value * 100
ELSIF PaymentAmount/Currency/Code IN ('BHD', 'IQD', 'KWD') THEN
    PayeeAccount/FundsCaptureOrder/OrderAmount/Value * 1000
ELSE
    PayeeAccount/FundsCaptureOrder/OrderAmount/Value
END IF;
```

## Functions

Following is the list of supported functions:

- **SEQUENCE\_NUMBER** - is a record element index. It is used in conjunction with the Define Sequence command. It has one parameter, which is the sequence defined by the Define Sequence command. At runtime it will increase its sequence value by one each time it is referenced in a record.
- **COUNT** - counts the child level extract instances or child level records of a specific type. Declare the COUNT function on a level above the entity to be counted. The function has one argument. If the argument is a level, the function will count all the instances of the (child) level belonging to the current (parent) level instance.  
  
For example, if the level to be counted is Payment and the current level is Batch, then the COUNT will return the total number of payments in the batch. However, if the current level is RequestHeader, the COUNT will return the total number of payments in the file across all batches. If the argument is a record type, the count function will count all the generated records of the (child level) record type belonging to the current level instance.
- **INTEGER\_PART, DECIMAL\_PART** - returns the integer or decimal portion of a numeric value. This is used in nested expressions and in commands (display condition and group by). For the final formatting of a numeric field in the data column, use the Integer/Decimal format.
- **IS\_NUMERIC** - boolean test whether the argument is numeric. Used only with the "IF" control structure.
- **TRUNCATE** - truncate the first argument - a string to the length of the second argument. If the first argument is shorter than the length specified by the second argument, the first argument is returned unchanged. This is a user-friendly version for a subset of the SQL substr() functionality.
- **SUM** - sums all the child instance of the XML extract field argument. The field must be a numeric value. The field to be summed must always be at a lower level than the level on which the SUM function was declared.

- MIN, MAX - find the minimum or maximum of all the child instances of the XML extract field argument. The field must be a numeric value. The field to be operated on must always be at a lower level than the level on which the function was declared.
- FORMAT\_DATE - Formats a date string to any desirable date format. For example:  
`FORMAT_DATE("1900-01-01T18:19:20", "YYYY/MM/DD HH24:MI:SS")`  
will produce the following output:  
1900/01/01 18:19:20
- FORMAT\_NUMBER - Formats a number to display in desired format. For example:  
`FORMAT_NUMBER("1234567890.0987654321", "999,999.99")`  
produces the following output:  
1,234,567,890.10
- MESSAGE\_LENGTH - returns the length of the message in the EFT message.
- RECORD\_LENGTH - returns the length of the record in the EFT message.
- INSTR - returns the numeric position of a named character within a text field.
- SYSDATE, DATE - gets Current Date and Time.
- POSITION - returns the position of a node in the XML document tree structure.
- REPLACE - replaces a string with another string.s
- CONVERT\_CASE - converts a string or a character to UPPER or LOWER case.
- CHR - gets the character representation of an argument, which is an ASCII value.
- LPAD, RPAD - generates left or right padding for string values.
- AND, OR, NOT - operator functions on elements.
- DISTINCT\_VALUES - equivalent to the XPATH function DISTINCT-VALUES. Returns a sequence in which all but one of a set of duplicate values, based on value equality, have been deleted. Usage: `distinct_values(fieldname)`.
- INCREASE\_DATE - increments a date by the number of days specified.  
Usage:  
`increase_date(./date, 2)`

returns a date value two days after the value of `./date`

- `DECREASE_DATE` - decreases a date by the number of days specified.

Usage:

```
decrease_date(./date, 2)
```

returns a date value two before the value of `./date`

- Other SQL functions include the following. Use the syntax corresponding to the SQL function.
  - `TO_DATE`
  - `LOWER`
  - `UPPER`
  - `LENGTH`
  - `GREATEST`
  - `LEAST`
  - `DECODE`
  - `CEIL`
  - `ABS`
  - `FLOOR`
  - `ROUND`
  - `CHR`
  - `TO_CHAR`
  - `SUBSTR`
  - `LTRIM`
  - `RTRIM`
  - `TRIM`
  - `IN`
  - `TRANSLATE`



## Identifiers, Operators, and Literals

This section lists the reserved key word and phrases and their usage. The supported operators are defined and the rules for referencing XML extract fields and using literals.

### Key Words

There are four categories of key words and key word phrases:

- Command and column header key words
- Command parameter and function parameter key words
- Field-level key words
- Expression key words

### Command and Column Header Key Words

The following key words must be used as shown: enclosed in <>s and in all capital letters with a bold font.

- **<LEVEL>**- the first entry of a data table. Associates the table with an XML element and specifies the hierarchy of the table.
- **<END LEVEL>** - declares the end of the current level. Can be used at the end of a table or in a standalone table.
- **<POSITION>** - column header for the first column of data field rows, which specifies the starting position of the data field in a record.
- **<LENGTH>** - column header for the second column of data field rows, which specifies the length of the data field.
- **<FORMAT>** - column header for the third column of data field rows, which specifies the data type and format setting.
- **<PAD>** - column header for the fourth column of data field rows, which specifies the padding style and padding character.
- **<DATA>** - column header for the fifth column of data field rows, which specifies the data source.
- **<COMMENT>** - column header for the sixth column of data field rows, which allows for free form comments.
- **<NEW RECORD>** - specifies a new record.

- **<DISPLAY CONDITION>** - specifies the condition when a record should be printed.
- **<TEMPLATE TYPE>** - specifies the type of the template, either `FIXED_POSITION_BASED` or `DELIMITER_BASED`.
- **<OUTPUT CHARACTER SET>** - specifies the character set to be used when generating the output.
- **<NEW RECORD CHARACTER>** - specifies the character(s) to use to signify the explicit and implicit new records at runtime.
- **<DEFINE LEVEL>** - defines a format-specific level in the template.
- **<BASE LEVEL>** - subcommand for the define level and define concatenation commands.
- **<GROUPING CRITERIA>** - subcommand for the define level command.
- **<END DEFINE LEVEL>** - signifies the end of a level.
- **<DEFINE SEQUENCE>** - defines a record or extract element based sequence for use in the template fields.
- **<RESET AT LEVEL>** - subcommand for the define sequence command.
- **<INCREMENT BASIS>** - subcommand for the define sequence command.
- **<START AT>** - subcommand for the define sequence command.
- **<MAXIMUM>** - subcommand for the define sequence command.
- **<MAXIMUM LENGTH>** - column header for the first column of data field rows, which specifies the maximum length of the data field. For `DELIMITER_BASED` templates only.
- **<END DEFINE SEQUENCE>** - signifies the end of the sequence command.
- **<DEFINE CONCATENATION>** - defines a concatenation of child level elements that can be referenced as a string in the parent level fields.
- **<ELEMENT>** - subcommand for the define concatenation command.
- **<DELIMITER>** - subcommand for the define concatenation command.
- **<END DEFINE CONCATENATION>** - signifies the end of the define concatenation command.

- **<SORT ASCENDING>** - format-specific sorting for the instances of a level.
- **<SORT DESCENDING>** - format-specific sorting for the instances of a level.

### **Command Parameter and Function Parameter Key Words**

These key words must be entered in all capital letters, nonbold fonts.

- **PERIODIC\_SEQUENCE** - used in the reset at level subcommand of the define sequence command. It denotes that the sequence number is to be reset outside the template.
- **FIXED\_POSITION\_BASED, DELIMITER\_BASED** - used in the template type command, specifies the type of template.
- **RECORD, LEVEL** - used in the increment basis subcommand of the define sequence command. **RECORD** increments the sequence each time it is used in a new record. **LEVEL** increments the sequence only for a new instance of the level.

### **Field-Level Key Words**

- **Alpha** - in the **<FORMAT>** column, specifies the data type is alphanumeric.
- **Number** - in the **<FORMAT>** column, specifies the data type is numeric.
- **Integer** - in the **<FORMAT>** column, used with the **Number** key word. Takes the integer part of the number. This has the same functionality as the **INTEGER** function, except the **INTEGER** function is used in expressions, while the **Integer** key word is used in the **<FORMAT>** column only.
- **Decimal** - in the **<FORMAT>** column, used with the **Number** key word. Takes the decimal part of the number. This has the same functionality as the **DECIMAL** function, except the **DECIMAL** function is used in expressions, while the **Decimal** key word is used in the **<FORMAT>** column only.
- **Date** - in the **<FORMAT>** column, specifies the data type is date.
- **L, R** - in the **<PAD>** column, specifies the side of the padding (Left or Right).

### **Expression Key Words**

Key words and phrases used in expressions must be in capital letters and bold fonts.

- **IF THEN ELSE IF THEN ELSE END IF** - these key words are always used as a group. They specify the "IF" control structure expressions.
- **IS NULL, IS NOT NULL** - these phrases are used in the IF control structure. They form part of boolean predicates to test if an expression is NULL or not NULL.

## Operators

There are two groups of operators: the boolean test operators and the expression operators. The boolean test operators include: "=", "<>", "<", ">", ">=", and "<=". They can be used only with the IF control structure. The expression operators include: "()", "||", "+", "-", and "\*". They can be used in any expression.

Symbol	Usage
=	Equal to test. Used in the IF control structure only.
<>	Not equal to test. Used in the IF control structure only.
>	Greater than test. Used in the IF control structure only.
<	Less than test. Used in the IF control structure only.
>=	Greater than or equal to test. Used in the IF control structure only.
<=	Less than or equal to test. Used in the IF control structure only.
()	Function argument and expression group delimiter. The expression group inside "()" will always be evaluated first. "()" can be nested.
	String concatenation operator.
+	Addition operator. Implicit type conversion may be performed if any of the operands are not numbers.
-	Subtraction operator. Implicit type conversion may be performed if any of the operands are not numbers.
*	Multiplication operator. Implicit type conversion may be performed if any of the operands are not numbers.

Symbol	Usage
DIV	Division operand. Implicit type conversion may be performed if any of the operands are not numbers. Note that "/" is not used because it is part of the XPATH syntax.
IN	Equal-to-any-member-of test.
NOT IN	Negates the IN operator. Not-Equal-to-any-member-of test.

### Reference to XML Extract Fields and XPATH Syntax

XML elements can be used in any expression. At runtime they will be replaced with the corresponding field values. The field names are case-sensitive.

When the XML extract fields are used in the template, they must follow the XPATH syntax. This is required so that the BI Publisher engine can correctly interpret the XML elements.

There is always an extract element considered as the context element during the BI Publisher formatting process. When BI Publisher processes the data rows in a table, the level element of the table is the context element. For example, when BI Publisher processes the data rows in the Payment table, Payment is the context element. The relative XPATH you use to reference the extract elements are specified in terms of the context element.

For example if you need to refer to the PayeeName element in a Payment data table, you will specify the following relative path:

Payee/PayeeInfo/PayeeName

Each layer of the XML element hierarchy is separated by a backslash "/". You use this notation for any nested elements. The relative path for the immediate child element of the level is just the element name itself. For example, you can use TransactionID element name as is in the Payment table.

To reference a parent level element in a child level table, you can use the "../" notation. For example, in the Payment table if you need to reference the BatchName element, you can specify ../BatchName. The "../" will give you Batch as the context; in that context you can use the BatchName element name directly as BatchName is an immediate child of Batch. This notation goes up to any level for the parent elements. For example if you need to reference the RequesterParty element (in the RequestHeader) in a Payment data table, you can specify the following:

../TrxnParties/RequesterParty

You can always use the absolute path to reference any extract element anywhere in the

template. The absolute path starts with a backslash "/". For the PayeeName in the Payment table example above, you will have the following absolute path:  
/BatchRequest/Batch/Payment/Payee/PayeeInfo/PayeeName

The absolute path syntax provides better performance.

The identifiers defined by the setup commands such as define level, define sequence and define concatenation are considered to be global. They can be used anywhere in the template. No absolute or relative path is required. The base level and reset at level for the setup commands can also be specified. BI Publisher will be able to find the correct context for them.

If you use relative path syntax, you should specify it relative to the base levels in the following commands:

- The element subcommand of the define concatenation command
- The grouping criteria subcommand of the define level command

The extract field reference in the start at subcommand of the define sequence command should be specified with an absolute path.

The rule to reference an extract element for the level command is the same as the rule for data fields. For example, if you have a Batch level table and a nested Payment level table, you can specify the Payment element name as-is for the Payment table. Because the context for evaluating the Level command of the Payment table is the Batch.

However, if you skip the Payment level and you have an Invoice level table directly under the Batch table, you will need to specify Payment/Invoice as the level element for the Invoice table.

The XPATH syntax required by the template is very similar to UNIX/LINUX directory syntax. The context element is equivalent to the current directory. You can specify a file relative to the current directory or you can use the absolute path which starts with a "/".

Finally, the extract field reference as the result of the grouping criteria sub-command of the define level command must be specified in single quotes. This tells the BI Publisher engine to use the extract fields as the grouping criteria, not their values.

### **Notes on Viewing eText Output from a Browser**

If the report data contains Simplified Chinese characters and the <OUTPUT CHARACTER SET> is set to GBK, the Chinese characters will not display properly in Internet Explorer 7 with gbk2312 encoding. This issue may also occur in other non-English encodings as well, such as native Japanese and Korean. The output renders appropriately with Firefox 3.5, when setting the character set to be GBK in the eText template and setting the browser encoding to be GBK or GB2312. You can work around this issue by setting <OUTPUT CHARACTER SET> to utf-8. Note that this is a browser display issue only. The text file is generated correctly.

---

## Setting Report Processing and Output Document Properties

This chapter covers the following topics:

- Introduction
- PDF Output Properties
- PDF Security Properties
- PDF Digital Signature Properties
- RTF Output Properties
- HTML Output Properties
- FO Processing Properties
- RTF Template Properties
- PDF Template Properties
- Flash Template Properties
- CSV Output Properties
- Excel 2007 Output Properties
- All Outputs
- Defining Font Mappings

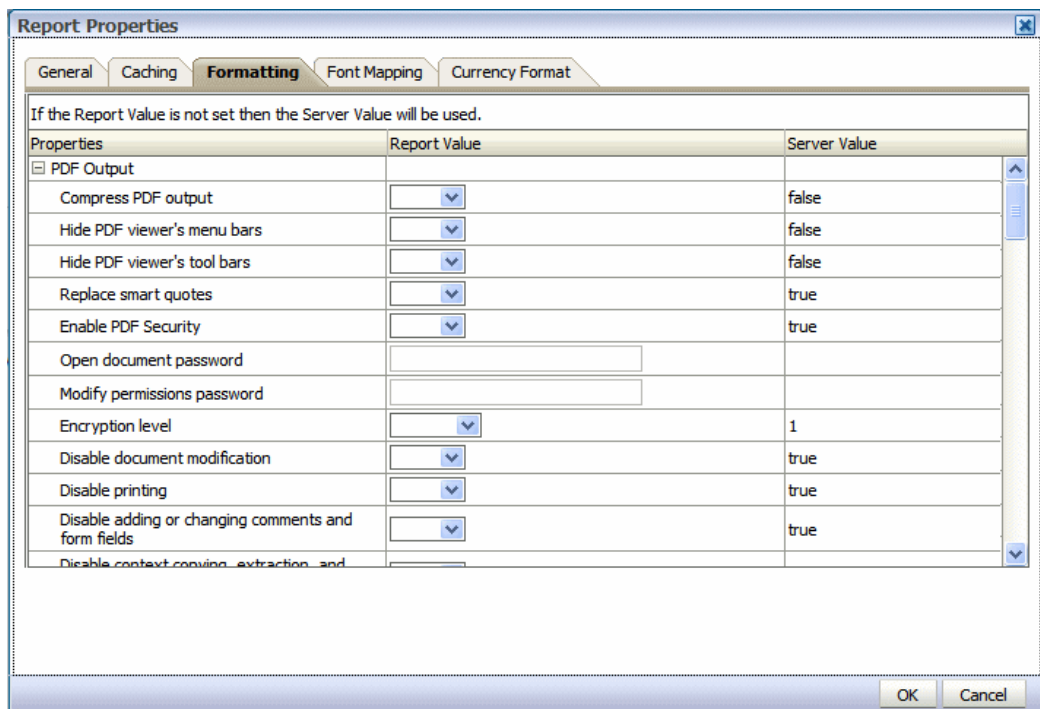
### Introduction

The **Formatting tab** of the Report Properties dialog enables you to set runtime formatting properties at the report level. These properties are also set at the system level. If conflicting values are set for a property at each level, the report level will take precedence.

To set a property at the report level:

1. Open the report in the Report Editor, and then
2. Click **Properties**. This will launch the **Report Properties** dialog.
3. Click the **Formatting** tab to display the formatting properties.

For each property, **Report Value** is updateable and the **Server Value** is shown for reference.



For information on setting the properties at the server level, see *Defining Runtime Configurations, Oracle Fusion Middleware Administrator's Guide for Oracle Business Intelligence Publisher*

## PDF Output Properties

The following table describes the properties that are available for PDF output:



Property Name	Description
Compress PDF output	<p><b>Default:</b> true</p> <p><b>Description:</b> Specify "true" or "false" to control compression of the output PDF file.</p> <p><b>Internal Name:</b>pdf-compression</p>
Hide PDF viewer's menu bars	<p><b>Default:</b> false</p> <p><b>Description:</b> Specify "true" to hide the viewer application's menu bar when the document is active. The menu bar option is only effective when using the Export button, which displays the output in a standalone Acrobat Reader application outside of the browser.</p> <p><b>Internal Name:</b> pdf-hide-menubar</p>
Hide PDF viewer's tool bars	<p><b>Default:</b> false</p> <p><b>Description:</b> Specify "true" to hide the viewer application's toolbar when the document is active.</p> <p><b>Internal Name:</b>pdf-hide-toolbar</p>
Replace smart quotes	<p><b>Default:</b> true</p> <p><b>Description:</b> Set to "false" if you do not want curly quotes replaced with straight quotes in your PDF output.</p> <p><b>Internal Name:</b> pdf-replace-smartquotes</p>
Use only one shared resources object for all pages	<p><b>Default:</b> true</p> <p><b>Description:</b> The default mode of BI Publisher creates one shared resources object for all pages in a PDF file. This mode has the advantage of creating an overall smaller file size. However, the disadvantages are the following:</p> <ul style="list-style-type: none"> <li>• Viewing may take longer for a large file with many SVG objects</li> <li>• If you choose to break the file up by using Adobe Acrobat to extract or delete portions, the edited PDF files will be larger because all of the single shared resource object (containing all of the SVG objects for the entire file) will be included with each extracted portion.</li> </ul> <p>Setting this property to "false" will create a resource object for each page. The file size will be bigger, but the PDF viewing will be faster and the PDF can be broken up into smaller files more easily.</p> <p><b>Internal Name:</b> pdf-use-one-resources</p>

## PDF Security Properties

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

---

Property Name	Description
Enable PDF Security	<p><b>Default:</b> false</p> <p><b>Description:</b> If you specify "true," the output PDF file will be encrypted. You can then also specify the following properties:</p> <ul style="list-style-type: none"><li>• Open document password</li><li>• Modify permissions password</li><li>• Encryption Level</li></ul> <p><b>Internal Name:</b> pdf-security</p>
Open document password	<p><b>Default:</b> N/A</p> <p><b>Description:</b> This password will be required for opening the document. It will enable users to open the document only. This property is enabled only when "Enable PDF Security" is set to "true".</p> <p>Note that BI Publisher follows Adobe's password restrictions. The password must contain only Latin 1 characters and must be no more than 32 bytes long.</p> <p><b>Internal Name:</b> pdf-open-password</p>
Modify permissions password	<p><b>Default:</b> N/A</p> <p><b>Description:</b> This password enables users to override the security setting. This property is effective only when "Enable PDF Security" is set to "true".</p> <p>Note that BI Publisher follows the Adobe's password restrictions. The password must contain only Latin 1 characters and must be no more than 32 bytes long.</p> <p><b>Internal Name:</b> pdf-permissions-password</p>

---

Property Name	Description
Encryption level	<p><b>Default:</b> 2 - high</p> <p><b>Description:</b> Specify the encryption level for the output PDF file. The possible values are:</p> <ul style="list-style-type: none"> <li>• 0: Low (40-bit RC4, Acrobat 3.0 or later)</li> <li>• 1: Medium (128-bit RC4, Acrobat 5.0 or later)</li> <li>• 2: High (128-bit AES, Acrobat 7.0 or later)</li> </ul> <p>This property is effective only when "Enable PDF Security" is set to "true". When Encryption level is set to 0, you can also set the following properties:</p> <ul style="list-style-type: none"> <li>• Disable printing</li> <li>• Disable document modification</li> <li>• Disable context copying, extraction, and accessibility</li> <li>• Disable adding or changing comments and form fields</li> </ul> <p>When Encryption level is set to 1 or higher, the following properties are available:</p> <ul style="list-style-type: none"> <li>• Enable text access for screen readers</li> <li>• Enable copying of text, images, and other content</li> <li>• Allowed change level</li> <li>• Allowed printing level</li> </ul> <p><b>Internal Name:</b> pdf-encryption-level</p>
Disable document modification	<p><b>Default:</b> false</p> <p><b>Description:</b> Permission available when "Encryption level" is set to 0. When set to "true", the PDF file cannot be edited.</p> <p><b>Internal Name:</b> pdf-no-changing-the-document</p>

Property Name	Description
Disable printing	<p><b>Default:</b> false</p> <p><b>Description:</b> Permission available when "Encryption level" is set to 0. When set to "true", printing is disabled for the PDF file.</p> <p><b>Internal Name:</b>pdf-no-printing</p>
Disable adding or changing comments and form fields	<p><b>Default:</b> false</p> <p><b>Description:</b> Permission available when "Encryption level" is set to 0. When set to "true", the ability to add or change comments and form fields is disabled.</p> <p><b>Internal Name:</b>pdf-no-accff</p>
Disable context copying, extraction, and accessibility	<p><b>Default:</b> false</p> <p><b>Description:</b> Permission available when "Encryption level" is set to 0. When set to "true", the context copying, extraction, and accessibility features are disabled.</p> <p><b>Internal Name:</b> pdf-no-cceda</p>
Enable text access for screen readers	<p><b>Default:</b> true</p> <p><b>Description:</b> Permission available when "Encryption level" is set to 1 or higher. When set to "true", text access for screen reader devices is enabled.</p> <p><b>Internal Name:</b> pdf-enable-accessibility</p>
Enable copying of text, images, and other content	<p><b>Default:</b> false</p> <p><b>Description:</b> Permission available when "Encryption level" is set to 1 or higher. When set to "true", copying of text, images, and other content is enabled.</p> <p><b>Internal Name:</b> pdf-enable-copying</p>

Property Name	Description
Allowed change level	<p><b>Default:</b> 0</p> <p><b>Description:</b> Permission available when "Encryption level" is set to 1 or higher. Valid Values are:</p> <ul style="list-style-type: none"> <li>• 0: none</li> <li>• 1: Allows inserting, deleting, and rotating pages</li> <li>• 2: Allows filling in form fields and signing</li> <li>• 3: Allows commenting, filling in form fields, and signing</li> <li>• 4: Allows all changes except extracting pages</li> </ul> <p><b>Internal Name:</b>pdf-changes-allowed</p>
Allowed printing level	<p><b>Default:</b> 0</p> <p><b>Description:</b> Permission available when "Encryption level" is set to 1 or higher. Valid values are:</p> <ul style="list-style-type: none"> <li>• 0: None</li> <li>• 1: Low resolution (150 dpi)</li> <li>• 2: High resolution</li> </ul> <p><b>Internal Name:</b>pdf-printing-allowed</p>

## PDF Digital Signature Properties

The following properties should only be set at the report level to enable digital signature for a report and to define the placement of the signature in the output PDF document. For more information on how to enable digital signature for your output PDF documents, see *Implementing a Digital Signature, Oracle Fusion Middleware Administrator's Guide for Oracle Business Intelligence Publisher*.

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

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

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

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

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

Property Name	Description
Enable Digital Signature	<p><b>Default:</b> false</p> <p><b>Description:</b> Set this to "true" to enable digital signature for the report.</p> <p><b>Internal Name:</b> signature-enable</p>
Existing signature field name	<p><b>Default:</b> N/A</p> <p><b>Description:</b> This property applies to PDF layout templates only. If your report is based on a PDF template, you can enter a field from the PDF template in which to place the digital signature. For more information on defining a field for the signature in a PDF template, see Adding or Designating a Field for a Digital Signature, page 7-19.</p> <p><b>Internal Name:</b> signature-field-name</p>
Signature field location	<p><b>Default:</b> top-left</p> <p><b>Description:</b> This property can apply to RTF or PDF layout templates. This property provides a list containing the following values: Top Left, Top Center, Top Right. Choose one of these general locations and BI Publisher will insert the digital signature to the output document, sized and positioned appropriately. If you choose to set this property, do not enter X and Y coordinates or width and height properties.</p> <p><b>Internal Name:</b> signature-field-location</p>

Property Name	Description
Signature field X coordinate	<p><b>Default:</b> 0</p> <p><b>Description:</b> This property can apply to RTF or PDF layout templates. Using the left edge of the document as the zero point of the X axis, enter the position in points that you want the digital signature to be placed from the left. For example, if you want the digital signature to be placed horizontally in the middle of an 8.5 inch by 11 inch document (that is, 612 points in width and 792 points in height), enter 306.</p> <p><b>Internal Name:</b> signature-field-pos-x</p>
Signature field Y coordinate	<p><b>Default:</b> 0</p> <p><b>Description:</b> This property can apply to RTF or PDF layout templates. Using the bottom edge of the document as the zero point of the Y axis, enter the position in points that you want the digital signature to be placed from the bottom. For example, if you want the digital signature to be placed vertically in the middle of an 8.5 inch by 11 inch document (that is, 612 points in width and 792 points in height), enter 396.</p> <p><b>Internal Name:</b> signature-field-pos-y</p>
Signature field width	<p><b>Default:</b> 0</p> <p><b>Description:</b> Enter in points (72 points equal one inch) the desired width of the inserted digital signature field. This applies only if you are also setting the properties <b>Signature field x coordinate</b> and <b>Signature field Y coordinate</b>.</p> <p><b>Internal Name:</b> signature-field-width</p>
Signature field height	<p><b>Default:</b> 0</p> <p><b>Description:</b> Enter in points (72 points equal one inch) the desired height of the inserted digital signature field. This applies only if you are also setting the properties <b>Signature field x coordinate</b> and <b>Signature field Y coordinate</b>.</p> <p><b>Internal Name:</b> signature-field-height</p>

## RTF Output Properties

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

Property Name	Description
Enable change tracking	<p><b>Default:</b> false</p> <p><b>Description:</b> Set to "true" to enable change tracking in the output RTF document.</p> <p><b>Internal Name:</b> rtf-track-changes</p>
Protect document for tracked changes	<p><b>Default:</b> false</p> <p><b>Description:</b> Set to "true" to protect the document for tracked changes.</p> <p><b>Internal Name:</b> rtf-protect-document-for-tracked-changes</p>
Default font	<p><b>Default:</b> Arial:12</p> <p><b>Description:</b> Use this property to define the font style and size in RTF output when no other font has been defined. This is particularly useful to control the sizing of empty table cells in generated reports.</p> <p>Enter the font name and size in the following format &lt;FontName&gt;:&lt;size&gt; for example: Arial:12.</p> <p>Note that the font you choose must be available to the BI Publisher processing engine at runtime. See Defining Font Mappings, page 10-19 for information on installing fonts for the BI Publisher server and also for the list of fonts predefined for BI Publisher.</p> <p><b>Internal Name:</b> rtf-output-default-font</p>

## HTML Output Properties

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

Property Name	Description
Show header	<p><b>Default:</b> true</p> <p><b>Description:</b> Set to "false" to suppress the template header in HTML output.</p> <p><b>Internal Name:</b> html-show-header</p>



Property Name	Description
Show footer	<p><b>Default:</b> true</p> <p><b>Description:</b> Set to "false" to suppress the template footer in HTML output.</p> <p><b>Internal Name:</b> <code>html-show-footer</code></p>
Replace smart quotes	<p><b>Default:</b> true</p> <p><b>Description:</b> Set to "false" if you do not want curly quotes replaced with straight quotes in your HTML output.</p> <p><b>Internal Name:</b> <code>html-replace-smartquotes</code></p>
Character set	<p><b>Default:</b> UTF-8</p> <p><b>Description:</b> Specifies the output HTML character set.</p> <p><b>Internal Name:</b> <code>html-output-charset</code></p>
Make HTML output accessible	<p><b>Default:</b> false</p> <p><b>Description:</b> Specify true if you want to make the HTML output accessible.</p> <p><b>Internal Name:</b> <code>make-accessible</code></p>
Use percentage width for table columns	<p><b>Default:</b> true</p> <p><b>Description:</b> Set this property to true to render table columns according to a percentage value of the total width of the table rather than as a value in points.</p> <p>This property is especially useful if your browser renders tables with extremely wide columns. Setting this property to true will improve readability of the tables.</p> <p><b>Internal Name:</b> <code>html-output-width-in-percentage</code></p>

## FO Processing Properties

The following properties can be set to govern FO processing:

Property Name	Description
Use BI Publisher's XSLT processor	<p><b>Default:</b> true</p> <p><b>Description:</b> Controls BI Publisher's parser usage. If set to false, XSLT will not be parsed.</p> <p><b>Internal Name:</b> <code>xslt-xdoparser</code></p>
Enable scalable feature of XSLT processor	<p><b>Default:</b> false</p> <p><b>Description:</b> Controls the scalable feature of the XDO parser. The property "Use BI Publisher's XSLT processor" must be set to "true" for this property to be effective.</p> <p><b>Internal Name:</b> <code>xslt-scalable</code></p>
Enable XSLT runtime optimization	<p><b>Default:</b> true</p> <p><b>Description:</b> When set to "true", the overall performance of the FO processor is increased and the size of the temporary FO files generated in the temp directory is significantly decreased. Note that for small reports (for example 1-2 pages) the increase in performance is not as marked.</p> <p>To further enhance performance when you set this property to true, it is recommended that you set the property <b>Extract attribute sets</b> to "false". See RTF Template Properties, page 10-14.</p> <p><b>Internal Name:</b> <code>xslt-runtime-optimization</code></p>
Enable XPath Optimization	<p><b>Default:</b> false</p> <p><b>Description:</b> When set to "true", the XML data file will be analyzed for element frequency. The information is then used to optimize XPath in XSL.</p> <p><b>Internal Name:</b> <code>xslt-xpath-optimization</code></p>
Pages cached during processing	<p><b>Default:</b> 50</p> <p><b>Description:</b> This property is enabled only when you have specified a Temporary Directory (under General properties). During table of contents generation, the FO Processor caches the pages until the number of pages exceeds the value specified for this property. It then writes the pages to a file in the Temporary Directory.</p> <p><b>Internal Name:</b> <code>system-cache-page-size</code></p>

Property Name	Description
Bidi language digit substitution type	<p><b>Default:</b> None</p> <p><b>Description:</b> Valid values are "None" and "National". When set to "None", Eastern European numbers will be used. When set to "National", Hindi format (Arabic-Indic digits) will be used. This setting is effective only when the locale is Arabic, otherwise it is ignored.</p> <p><b>Internal Name:</b> <code>digit-substitution</code></p>
Disable variable header support	<p><b>Default:</b> false</p> <p><b>Description:</b> If "true", prevents variable header support. Variable header support automatically extends the size of the header to accommodate the contents.</p> <p><b>Internal Name:</b> <code>fo-prevent-variable-header</code></p>
Add prefix to IDs when merging FO	<p><b>Default:</b> false</p> <p><b>Description:</b> When merging multiple XSL-FO inputs, the FO Processor automatically adds random prefixes to resolve conflicting IDs. Setting this property to "true" disables this feature.</p> <p><b>Internal Name:</b> <code>fo-merge-conflict-resolution</code></p>
Enable multithreading	<p><b>Default:</b> false</p> <p><b>Description:</b> If you have a multiprocessor machine or a machine with a dual-core single processor, you may be able to achieve faster document generation by setting this option to True.</p> <p><b>Internal Name:</b> <code>fo-multi-threads</code></p>
Disable external references	<p><b>Default:</b> true</p> <p><b>Description:</b> A "true" setting (default) disallows the importing of secondary files such as subtemplates or other XML documents during XSL processing and XML parsing. This increases the security of your system. Set this to "false" if your report or template calls external files.</p> <p><b>Internal Name:</b> <code>xdk-secure-io-mode</code></p>

Property Name	Description
FO Parsing Buffer Size	<p><b>Default:</b> 1000000</p> <p><b>Description:</b> Sets the size of the buffer for the FO Processor. When the buffer is full, the elements from the buffer will be rendered in the report. Reports with large tables or pivot tables that require complex formatting and calculations may require a larger buffer to properly render those objects in the report. Increase the size of the buffer at the report level for these reports. Note that increasing this value will affect the memory consumption of your system.</p> <p><b>Internal Name:</b> fo-chunk-size</p>
Enable XSLT runtime optimization for sub-template	<p><b>Default:</b> true</p> <p><b>Note:</b> The default is true on the BI Publisher server. If you call the FOProcessor directly, the default is false.</p> <p><b>Description:</b> Provides an option to perform XSL import in FOProcessor before passing only one XSL to XDK for further processing. This allows xslt-optimization to be applied to the entire main XSL template which already includes all its subtemplates.</p> <p><b>Internal Name:</b> xslt-do-import</p>
Enable PPTX native chart support	<p><b>Default:</b> true</p> <p><b>Description:</b> This property applies to PowerPoint 2007 output. When set to true, charts in PowerPoint 2007 output will be rendered as native PowerPoint (PPTX) charts. If this property is set to false, the chart will be rendered as an embedded PNG image.</p> <p><b>Internal Name:</b> pptx-native-chart</p>

## RTF Template Properties

The following properties can be set to govern RTF templates:

Property Name	Description
Extract attribute sets	<p><b>Default:</b> Auto</p> <p><b>Description:</b> The RTF processor will automatically extract attribute sets within the generated XSL-FO. The extracted sets are placed in an extra FO block, which can be referenced. This improves processing performance and reduces file size.</p> <p>Valid values are:</p> <ul style="list-style-type: none"> <li>• Enable - extract attribute sets for all templates and subtemplates</li> <li>• Auto - extract attribute sets for templates, but not subtemplates</li> <li>• Disable - do not extract attribute sets</li> </ul> <p><b>Internal Name:</b> rtf-extract-attribute-sets</p>
Enable XPath rewriting	<p><b>Default:</b> true</p> <p><b>Description:</b> When converting an RTF template to XSL-FO, the RTF processor will automatically rewrite the XML tag names to represent the full XPath notations. Set this property to "false" to disable this feature.</p> <p><b>Internal Name:</b> rtf-rewrite-path</p>
Characters used for checkbox	<p><b>Default:</b> Albany WT J;9746;9747/A</p> <p><b>Description:</b> The BI Publisher default PDF output font does not include a glyph to represent a checkbox. If your template contains a checkbox, use this property to define a Unicode font for the representation of checkboxes in your PDF output. You must define the Unicode font number for the "checked" state and the Unicode font number for the "unchecked" state using the following syntax: <code>fontname;&lt;unicode font number for true value's glyph &gt;;&lt;unicode font number for false value's glyph&gt;</code></p> <p>Example: Albany WT J;9746;9747/A</p> <p>Note that the font that you specify must be made available to BI Publisher at runtime.</p> <p><b>Internal Name:</b> rtf-checkbox-glyph</p>

## PDF Template Properties

The following properties can be set to govern PDF templates:

Property Name	Description
Remove PDF fields from output	<p><b>Default:</b> false</p> <p><b>Description:</b> Specify "true" to remove PDF fields from the output. When PDF fields are removed, data entered in the fields cannot be extracted. For more information, see Setting Fields as Updateable or Read Only, page 7-16.</p> <p><b>Internal Name:</b> remove-pdf-fields</p>
Set all fields as read only in output	<p><b>Default:</b> true</p> <p><b>Description:</b> By default, BI Publisher sets all fields in the output PDF of a PDF template to be read only. If you want to set all fields to be updateable, set this property to "false". For more information, see Setting Fields as Updateable or Read Only, page 7-16.</p> <p><b>Internal Name:</b> all-field-readonly</p>
Maintain each field's read only setting	<p><b>Default:</b> false</p> <p><b>Description:</b> Set this property to "true" if you want to maintain the "Read Only" setting of each field as defined in the PDF template. This property overrides the settings of "Set all fields as read only in output." For more information, see Setting Fields as Updateable or Read Only, page 7-16.</p> <p><b>Internal Name:</b> all-fields-readonly-asis</p>

## Flash Template Properties

The following properties can be set to govern Flash templates:

Property Name	Description
Page width of wrapper document	<p><b>Default:</b> 792</p> <p><b>Description:</b> Specify in points the width of the output PDF document. The default is 792, or 11 inches.</p> <p><b>Internal Name:</b> flash-page-width</p>
Page height of wrapper document	<p><b>Default:</b> 612</p> <p><b>Description:</b> Specify in points the height of the output PDF document. The default is 612, or 8.5 inches</p> <p><b>Internal Name:</b> flash-page-height</p>

Property Name	Description
Start x position of Flash area in PDF	<p><b>Default:</b> 18</p> <p><b>Description:</b> Using the left edge of the document as the 0 axis point, specify in points the beginning horizontal position of the Flash object in the PDF document. The default is 18, or .25 inch</p> <p><b>Internal Name:</b> <code>flash-startx</code></p>
Start y position of Flash area in PDF	<p><b>Default:</b> 18</p> <p><b>Description:</b> Using the upper left corner of the document as the 0 axis point, specify in points the beginning vertical position of the Flash object in the PDF document. The default is 18, or .25 inch.</p> <p><b>Internal Name:</b> <code>flash-starty</code></p>
Width of Flash area	<p><b>Default:</b> Same as flash width in points in swf</p> <p><b>Description:</b> Enter in points the width of the area in the document for the Flash object to occupy. The default is the width of the SWF object.</p> <p><b>Internal Name:</b> <code>flash-width</code></p>
Height of Flash area	<p><b>Default:</b> Same as flash height in points in swf</p> <p><b>Description:</b> Enter in points the height of the area in the document for the Flash object to occupy. The default is the height of the SWF object.</p> <p><b>Internal Name:</b> <code>flash-height</code></p>

## CSV Output Properties

Use the following properties to control comma-separated value output:

Property Name	Description
CSV delimiter	<p><b>Default:</b> ,</p> <p><b>Description:</b> Specifies the character used to delimit the data in comma-separated value output. Other options are: Semicolon (;), Tab (\t) and Pipe ( ).</p>

Property Name	Description
Remove leading and trailing white space	<p><b>Default:</b> false</p> <p><b>Description:</b> Specify "True" to remove leading and trailing white space between data elements and the delimiter.</p>

## Excel 2007 Output Properties

Use the following properties to control Excel 2007 output:

Property Name	Description
Show grid lines	<p><b>Default:</b> false</p> <p><b>Description:</b> Set to true to show the Excel table grid lines in the report output.</p>
Page break as a new sheet	<p><b>Default:</b> true</p> <p><b>Description:</b> When set to "True" a page break specified in the report template will generate a new sheet in the Excel workbook.</p>
Minimum column width	<p><b>Default:</b> 3 (in points, 0.04 inch)</p> <p><b>Description:</b> When the column width is less than the specified minimum and it contains no data, the column will be merged with the preceding column.</p>
Minimum row height	<p><b>Default:</b> 1 (in points, 0.01 inch)</p> <p><b>Description:</b> When the row height is less than the specified minimum and it contains no data, the row will be removed.</p>

## All Outputs



Property Name	Description
Hide version number in output	<p><b>Default:</b> true</p> <p><b>Description:</b> Some report output documents display Oracle BI Publisher in the document properties. For example, PDF documents identify Oracle BI Publisher as the PDF Producer in the properties for the document. If you wish to include the version of BI Publisher (for example, Oracle BI Publisher 11.1.1.5.0) that generated the document, set this property to false.</p>

## Defining Font Mappings

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

There are two types of font mappings:

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

## Making Fonts Available to BI Publisher

BI Publisher provides a set of Type1 fonts and a set of TrueType fonts. You can select any of the fonts in these sets as a target font with no additional setup required. For a list of the predefined fonts see BI Publisher's Predefined Fonts, page 10-20.

The predefined fonts are located in the Oracle Business Intelligence Oracle home, in: *ORACLE\_HOME/common/fonts*. If you wish to map to another font, you must place the font in this directory to make it available to BI Publisher at runtime. If your environment is clustered, you must place the font on every server.

**Note:** The font location is set by the XDO\_FONT\_DIR variable. If this variable is not set in your environment the fonts will be located in *\$JAVA\_HOME/jre/lib/fonts*.

## Setting Font Mapping at the Site Level or Report Level

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

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

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

## Creating a Font Mapping

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

### To create a Font Mapping

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

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

For a list of the predefined fonts see BI Publisher's Predefined Fonts, page 10-20

## BI Publisher's Predefined Fonts

The following Type1 fonts are built-in to Adobe Acrobat and BI Publisher provides a mapping for these fonts by default. You can select any of these fonts as a target font with no additional setup required.

The Type1 fonts are listed in the following table:

### Type 1 Fonts

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

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

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

Number	Font Family Name	Style	Weight	Actual Font	Actual Font Type
1	Albany WT	normal	normal	ALBANYWT.ttf	TrueType (Latin1 only)
2	Albany WT J	normal	normal	ALBANWTJ.ttf	TrueType (Japanese flavor)
3	Albany WT K	normal	normal	ALBANWTK.ttf	TrueType (Korean flavor)
4	Albany WT SC	normal	normal	ALBANWTS.ttf	TrueType (Simplified Chinese flavor)
5	Albany WT TC	normal	normal	ALBANWTT.ttf	TrueType (Traditional Chinese flavor)

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

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

### Included Barcode Fonts

BI Publisher also includes the following barcode fonts:

Font File	Supported Algorithm
128R00.TTF	code128a, code128b, and code128c
B39R00.TTF	code39, code39mod43
UPCR00.TTF	upca, upce

For information on using barcode fonts in an RTF template, see [Using the Barcodes Shipped with BI Publisher](#), page 4-126.

# Part 3

---

## Creating and Implementing Style Templates





---

# Creating and Implementing Style Templates

This chapter covers the following topics:

- Understanding Style Templates
- Creating a Style Template RTF File
- Uploading a Style Template File to the Catalog
- Assigning a Style Template to a Report Layout
- Updating a Style Template
- Adding Translations to Your Style Template Definition

## Understanding Style Templates

A style template is an RTF template that contains style information that can be applied to RTF layouts. The style information in the style template is applied to RTF layouts at runtime to achieve a consistent look and feel across your enterprise reports. You associate a style template to a report layout in the report definition. Using a style template has the following benefits:

- Enables the same look and feel across your enterprise reports
- Enables same header and footer content, such as company logos, headings, and page numbering
- Simplifies changing the elements and styles across all reports

## About Styles Defined in the Style Template

The styles of the following elements can be defined in the style template:

### Paragraph and Heading Styles

You can create a paragraph style in your style template. When this same named style is

used in your report layout, the report layout will inherit the following from the style template definition: font family, font size, font weight (normal, bold), font style (normal, italic), font color, and text decoration (underline, overline, or strikethrough).

## Table Styles

Following are some of the style elements inherited from the table style definition: font style, border style, gridline definition, shading, and text alignment.

## Header and Footer Content

The header and footer regions of the style template will be applied to the report layout. This includes images, dates, page numbers, and any other text-based content. If the report layout also includes header and footer content, it will be overwritten.

## Style Template Process Overview

### Design Time

For the Style Template:

1. Open Microsoft Word.
2. Define named styles for paragraphs, tables, headings, and static header and footer content. This is your style template
3. Save this document as a .RTF file.
4. To ensure that you do not lose your custom styles in Microsoft Word, also save the document as a Word Template file (.dot) or save your styles to the Normal.dot file. This file can be shared with other report designers.
5. Upload the RTF style template file to the catalog.

For the layout template using the style template:

1. In your RTF template, use the same named styles for paragraph and table elements that you want to be inherited from the style template.
2. Open the report in BI Publisher's Report Editor and select the style template to associate to the report. Then enable the style template for the specific report layout.

### Runtime

When you run the report with the selected layout, BI Publisher will apply the styles, header, and footer from the style template.

## Creating a Style Template RTF File

The following sections describe how to define the style types in your Microsoft Word document. For more complete information see the Microsoft Word documentation.

### Defining Styles for Paragraphs and Headings

Use a paragraph style to define formatting such as font type, size, color, text positioning and spacing. A paragraph style can be applied to one or more paragraphs. Use a paragraph style to format headings and titles in your report as well.

#### To define a paragraph style type:

1. In your Microsoft Word document, from the Format menu, select Styles and Formatting.
2. From the Styles and Formatting task pane, select New Style.
3. In the New Style dialog, enter a name for your style. Choose style type: Paragraph. Format your style using the options presented in the dialog. To see additional paragraph options (such as font color and text effects), click Format.
4. When finished, click OK and your new style will appear in the list of available formats in the Styles and Formatting task pane.
5. Choose your new style and make an entry in your style template to display the style.

#### To apply the paragraph style type in your document:

1. Position your cursor within the paragraph (or text) to which you wish to apply the style.
2. Select the style from list of available formats in the Styles and Formatting task pane. The style will be applied to the paragraph.

#### To modify an existing style type:

1. In your Microsoft Word document, from the Format menu, select Styles and Formatting.
2. From the Styles and Formatting task pane, select and right-click the style to modify.
3. From the menu, select Modify.

## Defining Styles for Tables

### To define a table style type:

1. In your Microsoft Word document, from the Format menu, select Styles and Formatting.
2. From the Styles and Formatting task pane, select New Style.
3. In the New Style dialog, enter a name for your style. Choose style type: Table. Format your style using the options presented in the dialog. To see additional table options (such as Table Properties and Borders and Shading), click Format.
4. When finished, click OK and your new style will appear in the list of available formats in the Styles and Formatting task pane.
5. Choose your new style and make an entry in your style template to display the style.

### To apply the table style type in your document:

1. Position your cursor within the table to which you wish to apply the style.
2. Select the table style from list of available formats in the Styles and Formatting task pane. The style will be applied to the table.

## Defining a Header and Footer

You can define a header and footer in your style template. The contents and sizing of the header and footer in the style template will be applied to the report layouts.

**Important:** If a header and footer have been defined in the report layout, they will be overwritten. The header and footer from the style template will be applied.

### To define a header and footer:

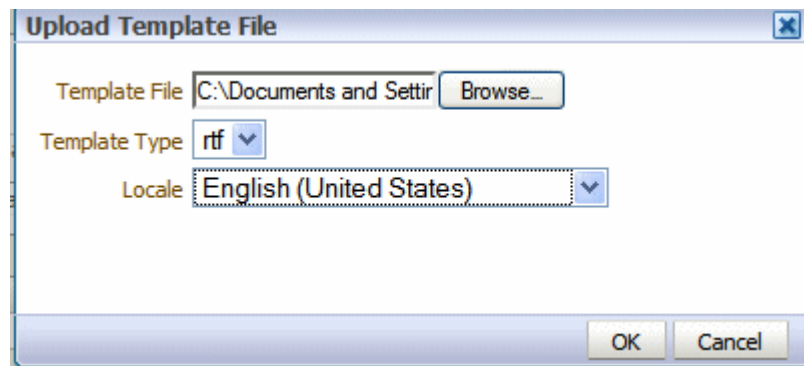
1. In your Microsoft Word document, from the View menu, select Header and Footer.
2. Enter header and footer content. This can include a logo or image file, static text, current date and time stamps, page numbers, or other content supported by Microsoft Word.

## Uploading a Style Template File to the Catalog

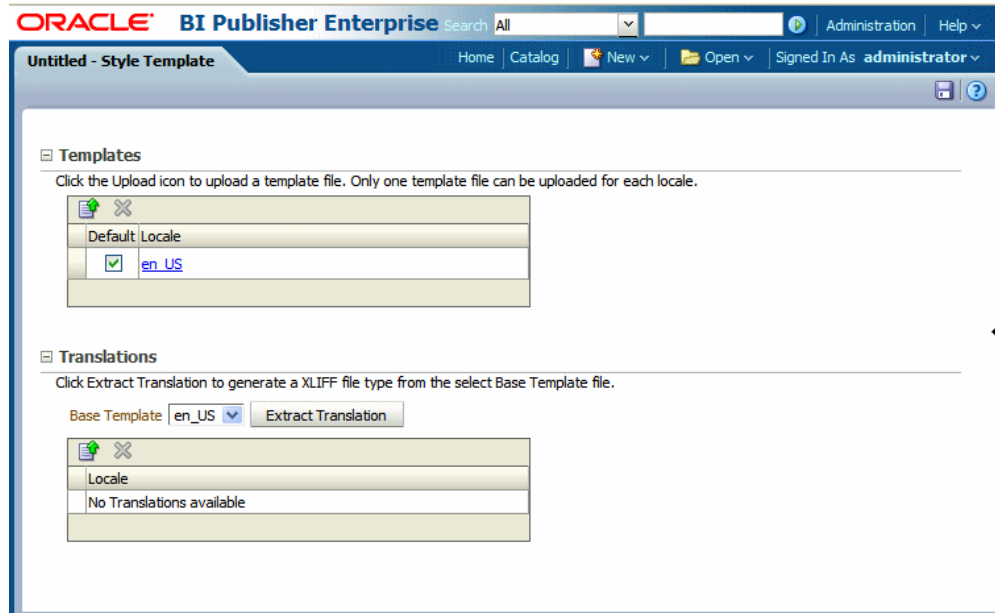
You can place a style template in any folder in the catalog to which you have access. Your organization may have a designated folder for style templates.

### To upload a style template file:

1. On the global header click **New** and then click **Style Template**. This launches an untitled Style Template properties page.
2. From the **Templates** region, click the **Upload** toolbar button.
3. In the **Upload Template File** dialog, use the **Browse** button to select your Template File. Select **rtf** as the **Type**, and select the appropriate **Locale**. Then click **OK**.



Your style template file will display in the Templates region as the locale name you selected (for example: en\_US).



4. Click **Save**.
5. In the **Save As** dialog choose the catalog folder in which to save the style template. Enter the **Name** and click **Save**.

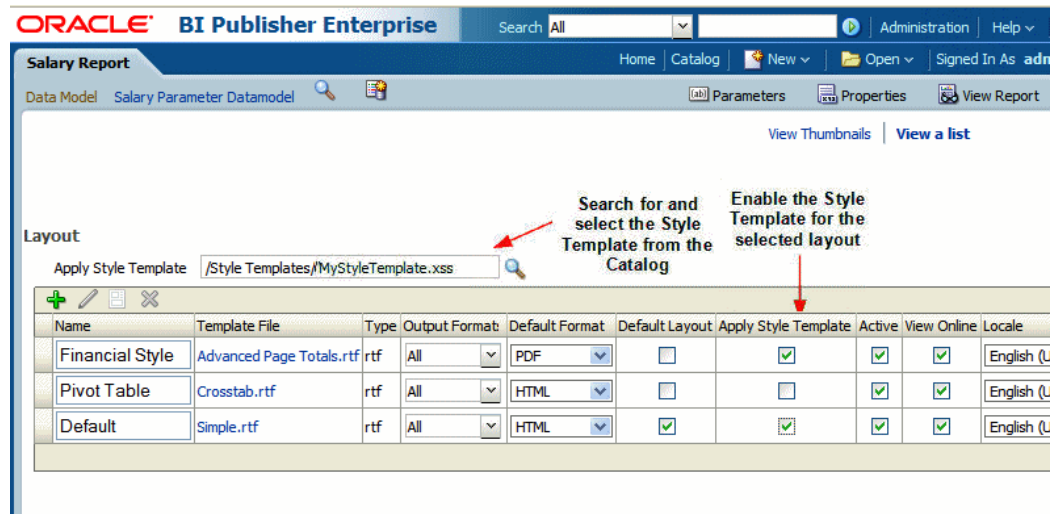
**Note:** You may only upload one RTF file per locale to a Style Template definition. If you upload additional template files to this Style Template, each file will be automatically named as the locale regardless of the name you give the file before upload.

6. If you are uploading multiple localized files, select the file that is to be used as the default. For more information on localization of template files see Adding Translations to Your Style Template, page 11-7.

## Assigning a Style Template to a Report Layout

1. Navigate to the report in the catalog and click **Edit** to open the report editor.
2. From the default thumbnail view, select **View a List**. In the Layout Region, click the **Choose** icon to search for and select your style template from the BI Publisher catalog. .
3. For the layout templates that you wish to use the style template, select the **Apply Style Template** box for the template. Note that the box is only enabled for RTF templates.

The following figure shows the actions required to enable a style template in the Report Editor.



## Updating a Style Template

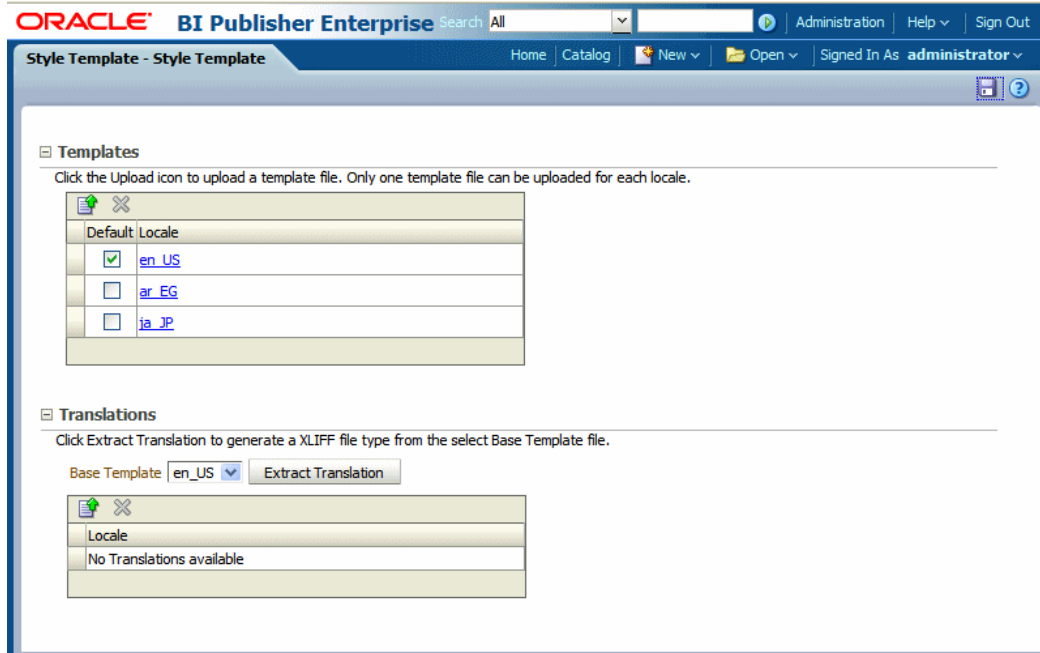
To update or edit a saved style template:

1. Navigate to the file in the catalog.
2. Click **Edit** to open the Style Template properties page.
3. Delete the existing file.
4. Upload the edited file, choosing the same locale.

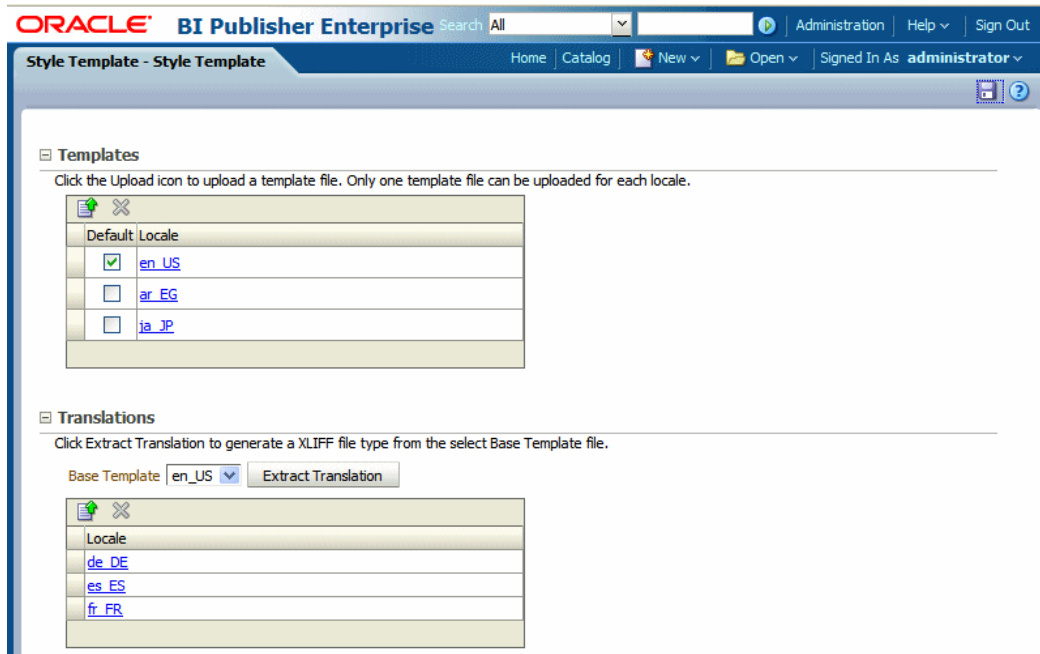
## Adding Translations to Your Style Template Definition

Style templates offer the same support for translations as RTF template files.

You can upload multiple translated RTF files under a single Style Template definition and assign the appropriate locale. These will display in the Templates region, as shown in the following figure:



Or you can generate an XLIFF (.xlf) file of the translatable strings, translate the strings, and upload the translated file. These will display in the Translations region, as shown in the following figure:



At runtime, the appropriate style template is applied based on the user's account Preference setting for Report Locale for reports viewed online; or, for scheduled reports, based on the user's selection for Report Locale for the scheduled report.



The XLIFF files for style templates can be generated individually, then translated, and uploaded individually. Or, if you perform a catalog translation that includes the style template folders, the strings from the style template files will be extracted and included in the larger catalog translation file. When the catalog translation file is uploaded to BI Publisher, the appropriate translations from the catalog file will display in the Translations region of the Style Template definition.

For more information on translations, see the section *Translating Reports and Catalog Objects*, page 15-1.



# Part 4

---

## Creating and Implementing Sub Templates



---

## Understanding Subtemplates

This chapter covers the following topics:

- What is a Subtemplate
- Supported Locations for Subtemplates
- Designing a Subtemplate
- Testing Subtemplates from the Desktop
- Creating the Sub Template Object in the Catalog
- Calling a Subtemplate from an External Source

### What is a Subtemplate

A Subtemplate is a piece of formatting functionality that can be defined once and used multiple times within a single layout template or across multiple layout template files. This piece of formatting can be in an RTF file format or an XSL file format. RTF subtemplates are easy to design as you can use Microsoft Word native features. XSL subtemplates can be used for complex layout and data requirements.

Some common uses for subtemplates include:

- Reusing a common layout or component (such as a header, footer, or address block)
- Handling parameterized layouts
- Handling dynamic or conditional layouts
- Handling lengthy calculations or reusing formulae

### About RTF Subtemplates

An RTF subtemplate is an RTF file that consists of one or more `<?template:??>` definitions, each containing a block of formatting or commands.

This RTF file, when uploaded to BI Publisher as a Sub Template object in the Catalog, can be called from within another RTF Template.

## About XSL Subtemplates

An XSL subtemplate is an XSL file that contains formatting or processing commands in XSL for the BI Publisher formatting engine to execute. Use an XSL template to include complex calculations or formatting instructions not supported by the RTF standard.

This XSL file, when uploaded to BI Publisher as a Sub Template object in the Catalog, can be called from within an RTF Template.

## Supported Locations for Subtemplates

It is recommended that you upload your subtemplates to the BI Publisher catalog. This is the most secure location.

For compatibility with older versions of BI Publisher, it is also possible to call a subtemplate that resides in a file on the local server, or on a different server (that can be accessed by HTTP protocol). Using one of these methods requires specific import syntax and server settings to allow the communication. See *Calling a Subtemplate from an External Source*, page 12-4 for more information.

## Designing a Subtemplate

For information on designing an RTF subtemplate, see *Designing RTF Subtemplates*, page 13-1.

For information on designing an XSL subtemplate, see *Designing XSL Subtemplates*, page 14-1.

## Testing Subtemplates from the Desktop

If you have the BI Publisher Template Builder installed, you can preview the template and subtemplate combination before uploading them to the BI Publisher catalog. To test from your local environment, you must alter the import template syntax to enable the BI Publisher processor to locate the subtemplate file on your local directory. To test, enter the import template syntax as follows:

```
<?import:file:{local_template_path}?>
```

For example:

```
<?import:file:C:///Template_Directory/subtemplate_file.rtf?>
```

or for an XSL subtemplate file:

```
<?import:file:C:///Template_Directory/subtemplate_file.xsl?>
```

You can then select the **Preview** option in the Template Builder and the BI Publisher

processor can locate your subtemplate and render it from your local environment.

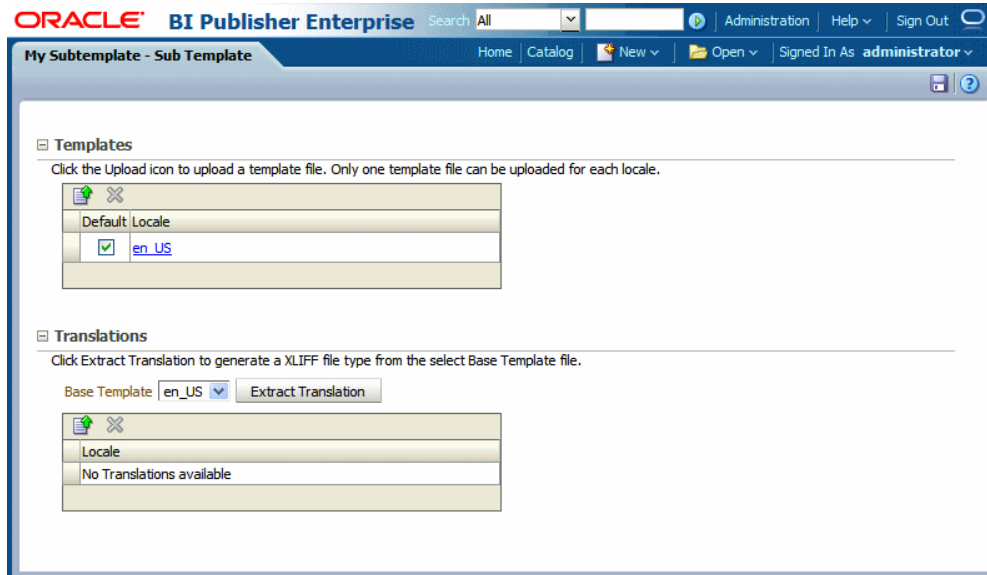
Note that before you upload the primary template to the BI Publisher catalog, you must change the import syntax to point to the appropriate location in the catalog.

## Creating the Sub Template Object in the Catalog

To upload your subtemplate file:

1. On the global header click **New** and then click **Sub Template**. This launches an untitled Sub Template page.
2. In the **Templates** region, click **Upload** to launch the **Upload Template File** dialog.
3. Browse for and select your subtemplate file.
  - **Type:** Select rtf for RTF subtemplate files or xsl for XSL subtemplate files.
  - **Locale:** Select the appropriate locale for the subtemplate file.
4. Click **Upload**.

Your subtemplate file will display in the **Templates** region as the locale name you selected (for example: en\_US).
5. Click **Save**. In the **Save As** dialog choose the catalog folder in which to save the Sub Template. Enter the **Name** and click **Save**. The following figure shows a Sub Template named "My Subtemplate":



- (RTF Sub Templates only) If you are uploading multiple localized files, select the file that is to be used as the default. For more information on localization of template files, see Adding Translations to Your Sub Template, page 11-7.

**Note:** You may upload only one RTF file per locale to a Sub Template definition. If you upload additional template files to this Sub Template, each file will be automatically named as the locale regardless of the name you give the file before upload.

**Important:** Translations are not supported for XSL Sub Templates.

Note that the Sub Template object is saved with the extension ".xsb". You will use the Name that you choose here with the .xsb extension when you import the Sub Template object (for example: MySubtemplate.xsb).

## Calling a Subtemplate from an External Source

This section describes how to call a subtemplate that resides outside the catalog.

**Important:** These instructions are provided for backward compatibility only. It is recommended that you place your subtemplates in the catalog.

Note that localization is not supported for subtemplates that are maintained outside the catalog.



## Importing a Subtemplate Outside the Catalog over HTTP or FTP

Use a standard protocol, such as http or ftp and enter the import statement as follows:

```
<?import:http://myhost:8080/subtemplate.rtf?>
```

## Importing Subtemplates Outside the Catalog on the Same Server

If your subtemplate is located on the server, but not in the BI Publisher catalog, enter the following:

```
<?import:file://{template_path}?>
```

where

*template\_path* is the path to the subtemplate file on your server

For example:

```
<?import:file://c:/Folder/mySubtemplate.rtf?>
```

## Required Settings To Run Subtemplates Stored Outside the Catalog

Using subtemplates requires the following FO processing configuration property setting for the report:

- Disable external references: Must be set to False



---

## Designing RTF Subtemplates

This chapter covers the following topics:

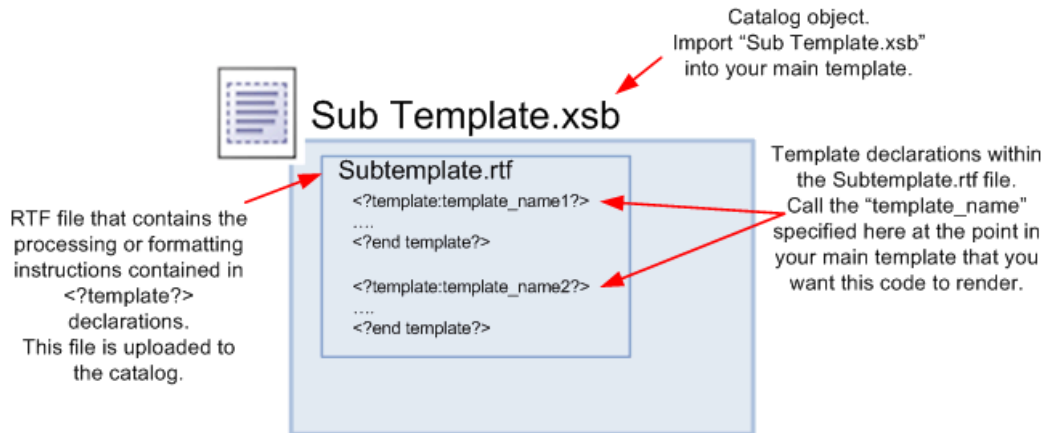
- Understanding RTF Subtemplates
- Process Overview for Creating and Implementing RTF Subtemplates
- Creating an RTF Subtemplate File
- Calling a Subtemplate from Your Main Template
- When to Use RTF Subtemplates
- Adding Translations to an RTF Subtemplate

### Understanding RTF Subtemplates

An RTF subtemplate is an RTF file that consists of one or more `<?template:??>` definitions, each containing a block of formatting or commands.

This RTF file, when uploaded to BI Publisher as a Sub Template object in the Catalog, can be called from other RTF templates.

The following figure illustrates the composition of an RTF Sub Template:



## Process Overview for Creating and Implementing RTF Subtemplates

Using a subtemplate consists of the following steps (described in the following sections):

1. Create the RTF file containing the common components or processing instructions that you wish to include in other templates.
2. Create the calling or "main" layout and include the following two commands:
  - `import` - to import the subtemplate file to the main layout template.
  - `call-template` - to execute or render the subtemplate contents in the main layout.
3. Test your template and subtemplate.

**Tip:** You can use the BI Publisher Desktop Template Viewer to test your main layout plus subtemplate before loading them to the catalog. To do so, you must alter the `import` template syntax to point to the location of the subtemplate in your local environment. See Testing Subtemplates from the Desktop, page 12-2.
4. Upload the main template to the report definition and create the Sub Template object in the catalog. See Creating the Sub Template Object in the Catalog, page 12-3

## Creating an RTF Subtemplate File

Enter the components or instructions in an RTF file. To define the instructions as a subtemplate, enclose the contents in the following tags:

```
<?template:template_name?>
  ..subtemplate contents...
<?end template?>
```

where


template\_name is the name you choose for the subtemplate.

Note that in a single RTF file, you can have multiple

```
<?template:template_name?>
<?end template?>
```

entries, to mark different "subtemplates" or segments to include in other files.

For example, the following sample RTF file contains two subtemplates, one named commonHeader and one named commonFooter:

<code>&lt;?template:commonHeader?&gt;</code>	
	Report Date: 17 August 2009
<code>&lt;?end template?&gt;</code>	
<code>&lt;?template:commonFooter?&gt;</code>	
Oracle Corporation 600 Oracle Parkway Redwood Shores CA 94065	Page 1 of 1
<code>&lt;?end template?&gt;</code>	

## Calling a Subtemplate from Your Main Template

To implement the subtemplate in your main template, you must make two entries in your main template:

First, import the subtemplate file to the main template. The import syntax tells the BI Publisher engine where to find the Sub Template in the catalog.

Second, enter a call command to render the contents of the subtemplate at the position desired.

## Importing the Subtemplate to the Main Template

Enter the import command anywhere in the main template prior to the call template command:

- If you do not require a locale, enter the following:

```
<?import:xdoxsl:/// {path to subtemplate.xsb}?>
```

where

*path to subtemplate.xsb* is the path to the subtemplate .xsb object in the catalog.

For example:

```
<?import:xdoxsl:///Executive/HR_Reports/mySubtemplate.xsb?>
```

## Calling the Subtemplate to Render Its Contents

1. In the position in the main template where you want the subtemplate to render, enter the call-template command, as follows:

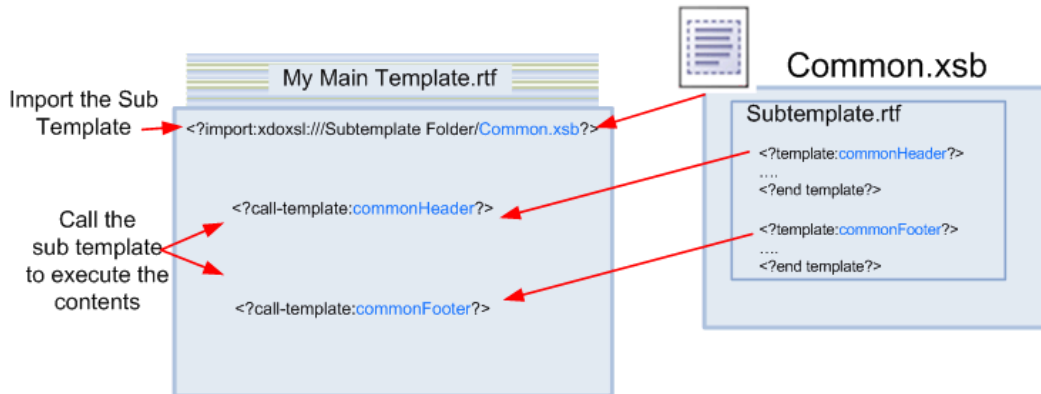
```
<?call-template:template_name?>
```

where

*template\_name* is the name you assigned to the contents in the template declaration statement within the subtemplate file (that is, the

`<?template:template_name?>` statement).

The following figure illustrates the entries required in your main template:



## Importing a Localized Subtemplate

To designate the locale of the imported subtemplate, append the locale to the import statement as follows:

```
<?import:xdoxsl:///{path to subtemplate.xsb}?loc={locale_name}?>
```

where

*path to subtemplate.xsb* is the path to the subtemplate .xsb object in the catalog

and

*locale\_name* is the language-territory combination which comprises the locale. The locale designation is optional.

For example:

```
<?import:xdoxsl:///Executive/HR_Reports/mySubtemplate.xsb?loc=en-US?>
```

Note that you can also use `$_XDOLOCALE` to import a localized subtemplate based on the runtime user locale. For example:

```
<?import:xdoxsl:///Executive/HR_Reports/mySubtemplate.xsb?loc=$_XDOLOCALE?>
```

## Example

In this example, your company address is a fixed string that appears in all your templates. Rather than reproduce the string in all your templates, you can place it in one subtemplate and reference it from all the others.

1. In an RTF file enter the following template declaration:

```
<?template:MyAddress?>
My Company
500 Main Street
Any City, CA 98765
<?end template?>
```

2. Create a Sub Template in the catalog in the following location: Customer Reports/Templates.
3. Upload this file to the Sub Template and save it as "Common Components" (BI Publisher will assign the object the .xsb extension).
4. In your main template, enter the following import statement in a form field or directly in the template:

```
<?import:xdoxsl:///Customer Reports/Templates/Common
Components.xsb?>
```

5. In your main template, in the location you want the address to appear, enter

```
<?call-template:MyAddress?>
```

At runtime the contents of the MyAddress subtemplate will be fetched and rendered in the layout of the main template.

This functionality is not limited to just strings, you can insert any valid RTF template functionality in a subtemplate, and even pass parameters from one to the other. For examples, see the next section: When to Use RTF Subtemplates, page 13-5.

## When to Use RTF Subtemplates

Following are several common use-cases for RTF subtemplates.

### Reusing a Common Layout

Frequently multiple reports require the same header and footer content. By using an RTF subtemplate to contain this content, any global changes will be simplified and

require updating only the subtemplate instead of each individual layout.

## Conditionally Displaying a Layout Based on a Value in the Data

Subtemplates can also be used to apply conditional layouts based on a value in the report data.

By using the RTF template "choose" command, you can instruct BI Publisher to apply a different `<?template?>` defined in your subtemplate file.

**IMPORTANT:** You cannot conditionalize the import statement for the subtemplate file. Instead, you import one subtemplate file and conditionalize the call statement. You define the multiple `<?template?>` options in the single subtemplate file.

### Example

Assume you have a report that will be sent to customers in India and the United States. You need to apply a different address layout depending on the country code (COUNTRY\_CODE) supplied in the data. This example uses the RTF templates "if" statement functionality to call the subtemplate with the appropriate address format.

Your subtemplate file may look like the following:

```
<?template:US_Address?>
  <?US_Address_Field1?>
  <?US_Address_Field2?>
  <?US_Address_Field3?>
<?end template?>

<?template:IN_Address?>
  <?IN_Address_Field1?>
  <?IN_Address_Field2?>
  <?IN_Address_Field3?>
<?end template?>
```

1. Create a Sub Template in the catalog in the following location:  
Customers/Invoice Reports  
Upload the RTF file and save the Sub Template as "Addresses".
2. In your main template enter the following to import the Sub Template:  
`<?import:xdoxsl:///Customers/Invoice Reports/Addresses.xsb?>`
3. In the location where you want the address to display, enter the following:  

```
<?if:COUNTRY_CODE='USA'?>
  <?call:US_Address?>
<?end if?>
<?if:COUNTRY_CODE='IN'?>
  <?call:IN_Address?>
<?end if?>
```

When the report is run, the address format will be properly applied depending on the value of COUNTRY\_CODE in the data.



## Conditionally Displaying a Layout Based on a Parameter Value

This example illustrates how to display a different layout based on a user parameter value or a selection from a list of values. The parameter can be passed to the RTF template and used to call a different `<?template?>` within the subtemplate file based on the value.

**IMPORTANT:** You cannot conditionalize the import statement for the subtemplate file.

### Example

Assume in your report data model you have defined a parameter named DeptName. You set up this parameter as type Menu and associated it to a list of values, enabling your user to make a selection from the list when he views the report in the Report Viewer (or when he schedules the report).

In your RTF main layout template, enter the following command to capture the value chosen by the user:

```
<?param@begin:DeptName?>
```

To display the layout based on this user selection, you can use an IF statement or a CHOOSE statement to evaluate the parameter value and call the associated subtemplate.

Use the CHOOSE statement when there are many conditional tests and a default action is expected for the rest of the values. For example, the Accounting, Sales, and Marketing departments each require a different layout. All other departments can use a default layout.

1. Create an RTF file and include the following template declarations:

```
<?template:tAccounting?>
- - - Specific Accounting Layout here - - -
<?end template?>

<?template:tSales?>
- - - Specific Sales Layout here - - -
<?end template?>

<?template:tMark?>
- - - Specific Marketing Layout here - - -
<?end template?>

<?template:tDefault?>
- - - Default Layout here - - -
<?end template?>
```

2. Create a Sub Template in the catalog in the following location:

Shared Folders/Executive/Department Expenses

Upload the RTF file and save the Sub Template as "DeptSubtemps".

3. In your main RTF template include the following commands:

```

<?import:xdoxsl:///Executive/Department
Expenses/DeptSubtemps.xsb?loc=en-US?>

<?param@begin:DeptName?>

<?choose:??>
  <?when:$DeptName='Accounting'?>
    <?call:tAccounting?>
  <?end when?>
  <?when:$DeptName='Sales'?>
    <?call:tSales?>
  <?end when?>
  <?when:$DeptName='Marketing'?>
    <?call:tMark?>
  <?end when?>
  <?otherwise:$?>
    <?call:tDefault?>
  <?end otherwise?>
<?end choose:??>

```

When the user runs the report, the layout applied will be determined based on the value of DeptName. For more information on CHOOSE statements in RTF templates, see Choose Statements, page 4-67.

## Handling Simple Calculations or Repeating Formulae

Simple calculations can also be handled using an RTF subtemplate. More complex formulae should be handled with an XSL subtemplate.

### Example

This example illustrates setting up a subtemplate to contain a formula to calculate interest.

The subtemplate will perform the interest calculation on the data in this report and pass the result back to the main template. The subtemplate will accommodate the possibility that multiple reports that call this functionality may have different tag names for the components of the formula.

Assume you have the following XML data:

```

<LOAN_DATA>
  <LOAN_AMOUNT>6000000</LOAN_AMOUNT>
  <INTEREST_RATE>.053</INTEREST_RATE>
  <NO_OF_YEARS>30</NO_OF_YEARS>
</LOAN_DATA>

```

1. In an RTF file, create a template declaration called "calcInterest". In this subtemplate you will define a parameter for each of the elements (principal, interest rate, and years) in the formula. Note that you must set the default value for each parameter.

```

<?template:calcInterest?>
  <?param:principal;0?>
  <?param:intRate;0?>
  <?param:years;0?>
  <?number($principal) * number($intRate) * number($years)?>
<?end template?>

```

2. Create a Sub Template in the catalog in the following location:

Shared Folders/Subtemplates

Upload the RTF file and save the Sub Template as "calculations".

3. In your main template enter the following to import the subtemplate:

```
<?import:xdoxsl:///Subtemplates/calculations.xsb?>
```

4. In the location where you want the results of the calculation to display, enter the following in a BI Publisher field:

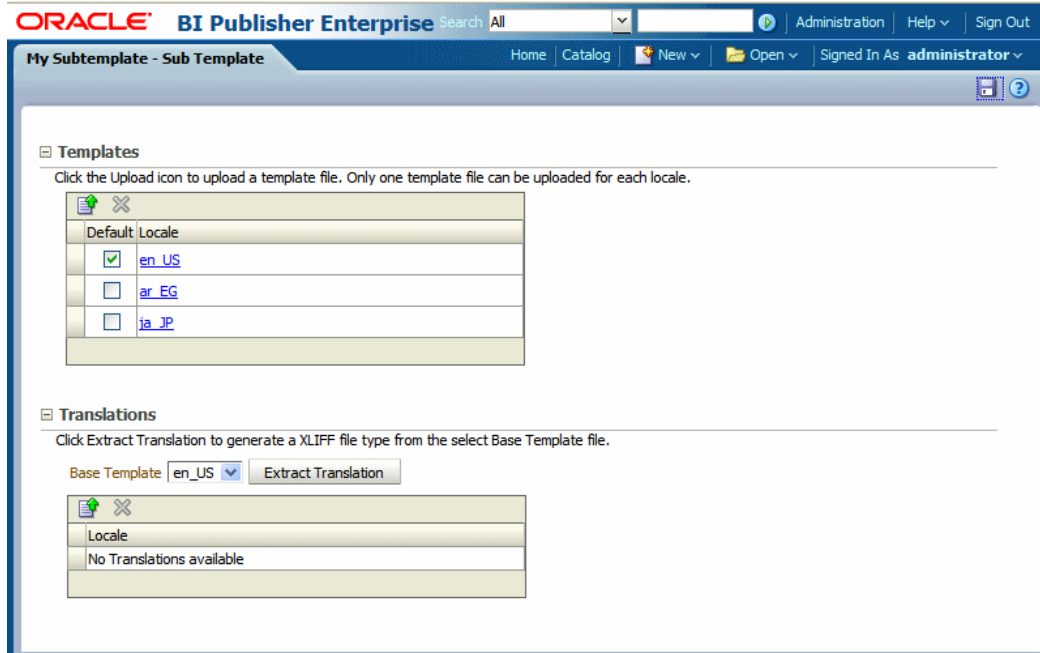
```
<?call@inlines:calcInterest?>  
  <?with-param:principal;./LOAN_AMOUNT?>  
  <?with-param:intRate;./INTEREST_RATE?>  
  <?with-param:years;./NO_OF_YEARS?>  
<?end call?>
```

Note the use of the `@inlines` command here. This is optional. The `@inlines` command forces the results to be rendered inline at the location in the template where the call to the subtemplate is made. Use this feature, for example, if you want to keep the results on the same line as a string of text that precedes the call.

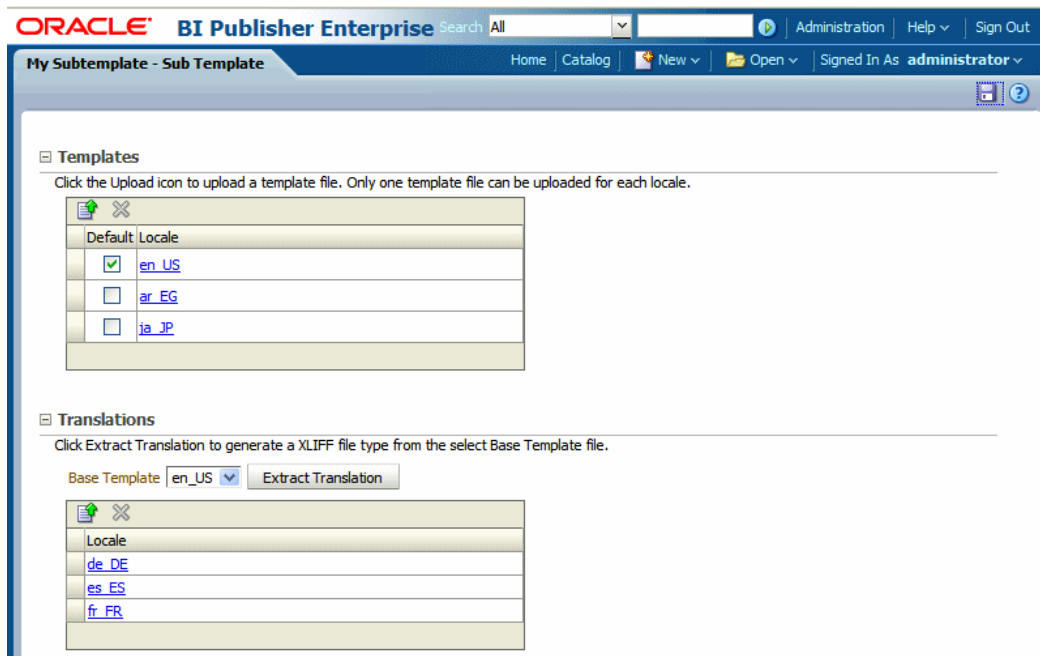
## Adding Translations to an RTF Subtemplate

RTF subtemplates offer the same support for translations as RTF template files.

You can upload multiple translated RTF files under a single Sub Template definition and assign the appropriate locale. These will display in the Templates region, as shown in the following figure:



Or you can generate an XLIFF (.xlf) file of the translatable strings, translate the strings, and upload the translated file. These will display in the Translations region, as shown in the following figure:



At runtime, the appropriate subtemplate localization is applied based on the user's account Preference setting for Report Locale for reports viewed online; or, for scheduled reports, based on the user's selection for Report Locale for the scheduled report.

The XLIFF files for subtemplates can be generated individually, then translated, and uploaded individually. Or, if you perform a catalog translation that includes the Sub Template folders, the strings from the subtemplate files will be extracted and included in the larger catalog translation file. When the catalog translation file is uploaded to BI Publisher, the appropriate translations from the catalog file will display in the Translations region of the Sub Template definition.

For more information on translations, see the section *Translating Reports and Catalog Objects*, page 15-1.



---

## Designing XSL Subtemplates

This chapter covers the following topics:

- Understanding XSL Subtemplates
- Process Overview for Creating and Implementing XSL Subtemplates
- Creating an XSL Subtemplate File
- Calling an XSL Subtemplate from Your Main Template
- Creating the Sub Template Object in the Catalog
- Example Uses of XSL Subtemplates

### Understanding XSL Subtemplates

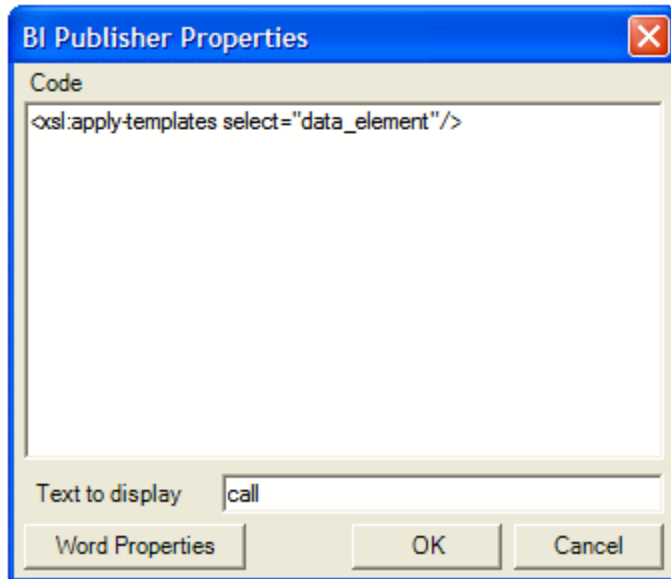
An XSL subtemplate is an XSL file that consists of one or more `<xsl:template>` definitions, each containing a block of formatting or processing commands.

This XSL file, when uploaded to BI Publisher as a Sub Template object in the Catalog, can be called from other RTF templates to execute the formatting or processing commands.

XSL subtemplates empower the report designer to handle complex data and layout requirements. You can transform the data structure for a section of a report (for example, for a chart) or you can create a style sheet to manage a complex layout.

### Where to Put XSL Code in the RTF Main Template

When you call the XSL subtemplate within your main RTF subtemplate you use XSL commands. You must put this code inside a BI Publisher field (or Microsoft Word form field). You cannot enter XSL code directly in the body of your RTF template. The BI Publisher properties dialog is shown in the following figure:



For more information on inserting form fields in an RTF template see:

Inserting a Field, page 5-9 in the chapter: Creating RTF Templates Using the Template Builder for Word

or

Inserting a Form Field, page 4-9 in the chapter: Creating RTF Templates.

## Process Overview for Creating and Implementing XSL Subtemplates

Creating and implementing an XSL subtemplate consists of the following steps:

1. Create the XSL file containing the common components or processing instructions that you wish to include in other templates.  
  
XSL sub template consists of one or more XSL template definitions. These templates contain rules to apply when a specified node is matched.
2. Create the calling or "main" layout and include a command to "import" the subtemplate to the main template, and a command to apply the XSL subtemplate to the appropriate data element.
3. Upload the main template to your report definition and create the Sub Template object in the catalog.

## Creating an XSL Subtemplate File

Enter the instructions in an editor that will enable you to save the file as type ".xml". An XSL subtemplate consists of one or more XSL template definitions. These templates



contain rules to apply when a specified node is matched.

The syntax of the subtemplate definition is as follows:

```
<xsl:template
  name="name"
  match="pattern"
  mode="mode"
  priority="number">
<!--Content: (<xsl:param>*, template) -->
</xsl:template>
```

The following table describes the components of the template declaration:

Component	Description
xsl:template	The xsl:template element is used to define a template that can be applied to a node to produce a desired output display.
name="name"	Optional. Specifies a name for the template. Note: If this attribute is omitted, a match attribute is required.
match="pattern"	Optional. The match pattern for the template. Note: If this attribute is omitted a name attribute is required.
priority="number"	Optional. A number which indicates the numeric priority of the template. It is possible that more than one template can be applied to a node. The highest priority value template is always chosen. The value ranges from -9.0 to 9.0.

Example:

```
<xsl:template match="P|p">
  <fo:block white-space-collapse="false" padding-bottom="3pt"
  linefeed-treatment="preserve">
    <xsl:apply-templates select="text()|*|@*" />
  </fo:block>
</xsl:template>

<xsl:template match="STRONG|B|b">
  <fo:inline font-weight="bold">
    <xsl:apply-templates />
  </fo:inline>
</xsl:template>
```

## Calling an XSL Subtemplate from Your Main Template

To implement the subtemplate in your main template, you must make two entries in your main template:

First, import the subtemplate file to the main template. The import syntax tells the BI Publisher engine where to find the Sub Template in the catalog.

Second, enter a call command to render the contents of the subtemplate at the position desired.

## Importing the Subtemplate

Enter the import command anywhere in the main template prior to the call template command as follows:

```
<?import:xdoxsl:://{path to subtemplate.xsb}?>
```

where

*path to subtemplate.xsb* is the path to the subtemplate .xsb object in the catalog.

For example:

```
<?import:xdoxsl:///Executive/Financial Reports/mySubtemplate.xsb?>
```

## Calling the Subtemplate

The template statements that you defined within the XSL subtemplate file are applied on data elements. There are two ways you can call a template defined in the imported XSL subtemplate:

- By matching the data content with the match criteria:

```
<xsl:apply-templates select="data_element"/>
```

This will apply all the templates defined in the XSL subtemplate to the `data_element` specified. Based on the data content of `data_element`, appropriate functions in those templates will be applied. See the following use case for a detailed example:

- Handling XML Data with HTML Formatting, page 14-6
- By calling a template by name:

```
<xsl:call-template name="templateName"/>
```

This will simply call the template by name and the template will execute similar to a function call. Here also parameters can be passed to the template call similar to an RTF subtemplate. See the next section: Passing Parameters to an XSL Subtemplate, page 14-5.

See the following use case for a detailed example:

- Dynamically Applying Formatting to a Portion of Data, page 14-9

## Passing Parameters to an XSL Subtemplate

To pass parameters to the XSL subtemplate, first declare the parameter in the `<xsl:template>` definition, as follows:

```
<xsl:template name="templateName" match="/">
  <xsl:param name="name" />
</xsl:template>
```

Then call this template using the following syntax:

```
<xsl:call-template name="templateName">
  <xsl:with-param name="name" select="expression">
    <!-- Content:template -->
  </xsl:with-param>
</xsl:call-template>
```

## Creating the Sub Template Object in the Catalog

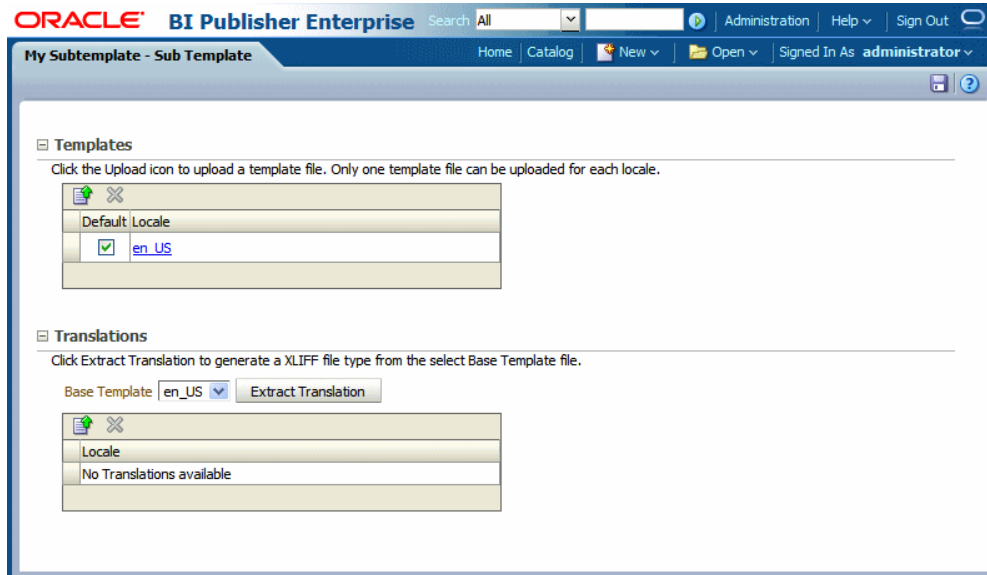
To upload your subtemplate file:

1. On the global header click **New** and then click **Sub Template**. This launches an untitled Sub Template page.
2. In the **Templates** region, click **Upload** to launch the **Upload Template File** dialog.
3. Browse for and select your subtemplate file.
  - **Type**: Select `xsl` for an XSL subtemplate file.
  - **Locale**: Select the appropriate locale for the subtemplate file.

4. Click **Upload**.

Your subtemplate file will display in the **Templates** region as the locale name you selected (for example: `en_US`).

5. Click **Save**. In the **Save As** dialog choose the catalog folder in which to save the Sub Template. Enter the **Name** and click **Save**. The following figure shows a Sub Template named "My Subtemplate":



**Important:** Translations are not supported for XSL Sub Templates.

Note that the Sub Template object is saved with the extension ".xsb". You will use the Name that you choose here with the .xsb extension when you import the Sub Template to your report (for example: MySubtemplate.xsb).

## Example Uses of XSL Subtemplates

The following are examples of formatting that can be achieved in your report by using XSL subtemplates:

- Handling XML Data with HTML Formatting, page 14-6
- Dynamically Applying Formatting to a Portion of Data, page 14-9

## Handling XML Data with HTML Formatting

If you have XML data that already contains HTML formatting and you wish to preserve that formatting in your report, you can preserve that formatting by using an XSL subtemplate to map the HTML formatting commands to XSL equivalents that can be handled by BI Publisher.

Note that the HTML must be in XHTML format. This means that all HTML tags must have start and end tags in the data. For example, if your data uses a simple <BR> for a break, you will have to add the closing </BR> before you can use this solution.

Following is some sample data with HTML formatting:

```

<DATA>
  <ROW>
    <PROJECT_NAME>Project Management</PROJECT_NAME>
    <PROJECT_SCOPE>
      <p>Develop an application to produce <i>executive-level
summaries</i> and detailed project reports. The application will allow
users to: </p>
      <p>Import existing MS Project files </p>
      <p>Allow the user to map file-specific resources to a central
database entities (i.e., people) and projects; </p>
      <p>Provide structured output that can be viewed by staff and
executives. </p>
    </PROJECT_SCOPE>
    <PROJECT_DEFINITION><b>Information about current projects is not
readily available to executives.</b> Providing this information creates
a reporting burden for IT staff, who may already maintain this
information in Microsoft Project files. </PROJECT_DEFINITION>
  </ROW>
</DATA>

```

Note in this sample the following HTML tags:

- <p> - paragraph tag
- <i> - italics tag
- <b> - bold tag

Assume you want to display this data in a report retain the formatting supplied by these tags as shown in the following figure:

<b>Project Name</b>	Project Management
<b>Project Scope</b>	Develop an application to produce <i>executive-level summaries</i> and detailed project reports. The application will allow users to: Import existing MS Project files Allow the user to map file-specific resources to a central database entities (i.e., people) and projects; Provide structured output that can be viewed by staff and executives.
<b>Project Definition</b>	<b>Information about current projects is not readily available to executives.</b> Providing this information creates a reporting burden for IT staff, who may already maintain this information in Microsoft Project files.

The following subtemplate uses XSL syntax to match the three HTML tags in the XML data. The template then replaces the matched HTML string with its XSLFO equivalent.

```

<xsl:template match="P|p">
  <fo:block white-space-collapse="false" padding-bottom="3pt"
linefeed-treatment="preserve">
  <xsl:apply-templates select="text()|*|@*" />
  </fo:block>
</xsl:template>

<xsl:template match="STRONG|B|b">
  <fo:inline font-weight="bold">
  <xsl:apply-templates />
  </fo:inline>
</xsl:template>

<xsl:template match="EM|I|i">
  <fo:inline font-style="italic">
  <xsl:apply-templates />
  </fo:inline>
</xsl:template>

```

1. Upload this XSL subtemplate file to the BI Publisher catalog location: Shared Folders/Projects. Save this subtemplate file as "htmlmarkup.xsb".

2. In your main template enter the following to import the subtemplate file:

```
<?import:xdoxsl:///Projects/htmlmarkup.xsb?>
```

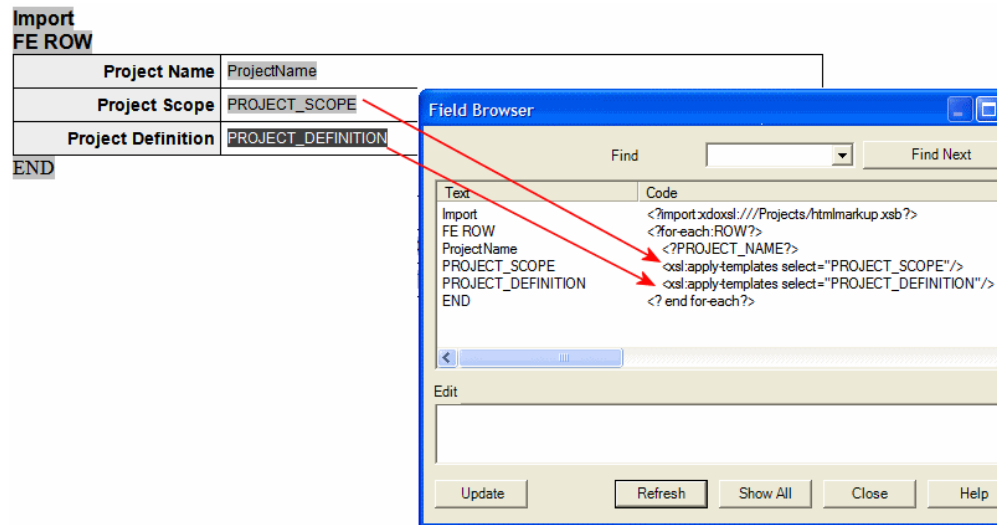
3. For each field that has HTML markup, call the xsl apply-template command. In this example, there are two fields:

```

<xsl:apply-templates select="PROJECT_SCOPE"/>
<xsl:apply-templates select="PROJECT_DEFINITION"/>

```

The following figure shows the field definitions in the template:



The command tells the processor to apply all templates to the value of the element PROJECT\_SCOPE and PROJECT\_DEFINITION. It will then cycle through the subtemplate functions looking for a match.

## Dynamically Applying Formatting to a Portion of Data

This application of subtemplates is useful for documents that require chemical formulae, mathematical calculations, or superscripts and subscripts.

For example, in the sample XML data below CO<sub>2</sub> is expected to display as CO<sub>2</sub> and H<sub>2</sub>O is expected to display as H<sub>2</sub>O.

```
<ROWSET>
  <ROW>
    <FORMULA>CO2</FORMULA>
  </ROW>
  <ROW>
    <FORMULA>H2O</FORMULA>
  </ROW>
</ROWSET>
```

This can be achieved by using an XSL subtemplate. Using XSL syntax you can define a template with any name, for example, "chemical\_formatter" that will accept the FORMULA field as a parameter, and then read one character at a time. It will compare the character with 0 – 9 digits, and if there is a match, then that character will be subscripted using the following XSL FO syntax:

```
<fo:inline baseline-shift="sub" font-size="75%">
```

Here is sample code for the XSL template statement:

```

<xsl:template name="chemical_formatter">
<! - accepts a parameter e.g. H2O - >
<xsl:param name="formula"/>
<! - Takes the first character of the string and tests it to see if it
is a number between 0-9 - > <xsl:variable name="each_char"
select="substring($formula,1,1)"/>
<xsl:choose>
  <xsl:when test="$each_char='1' or $each_char='2'
or $each_char='3' or $each_char='4' or $each_char='5'
or $each_char='6' or $each_char='7' or $each_char='8'
or $each_char='9' or $each_char='0'">
    <! - if it is numeric it sets the FO subscripting properties - >
    <fo:inline baseline-shift="sub" font-size="75%">
      <xsl:value-of select="$each_char"/>
    </fo:inline>
  </xsl:when>
  <xsl:otherwise>
    <! - otherwise the charater is left as is - >
    <fo:inline baseline-shift="normal">
      <xsl:value-of select="$each_char"/>
    </fo:inline>
  </xsl:otherwise>
</xsl:choose>
<! - test if there are other chars in the string, if so the recall the
template - >
  <xsl:if test="substring-after($formula,$each_char) !=''">
    <xsl:call-template name="chemical_formatter">
      <xsl:with-param name="formula">
        <xsl:value-of select="substring-after($formula,$each_char)"/>
      </xsl:with-param>
    </xsl:call-template>
  </xsl:if>
</xsl:template>

```

1. Save this file as chemical.xml.
2. Follow the instructions in the section Uploading a Subtemplate to the Catalog, page 12-3 to create a Sub Template object in the catalog. Assume you name the Sub Template "Chemical" (it will be saved as Chemical.xml) and place it in the following location: Shared Folders/Subtemplates.

3. In the main RTF template enter the import syntax:

```
<?import:xdoxsl:///Subtemplates/Chemical.xml?>
```

4. To render the XSL code in the report, create a loop over the data and in the VALUE field use:

```
<xsl:call-template name="chemical_formatter">
<xsl:with-param name="formula" select="VALUE"/> </xsl:call-template>
```

This calls the formatting template with the FORMULA value that is, H2O. Once rendered, the formulae will be shown as expected: H<sub>2</sub>O.



# Part 5

---

## Translating Reports and Catalog Objects



---

## Translation Support Overview and Concepts

This chapter covers the following topics:

- Translation Support Overview
- Working with Translation Files
- What Is an XLIFF?
- Structure of the XLIFF File
- Locale Selection Logic

### Translation Support Overview

BI Publisher supports translation in two ways:

- Catalog translations
- Template translations

### What Is Catalog Translation?

**Important:** If BI Publisher is integrated with Oracle Business Intelligence Enterprise Edition, BI Publisher catalog translation (folder and report, data model, style template and sub template names) is ignored. The Oracle BI Enterprise Edition catalog translation mechanism is applied instead. See "Localizing Oracle Business Intelligence Deployments" in the *Oracle Fusion Middleware System Administrator's Guide for Oracle Business Intelligence Enterprise Edition* for information on components that can be translated in an integrated catalog.

Catalog translation is a feature of BI Publisher that enables you to extract the translatable strings from all objects contained in a selected catalog folder into a separate

file; this file can then be translated and uploaded back to BI Publisher and assigned the appropriate language code.

Catalog translation extracts not only translatable strings from the report layouts, but also the user interface strings that are displayed to users, such as catalog object descriptions, report parameter names, and data display names.

Users viewing the catalog will see the item translations appropriate for the UI Language they selected in their My Account preferences. Users will see report translations appropriate for the Report Locale they selected in their My Account preferences.

## What Is Template Translation?

Template translation is a feature of BI Publisher that enables you to extract the translatable strings from a single RTF-based template (including sub templates and style templates) or a single BI Publisher layout template (.xpt file).

Use this option when you only need the final report documents translated. For example, you need to generate translated invoices to send to German and Japanese customers.

## Working with Translation Files

When you extract the translatable strings for a catalog or template translation, BI Publisher creates an XLIFF file that contains the strings.

You can translate these strings within your organization or send the file to a localization provider. You then upload the translated XLIFF file back to the catalog or the individual layout and assign it the appropriate locale.

This section describes how to work with an XLIFF file. It contains the following topics:

- What is an XLIFF?
- Structure of the XLIFF File

## What Is an XLIFF?

XLIFF is the XML Localization Interchange File Format. It is the standard format used by localization providers. For more information about the XLIFF specification, see <http://www.oasis-open.org/committees/xliff/documents/xliff-specification.htm>

## Structure of the XLIFF File

The generated XLIFF file has the following structure:

```

<xliff>
  <file>
    <header>
      <body>
        <trans-unit>
          <source>
          <target>
          <note>

```

The following figure shows an excerpt from an untranslated XLIFF file:

```

<?xml version = '1.0' encoding = 'utf-8 ?>
<xliff version="1.0">
  <file source-language="en-US" target-language="en-US" datatype="XDO" original="orpher">
    <header/>
    <body>
      <trans-unit id="d678c24b" maxbytes="4000" maxwidth="90" size-unit="char" translatable="true">
        <source>Italian Purchase VAT Register - [&#1;]</source>
        <target>Italian Purchase VAT Register - [&#1;]</target>
        <note>Text located: header/table, token &#1;:anonymous placeholder(s)</note>
      </trans-unit>
      <trans-unit id="4d3eb24" maxbytes="4000" maxwidth="15" size-unit="char" translatable="true">
        <source>Total</source>
        <target>Total</target>
        <note>Text located: body/table</note>
      </trans-unit>
      <trans-unit id="aec17e" maxbytes="4000" maxwidth="37" size-unit="char" translatable="true">
        <source>Non-Recoverable</source>
        <target>Non-Recoverable</target>
        <note>Text located: body/table</note>
      </trans-unit>
      .
      .
    </body>
  </file>
</xliff>

```

## source-language and target-language attributes

The `<file>` element includes the attributes `source-language` and `target-language`. The valid value for `source-language` and `target-language` is a combination of the language code and country code as follows:

- the two-letter ISO 639 language code
- the two-letter ISO 3166 country code

**Note:** For more information on the International Organization for Standardization (ISO) and the code lists, see International Organization for Standardization [<http://www.iso.org/iso/en/ISOOnline.frontpage>].

For example, the value for English-United States is "en-US". This combination is also referred to as a *locale*.

When you edit the exported XLIFF file you must change the `target-language`

attribute to the appropriate locale value of your target language. The following table shows examples of source-language and target-language attribute values appropriate for the given translations:

<b>Translation (Language/Territory)</b>	<b>source-language value</b>	<b>target-language value</b>
From English/US To English/Canada	en-US	en-CA
From English/US To Chinese/China	en-US	zh-CN
From Japanese/Japan To French/France	ja-JP	fr-FR

## Embedded Data Fields

Some templates contain placeholders for data fields embedded in the text display strings of the report. For example, the title of the sample report is

### **Italian Purchase VAT Register - (year)**

where (year) is a placeholder in the RTF template that will be populated at runtime by data from an XML element. These fields are not translatable, because the value comes from the data at runtime.

To identify embedded data fields, the following token is used in the XLIFF file:

```
[ & n ]
```

where *n* represents the numbered occurrence of a data field in the template.

For example, in the preceding XLIFF sample, the first translatable string is

```
<source>Italian Purchase VAT Register - [ & 1 ]</source>
```

**Warning:** Do not edit or delete the embedded data field tokens or you will affect the merging of the XML data with the template.

## <source> and <target> Elements

Each <source> element contains a translatable string from the template in the source language of the template. For example,

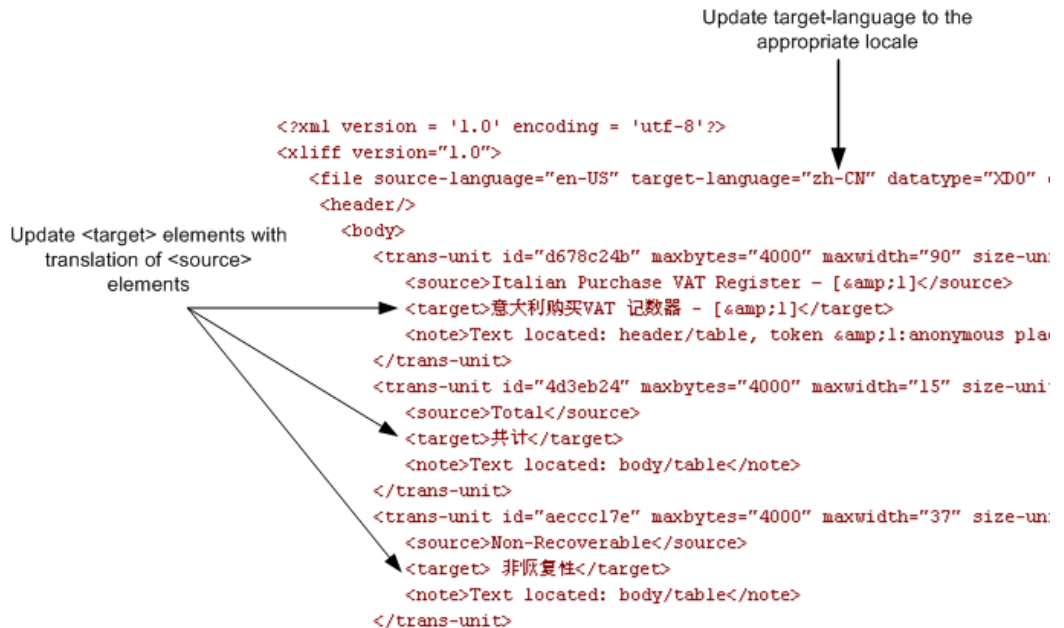
```
<source>Total</source>
```

When you initially export the XLIFF file for translation, the source and target elements are all identical. To create the translation for this template, enter the appropriate translation for each source element string in its corresponding <target> element.

Therefore if you were translating the sample template into German, you would enter the following for the Total string:

```
<source>Total</source>
<target>Gesamtbetrag</target>
```

The following figure shows the sample XLIFF file from the previous figure updated with the Chinese translation:



## Locale Selection Logic

BI Publisher applies a translation based on the user's selected Report Locale. BI Publisher will first try to match an RTF template named for the locale, then an XLIFF file named for the locale. If an exact match on language-territory is not found, BI Publisher will try to match on language only.

For example, if you have a report for which the base template is called EmployeeTemplate.rtf and the locale selected is French (France), BI Publisher will select the translation to apply according to the following hierarchy:

EmployeeTemplate.rtf (fr\_FR)

EmployeeTemplate.xlf (fr\_FR)

EmployeeTemplate.rtf (fr)

EmployeeTemplate.xlf (fr)

EmployeeTemplate.rtf (default)

Note that with the same set of translations, if the locale selected is French (Switzerland), the EmployeeTemplate.rtf (fr) would be applied. Now if the available translations were limited to the following set:

EmployeeTemplate.rtf (fr\_FR)

EmployeeTemplate.xlf (fr\_FR)

EmployeeTemplate.rtf (default)

and the locale selected is French (Switzerland), then the EmployeeTemplate.rtf (default) will be applied. Even though there is a language match, BI Publisher will not match the different locales.

Therefore, if you want to ensure that a French language translation is used when French is the selected language, regardless of the selected locale, then you must include either an rtf or xlf file named for the language only (that is, EmployeeTemplate\_fr.rtf or EmployeeTemplate\_fr.xlf).



---

## Translating Individual Templates

This chapter covers the following topics:

- Overview
- Types of Translations
- Using the XLIFF Option
- Using the Localized Template Option

### Overview

Individual files that can be translated are:

- RTF layout files
- style templates
- subtemplates
- BI Publisher layouts (.xpt)

This chapter describes how to create and upload translated template files when you only want to provide translations for specific templates.

If you wish to translate the layouts within the broader scope of the catalog, see *Translating Catalog Objects, Data Models, and Templates*, page 17-1.

### Types of Translations

There are two options for adding translations for your templates:

- Create a separate RTF template that is translated (a localized template). Note that this option is supported for RTF templates only.

- Generate an XLIFF from the original template (at runtime the original template is applied for the layout and the XLIFF is applied for the translation)

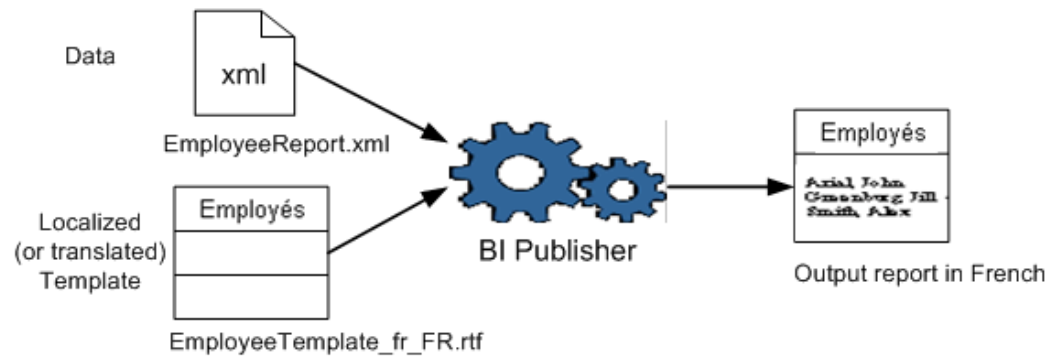
Use the first option if the translated template requires a different layout from the original template.

If you only require translation of the text strings of the template layout, use the XLIFF option.

The following diagrams illustrate the translation concepts:

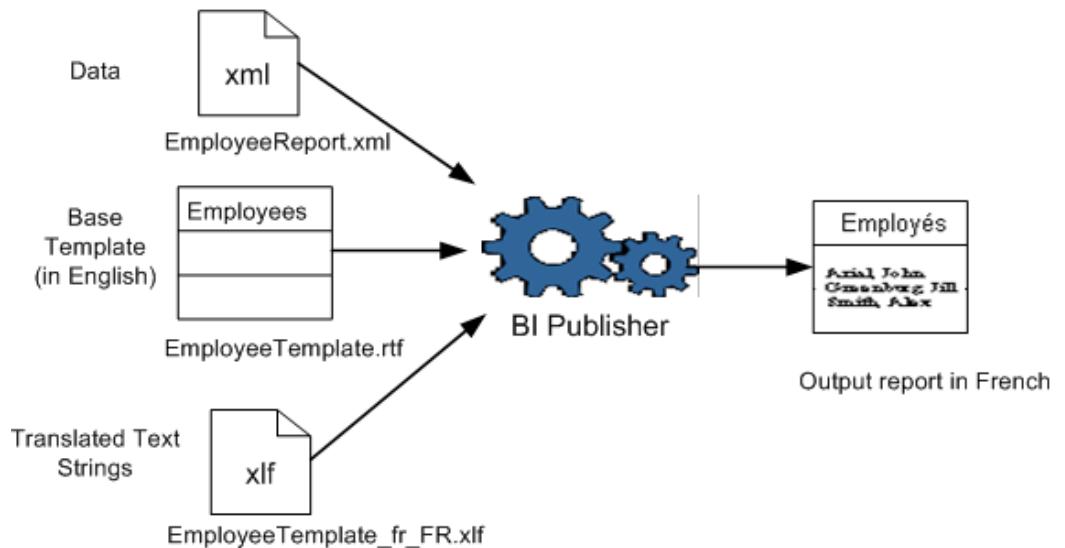
## Localized Template Option

Use this option when you want a specific translation to have a different layout.



## XLIFF File Option

Use this option when you want to use the same layout and apply specific translations.



## Using the XLIFF Option

The process overview for using the XLIFF option is:

1. Generate the XLIFF from the RTF or XPT template.
2. Translate the strings.
3. Upload the translation.

## Generating the XLIFF from a Template

There are two methods for generating an XLIFF for a single template file:

- Generate the XLIFF using the Template Builder for Microsoft Word (not supported for XPT templates)
- Generate the XLIFF from the Layout Properties page

## Generating the XLIFF from the Template Builder

**Note:** This procedure assumes you have installed the BI Publisher Template Builder for Microsoft Word. See Download BI Publisher Tools, *Oracle Fusion Middleware User's Guide for Oracle Business Intelligence Publisher* for information on downloading the add-in for Microsoft Word.

To generate an XLIFF from the Template Builder:

1. Open your template in Microsoft Word with the Template Builder for Word installed.
2. On the Template Builder tab, in the Tools group, click Translation, and then click Extract Text.

BI Publisher extracts the translatable strings from the template and exports them to an XLIFF (.xlf file).

3. Save the XLIFF to a local directory.

## Generating the XLIFF from the Layout Properties Page

**For report layout templates:**

1. Navigate to the report in the catalog and click **Edit** to open it for editing.
2. From the thumbnail view of the report layouts, click the **Properties** link of the layout (RTF or XPT) to open the **Layout Properties** page.
3. In the **Translations** region, click **Extract Translation**.

BI Publisher extracts the translatable strings from the template and exports them to an XLIFF (.xlf file).

4. Save the XLIFF to a local directory.

**For style templates and sub templates:**

1. Navigate to the style template or sub template in the catalog and click **Edit** to open

the Template Manager.

2. In the **Translations** region, click **Extract Translation**.

BI Publisher extracts the translatable strings from the template and exports them to an XLIFF (.xlf file).

3. Save the XLIFF to a local directory.

## Translating the XLIFF

Once you have downloaded the XLIFF, it can be sent to a translation provider, or using a text editor, you can enter the translation for each string. See Structure of the XLIFF File, page 15-2 for instructions on how to edit the XLIFF file.

A "translatable string" is any text in the template that is intended for display in the published report, such as table headers and field labels. Text supplied at runtime from the data is not translatable, nor is any text that you supply in the Microsoft Word form fields.

You can translate the template XLIFF file into as many languages as desired and then associate these translations to the original template. See Uploading the Translated XLIFF to BI Publisher, page 16-5.

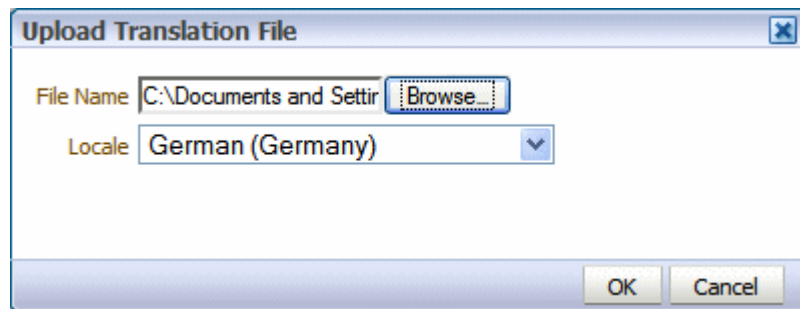
## Uploading the Translated XLIFF to BI Publisher

1. Navigate to the report, sub template, or style template in the catalog and click **Edit** to open it for editing.

**For reports only:**

From the thumbnail view of the report layouts, click the **Properties** link of the layout to open the Template Manager.

2. In the **Translations** region, click the **Upload** toolbar button.



3. In the **Upload Translation File** dialog, locate the file in your local directory and select the **Locale** for this translation.

4. Click OK to upload the file and view it in the **Translations** table.

## Using the Localized Template Option

If you need to design a different layout for the reports that you present for different localizations, you can create new RTF file that is designed and translated for the locale and upload this file to the Template Manager.

**Note:** The localized template option is not supported for XPT templates.

The process overview for using the localized template option is:

1. Design the localized RTF layout template, subtemplate, or style template.
2. Upload the localized file to the Template Manager.

## Designing the Localized Template File

Use the same tools you used to create the base template file, translating the strings and customizing the layout as desired for the locale.

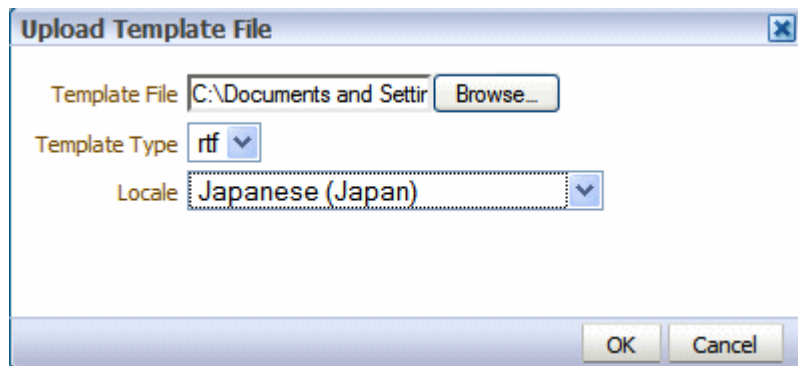
## Uploading the Localized Template to BI Publisher

1. Navigate to the report, subtemplate, or style template in the catalog and click **Edit** to open it for editing.

**For reports only:**

From the thumbnail view of the report layouts, click the **Properties** link of the layout to open the Template Manager.

2. In the **Templates** region, click the **Upload** toolbar button.



3. In the **Upload Template File** dialog, locate the file in your local directory, select rtf as the **Template Type** and select the **Locale** for this template file.
4. Click OK to upload the file and view it in the **Templates** table.





---

# Translating Catalog Objects, Data Models, and Templates

This chapter covers the following topics:

- Overview
- What Can Be Translated
- Exporting the XLIFF File
- Identifying and Updating the Object Tags
- Importing the XLIFF File

## Overview

This chapter describes how to use the Export XLIFF function that is available at the catalog level.

When you select a folder and choose this option, a single XLIFF file is generated that contains the translatable strings from the catalog objects contained in the folder; and the RTF and XPT templates contained in the folder. See the following section for the detailed list of what is translatable.

The target strings in the generated XLIFF file can be translated into the desired language. The XLIFF can then be uploaded back to the BI Publisher repository and assigned the appropriate locale. The translated strings from the XLIFF will be displayed when a user selects the target language as their UI language (for catalog object strings) or selects the target language as their Report Locale (for report template strings).

Note that if BI Publisher is integrated with Oracle Business Intelligence Enterprise Edition, BI Publisher catalog object string translation (folder and report, data model, style template and sub template names) is ignored. The Oracle BI Enterprise Edition catalog translation mechanism is applied instead. See "Localizing Oracle Business Intelligence Deployments" in the *Oracle Fusion Middleware System Administrator's Guide for Oracle Business Intelligence Enterprise Edition* for information on components that can

be translated in an integrated catalog.

## What Can Be Translated

The following table shows what strings can be translated:

Object	What Can Be Translated	Preference That Determines Translation Displayed
Folder	Name	UI Language (applies to all)
	Description	
Data Model	Name	UI Language (applies to all)
	Description	
	Data Display Name	
Report	Name	UI Language (applies to all)
	Description	
	Layout Names	
	Data Model Reference	
	Parameter Name	
Style Template	Name	UI Language
	Static text in the template	Report Locale
Sub Template	Name	UI Language
	Static text in the template	Report Locale
BI Publisher Layouts (.xpt)	Static text in the layout	Report Locale
RTF Layouts	Static text in the layout	Report Locale

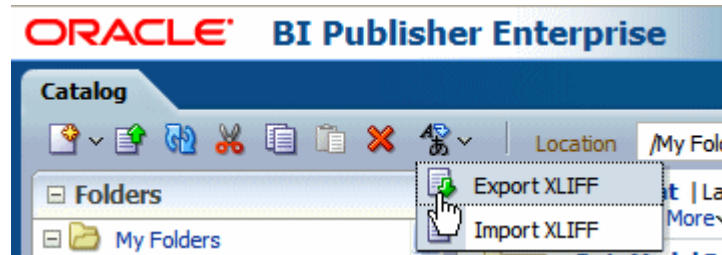
### About Source Language Limitations

For catalog translation, the source language is limited to "en". You must create catalog and data model objects in English locale to be able to translate them.

## Exporting the XLIFF File

To export an XLIFF file for a catalog folder:

1. Select the folder in the catalog.
2. Click the **Translation** toolbar button and then click **Export XLIFF**.



BI Publisher extracts the translatable strings from the template and exports them to an XLIFF (.xlf file).

3. Save the XLIFF file to a local directory.

## Identifying and Updating the Object Tags

For information on how to manually update the XLIFF files with translation strings, see Translation Concepts, page 15-2.

Note that in the XLIFF file generated for a catalog object the `source-language` and `target-language` attributes contain values for the two-letter language code only, as shown in the following figure:

```
<?xml version = '1.0' encoding = 'utf-8'?>
<xliff version="1.0">
  <file source-language="en" target-language="en" datatype="xml" product-version="11.1.1.3.0" pr
  <body>
    <trans-unit id="xdo#%2F%7Eadministrator%2FMy+Folder%2FReport.xdo#tmp_Salary.xpt">
      <source>Salary</source>
      <target>Salary</target>
    </trans-unit>
```

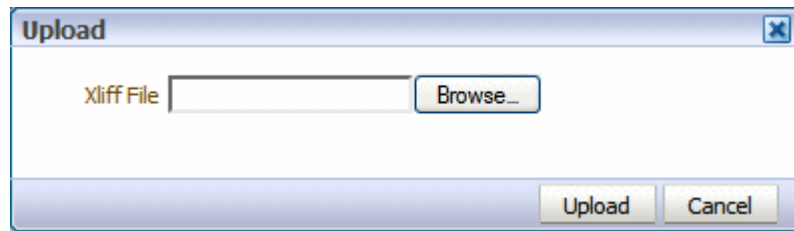
## Importing the XLIFF File

When the target tags have been translated you are ready to import the XLIFF file back to BI Publisher.

### To Import an XLIFF File

1. Navigate to the folder from which the XLIFF file was generated.
2. From the toolbar, click the **Translation** button and select **Import XLIFF**. This

launches the **Upload** dialog shown in the following figure:



3. Click **Browse** to locate the translated file and then select the appropriate locale from the list.
4. Click **Upload**.

---

## Techniques for Handling Large Output Files

This appendix covers the following topics:

- Techniques for Handling Large PDF Output Files
- Reusing Static Content
- Generating Zipped PDF Output
- Implementing PDF Splitting for an RTF Template
- Implementing PDF Splitting for a PDF Template

### Techniques for Handling Large PDF Output Files

This section describes techniques available to improve performance when your report generates very large PDF output files. The techniques discussed in this chapter are:

- Reusing Static Content to Reduce Output File Size, page A-1
- Generating Zipped PDF Output, page A-4

### Reusing Static Content

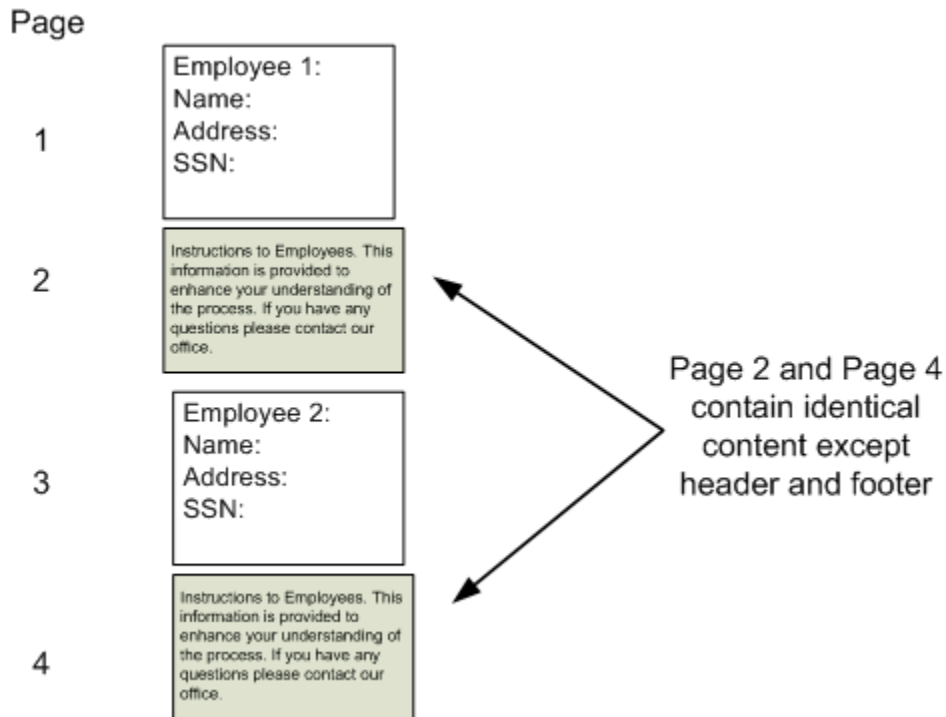
This section describes how to reuse static, repeating content in your PDF report output to reduce the overall PDF file size. This section contains the following topics:

- What Is Static Content Reuse?
- Limitations of This Feature
- Defining Reusable Content in an RTF Template
- Example

## What Is Static Content Reuse?

If your report contains static content and the placement of that content in the report is also fixed (for example, a set of instructions on the back of a Federal W-2 form), you can use this feature of BI Publisher to reduce the size of the generated PDF file.

Using the W-2 form as an example, the report has the following expected output:



For each employee, specific content is rendered, but the back (or second) page of each contains an identical set of instructions.

This set of instructions can be defined as reusable static content. When content is identified as reusable static content, BI Publisher will include the static content in the generated PDF document only once and reference it in other places when needed, thereby reducing the overall output file size.

## Limitations of this Feature

This feature has the following limitations:

- The static content to be reused in the generated report must fit onto one page of the generated PDF output.
- The contents of the report before the static content must have a fixed height. For example, the W-2 form has a fixed set of fields that occur before the static content is to be rendered. The reusable static contents are placed in the same position from the

page origin for each occurrence.

- This feature can only be used with RTF templates generating PDF output.

## Defining Reusable Content in an RTF Template

To define the static content to be reused, use the following tags around the content in your template as follows:

```
<?reusable-static-content:?>
```

```
.... static content here ...
```

```
<?end reusable-static-content?>
```

Inserting these tags around the static content signals BI Publisher to include this content only once in the generated file and then reference it in the same position for each occurrence.

## Example

The following example illustrates an implementation of this feature. The sample report generates one occurrence per employee. The generated report will have employee-specific information on the front page of each occurrence, and static instructions that will print on the back of each occurrence. A section break will occur after each employee to reset page numbering.

The following illustrates this template structure:

<?for-each@section:employee?>

<b>Lastname</b>	<b>Firstname</b>	<b>MI</b>
<b>Address</b>	<b>City</b>	<b>State</b>
<b>Year</b>	<b>Employer</b>	<b>Earnings</b>

Page Break

<?reusable-static-content:?>

<b>Notice to Employee</b> Refund. Even if you do not have to file a tax return, you should file to get a refund if box 2 shows federal income tax withheld or if you can take the earned income credit. Earned income credit (EIC). You must file a tax return if any amount is shown in box 9. You may be able to take the EIC for 2009 if (a) you do not have a qualifying child and you earned less than \$13,440 (\$16,560 if married filing jointly), (b) you have one qualifying child and you earned less than \$35,463 (\$38,583 if married filing	or ( c ) you have one than one Qualifying child and you earned less than \$40,295 (\$43,415 if married filing jointly). You and any qualifying children must have valid social security numbers (SSNs). You cannot take the EIC if your investment income is more than \$3,100. Any EIC that is more than your tax liability is refunded to you, but only if you file a tax return. If you have at least one qualifying child, you may get as much as \$1,826 of the EIC in advance by completing Form W-5, Earned Income Credit Advance Payment Certificate, and giving it to your employer
---	--

<?end reusable-static-content?>

<?end for-each?>

## Generating Zipped PDF Output

When generating PDF output, BI Publisher does not limit the size of the output file. However, when the size of the file approaches 2 GB, Adobe Acrobat Reader may no longer be able to open or handle the file.

BI Publisher provides a feature to split a large PDF output file into smaller, more manageable files, while still maintaining the integrity of the report as one logical unit.

When PDF output splitting is enabled for a report, the report is split into multiple files generated in one zip file. The output type is PDFZ. For easy access to the component files, BI Publisher also generates an index file that specifies from and to elements contained in each component PDF file.

To enable this feature, the report designer must set up the report using the methods described in this section.

## Limitations and Prerequisites

- This feature is supported only for PDF output that is generated from an RTF template or a PDF template.
- Data set input to the report must be flat XML data (that is, ROWSET/ROW). The data set cannot be hierarchical or concatenated.
- The data set must be sorted by the element designated as the "repeat" element (as described below).



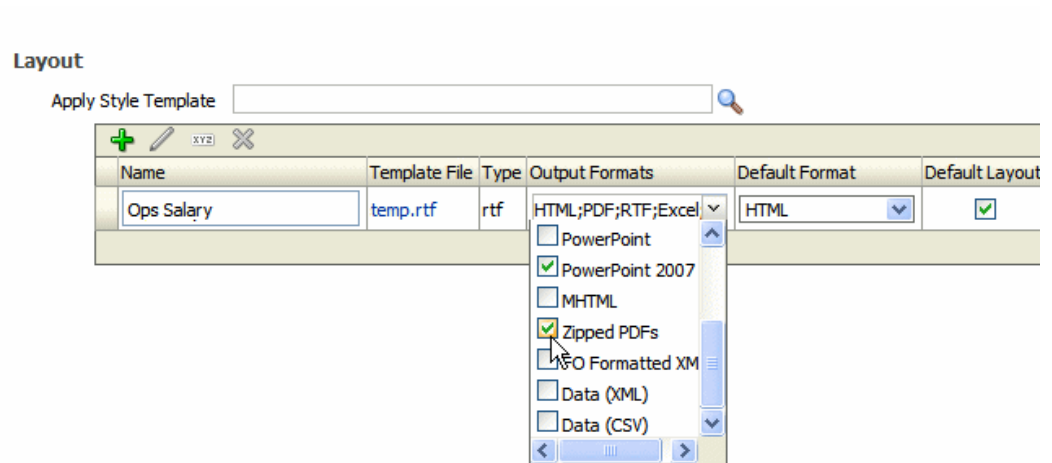
## Design Time Considerations

To enable report splitting, the report designer must determine the following:

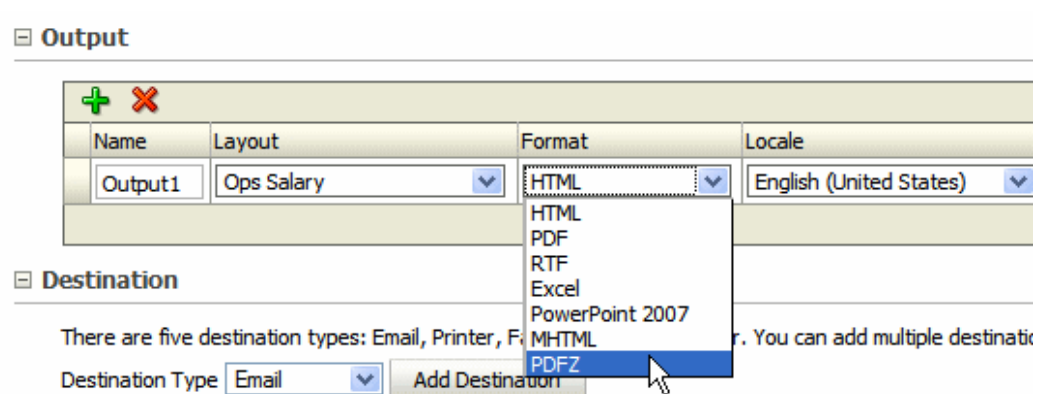
- Select a "repeat" element that will serve as the counter.
- Determine how many instances of the repeat element will occur per PDF file.
- Select which data elements to include in the generated index file.

## Selecting the Output Type

After uploading the template to the report definition, enable Zipped PDFs as an output type:



When scheduling the report, select PDFZ as the output type.



## Implementing PDF Splitting for an RTF Template

This section describes how to enable PDF splitting for reports generated from RTF templates. This section includes the following topics:

- Entering the Commands in Your RTF Template
- Examples

### Entering the Commands in Your RTF Template

When you design your template to use this feature, you must add commands to specify the following:

- What element in the data will be repeated (using the simple for-each command)
- How many occurrences of the element will be included in each PDF file
- What information (data elements) to include in the index file

To achieve this, the following two commands must be entered in your template within the for-each loop of the element by which you want the document to split:

- `<?catalog-index-info:name;element_name?>`

where

*name* is the name you choose that will be used in the index file to identify the from and to records included in each document.

*element\_name* is the XML tag name of the element that will provide the value for *name* that you identify above.

The `catalog-index-info` command defines the construction of the index file that is created.

- `<?if:position() mod n = 0?><?document-split:?><?end if?>`

where

*n* is the number of records you want included per PDF file.

This command must be placed within the for-each loop of the element that is to be counted. This command instructs BI Publisher to split the document after the next page break when the number of records equals the value you have supplied for *n*.

Each time the document-split is performed, the name-value pairs defined in the `catalog-index-info` command will be written to the index files.

## Example - split by each department

This example is based on the following XML data:

```
<DATA_DS>
  <G_EMP>
    <DEPARTMENT_NAME>Sales</DEPARTMENT_NAME>
    <FIRST_NAME>Ellen</FIRST_NAME>
    <LAST_NAME>Abel</LAST_NAME>
    <HIRE_DATE>1996-05-11T00:00:00.000-07:00</HIRE_DATE>
    <SALARY>11000</SALARY>
  </G_EMP>
  <G_EMP>
    <DEPARTMENT_NAME>Sales</DEPARTMENT_NAME>
    <FIRST_NAME>Sundar</FIRST_NAME>
    <LAST_NAME>Ade</LAST_NAME>
    <HIRE_DATE>2000-03-24T00:00:00.000-08:00</HIRE_DATE>
    <SALARY>6400</SALARY>
  </G_EMP>
  <G_EMP>
    <DEPARTMENT_NAME>Shipping</DEPARTMENT_NAME>
    <FIRST_NAME>Mozhe</FIRST_NAME>
    <LAST_NAME>Atkinson</LAST_NAME>
    <HIRE_DATE>1997-10-30T00:00:00.000-08:00</HIRE_DATE>
    <SALARY>2800</SALARY>
  </G_EMP>
  <G_EMP>
    <DEPARTMENT_NAME>IT</DEPARTMENT_NAME>
    <FIRST_NAME>David</FIRST_NAME>
    <LAST_NAME>Austin</LAST_NAME>
    <HIRE_DATE>1997-06-25T00:00:00.000-07:00</HIRE_DATE>
    <SALARY>4800</SALARY>
  </G_EMP>
  ...
</DATA_DS>
```

In this example, the output PDF report includes a document for each employee. You want a new PDF file generated for each department. You want the index to list the FIRST\_NAME and LAST\_NAME from each record that is included in the PDF file.

To achieve this output, enter the following in your template

```
<?for-each-group:ROW;./DEPARTMENT_NAME?>
  <?for-each:current-group()?>
  <?catalog-index-info:'First Name';FIRST_NAME?>
  <?catalog-index-info:'Last Name';LAST_NAME?>
  ...
<?end for-each?>
<?document-split:?>
<?end for-each-group?>
```

## Implementing PDF Splitting for a PDF Template

This section describes the commands required in a PDF template to split the output into multiple PDF files.

## Entering the Commands in the PDF Template

To enable this feature for a PDF template, enter the following three form fields in your template with the specified commands in the "Tooltip" field:

---

Form Field Name	Tooltip Command
REPEAT-ELEMENT	<pre>&lt;?repeat-element:element name?&gt;</pre> <p>where</p> <p>element_name is the XML tag name of the repeating element that will be counted.</p> <p>Example:</p> <pre>&lt;?repeat-element:emp_id?&gt;</pre>
CATALOG-INDEX-INFO	<pre>&lt;?catalog-index-info:'Name';element_name?&gt;</pre> <p>where</p> <p>'Name' is the label that will appear in the index file for the element_name that you specify. The index will generate a "From" and "To" listing for each file in the zipped set.</p> <p>Example:</p> <pre>&lt;?catalog-index-info:'Last Name';LAST_NAME?&gt;</pre> <p>Note that you can include multiple occurrences of the catalog-index-info command to include multiple data elements in the index file.</p>
SPLIT-COUNT	<pre>&lt;?split-count:n?&gt;</pre> <p>where</p> <p>n is the number of occurrences of the repeat-element that will trigger the creation of a new file.</p> <p>Example:</p> <pre>&lt;?split-count:10000?&gt;</pre>

---

---

## Extended Function Support in RTF Templates

This appendix covers the following topics:

- Extended SQL and XSL Functions
- XSL Equivalents
- Using FO Elements

### Extended SQL and XSL Functions

BI Publisher has extended a set of SQL and XSL functions for use in RTF templates. The syntax for these extended functions is

```
<?xdofx:expression?>
```

for extended SQL functions or

```
<?xdoxslt:expression?>
```

for extended XSL functions.

**Note:** You cannot mix xdofx statements with XSL expressions in the same context. For example, assume you had two elements, FIRST\_NAME and LAST\_NAME that you wanted to concatenate into a 30-character field and right pad the field with the character "x", you could NOT use the following:

**INCORRECT:**

```
<?xdofx: rpad (concat (FIRST_NAME, LAST_NAME) , 30, 'x') ?>
```

because concat is an XSL expression. Instead, you could use the following:

**CORRECT:**

```
<?xdoFx: rpad (FIRST_NAME || LAST_NAME) , 30, 'x') ?>
```

The supported functions are shown in the following table:

SQL Statement or XSL Expression	Usage	Description
2+3	<b>Usage:</b> <?xdoFx:2+3?> <b>Description:</b> Addition	Addition
2-3	<b>Usage:</b> <?xdoFx:2-3?> <b>Description:</b> Subtraction	Subtraction
2*3	<b>Usage:</b> <?xdoFx:2*3?> <b>Description:</b> Multiplication	Multiplication
2 div 3	<b>Usage:</b> <?xdoFx:2 div 3?> <b>Description:</b> Division	Division
2**3	<b>Usage:</b> <?xdoFx:2**3?> <b>Description:</b> Exponential	Exponential
3  2	<?xdoFx:3  2?>	Concatenation
sdiv()	<?xdoXslt:sdiv(num1,num2,string)?>	Safe divide function returns a specified value if the result of the function is NaN. In the syntax shown,  num1 is the dividend; num2 is the divisor and string is the value to be returned if NaN is returned.  Examples: <?xdoXslt:sdiv(10,0,'0')?> would yield '0'  <?xdoXslt:sdiv(10,0,'None')?> would yield 'None'

SQL Statement or XSL Expression	Usage	Description
lpad('aaa',10,'.')	<?xdoxsl:lpad('aaa',10,'.')?>	<p>The lpad function pads the left side of a string with a specific set of characters. The syntax for the lpad function is:</p> <pre>lpad(   string1,padded_length,[pad_string])</pre> <p><i>string1</i> is the string to pad characters to (the left-hand side).</p> <p><i>padded_length</i> is the number of characters to return.</p> <p><i>pad_string</i> is the string that will be padded to the left-hand side of <i>string1</i>.</p>
rpad('aaa',10,'.')	<?xdoxsl:rpad('aaa',10,'.')?>	<p>The rpad function pads the right side of a string with a specific set of characters.</p> <p>The syntax for the rpad function is:</p> <pre>rpad(   string1,padded_length,[pad_string]).</pre> <p><i>string1</i> is the string to pad characters to (the right-hand side).</p> <p><i>padded_length</i> is the number of characters to return.</p> <p><i>pad_string</i> is the string that will be padded to the right-hand side of <i>string1</i>.</p>
trim()	<?xdoxsl:trim(' a ')?>	Removes spaces in a string. Enter the text to be trimmed, the function returns the trimmed text.
ltrim()	<?xdoxsl:ltrim(' a ')?>	Removes the leading white spaces in a string.
rtrim()	<?xdoxsl:rtrim(' a ')?>	Removes the trailing white spaces in a string.

SQL Statement or XSL Expression	Usage	Description
truncate	<?xdoxslt:truncate ( <i>number</i> [, <i>integer</i> ] )?>	<p>The truncate function returns <i>number</i> truncated to <i>integer</i> places right of the decimal point. If <i>integer</i> is omitted, then <i>number</i> is truncated to the whole integer value. <i>integer</i> can be negative to truncate values left of the decimal point. <i>integer</i> must be an integer.</p> <p>Example:</p> <pre>&lt;?xdoxslt:truncate(-2.3333)?&gt;</pre> <p>returns</p> <p>-2</p> <p>Example:</p> <pre>&lt;?xdoxslt:truncate(2.7777, 2)?&gt;</pre> <p>returns</p> <p>2.77</p> <p>Example:</p> <pre>&lt;?xdoxslt:truncate(27.7777, -1)?&gt;</pre> <p>returns</p> <p>20</p>
replicate	<?xdoxslt:replicate('string' , <i>integer</i> )?>	<p>The replicate function will replicate the specified string the specified number of times.</p> <p>Example:</p> <pre>&lt;?xdoxslt:replicate('oracle', 3)?&gt;</pre> <p>returns</p> <p>oracleoracleoracle</p>



SQL Statement or XSL Expression	Usage	Description
<code>decode('xxx','bbb','ccc','xxx','ddd')</code>	<code>&lt;?xdoxfx:decode('xxx','bbb','ccc','xxx','ddd')?&gt;</code>	<p>The <i>decode</i> function has the functionality of an IF-THEN-ELSE statement. The syntax for the <i>decode</i> function is:</p> <pre>decode(expression, search, result [,search, result]...[, default])</pre> <p><i>expression</i> is the value to compare.</p> <p><i>search</i> is the value that is compared against <i>expression</i>.</p> <p><i>result</i> is the value returned, if <i>expression</i> is equal to <i>search</i>.</p> <p><i>default</i> is returned if no matches are found.</p>
<code>Instr('abcabcabc','a',2)</code>	<code>&lt;?xdoxfx:Instr('abcabcabc','a',2)?&gt;</code>	<p>The <i>instr</i> function returns the location of a substring in a string. The syntax for the <i>instr</i> function is:</p> <pre>instr(string1,string2,[start_position],[nth_appearance])</pre> <p><i>string1</i> is the string to search.</p> <p><i>string2</i> is the substring to search for in <i>string1</i>.</p> <p><i>start_position</i> is the position in <i>string1</i> where the search will start. The first position in the string is 1. If the <i>start_position</i> is negative, the function counts back <i>start_position</i> number of characters from the end of <i>string1</i> and then searches towards the beginning of <i>string1</i>.</p> <p><i>nth_appearance</i> is the <i>nth</i> appearance of <i>string2</i>.</p>
<code>substr('abcdefg',2,3)</code>	<code>&lt;?xdoxfx:substr('abcdefg',2,3)?&gt;</code>	<p>The <i>substr</i> function allows you to extract a substring from a string. The syntax for the <i>substr</i> function is:</p> <pre>substr(string, start_position, length)</pre> <p><i>string</i> is the source string.</p> <p><i>start_position</i> is the position for extraction. The first position in the string is always 1.</p> <p><i>length</i> is the number of characters to extract.</p>

SQL Statement or XSL Expression	Usage	Description
left	<code>&lt;?xdoxslt:left('abcdefg', 3)?&gt;</code>	<p>Enables you to extract the specified number of characters from a string, starting from the left. The syntax is <code>left(string, Numchars)</code></p> <p>For example, <code>&lt;?xdoxslt:left('abcdefg', 3)?&gt;</code></p> <p>returns</p> <p>abc</p>
right	<code>&lt;?xdoxslt:right('abcdefg', 3)?&gt;</code>	<p>Enables you to extract the specified number of characters from a string, starting from the right. The syntax is <code>right(string, Numchars)</code></p> <p>For example, <code>&lt;?xdoxslt:right('abcdefg', 3)?&gt;</code></p> <p>returns</p> <p>efg</p>
<code>replace(name,'John','Jon')</code>	<code>&lt;?xdofx:replace(name, 'John', 'Jon')?&gt;</code>	<p>The replace function replaces a sequence of characters in a string with another set of characters. The syntax for the replace function is:</p> <p><code>replace(string1,string_to_replace,[replacement_string])</code></p> <p><i>string1</i> is the string to replace a sequence of characters with another set of characters.</p> <p><i>string_to_replace</i> is the string that will be searched for in <i>string1</i>.</p> <p><i>replacement_string</i> is optional. All occurrences of <i>string_to_replace</i> will be replaced with <i>replacement_string</i> in <i>string1</i>.</p>
<code>to_number('12345')</code>	<code>&lt;?xdofx:to_number('12345')?&gt;</code>	<p>Function <code>to_number</code> converts <code>char</code>, a value of <code>CHAR</code>, <code>VARCHAR2</code>, <code>NCHAR</code>, or <code>NVARCHAR2</code> datatype containing a number in the format specified by the optional format model <code>fmt</code>, to a value of <code>NUMBER</code> datatype.</p>

SQL Statement or XSL Expression	Usage	Description
format_number	<code>&lt;?xdoxslt:format_number(12345, n, \$_XDOLOCALE)?&gt;</code>	<p>Converts a number to a string and formats the number according to the locale specified in <code>\$_XDOLOCALE</code> and to the number of decimal positions specified in <code>n</code> using Java's default symbols. For example:</p> <pre>&lt;?xdoxslt:format_number(-12345, 2, 'fr-FR')?&gt;</pre> <p>returns</p> <p>-12 345,00</p>
format_number	<code>&lt;?xdoxslt:format_number(12345, n, s1, s2, \$_XDOLOCALE)?&gt;</code>	<p>Converts a number to a string and uses the specified separators: <code>s1</code> for the thousand separator and <code>s2</code> for the decimal separator. For example:</p> <pre>&lt;?xdoxslt:format_number(12345, 2, 'g', 'd', \$_XDOLOCALE)?&gt;</pre> <p>returns</p> <p>12g345d00</p>
pat_format_number	<code>&lt;?xdoxslt:pat_format_number(12345, '##,##0.00', \$_XDOLOCALE)?&gt;</code>	<p>Returns a number formatted with the specified pattern.</p> <p>For example:</p> <pre>&lt;?xdoxslt:pat_format_number(12345, '##,##0.00', \$_XDOLOCALE)?&gt;</pre> <p>returns</p> <p>12,345.00</p>
to_char(12345)	<code>&lt;?xdofx:to_char('12345')?&gt;</code>	<p>Use the TO_CHAR function to translate a value of NUMBER datatype to VARCHAR2 datatype.</p>
to_date	<code>&lt;?xdofx:to_date ( char [, fmt [, 'nlsparam']] )</code>	<p>TO_DATE converts char of CHAR, VARCHAR2, NCHAR, or NVARCHAR2 datatype to a value of DATE datatype. The <i>fmt</i> is a date format specifying the format of <i>char</i>. If you omit <i>fmt</i>, then <i>char</i> must be in the default date format. If <i>fmt</i> is 'J', for Julian, then <i>char</i> must be an integer.</p>

SQL Statement or XSL Expression	Usage	Description
format_date()	<?xdoxslt:format_date(./AnyDate,'yy-yy-MM-dd','MM/dd/yyyy',\$_XDOLOCALE,\$_XDOTIMEZONE)?>	Reads date in one format and creates in another format.
sysdate()	<?xdofx:sysdate() ?>	<p>SYSDATE returns the current date and time in XML canonical date format (for example: 1997-07-16T19:20:30.45+01:00). The datatype of the returned value is DATE. The function requires no arguments.</p> <p>See <a href="#">Displaying the System Date (sysdate) in Reports</a>, page 4-120 for information on properly formatting the sysdate in report output.</p>
current_date()	<p>&lt;?xdoxslt:current_date(\$_XDOLOCALE, \$_XDOTIMEZONE) ?&gt;</p> <p>Example: &lt;?xdoxslt:current_date('ja-JP', 'Asia/Tokyo') ?&gt;</p>	Returns the current date in "yyyy-MM-dd" format in the given locale and timezone. This function supports only the Gregorian calendar.
current_time()	<p>&lt;?xdoxslt:current_time(\$_XDOLOCALE, \$_XDOTIMEZONE) ?&gt;</p> <p>Example: &lt;?xdoxslt:current_time('ja-JP', 'Asia/Tokyo') ?&gt;</p>	Returns the current time in the given locale and timezone. This function supports only the Gregorian calendar.
minimum	<?xdoxslt:minimum(ELEMENT_NAME) ?>	Returns the minimum value of the element in the set.

SQL Statement or XSL Expression	Usage	Description
date_diff	<pre>&lt;?xdoxslt:date_diff('y', 'YYYY-MM-DD', 'YYYY-MM-DD', \$_XDOLOCALE, \$_XDOTIMEZONE) ?&gt;</pre>	<p>This function provides a method to get the difference between two dates in the given locale. The dates need to be in "yyyy-MM-dd" format. This function supports only the Gregorian calendar. The syntax is as follows:</p> <pre>&lt;?xdoxslt:date_diff('format', 'YYYY-MM-DD', 'YYYY-MM-DD', \$_XDOLOCALE, \$_XDOTIMEZONE) ?&gt;</pre> <p>where</p> <p><i>format</i> is the time value for which the difference is to be calculated. Valid values are :</p> <ul style="list-style-type: none"> <li>• y - for year</li> <li>• m - for month</li> <li>• w - for week</li> <li>• d - for day</li> <li>• h - for hour</li> <li>• mi - for minute</li> <li>• s - for seconds</li> <li>• ms - for milliseconds</li> </ul> <p>Example:</p> <pre>&lt;?xdoxslt:date_diff('y', '2000-04-08', '2001-05-01', \$_XDOLOCALE, \$_XDOTIMEZONE) ?&gt;</pre> <p>returns</p> <p>1</p> <p>Example:</p> <pre>&lt;?xdoxslt:date_diff('m', '2001-04-08', '2000-02-01', \$_XDOLOCALE, \$_XDOTIMEZONE) ?&gt;</pre> <p>returns</p>

SQL Statement or XSL Expression	Usage	Description
sec_diff	<pre>&lt;?xdoxslt:sec_diff('2000-04-08T20:00:00', '2000-04-08T21:00:00', \$_XDOLOCALE, \$_XDOTIMEZONE?)&gt;</pre>	<p data-bbox="857 338 894 363">-14</p> <p data-bbox="857 394 959 420">Example:</p> <pre data-bbox="857 443 1243 548">&lt;?xdoxslt:date_diff('d', '2006-04-08', '2006-04-01', \$_XDOLOCALE, 'America/Los_Angeles')?&gt;</pre> <p data-bbox="857 579 935 604">returns</p> <p data-bbox="857 632 878 657">-7</p> <p data-bbox="857 701 1354 856">This function provides a method to get the difference between two dates in seconds in the given locale. The dates need to be in "yyyy-MM-dd'T'HH:mm:ss". This function supports only Gregorian calendar.</p>
get_day	<pre>&lt;?xdoxslt:get_day('2000-04-08', \$_XDOLOCALE)?&gt;</pre>	<p data-bbox="857 888 959 913">Example:</p> <pre data-bbox="857 936 1354 1020">&lt;?xdoxslt:sec_diff('2000-04-08T20:00:00', '2000-04-08T21:00:00', \$_XDOLOCALE, \$_XDOTIMEZONE?)&gt;</pre> <p data-bbox="857 1052 935 1077">returns</p> <p data-bbox="857 1104 907 1129">3600</p> <p data-bbox="857 1167 1354 1293">This function provides a method to get the day value of a date in "yyyy-MM-dd" format in the given locale. This function supports only the Gregorian calendar.</p> <p data-bbox="857 1325 959 1350">Example:</p> <pre data-bbox="857 1373 1300 1430">&lt;?xdoxslt:get_day('2000-04-08', \$_XDOLOCALE)?&gt;</pre> <p data-bbox="857 1461 935 1486">returns</p> <p data-bbox="857 1514 870 1539">8</p>

SQL Statement or XSL Expression	Usage	Description
get_month	<?xdoxslt:get_month('2000-04-08', \$_XDOLOCALE)?>	<p>This function provides a method to get the month value of a date in "yyyy-MM-dd" format in the given locale. This function supports only the Gregorian calendar.</p> <p>Example:</p> <pre data-bbox="954 537 1425 596">&lt;?xdoxslt:get_month('2000-04-08', \$_XDOLOCALE)?&gt;</pre> <p>returns</p> <p>4</p>
get_year	<?xdoxslt:get_year('2000-04-08', \$_XDOLOCALE)?>	<p>This function provides a method to get the year value of a date in "yyyy-MM-dd" format in the given locale. This function supports only the Gregorian calendar.</p> <p>Example:</p> <pre data-bbox="954 947 1425 1005">&lt;?xdoxslt:get_year('2000-04-08', \$_XDOLOCALE)?&gt;</pre> <p>returns</p> <p>2000</p>

SQL Statement or XSL Expression	Usage	Description
month_name	<?xdoxslt:month_name(1, 0, \$_XDOLOCALE)?>	<p>This function provides a method to get the name of the month in the given locale. This function supports only the Gregorian calendar.</p> <p>The syntax for this function is:</p> <pre data-bbox="857 506 1276 564">&lt;?xdoxslt:month_name(month, [abbreviate?], \$_XDOLOCALE)?&gt;</pre> <p>where</p> <p>month is the numeric value of the month (January = 1)</p> <p>and</p> <p>[abbreviate?] is the value 0 for do not abbreviate or 1 for abbreviate.</p> <p>Example:</p> <pre data-bbox="857 911 1243 970">&lt;?xdoxslt:month_name(12, 1, 'fr-FR')?&gt;</pre> <p>returns</p> <p>dec.</p> <p>Example"</p> <pre data-bbox="857 1148 1230 1207">&lt;?xdoxslt:month_name(1, 0, \$_XDOLOCALE)?&gt;</pre> <p>returns</p> <p>January</p>
maximum	<?xdoxslt:maximum(ELEMENT_NAME)?>	Returns the maximum value of the element in the set.
abs	<?xdoxslt:abs(-123.45)?>	<p>Returns the absolute value of the number entered.</p> <p>Example:</p> <pre data-bbox="857 1598 1203 1625">&lt;?xdoxslt:abs(-123.45)?&gt;</pre> <p>Returns:</p> <p>123.45</p>



SQL Statement or XSL Expression	Usage	Description
chr	<?xdoxfx:chr( <i>n</i> )?>	CHR returns the character having the binary equivalent to <i>n</i> in either the database character set or the national character set.
ceil	<?xdoxfx:ceil( <i>n</i> )?>	CEIL returns smallest integer greater than or equal to <i>n</i> .
floor	<?xdoxfx:floor( <i>n</i> )?>	FLOOR returns largest integer equal to or less than <i>n</i> .
round (SQL function)	<?xdoxfx:round ( <i>number</i> [, <i>integer</i> ] )?>	<p>ROUND returns <i>number</i> rounded to <i>integer</i> places right of the decimal point. If <i>integer</i> is omitted, then <i>number</i> is rounded to 0 places. <i>integer</i> can be negative to round off digits left of the decimal point. <i>integer</i> must be an integer.</p> <p>Example:</p> <pre>&lt;?xdoxfx:round (2.777)?&gt;</pre> <p>returns</p> <p>3</p> <p>Example:</p> <pre>&lt;?xdoxfx:round (2.777, 2)?&gt;</pre> <p>returns</p> <p>2.78</p>

SQL Statement or XSL Expression	Usage	Description
round (XSLT function)	<code>&lt;?xdoxslt:round ( number [, integer ] )?&gt;</code>	<p>ROUND returns <i>number</i> rounded to <i>integer</i> places right of the decimal point. If <i>integer</i> is omitted, then <i>number</i> is rounded to 0 places. <i>integer</i> can be negative to round off digits left of the decimal point. <i>integer</i> must be an integer.</p> <p>Example:</p> <pre>&lt;?xdoxslt:round (2.777)?&gt;</pre> <p>returns</p> <p>3</p> <p>Example:</p> <pre>&lt;?xdoxslt:round (2.777, 2)?&gt;</pre> <p>returns</p> <p>2.78</p>
lower	<code>&lt;?xdofx:lower (char)?&gt;</code>	<p>LOWER returns <i>char</i>, with all letters lowercase. <i>char</i> can be any of the datatypes CHAR, VARCHAR2, NCHAR, NVARCHAR2, CLOB, or NCLOB. The return value is the same datatype as <i>char</i>.</p>
upper	<code>&lt;?xdofx:upper (char)?&gt;</code>	<p>UPPER returns <i>char</i>, with all letters uppercase. <i>char</i> can be any of the datatypes CHAR, VARCHAR2, NCHAR, NVARCHAR2, CLOB, or NCLOB. The return value is the same datatype as <i>char</i>.</p>
length	<code>&lt;?xdofx:length (char)?&gt;</code>	<p>The "length" function returns the length of <i>char</i>. LENGTH calculates length using characters as defined by the input character set.</p>
greatest	<code>&lt;?xdofx:greatest ( expr [, expr]... )?&gt;</code>	<p>GREATEST returns the greatest of the list of <i>exprs</i>. All <i>exprs</i> after the first are implicitly converted to the datatype of the first <i>expr</i> before the comparison.</p>
least	<code>&lt;?xdofx:least ( expr [, expr]... )?&gt;</code>	<p>LEAST returns the least of the list of <i>exprs</i>. All <i>exprs</i> after the first are implicitly converted to the datatype of the first <i>expr</i> before the comparison.</p>

SQL Statement or XSL Expression	Usage	Description
next_element	<pre>&lt;?xdoxslt:next_element(current-group(),..,'&lt;element-name&gt;')?&gt;</pre>	<p>Method to get the next element in the current group. Will return the element that occurs after the element named. For example:</p> <pre>&lt;?xdoxslt:next_element(current-group(),..,'employee')?&gt;</pre> <p>will return the element that occurs in the current group after "employee".</p>
prev_element	<pre>&lt;?xdoxslt:prev_element(current-group(),..,'&lt;element-name&gt;')?&gt;</pre>	<p>Method to get the previous element in the current group. Will return the element that occurs before the element named. For example:</p> <pre>&lt;?xdoxslt:prev_element(current-group(),..,'employee')?&gt;</pre> <p>will return the element that occurs in the current group before "employee".</p>
set_array	<pre>&lt;?xdoxslt:set_array(\$_XDOCTX, '&lt;name of hash table&gt;', n, '&lt;value&gt;')?&gt;</pre>	<p>Sets a value in a hash table. Syntax is</p> <pre>&lt;?xdoxslt:set_array(\$_XDOCTX, '&lt;name of hash table&gt;', n, '&lt;value&gt;')?&gt;</pre> <p>where</p> <ul style="list-style-type: none"> <li><code>\$_XDOCTX</code> is required to set the context,</li> <li><code>&lt;name of hash table&gt;</code> is the name you supply for your table</li> <li><code>n</code> is the index of the hash table</li> <li><code>&lt;value&gt;</code> is the value to set in the hash table.</li> </ul> <p>For example:</p> <pre>&lt;?xdoxslt:set_array(\$_XDOCTX, 'Employee', 2, 'Jones')?&gt;</pre> <p>See <code>get_array</code> below.</p>

SQL Statement or XSL Expression	Usage	Description
get_array	<pre>&lt;?xdoxslt:get_array(\$_XDOCTX, '&lt;name of hash table&gt;', n)??&gt;</pre>	<p>Returns the value at the specified index of the hash table.</p> <p>Syntax is  <pre>&lt;?xdoxslt:get_array(\$_XDOCTX, '&lt;name of hash table&gt;', n)??&gt;</pre></p> <p>where</p> <p><code>\$_XDOCTX</code> is required to set the context,</p> <p><code>&lt;name of hash table&gt;</code> is the name you supplied for your table in <code>set_array</code></p> <p><code>n</code> is the index value of the element you want returned.</p> <p>For example, used in conjunction with the <code>set_array</code> example above,</p> <pre>&lt;?xdoxslt:get_array(\$_XDOCTX, 'Employee', 2)??&gt;</pre> <p>returns</p> <p>Jones</p>

The following table shows supported combination functions:

SQL Statement	Usage
$(2+3/4-6*7)/8$	<pre>&lt;?xdofx:(2+3/4-6*7)/8?&gt;</pre>
<code>lpad(substr('1234567890',5,3),10,'^')</code>	<pre>&lt;?xdofx:lpad(substr('1234567890',5,3),10,'^')?&gt;</pre>
<code>decode('a','b','c','d','e','1')  instr('321',1,1)</code>	<pre>&lt;?xdofx:decode('a','b','c','d','e','1')  instr('321',1,1)?&gt;</pre>

## Number-To-Word Conversion

This function 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	Valid 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_FRACTION 1', 'DECIMAL_STYLE_FRACTION 2', 'DECIMAL_STYLE_WORD'

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

RTF Template Entry	Returned Output
<?xdofx:to_check_number(12345.67, 2)?>	Twelve thousand three hundred forty-five and 67/100
<?xdofx:to_check_number(12345.67, 'USD')?>	Twelve thousand three hundred forty-five and 67/100
<?xdofx:to_check_number(12345, 'JPY', 'CASE_UPPER')?>	TWELVE THOUSAND THREE HUNDRED FORTY-FIVE
<?xdofx:to_check_number(12345.67, 'EUR', 'CASE_LOWER', 'DECIMAL_STYLE_WORDS')?>	twelve thousand three hundred forty-five and sixty-seven

## XSL Equivalents

The following table lists the BI Publisher simplified syntax with the XSL equivalents.

Supported XSL Elements	Description	BI Publisher Syntax
<code>&lt;xsl:value-of select="name"&gt;</code>	Placeholder syntax	<code>&lt;?name?&gt;</code>
<code>&lt;xsl:apply-templates select="name"&gt;</code>	Applies a template rule to the current element's child nodes.	<code>&lt;?apply:name?&gt;</code>
<code>&lt;xsl:copy-of select="name"&gt;</code>	Creates a copy of the current node.	<code>&lt;?copy-of:name?&gt;</code>
<code>&lt;xsl:call-template name="name"&gt;</code>	Calls a named template to be inserted into/applied to the current template.	<code>&lt;?call:name?&gt;</code>
<code>&lt;xsl:sort select="name"&gt;</code>	Sorts a group of data based on an element in the dataset.	<code>&lt;?sort:name?&gt;</code>
<code>&lt;xsl:for-each select="name"&gt;</code> >	Loops through the rows of data of a group, used to generate tabular output.	<code>&lt;?for-each:name?&gt;</code>
<code>&lt;xsl:choose&gt;</code>	Used in conjunction with <code>when</code> and <code>otherwise</code> to express multiple conditional tests.	<code>&lt;?choose?&gt;</code>
<code>&lt;xsl:when test="exp"&gt;</code>	Used in conjunction with <code>choose</code> and <code>otherwise</code> to express multiple conditional tests	<code>&lt;?when:expression?&gt;</code>
<code>&lt;xsl:otherwise&gt;</code>	Used in conjunction with <code>choose</code> and <code>when</code> to express multiple conditional tests	<code>&lt;?otherwise?&gt;</code>
<code>&lt;xsl:if test="exp"&gt;</code>	Used for conditional formatting.	<code>&lt;?if:expression?&gt;</code>
<code>&lt;xsl:template name="name"&gt;</code>	Template declaration	<code>&lt;?template:name?&gt;</code>
<code>&lt;xsl:variable name="name"&gt;</code>	Local or global variable declaration	<code>&lt;?variable:name?&gt;</code>

Supported XSL Elements	Description	BI Publisher Syntax
<code>&lt;xsl:import href="url"&gt;</code>	Import the contents of one stylesheet into another	<code>&lt;?import:url?&gt;</code>
<code>&lt;xsl:include href="url"&gt;</code>	Include one stylesheet in another	<code>&lt;?include:url?&gt;</code>
<code>&lt;xsl:stylesheet xmlns:x="url"&gt;</code>	Define the root element of a stylesheet	<code>&lt;?namespace:x=url?&gt;</code>

## Using FO Elements

You can use most FO elements in an RTF template inside the Microsoft Word form fields. The following FO elements have been extended for use with BI Publisher RTF templates. The BI Publisher syntax can be used with either RTF template method.

The full list of FO elements supported by BI Publisher can be found in the Appendix: Supported XSL-FO Elements, page D-1.

FO Element	BI Publisher Syntax
<code>&lt;fo:page-number-citation ref-id="id"&gt;</code>	<code>&lt;?fo:page-number-citation:id?&gt;</code>
<code>&lt;fo:page-number&gt;</code>	<code>&lt;?fo:page-number?&gt;</code>
<code>&lt;fo:ANY NAME WITHOUT ATTRIBUTE&gt;</code>	<code>&lt;?fo:ANY NAME WITHOUT ATTRIBUTE?&gt;</code>





---

# Designing Accessible Reports

## Introduction

This appendix describes techniques for designing reports to increase accessibility of report output to users with disabilities.

Note that accessibility support is for HTML output only.

The following topics are included:

- [Avoiding Nested Tables or Separated Tables](#), page C-1
- [Defining a Document Title](#), page C-3
- [Defining Alternative Text for an Image](#), page C-3
- [Defining a Table Summary](#), page C-4
- [Defining a Table Column Header](#), page C-4
- [Defining a Table Row Header](#), page C-4
- [Sample Supported Tables](#), page C-5

## Avoiding Nested Tables or Separated Tables

Avoid using nested tables in a report. For a complex report, try breaking down complex tables into several simple, straightforward tables.

The following figure shows a simple table:

Header1	Header2	Header3

The following figure shows an example of a nested table: A table is inserted inside a table-cell:

Column Header1	Column Header2	Column Header3						
	Cell with inserted table							
	<table border="1"> <thead> <tr> <th colspan="2">Nested table</th> </tr> </thead> <tbody> <tr> <td></td> <td></td> </tr> <tr> <td></td> <td></td> </tr> </tbody> </table>		Nested table					
Nested table								

## Examples

The following are examples of table structures that BI Publisher does and does not support for accessibility:

### Nested Tables

BI Publisher does not support accessibility when nested tables are used in a report. In the following example, BI Publisher cannot tell to which column data "C1R1data" belongs:

**Unsupported table layout:**

	Column header 1	Column Header2
Row Header1	C1R1data	C2R1data
	C1R2data	C2R2data

Remove the nested table as shown:

**Supported table layout:**

	Column header 1	Column Header2
Row Header1	C1R1data	C2R1data
	C1R2data	C2R2data

## Table Headers Must Not Be Separated from the Table Body

The following example is not supported because the header, table body and accessibility fields exist in 3 different tables.

### Unsupported table layout

	Column Header 1	Column Header 2	Column Header 3
UC1			
Row Header 1	C1R1Data	C2R1Data	C3R1Data
Row Header 2	C1R2Data	C2R2Data	C3R2Data
UC2			
Subtotal		Total 1	Total 2

These three tables should be joined into one to support accessibility:

### Supported table layout

ACC	Column Header 1	Column Header 2	Column Header 3
UC1Row Header 1	C1R1Data	C2R1Data	C3R1Data
Row Header 2	C1R2Data	C2R2Data	C3R2DataUC2
Subtotal		Total 1	Total 2

## Defining a Document Title

To define or change a document title in Microsoft Word:

In Word 2007: Click the **Office** button, click **Prepare**, and then click **Properties**.

In previous versions of Word: On the **File** menu, click **Properties**, and then click the **Summary** tab.

## Defining Alternative Text for an Image

To define alternative text for an image in your template:

1. Right-click the image.
2. On the menu, click **Format Picture**.
3. On the **Alt Text** tab, enter "alt:" followed by the alternative text. For example:  
`alt:flower picture`

**Note:** In versions of Word prior to 2007, enter the alt:text syntax on the **Web** tab.

## Defining a Table Summary

Add a table summary to a table by inserting the following command:

```
<?table-summary: 'My Table Test '?>
```

in the first column and first row position of the table.

## Defining a Table Column Header

To define a table column header:

In Word 2007:

1. Select the heading row or rows. The selection must include the first row of the table.
2. On the **Design** tab, in the **Table Style Options** group, select **Header Row**.
3. Right-click the table and select **Table Properties**.
4. In the **Table Properties** dialog, click the Row tab and then select Repeat as Header row at the top of each page.

In prior versions of Word:

1. Select the heading row or rows. The selection must include the first row of the table.
2. On the Table menu, click **Heading Rows Repeat**.

## Defining a Table Row Header

To define multiple row headers, use the BI Publisher command:

```
<?acc-row-header:col_index?>
```

Example Usage:

```
<?acc-row-header:'1,2,4'?> ==> column 1, 2 and 4 will be row-headers.
```

```
<?acc-row-header:'1,4'?> ==> column 1 and 4 will be row-headers.
```

In the following figure, the code behind the ACC field is:

```
ACC Field=<?table-summary:'My Table Test '?><?acc-row-header:'1,2'?>
```

which defines the first two columns as row headers:

ACC		Column header 1a	Column header 1b	
		Column header 2a	Column header 2b	Column Header 2c
Row header 1a	Row header 2a	Data col 1 row 1	Data col 2 row 1	Data col 3 row 1
Row header 1b	Row header 2b	Data col 1 row 2	Data col 2 row 2	Data col 3 row 2
	Row header 2c	Data col 1 row 3	Data col 2 row 3	Data col 3 row 3

## Sample Supported Tables

The following figures display sample tables for which accessibility is supported:

### Double Column and Row Headers

ACC		Column header 1a	Column header 1b	
		Column header 2a	Column header 2b	Column Header 2c
Row header 1a	Row header 2a	Data col 1 row 1	Data col 2 row 1	Data col 3 row 1
Row header 1b	Row header 2b	Data col 1 row 2	Data col 2 row 2	Data col 3 row 2
	Row header 2c	Data col 1 row 3	Data col 2 row 3	Data col 3 row 3

### Group Summary Total

ACC		Column header 1a	Column header 1b	Column header 1c	Column Header 1d
		Column header 2a	Column header 2b	Column header 2c	Column Header 2d
Row header 1a	Row header 2a	Data col 1 row 1	Data col 2 row 1	Data col 3 row 1	Data col 4 row 1
	Row header 2b	Data col 1 row 2	Data col 2 row 2	Data col 3 row 2	Data col 4 row 2
	Row header 2c	Data col 1 row 3	Data col 2 row 3	Data col 3 row 3	Data col 4 row 3
		Summary Sub-Total		Total 1a	Total 2a
Row header 1b	Row header 2d	Data col 1 row 1	Data col 2 row 1	Data col 3 row 1	Data col 4 row 1
	Row header 2e	Data col 1 row 2	Data col 2 row 2	Data col 3 row 2	Data col 4 row 2
		Summary Sub-Total		Total 1b	Total 2b
		Grant Total		Grand-total	Grand-total

### Separated Row Headers with Shared Column Header

ACC Paramaters Table			
Row header 1a	Data col 1 row 1	HA	Data col 3 row 1
Row header 1b	Data col 1 row 2	HB	Data col 3 row 2
Row header 1c	Data col 1 row 3	HC	Data col 3 row 3

### Separated Row Headers with Individual Column Header

ACC Paramaters Table	Column header 1a		Column Header 1c
Row header 1a	Data col 1 row 1	HA	Data col 3 row 1
Row header 1b	Data col 1 row 2	HB	Data col 3 row 2
Row header 1c	Data col 1 row 3	HC	Data col 3 row 3

---

## Supported XSL-FO Elements

### Supported XSL-FO Elements

The following table lists the XSL-FO elements supported in this release. For each element the supported content elements and attributes are listed. If elements have shared supported attributes, these are noted as a group and are listed in the subsequent table, Property Groups. For example, several elements share the content element `inline`. Rather than list the `inline` properties each time, each entry notes that "inline-properties" are supported. The list of inline-properties can then be found in the Property Groups table.

<b>Element</b>	<b>Supported Content Elements</b>	<b>Supported Attributes</b>
basic-link	external-graphic inline leader page-number page-number-citation basic-link block block-container table list-block wrapper marker retrieve-marker	inline-properties external-destination internal-destination
bidi-override	bidi-override external-graphic instream-foreign-object inline leader page-number page-number-citation basic-link	inline-properties



<b>Element</b>	<b>Supported Content Elements</b>	<b>Supported Attributes</b>
block	external-graphic inline page-number page-number-citation basic-link block block-container table list-block wrapper	block-properties
block-container	block block-container table list-block wrapper	block-properties
bookmark-tree	bookmark	N/A
bookmark	bookmark bookmark-title	external-destination internal-destination starting-state
bookmark-title	N/A	color font-style font-weight

Element	Supported Content Elements	Supported Attributes
conditional-page-master-reference	N/A	master-reference page-position <ul style="list-style-type: none"> <li>• first</li> <li>• last</li> <li>• rest</li> <li>• any</li> <li>• inherit</li> </ul> odd-or-even <ul style="list-style-type: none"> <li>• odd</li> <li>• even</li> <li>• any</li> <li>• inherit</li> </ul> blank-or-not-blank <ul style="list-style-type: none"> <li>• blank</li> <li>• not-blank</li> <li>• any</li> <li>• inherit</li> </ul>
external-graphic	N/A	graphic-properties src

<b>Element</b>	<b>Supported Content Elements</b>	<b>Supported Attributes</b>
flow	block block-container table list-block wrapper	flow-properties
inline	external-graphic inline leader page-number page-number-citation basic-link block block-container table wrapper	inline-properties
instream-foreign-object	N/A	graphic-properties
layout-master-set	page-sequence-master simple-page-master simple-page-master page-sequence-master	N/A
leader	N/A	inline-properties
list-block	list-item	block-properties
list-item	list-item-label list-item-body	block-properties

<b>Element</b>	<b>Supported Content Elements</b>	<b>Supported Attributes</b>
list-item-body	block block-container table list-block wrapper	block-properties
list-item-label	block block-container table list-block wrapper	block-properties
page-number	N/A	empty-inline-properties
page-number-citation	N/A	empty-inline-properties ref-id

Element	Supported Content Elements	Supported Attributes
page-sequence	static-content flow	inheritable-properties id master-reference initial-page-number <ul style="list-style-type: none"> <li>• auto</li> <li>• &lt;page-number&gt;</li> </ul> force-page-count <ul style="list-style-type: none"> <li>• auto</li> <li>• end-on-even</li> <li>• end-on-odd</li> <li>• end-on-even-layout</li> <li>• end-on-odd-layout</li> <li>• no-force</li> <li>• inherit</li> </ul> format
page-sequence-master	single-page-master-reference repeatable-page-master-reference repeatable-page-master-alternatives	master-name
region-after	N/A	side-region-properties
region-before	N/A	side-region-properties
region-body	N/A	region-properties margin-properties-CSS column-count

<b>Element</b>	<b>Supported Content Elements</b>	<b>Supported Attributes</b>
region-end	N/A	side-region-properties
region-start	N/A	side-region-properties
repeatable-page-master-alternatives	conditional-page-master-reference	maximum-repeats
repeatable-page-master-reference	N/A	master-reference maximum-repeats
root	bookmark-tree layout-master-set page-sequence	inheritable-properties

Element	Supported Content Elements	Supported Attributes
simple-page-master	region-body region-before region-after region-start region-end	margin-properties-CSS master-name page-height page-width reference-orientation <ul style="list-style-type: none"> <li>• 0</li> <li>• 90</li> <li>• 180</li> <li>• 270</li> <li>• -90</li> <li>• -180</li> <li>• -270</li> <li>• 0deg</li> <li>• 90deg</li> <li>• 180deg</li> <li>• 270deg</li> <li>• -90deg</li> <li>• -180deg</li> <li>• -270deg</li> <li>• inherit</li> </ul> writing-mode <ul style="list-style-type: none"> <li>• lr-tb</li> </ul>

<b>Element</b>	<b>Supported Content Elements</b>	<b>Supported Attributes</b>
single-page-master-reference	N/A	master-reference
static-content	block block-container table wrapper	flow-properties
table	table-column table-header table-footer table-body	block-properties
table-body	table-row	inheritable-properties id
table-cell	block block-container table list-block wrapper	block-properties number-columns-spanned number-rows-spanned
table-column	N/A	inheritable-properties column-number column-width number-columns-repeated
table-footer	table-row	inheritable-properties id
table-header	table-row	inheritable-properties id



<b>Element</b>	<b>Supported Content Elements</b>	<b>Supported Attributes</b>
table-row	table-cell	inheritable-properties id
wrapper	inline page-number page-number-citation basic-link block block-container table wrapper	inheritable-properties id

#### **Property Groups Table**

The following table lists the supported properties belonging to the attribute groups defined in the preceding table.

Property Group	Properties
area-properties	<p>clip</p> <p>overflow (visible, hidden)</p> <p>reference-orientation</p> <ul style="list-style-type: none"> <li>• 0</li> <li>• 90</li> <li>• 180</li> <li>• 270</li> <li>• -90</li> <li>• -180</li> <li>• -270</li> <li>• 0deg</li> <li>• 90deg</li> <li>• 180deg</li> <li>• 270deg</li> <li>• -90deg</li> <li>• -180deg</li> <li>• -270deg</li> <li>• inherit</li> </ul> <p>writing-mode (lr-tb, rl-tb, lr, rl)</p> <p>baseline-shift (baseline, sub, super)</p> <p>vertical-align</p>
block-properties	<p>inheritable-properties</p> <p>id</p>

Property Group	Properties
border-padding-background-properties	background-color background-image background-position-vertical background-position-horizontal border border-after-color border-after-style (none, dotted, dashed, solid, double) border-after-width border-before-color border-before-style (none, solid) border-before-width border-bottom border-bottom-color border-bottom-style (none, dotted, dashed, solid, double) border-bottom-width border-color border-end-color border-end-style (none, dotted, dashed, solid, double) border-end-width border-left border-left-color border-left-style (none, dotted, dashed, solid, double) border-left-width border-right border-right-color border-right-style (none, dotted, dashed, solid, double) border-right-width border-start-color

Property Group	Properties
	border-start-style (none, dotted, dashed, solid, double) border-start-width border-top border-top-color border-top-style (none, dotted, dashed, solid, double) border-top-width border-width padding padding-after padding-before padding-bottom padding-end padding-left padding-right padding-start padding-top
box-size-properties	height width
character-properties	font-properties text-decoration
empty-inline-properties	character-properties border-padding-background-properties id color

Property Group	Properties
flow-properties	inheritable-properties id flow-name
font-properties	font-family font-size font-style (normal, italic, oblique) font-weight (normal, bold) table-omit-header-at-break (TRUE, FALSE, inherit) table-omit-footer-at-break (TRUE, FALSE, inherit)
graphic-properties	border-padding-background-properties margin-properties-inline box-size-properties font-properties keeps-and-breaks-properties-atomic id

Property Group	Properties
inheritable-properties	border-padding-background-properties box-size-properties margin-properties-inline area-properties character-properties line-related-properties leader-properties keeps-and-breaks-properties-block color absolute-position <ul style="list-style-type: none"> <li>• auto</li> <li>• absolute</li> <li>• fixed</li> <li>• inherit</li> </ul>
inline-properties	inheritable-properties id
keeps-and-breaks-properties-atomic	break-after (auto, column, page) break-before (auto,column) keep-with-next keep-with-next.within-page
keeps-and-breaks-properties-block	keeps-and-breaks-properties-inline

Property Group	Properties
keeps-and-breaks-properties-inline	keeps-and-breaks-properties-atomic keep-together keep-together.within-line keep-together.within-column keep-together.within-page
leader-properties	leader-pattern (rule, dots) leader-length leader-length.optimum (dotted, dashed, solid, double) rule-thickness
line-related-properties	text-align (start, center, end, justify, left, right, inherit) text-align-last (start, center, end, justify, left, right, inherit) text-indent linefeed-treatment (ignore, preserve, treat-as-space, treat-as-zero-width-space, inherit ) white-space-treatment (ignore, preserve, ignore-if-before-linefeed, ignore-if-after-linefeed, ignore-if-surrounding-linefeed, inherit) white-space-collapse (FALSE, TRUE, inherit) wrap-option (no-wrap, wrap, inherit) direction (ltr)
margin-properties-block	margin-properties-CSS space-after space-after.optimum space-before space-before.optimum start-indent end-indent

Property Group	Properties
margin-properties-CSS	margin margin-bottom margin-left margin-right margin-top
margin-properties-inline	margin-properties-block space-start space-start.optimum space-end space-end.optimum position <ul style="list-style-type: none"> <li>• static</li> <li>• relative</li> <li>• absolute</li> <li>• fixed</li> <li>• inherit</li> </ul> top left
region-properties	border-padding-background-properties area-properties region-name
side-region-properties	region-properties extent



---

# Index

## A

---

- accessibility
  - checking for using Template Builder, 5-36
- Adobe Flash
  - designing templates, 8-1
  - required settings for PDF output, 8-2
- alignment
  - RTF template, 4-42

## B

---

- background support
  - RTF templates, 4-47
- barcode formatting
  - custom, 4-128
- barcodes
  - included in BI Publisher, 4-126
- bidirectional language alignment
  - RTF template, 4-42
- body tags
  - PDF layout, 7-8
  - RTF template, 4-16
- bookmarks
  - generating PDF bookmarks from an RTF template, 4-59
  - inserting in RTF templates, 4-55
- brought forward/carried forward page totals, 4-78

## C

---

- calculations in PDF layout, 7-13
- calendar profile option, 4-123

- calendar specification, 4-123
- cell highlighting
  - conditional in RTF templates, 4-73
- charts
  - building in RTF templates, 4-19
- check box placeholder
  - creating in PDF layout, 7-6
- check box support
  - RTF templates, 4-60
- choose statements, 4-67
- clip art support, 4-30
- columns
  - fixed width in tables, 4-43
- conditional columns
  - rtf template, 4-69
- conditional formatting, 4-64
  - table rows, 4-71
- conditional formatting features, 4-64
- configuration properties
  - precedence of levels, 10-1
- connections
  - setting maximum for an interactive report, 3-21
- context command, 4-129
- cross-tab reports, 4-100

## D

---

- date fields in RTF templates, 4-45
- dates
  - formatting in Excel templates, 6-14
- digital signature
  - adding signature field to a pdf layout, 7-19

- setting properties, 10-7
- drawing support, 4-30
- drop-down form field support
  - RTF templates, 4-61
- dynamic data columns, 4-103
  - example, 4-104
- dynamic table of contents in RTF template, 4-58

## E

---

- end on even page, 4-54
- etext data tables, 9-6
- etext template command rows, 9-6
- etext template setup command table, 9-16
- even page
  - force report to end on, 4-54
- Excel templates
  - formatting dates, 6-14

## F

---

- file size
  - techniques for reducing, A-1
- filler block
  - etext templates, 9-29
- fixed-width columns
  - RTF templates, 4-43
- Flash templates
  - configuration properties, 8-13
  - designing, 8-1
  - uploading to the BI Publisher server, 8-13
- FO
  - supported elements, D-1
- FO elements
  - using in RTF templates, 4-138, B-19
- fonts
  - external, 4-125
  - mapping, 10-19
  - setting up, 4-125
- footers
  - RTF template, 4-16
- for-each-group XSL 2.0 standard, 4-85
- formatting options in PDF templates, 7-5
- form field method
  - inserting placeholders, 4-9
- form field properties options in PDF template, 7-5
- form fields in the PDF template, 7-4

## G

---

- groups
  - basic RTF method, 4-13
  - defining in PDF layout, 7-8
  - defining in RTF template, 4-12
    - syntax, 4-12
  - defining in RTF templates, 4-6
  - form field method, 4-14
  - grouping scenarios in RTF template, 4-12
  - in RTF templates, 4-5

## H

---

- headers and footers
  - different first page , 4-17
  - different odd and even pages, 4-17
  - inserting placeholders, 4-16
  - multiple, 4-16
  - resetting within one output file, 4-97
  - RTF template, 4-16
- hidden text
  - support in RTF templates, 4-42
- horizontal table break, 4-104
- HTML output
  - controlling table widths, 10-11
- hyperlinks
  - bookmarks, 4-55
  - dynamic, 4-55
  - inserting in RTF template, 4-55
  - internal, 4-55
  - static, 4-55

## I

---

- if statements, 4-65, 4-65
- IF statements
  - in free-form text, 4-65
- if-then-else statements, 4-66
- images
  - including in RTF template, 4-18
- IN predicate
  - If-Then-Else control structure
    - e-text templates, 9-32

## L

---

- last page

- support for special content, 4-51

layout editor

- features, 3-1

list component

- layout editor, 3-74

## **M**

---

markup

- adding to the PDF template, 7-4
- adding to the RTF template, 4-7

max connections

- setting for an interactive report, 3-21

multicolumn page support, 4-46

multiple headers and footers

- RTF template, 4-16

## **N**

---

Namespace support in RTF template, 4-136

native page breaks and page numbering, 4-41

nulls

- how to test for in XML data, 4-85

number-to-word conversion, B-16

## **O**

---

output formats

- limiting by report, 5-6, 5-7

overflow data in PDF layouts, 7-17

## **P**

---

page breaks

- PDF layouts, 7-9
- RTF template, 4-41, 4-49

page breaks and page numbering

- native support, 4-41

page number

- setting initial
- RTF templates, 4-50

page numbers

- PDF layouts, 7-9
- restarting within one output file, 4-97
- RTF template, 4-42

page totals

- brought forward/carried forward, 4-78
- inserting in RTF template, 4-76

PDF layouts

- completed example, 7-14
- creating from downloaded file, 7-17
- defining groups, 7-8
- definition of, 7-1
- overflow data, 7-17
- page breaks, 7-9
- page numbering, 7-9
- placeholders
  - check box, 7-6
  - radio button group, 7-7
- placement of repeating fields at runtime, 7-15
- runtime behaviors, 7-15

pdf output

- handling large files, A-1

PDF output

- invalid, 4-99

PDF output properties, 10-2

PDF security properties, 10-4

PDF template

- adding markup, 7-4
- placeholders
  - types of, 7-5

PDF templates

- placeholders
  - text, 7-5
- sample purchase order template, 7-2
- saving as Adobe Acrobat 5.0 compatible, 7-1
- sources for document templates, 7-2
- supported modes, 7-1
- when to use, 7-1

pivot table

- designing in RTF templates, 4-100

placeholders

- basic RTF method, 4-8, 4-8
- form field RTF method, 4-8, 4-9
- in PDF templates, 7-4
- in RTF templates, 4-5
  - defining, 4-6, 4-8
- inserting in the header and footer of RTF template, 4-16
- PDF layouts
  - check box, 7-6
  - radio button group, 7-7
- PDF templates
  - text, 7-5
  - types of, 7-5

PowerPoint output

- design considerations, 4-138
- predefined fonts, 10-20
- properties
  - setting at template level, 4-95

## R

---

- radio button group
  - creating in PDF layouts, 7-7
- regrouping, 4-85
- repeating elements
  - See* groups
- Rich Text Format (RTF)
  - definition, 4-2
- row breaking
  - preventing in RTF templates, 4-43
- row formatting
  - conditional, 4-71
- RTF placeholders
  - syntax, 4-8
- RTF template
  - adding markup, 4-7
  - applying design elements, 4-7
  - definition, 4-2
  - designing, 4-4
  - groups, 4-5
  - including images, 4-18
  - native formatting features, 4-41
  - placeholders, 4-5
  - prerequisites, 4-3
  - sample template design, 4-4
  - supported modes, 4-3
    - basic method, 4-3
    - form field method, 4-3
    - using XSL or XSL:FO, 4-3
- RTF template design
  - headers and footers, 4-16
- RTF template placeholders, 4-8
- running totals
  - RTF templates, 4-82

## S

---

- sample RTF template
  - completed markup, 4-11
- section context command, 4-97
- security
  - PDF properties, 10-4

- setting the initial page number
  - RTF templates, 4-50
- shape support, 4-30
- sorting
  - RTF template, 4-84
- SQL functions
  - BI Publisher syntax for, B-1
  - using in RTF templates, 4-132
- SQL functions extended for BI Publisher, B-1
- style templates, 11-1
- syntax
  - RTF template placeholder, 4-8
- sysdate
  - displaying and formatting in RTF templates, 4-120

## T

---

- table features
  - fixed-width columns, 4-43
  - preventing rows breaking across pages
    - RTF template, 4-43
  - text truncation, 4-44
- table features
  - repeating table headers
    - RTF template, 4-43
  - RTF template, 4-42
- table of contents support
  - RTF template, 4-58
  - dynamic TOC, 4-58
- tables
  - controlling table widths in HTML output, 10-11
- tables
  - horizontal table break, 4-104
- Template Builder, 5-3
  - editing field properties, 5-32
  - prerequisites, 5-3
  - preview options, 5-31
- text placeholder
  - creating in PDF template, 7-5
- text truncation in tables, 4-44
- totals
  - brought forward/carried forward, 4-78
  - inserting page totals in RTF template, 4-76
  - running
    - RTF templates, 4-82

## **U**

---

updateable variables  
RTF templates, 4-92

## **V**

---

variables  
RTF templates, 4-92

## **W**

---

watermarks  
RTF templates, 4-47

## **X**

---

XML data file  
example, 4-4  
XML file  
how to read, 4-5  
XSL:FO elements  
using in RTF templates, 4-132  
XSL elements  
apply a template rule, 4-136  
BI Publisher syntax for , B-18  
call template, 4-136  
copy the current node, 4-136  
define the root element of the stylesheet, 4-137  
import stylesheet, 4-137  
template declaration, 4-137  
using in RTF templates, 4-136  
variable declaration, 4-137

