

Oracle® Fusion Middleware

Report Designer's Guide for Oracle Business Intelligence
Publisher

11g Release 1 (11.1.1)

E22254-02

April 2012

Explains how to create a report and how to design report layouts using a variety of template types to display the data from an Oracle Business Intelligence Publisher data model. Includes how to create visualizations, set run-time properties, define subtemplates, and enable translations.

Oracle Fusion Middleware Report Designer's Guide for Oracle Business Intelligence Publisher, 11g Release 1 (11.1.1)

E22254-02

Copyright © 2010, 2012, 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

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 is software or related documentation that 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 America, 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.

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Xeon are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Opteron, the AMD logo, and the AMD Opteron logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

This software or hardware 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

Preface	xix
Audience	xix
Documentation Accessibility	xix
Related Documentation and Other Resources	xx
System Requirements and Certification	xx
Conventions	xx
New Features for Report Designers	xxiii
New Features for Oracle BI Publisher 11g Release 1 (11.1.1.6)	xxiii
New Features for Oracle BI Publisher 11g Release 1 (11.1.1.5)	xxiv
New Features for Oracle BI Publisher 11g Release 1 (11.1.1.3)	xxv
Part I Creating Reports and Layouts	
1 Introduction to Designing Reports	
1.1 Overview for Report Designers	1-1
1.2 About the Layout Types	1-1
1.3 About Setting Run-Time Properties	1-2
1.4 About Translations	1-2
1.5 About Style Templates	1-3
1.6 About Sub Templates	1-3
2 Creating and Editing Reports	
2.1 About Report Components	2-1
2.2 Creating a Report: Process Overview	2-2
2.3 Creating a Report	2-2
2.3.1 Choose Data Source	2-3
2.3.2 Choose Guide or Create	2-3
2.3.3 Select Columns	2-4
2.3.4 Choose View Report or Customize Layout	2-4
2.4 Editing Reports	2-4
2.5 Adding Layouts to the Report Definition	2-5
2.5.1 Adding a Layout Using the Layout Editor	2-5
2.5.2 Adding a Layout by Uploading a Template File	2-6

2.5.3	Adding a Layout by Generating a Template File.....	2-6
2.6	Configuring Layouts Using the List View	2-7
2.6.1	Applying a Style Template to the Layout	2-7
2.6.2	About the Layouts Toolbar	2-7
2.6.3	Configuring the Layout Settings Using the List View	2-7
2.6.4	Selecting Output Formats.....	2-8
2.6.5	Editing a Layout.....	2-10
2.7	Configuring Parameter Settings for the Report.....	2-10
2.8	Configuring Report Properties	2-13
2.8.1	Setting the General Properties	2-14
2.8.1.1	Description	2-14
2.8.1.2	Run Report Online.....	2-14
2.8.1.3	Excel Analyzer Options	2-15
2.8.1.4	Advanced Options.....	2-16
2.8.2	Setting the Caching Properties.....	2-16
2.8.3	Setting the Formatting Properties	2-18
2.8.4	Configuring Font Mapping	2-18
2.8.5	Configuring Currency Formats	2-18
2.9	Generating PDF/A and PDF/X Output.....	2-19
2.9.1	Generating PDF/A Output	2-19
2.9.1.1	Requirements and Limitations	2-19
2.9.1.2	Additional Resources.....	2-20
2.9.2	Generating PDF/X Output.....	2-20
2.9.2.1	Prerequisites	2-21
2.9.2.2	Requirements and Limitations	2-21
2.9.2.3	Additional Resources.....	2-22

3 Creating BI Publisher Layout Templates

3.1	Overview of BI Publisher Layouts	3-1
3.1.1	When to Use a BI Publisher Layout	3-3
3.1.2	Prerequisites, Recommendations, and Limitations	3-3
3.2	Launching the Layout Editor	3-3
3.2.1	When Creating a New Report.....	3-3
3.2.2	When Editing a Report.....	3-3
3.2.3	When Viewing a Report.....	3-4
3.2.4	Selecting a Predefined Layout	3-4
3.2.4.1	Adding Shared Templates for All Users	3-4
3.2.4.2	Adding Personal Predefined Layouts	3-5
3.3	About the Layout Editor Interface.....	3-5
3.3.1	About the Data Source Pane	3-6
3.3.2	About the Components Pane	3-7
3.3.3	About the Properties Pane.....	3-8
3.3.4	About the Static Toolbar	3-9
3.3.5	About the Tabbed Toolbar	3-9
3.3.5.1	Selecting and Deleting Layout Objects.....	3-9
3.3.5.2	About the Insert Tab	3-10
3.4	Page Layout Tab.....	3-10

3.4.1	Paper Options.....	3-11
3.4.2	Header/Footer Options	3-11
3.4.2.1	Setting Properties for Headers and Footers.....	3-11
3.4.3	View Options.....	3-12
3.4.4	Display Unit.....	3-12
3.4.5	Interactivity: Event Configuration	3-12
3.4.5.1	Example of Filter Event Configuration	3-13
3.4.5.2	Configuring Automatic Filtering	3-14
3.4.5.3	Example: Show Selection Only.....	3-15
3.4.6	Setting Page Margins.....	3-16
3.4.7	Setting Maximum Connections for an Interactive Report	3-17
3.5	Inserting Layout Components	3-18
3.6	About Layout Grids.....	3-18
3.6.1	Adding a Border or Background Color	3-19
3.6.2	About the Insert Options	3-20
3.6.3	About the Join and Unjoin Options	3-20
3.6.4	Adding an Expand and Collapse Option.....	3-20
3.7	About Repeating Sections.....	3-21
3.7.1	Setting Page Break Options for a Repeating Section	3-22
3.7.2	How Repeating Sections Display in Interactive Mode.....	3-23
3.7.3	Showing All Values in a Repeating Section.....	3-25
3.8	About Data Tables	3-26
3.8.1	Inserting a Data Table	3-26
3.8.2	Setting Alternating Row Colors.....	3-28
3.8.3	About the Table Tab	3-29
3.8.3.1	Setting the Rows to Display Option.....	3-30
3.8.3.2	About Filters.....	3-30
3.8.3.3	Setting Filters for a Table.....	3-30
3.8.3.4	Managing Filters.....	3-31
3.8.3.5	About Conditional Formats	3-31
3.8.3.6	Applying Conditional Formats to a Table	3-32
3.8.3.7	Managing Formats	3-33
3.8.3.8	Controlling the Display of the Total Row	3-33
3.8.4	About the Table Column Header Tab	3-34
3.8.4.1	About Grouping	3-34
3.8.4.2	Example: Group Left.....	3-35
3.8.4.3	Applying Subtotals.....	3-36
3.8.4.4	Example: Group Above	3-36
3.8.5	About the Column Tab	3-37
3.8.5.1	About the Data Formatting Options for Columns.....	3-38
3.8.5.2	Applying Formatting to Numeric Data Columns	3-38
3.8.5.3	Applying Formatting to Date Type Data Columns.....	3-39
3.8.5.4	About the Formula Option	3-39
3.8.5.5	About the Sort Option.....	3-39
3.8.5.6	Removing a Sort Order	3-41
3.8.6	About the Total Cell Tab.....	3-41
3.8.6.1	Applying Data Formatting to a Total Cell	3-41

3.8.6.2	Applying a Formula	3-41
3.8.7	Inserting Dynamic Hyperlinks	3-41
3.9	About Charts.....	3-43
3.9.1	Inserting a Chart	3-44
3.9.1.1	About the Chart Tab.....	3-46
3.9.1.2	Applying and Managing Filters	3-46
3.9.1.3	Converting a Chart to a Pivot Table	3-46
3.9.2	Changing the Formula Applied to a Chart Measure Field.....	3-46
3.9.3	Sorting a Chart Field	3-47
3.10	About Gauge Charts.....	3-47
3.10.1	Inserting a Gauge Chart.....	3-48
3.10.2	Setting the Properties for a Gauge Chart	3-49
3.10.3	Applying and Managing Filters	3-49
3.11	About Pivot Tables	3-49
3.11.1	Inserting a Pivot Table	3-50
3.11.2	Customizing a Pivot Table Menu.....	3-52
3.11.3	About the Pivot Table Tab.....	3-52
3.11.3.1	Applying Filters	3-52
3.11.3.2	Customizing the Display of Totals.....	3-52
3.11.3.3	Converting a Pivot Table to a Chart	3-53
3.11.3.4	Switching Rows and Columns.....	3-53
3.11.4	Customizing the Pivot Table Headers	3-54
3.11.5	Customizing the Pivot Table Data	3-54
3.12	About Text Items.....	3-55
3.12.1	Displaying a Data Field Side by Side with a Text Item.....	3-55
3.12.2	About the Text Toolbar	3-56
3.12.2.1	Editing Font Properties.....	3-56
3.12.2.2	Inserting Page Numbers.....	3-57
3.12.2.3	Inserting the Date and Time	3-57
3.12.2.4	Inserting a Hyperlink.....	3-58
3.13	About Images.....	3-59
3.14	About Lists.....	3-60
3.14.1	Inserting a List.....	3-60
3.14.2	Customizing a List.....	3-62
3.14.2.1	Customizing the Font Style and the Selected Font Style Commands.....	3-63
3.14.2.2	Customizing Behavior of Selected Items	3-63
3.15	Setting Predefined or Custom Formulas	3-64
3.15.1	About the Predefined Formulas	3-65
3.15.2	Applying a Custom Formula	3-65
3.15.2.1	About the Basic Math Functions	3-66
3.15.2.2	About the Statistical Math Functions	3-67
3.15.3	Applying a Custom Formula: Examples.....	3-67
3.16	Saving a Layout.....	3-73

4 Creating RTF Templates

4.1	Getting Started.....	4-2
4.1.1	What Are RTF Templates?.....	4-2

4.1.2	Prerequisites for Designing Templates.....	4-2
4.1.3	What is XSLT Compatibility?.....	4-3
4.1.4	Key Concepts.....	4-3
4.1.5	Designing the Template Layout	4-3
4.1.6	About Adding BI Publisher Code	4-3
4.2	Associating the XML Data to the Template Layout.....	4-4
4.2.1	Using an XML Input File	4-4
4.2.2	Identifying Placeholders and Groups.....	4-5
4.2.3	Using Placeholders	4-5
4.2.4	Identifying the Groups of Repeating Elements.....	4-6
4.3	Adding Markup to the Template Layout	4-6
4.3.1	Creating Placeholders	4-7
4.3.1.1	Using the Basic RTF Method.....	4-7
4.3.1.2	Using the Form Field Method.....	4-7
4.3.1.3	Completing the Example.....	4-9
4.4	Defining Groups.....	4-10
4.4.1	Grouping Scenarios	4-10
4.4.2	Using the Basic RTF Method.....	4-11
4.4.3	Using the Form Field Method.....	4-11
4.4.4	Completing the Example	4-12
4.5	Defining Headers and Footers	4-13
4.5.1	Native Support for Headers and Footers	4-13
4.5.2	Inserting Placeholders in the Headers and Footers	4-13
4.5.3	Creating Multiple or Complex Headers and Footers	4-13
4.5.4	Defining Different First Page and Different Odd and Even Pages	4-14
4.6	Inserting Images and Charts	4-15
4.6.1	Directly Inserting Images	4-15
4.6.2	Inserting Images with URL References	4-15
4.6.3	Inserting Images with an Element Reference from an XML File.....	4-15
4.6.4	Rendering an Image Retrieved from BLOB Data.....	4-16
4.6.5	Adding Charts to Templates.....	4-16
4.6.5.1	Adding a Sample Chart	4-17
4.6.5.2	Inserting the Dummy Image.....	4-18
4.6.5.3	Adding Code to the Alternative Text Box	4-18
4.6.5.4	Additional Chart Samples.....	4-21
4.6.5.5	Horizontal Bar Chart Sample.....	4-22
4.6.5.6	Changing the Appearance of the Chart.....	4-24
4.7	Adding Drawings, Shapes, and Clip Art.....	4-25
4.7.1	Adding Freehand Drawings	4-26
4.7.2	Adding Hyperlinks.....	4-26
4.7.3	Layering Shapes	4-26
4.7.4	Using 3-D Effects.....	4-26
4.7.5	Adding Microsoft Equations.....	4-26
4.7.6	Adding Organization Charts	4-27
4.7.7	Adding WordArt	4-27
4.7.8	Adding Data-Driven Shapes	4-27
4.7.9	Including Manipulation Commands	4-28

4.7.10	Replicating Shapes.....	4-28
4.7.11	Adding Text to Shapes.....	4-29
4.7.12	Adding Text Along a Path.....	4-29
4.7.13	Moving a Shape.....	4-29
4.7.14	Rotating a Shape	4-29
4.7.15	Skewing a Shape	4-30
4.7.16	Changing the Size of Shapes	4-30
4.7.17	Combining Commands.....	4-31
4.7.18	CD Ratings Example	4-31
4.7.19	Grouped Shape Example.....	4-33
4.8	Supported Formatting Features of Microsoft Word	4-36
4.8.1	General Features of Microsoft Word.....	4-36
4.8.2	Aligning Objects.....	4-36
4.8.3	Inserting Tables.....	4-37
4.8.4	Inserting Date Fields	4-38
4.8.5	Inserting Multiple Columns on Pages	4-39
4.8.6	Inserting Backgrounds and Watermarks	4-40
4.8.6.1	Adding a Background Using Microsoft Word 2000	4-40
4.8.6.2	Adding a Text or Image Watermark Using Microsoft Word 2002 or Later	4-40
4.8.7	Microsoft Word Features That Are Not Supported.....	4-41
4.9	Template Features.....	4-41
4.9.1	Inserting Page Breaks	4-41
4.9.2	Inserting an Initial Page Number	4-43
4.9.3	Specifying Last Page Only Content.....	4-44
4.9.4	Ending on Even or Odd Pages.....	4-46
4.9.5	Inserting Hyperlinks	4-47
4.9.5.1	Inserting Internal Links	4-50
4.9.6	Including a Table of Contents.....	4-50
4.9.7	Generating Bookmarks in PDF Output	4-50
4.9.8	Inserting Check Boxes	4-52
4.9.9	Inserting Drop-Down Lists.....	4-53
4.10	Using Conditional Formatting.....	4-56
4.10.1	Using If Statements.....	4-56
4.10.2	Using If Statements in Boilerplate Text	4-57
4.10.3	Using If-Then-Else Statements.....	4-58
4.10.4	Inserting Choose Statements.....	4-58
4.10.4.1	Conditional Formatting Example.....	4-59
4.10.5	Formatting Columns	4-60
4.10.6	Formatting Rows.....	4-62
4.10.7	Highlighting Cells.....	4-63
4.11	Inserting Page-Level Calculations	4-65
4.11.1	Displaying Page Totals	4-65
4.11.2	Inserting Brought Forward and Carried Forward Totals	4-68
4.11.3	Inserting Running Totals	4-71
4.12	Handling Data	4-72
4.12.1	Sorting Data	4-72
4.12.2	Checking for Null Values	4-73

4.12.3	Regrouping the XML Data	4-73
4.12.3.1	XML Sample	4-73
4.12.3.2	Regrouping Syntax	4-74
4.12.3.3	Template Example	4-75
4.12.3.4	Regrouping by an Expression	4-77
4.13	Setting Variables, Parameters, and Properties	4-79
4.13.1	Setting Variables	4-79
4.13.2	Setting Parameters	4-80
4.13.3	Setting Properties	4-82
4.14	Using Advanced Report Layouts	4-83
4.14.1	Creating Batch Reports	4-83
4.14.2	Handling No Data Found Conditions	4-85
4.14.3	Inserting Pivot Tables	4-85
4.14.4	Constructing Dynamic Data Columns	4-88
4.14.4.1	Defining Columns to Repeat Across Pages	4-89
4.14.4.2	Example of Dynamic Data Columns	4-89
4.15	Formatting Numbers, Dates, and Currencies	4-91
4.15.1	Formatting Numbers	4-91
4.15.2	Data Source Requirements	4-91
4.15.3	Localization Considerations	4-91
4.15.4	Using the Microsoft Number Format Mask	4-91
4.15.5	Supported Microsoft Format Mask Definitions	4-92
4.15.6	Using the Oracle Format Mask	4-93
4.15.7	Formatting Dates	4-94
4.15.8	Data Source Requirements	4-94
4.15.9	Using the Microsoft Date Format Mask	4-95
4.15.10	Using the Oracle Format Mask	4-96
4.15.11	Default Format Mask	4-98
4.15.12	Oracle Abstract Format Masks	4-98
4.15.13	Displaying the System Date (sysdate) in Reports	4-99
4.15.14	Formatting Currencies	4-99
4.15.15	Applying a Currency Format to a Field	4-100
4.15.16	Example: Displaying Multiple Currency Formats in a Report	4-100
4.15.17	Example: Displaying Multiple Currency Codes in a Single Report	4-101
4.16	Supporting Calendars and Time Zones	4-102
4.16.1	Calendar Specification	4-102
4.16.2	Time Zone Specification	4-103
4.17	Using External Fonts	4-103
4.17.1	Using the Barcode Fonts Shipped with BI Publisher	4-104
4.17.2	Implementing Custom Barcode Formats	4-106
4.17.2.1	Registering the Barcode Encoding Class	4-106
4.17.2.2	Encoding the Data	4-106
4.18	Controlling the Placement of Instructions Using the Context Commands	4-107
4.19	Using XPath Commands	4-109
4.19.1	Locating Data	4-110
4.19.2	Starting Reference	4-111
4.19.3	Specifying Context and Parents	4-111

4.20	Declaring Namespaces	4-112
4.21	Using FO Elements and XSL Elements	4-112
4.21.1	Using FO Elements	4-112
4.21.2	Using XSL Elements	4-112
4.21.2.1	Applying a Template Rule	4-112
4.21.2.2	Copying the Current Node	4-113
4.21.2.3	Calling a Named Template	4-113
4.21.2.4	Declaring a Template	4-113
4.21.2.5	Declaring a Variable	4-113
4.21.2.6	Importing a Style Sheet.....	4-113
4.21.2.7	Defining the Root Element of the Style Sheet.....	4-113
4.21.2.8	Formatting Native XSL Numbers	4-114
4.22	Guidelines for Designing RTF Templates for Microsoft PowerPoint Output	4-114
4.22.1	Limitations	4-114
4.22.2	Usage Guidelines	4-115
4.22.3	About Charts in PowerPoint Output	4-115
4.22.4	Configuring Fonts for the BI Publisher Server	4-116
4.22.5	Configuring Fonts for the BI Publisher Template Builder.....	4-116
4.23	Rendering HTML Formatted Data in a Report	4-117
4.23.1	Supported HTML Features	4-117
4.23.2	Data Model Requirements.....	4-118
4.23.3	RTF Template Requirements.....	4-118
4.23.4	Example.....	4-118

5 Creating RTF Templates Using the Template Builder for Word

5.1	Overview	5-1
5.1.1	Before You Get Started.....	5-2
5.1.2	Prerequisites and Limitations	5-2
5.2	Getting Started.....	5-3
5.2.1	Features of the Oracle BI Publisher Template Builder for Word.....	5-3
5.2.2	Building and Uploading a Template	5-4
5.2.2.1	Working in Connected Mode.....	5-4
5.2.2.2	Working in Disconnected Mode.....	5-5
5.3	Accessing Data for Building Templates.....	5-6
5.3.1	Loading XML Data from a Local File.....	5-6
5.3.2	Loading Data from the BI Publisher Catalog.....	5-7
5.4	Inserting Components to the Template	5-7
5.4.1	Inserting a Field	5-8
5.4.1.1	About the Insert Field Dialog	5-8
5.4.1.2	Find	5-8
5.4.1.3	Example.....	5-8
5.4.1.4	Force LTR (Left-to-Right) Direction.....	5-8
5.4.1.5	Calculation.....	5-9
5.4.2	Inserting a Table Using the Table Wizard.....	5-9
5.4.2.1	Step 1: Selecting Report Format.....	5-10
5.4.2.2	Step 2: Selecting Table Data	5-10
5.4.2.3	Step 3: Selecting Data Fields	5-11

5.4.2.4	Step 4: Grouping the Table.....	5-11
5.4.2.5	Step 5: Inserting a Break for the Group	5-13
5.4.2.6	Step 6: Sorting the Table	5-13
5.4.2.7	Step 7: Clicking Finish.....	5-14
5.4.2.8	Step 8: Customizing the Table Using Microsoft Word Functionality	5-14
5.4.3	Inserting a Table or Form Using the Insert Table/Form Dialog.....	5-14
5.4.3.1	Selecting Data Fields	5-14
5.4.3.2	Defining the Layout	5-15
5.4.3.3	Data Field Properties.....	5-15
5.4.3.4	Data Group Properties.....	5-15
5.4.3.5	Inserting Tables and Forms.....	5-16
5.4.3.6	Grouping.....	5-16
5.4.3.7	Understanding Fields Inserted to the Template	5-17
5.4.4	Inserting a Chart	5-17
5.4.4.1	Chart Type	5-18
5.4.4.2	Values	5-18
5.4.4.3	Aggregation.....	5-18
5.4.4.4	Labels.....	5-18
5.4.4.5	Color	5-19
5.4.4.6	Chart is Inside Group.....	5-19
5.4.4.7	Style	5-19
5.4.4.8	Properties.....	5-19
5.4.4.9	Preview	5-19
5.4.4.10	Group Data	5-19
5.4.4.11	Editing an Inserted Chart	5-19
5.4.5	Inserting a Pivot Table	5-19
5.4.6	Inserting a Repeating Group.....	5-22
5.4.7	Creating Grouping Fields Around an Existing Block	5-24
5.4.8	Inserting and Editing Conditional Regions	5-24
5.4.9	Inserting Conditional Formatting	5-25
5.5	Previewing a Template	5-27
5.6	Template Editing Tools	5-27
5.6.1	Editing and Viewing Field Properties	5-27
5.6.1.1	About the Properties Tab	5-28
5.6.1.2	About the Advanced Tab	5-29
5.6.1.3	About the Word Properties Button.....	5-29
5.6.2	Validating a Template.....	5-29
5.6.3	Using the Field Browser.....	5-29
5.6.4	Checking Accessibility	5-30
5.7	Uploading a Template to the BI Publisher Server.....	5-31
5.8	Using the Template Builder Translation Tools.....	5-31
5.8.1	About Translations	5-31
5.8.2	Extracting Text to an XLIFF File for Translation.....	5-32
5.8.3	Previewing the Template and Translation File	5-32
5.8.4	Localizing a Template	5-32
5.9	Setting Options for the Template Builder	5-33
5.9.1	Setting UI Options	5-33

5.9.2	Setting Preview Options	5-33
5.9.3	Setting Build Options	5-34
5.9.4	Setting Connection Options	5-36
5.10	Setting Up a Configuration File	5-36
5.11	BI Publisher Menu Reference	5-36
5.11.1	About the Online Group	5-36
5.11.2	About the Load Data Group	5-37
5.11.3	About the Insert Group	5-38
5.11.4	About the Preview Group	5-39
5.11.5	About the Tools Group	5-39
5.11.6	About the Options Group	5-40

6 Creating Excel Templates

6.1	Overview	6-1
6.1.1	Features of Excel Templates	6-1
6.1.2	Limitations of Excel Templates	6-1
6.1.3	Prerequisites	6-2
6.1.4	Supported Output	6-2
6.1.5	Desktop Tools for Excel Templates	6-2
6.1.6	Sample Excel Templates	6-2
6.2	Concepts	6-2
6.2.1	Identifying Data Field Placeholders and Groups	6-2
6.2.2	Use of Excel Defined Names	6-3
6.2.3	About the XDO_ Defined Names	6-3
6.2.4	Using Native Excel Functions	6-3
6.2.5	About the XDO_METADATA Sheet	6-3
6.3	Building a Simple Template	6-4
6.3.1	Step 1: Obtain Sample XML Data from the Data Model	6-4
6.3.2	Step 2: Open the BlankExcelTemplate.xls File and Save As the Template Name	6-5
6.3.3	Step 3: Design the Layout in Excel	6-5
6.3.4	Step 4: Assign the BI Publisher Defined Names	6-6
6.3.4.1	Applying a Defined Name to a Cell	6-6
6.3.4.2	Understanding Groups	6-8
6.3.4.3	Creating Groups in the Template	6-8
6.3.4.4	Deleting Defined Names and Groups	6-9
6.3.5	Step 5: Prepare the XDO_METADATA Sheet	6-9
6.3.5.1	Format of the XDO_METADATA Sheet	6-9
6.3.5.2	Creating the XDO_METADATA Sheet	6-10
6.3.5.3	Adding the Calculation for the XDO_?TOTAL_SALARY? Field	6-10
6.3.6	Step 6: Test the Template	6-11
6.4	Formatting Dates	6-12
6.5	Defining BI Publisher Functions	6-15
6.5.1	Reporting Functions	6-16
6.5.1.1	Splitting the Report into Multiple Sheets	6-16
6.5.1.2	Declaring and Passing Parameters	6-19
6.5.1.3	Defining a Link	6-20
6.5.1.4	Importing and Calling a Subtemplate	6-21

6.5.1.5	Referencing Java Extension Libraries	6-23
6.5.2	Formatting Functions That Rely on Specific Data Attribute Values	6-24
6.5.2.1	Defining Border and Underline Styles	6-24
6.5.2.2	Skipping a Row	6-29
6.5.3	Grouping Functions	6-31
6.5.3.1	Grouping the data	6-31
6.5.3.2	Handling the Generated XDO Define Names in Nested Groups	6-31
6.5.3.3	Regrouping the Data	6-32
6.6	Preprocessing the Data Using an XSL Transformation (XSLT) File	6-33
6.6.1	XSLT Preprocessing Examples: Splitting Flat Data into Multiple Sheets.....	6-34
6.6.1.1	Splitting the Data by a Specific Field	6-34
6.6.1.2	Splitting the Data by Count of Rows	6-36
6.7	Using the Template Viewer to Debug a Template	6-37

7 Creating PDF Templates

7.1	Overview	7-1
7.2	Requirements.....	7-2
7.3	Designing the Template	7-2
7.4	Adding Markup to the Template.....	7-4
7.5	Creating a Placeholder	7-4
7.5.1	Naming the Placeholder	7-4
7.5.2	Creating a Text Placeholder	7-4
7.5.3	Supported Field Properties Options	7-5
7.5.4	Creating a Check Box	7-5
7.5.5	Creating a Radio Button Group	7-6
7.6	Defining Groups of Repeating Fields.....	7-7
7.7	Adding Page Numbers and Breaks	7-8
7.7.1	Adding Page Numbers	7-8
7.7.2	Adding Page Breaks	7-8
7.8	Performing Calculations	7-11
7.9	Completed PDF Layout Example	7-12
7.10	Runtime Behavior	7-12
7.10.1	Placement of Repeating Fields	7-12
7.10.2	Setting Fields as Updatable or Read Only	7-13
7.10.3	Overflow Data	7-14
7.11	Creating a Layout from a Predefined PDF Form	7-14
7.11.1	Determining If a PDF Has Form Fields Defined	7-14
7.11.2	Using a Predefined PDF Form as a Layout by Renaming the Form Fields	7-14
7.11.3	Using the Comb of Characters Option	7-15
7.12	Adding or Designating a Field for a Digital Signature	7-17
7.12.1	About Signature Field Options	7-17
7.12.2	Adding a Signature Field.....	7-18
7.12.3	Configuring the Report to Insert the Digital Signature at Runtime	7-18

8 Creating eText Templates

8.1	Overview	8-1
-----	----------------	-----

8.1.1	Prerequisites	8-2
8.2	Structure of eText Templates.....	8-2
8.2.1	Command Rows, Data Rows, and Data Column Header Rows.....	8-3
8.2.2	Data Column Header Rows	8-4
8.2.3	Data Rows	8-4
8.3	Constructing the Data Tables.....	8-4
8.4	Command Rows.....	8-5
8.4.1	Level Command.....	8-5
8.4.2	New Record Command	8-8
8.4.3	Sort Ascending and Sort Descending Commands.....	8-8
8.4.4	Display Condition Command.....	8-8
8.5	Structure of the Data Rows.....	8-9
8.5.1	Position.....	8-9
8.5.2	Length/Maximum Length	8-9
8.5.3	Format Column	8-9
8.5.4	Number Data Type.....	8-10
8.5.5	Date Data Type.....	8-10
8.5.6	Mapping EDI Delimiter-Based Data Types to eText Data Types	8-10
8.5.7	Pad	8-11
8.5.8	Data.....	8-12
8.5.9	Tag.....	8-12
8.5.10	Comments	8-12
8.6	Setting Up Command Tables	8-12
8.6.1	TEMPLATE TYPE Command.....	8-13
8.6.2	DEFINE LEVEL Command.....	8-14
8.6.3	DEFINE SEQUENCE Command.....	8-17
8.6.4	RESET AT LEVEL.....	8-17
8.6.5	INCREMENT BASIS.....	8-18
8.6.6	Define Concatenation Command	8-18
8.6.7	Base Level Subcommand	8-18
8.6.8	Element Subcommand	8-18
8.6.9	Delimiter Subcommand	8-19
8.6.10	Using the SUBSTR Function.....	8-19
8.6.11	Invalid Characters and Replacement Characters Commands	8-20
8.6.12	Output Character Set and New Record Character Commands.....	8-20
8.6.13	Output Length Mode	8-21
8.6.14	Number Thousands Separator and Number Decimal Separator	8-21
8.6.15	CASE CONVERSION.....	8-21
8.7	Creating a Filler Block	8-21
8.8	Expressions, Control Structures, and Functions	8-23
8.8.1	Expressions	8-23
8.8.2	Control Structures.....	8-24
8.8.3	Functions.....	8-24
8.9	Identifiers, Operators, and Literals	8-26
8.9.1	Key Words	8-26
8.9.2	Command and Column Header Key Words	8-26
8.9.3	Command Parameter and Function Parameter Key Words	8-28

8.9.4	Field-Level Key Words	8-28
8.9.5	Expression Key Words.....	8-28
8.9.6	Operators	8-28
8.9.7	Reference to XML Extract Fields and XPATH Syntax.....	8-29
8.9.8	Notes on Viewing eText Output from a Browser	8-30

9 Creating Flash Templates

9.1	Overview	9-1
9.1.1	Prerequisites for Building and Viewing Flash Templates	9-1
9.1.2	Required Configuration Settings for Viewing PDF Output	9-2
9.2	Building a Flash Template	9-2
9.2.1	Adding the Data Source.....	9-2
9.2.2	Creating the Layout.....	9-5
9.2.2.1	Adding a Chart	9-7
9.2.3	Binding the Layout Objects to the Data Source.....	9-7
9.2.3.1	Binding the DataGrid.....	9-7
9.2.3.2	Binding the Chart	9-9
9.3	Uploading the Flash Template to the Report Definition.....	9-10
9.4	Setting Properties for PDF Output	9-10

10 Setting Report Processing and Output Document Properties

10.1	Overview	10-1
10.2	PDF Output Properties.....	10-2
10.3	PDF Security Properties	10-3
10.4	PDF Digital Signature Properties	10-6
10.5	PDF/A Output Properties	10-8
10.6	PDF/X Output Properties	10-8
10.7	RTF Output Properties	10-10
10.8	HTML Output Properties	10-10
10.9	FO Processing Properties.....	10-11
10.10	RTF Template Properties	10-13
10.11	PDF Template Properties.....	10-14
10.12	Flash Template Properties	10-15
10.13	CSV Output Properties.....	10-15
10.14	Excel 2007 Output Properties.....	10-16
10.15	All Outputs	10-16
10.16	Defining Font Mappings.....	10-16
10.16.1	Making Fonts Available to BI Publisher.....	10-17
10.16.2	Setting Font Mapping at the Site Level or Report Level	10-17
10.16.3	Creating a Font Mapping.....	10-17
10.16.4	BI Publisher's Predefined Fonts.....	10-18
10.16.4.1	Included Barcode Fonts	10-19

Part II Creating Style Templates and Sub Templates

11 Creating and Implementing Style Templates

11.1	Understanding Style Templates.....	11-1
11.1.1	About Styles Defined in the Style Template.....	11-1
11.1.1.1	Paragraph and Heading Styles.....	11-1
11.1.1.2	Table Styles.....	11-2
11.1.1.3	Header and Footer Content.....	11-2
11.1.2	Style Template Process Overview.....	11-2
11.2	Creating a Style Template RTF File.....	11-2
11.2.1	Defining Styles for Paragraphs and Headings.....	11-2
11.2.2	Defining Styles for Tables.....	11-3
11.2.3	Defining a Header and Footer.....	11-3
11.3	Uploading a Style Template File to the Catalog.....	11-4
11.4	Assigning a Style Template to a Report Layout.....	11-5
11.5	Updating a Style Template.....	11-5
11.6	Adding Translations to a Style Template Definition.....	11-6

12 Understanding Subtemplates

12.1	What is a Subtemplate?.....	12-1
12.1.1	About RTF Subtemplates.....	12-1
12.1.2	About XSL Subtemplates.....	12-2
12.2	Supported Locations for Subtemplates.....	12-2
12.3	Testing Subtemplates from the Desktop.....	12-2
12.4	Creating the Sub Template Object in the Catalog.....	12-2
12.5	Calling a Subtemplate from an External Source.....	12-3
12.5.1	Importing a Subtemplate Outside the Catalog over HTTP or FTP.....	12-4
12.5.2	Importing Subtemplates Outside the Catalog on the Same Server.....	12-4
12.5.3	Required Settings To Run Subtemplates Stored Outside the Catalog.....	12-4

13 Designing RTF Subtemplates

13.1	Understanding RTF Subtemplates.....	13-1
13.2	Process Overview for Creating and Implementing RTF Subtemplates.....	13-2
13.3	Creating an RTF Subtemplate File.....	13-2
13.4	Calling a Subtemplate from a Main Template.....	13-3
13.4.1	Importing the Subtemplate to the Main Template.....	13-3
13.4.2	Calling the Subtemplate to Render Its Contents.....	13-4
13.4.3	Importing a Localized Subtemplate.....	13-4
13.4.4	Example.....	13-5
13.5	When to Use RTF Subtemplates.....	13-5
13.5.1	Reusing a Common Layout.....	13-5
13.5.2	Conditionally Displaying a Layout Based on a Value in the Data.....	13-5
13.5.2.1	Example.....	13-6
13.5.3	Conditionally Displaying a Layout Based on a Parameter Value.....	13-6
13.5.3.1	Example.....	13-6
13.5.3.2	Handling Simple Calculations or Repeating Formulae.....	13-8
13.5.3.3	Example.....	13-8
13.6	Adding Translations to an RTF Subtemplate.....	13-9

14 Designing XSL Subtemplates

14.1	Understanding XSL Subtemplates	14-1
14.1.1	Where to Put XSL Code in the RTF Main Template	14-1
14.2	Process Overview for Creating and Implementing XSL Subtemplates	14-1
14.3	Creating an XSL Subtemplate File	14-2
14.4	Calling an XSL Subtemplate from the Main Template	14-3
14.4.1	Importing the Subtemplate	14-3
14.4.2	Calling the Subtemplate	14-3
14.4.3	Passing Parameters to an XSL Subtemplate	14-3
14.5	Creating the Sub Template Object in the Catalog	14-4
14.6	Example Uses of XSL Subtemplates	14-5
14.6.1	Handling XML Data with HTML Formatting	14-5
14.6.2	Dynamically Applying Formatting to a Portion of Data	14-7

Part III Translating Reports and Catalog Objects

15 Translation Support Overview and Concepts

15.1	Translation Support Overview	15-1
15.1.1	What Is Catalog Translation?	15-1
15.1.2	What Is Template Translation?	15-2
15.2	Working with Translation Files	15-2
15.2.1	What Is an XLIFF?	15-2
15.2.2	Structure of the XLIFF File	15-2
15.2.2.1	source-language and target-language attributes	15-3
15.2.2.2	Embedded Data Fields	15-4
15.2.2.3	<source> and <target> Elements	15-4
15.3	Locale Selection Logic	15-5

16 Translating Individual Templates

16.1	Overview	16-1
16.2	Types of Translations	16-1
16.3	Using the XLIFF Option	16-2
16.3.1	Generating the XLIFF from a Template	16-2
16.3.1.1	Generating the XLIFF from the Template Builder	16-3
16.3.1.2	Generating the XLIFF from the Layout Properties Page	16-3
16.3.2	Translating the XLIFF	16-3
16.3.3	Uploading the Translated XLIFF to BI Publisher	16-4
16.4	Using the Localized Template Option	16-4
16.4.1	Designing the Localized Template File	16-4
16.4.2	Uploading the Localized Template to BI Publisher	16-5

17 Translating Catalog Objects, Data Models, and Templates

17.1	Overview	17-1
17.2	What Can Be Translated?	17-1
17.2.1	About Source Language Limitations	17-2

17.3	Exporting the XLIFF File.....	17-2
17.4	Identifying and Updating the Object Tags.....	17-2
17.5	Importing the XLIFF File	17-3

Part IV Reference Information

A Techniques for Handling Large Output Files

A.1	Reusing Static Content	A-1
A.1.1	What Is Static Content Reuse?	A-1
A.1.2	Limitations of this Feature.....	A-2
A.1.3	Defining Reusable Content in an RTF Template.....	A-2
A.1.4	Example.....	A-2
A.2	Generating Zipped PDF Output.....	A-3
A.2.1	Limitations and Prerequisites	A-3
A.2.2	Design Time Considerations	A-3
A.2.3	Selecting the Output Type.....	A-4
A.3	Implementing PDF Splitting for an RTF Template.....	A-4
A.3.1	Entering the Commands in an RTF Template	A-4
A.3.2	Example - split by each department	A-5
A.4	Implementing PDF Splitting for a PDF Template.....	A-6
A.4.1	Entering the Commands in the PDF Template	A-6

B Extended Function Support in RTF Templates

B.1	Extended SQL and XSL Functions.....	B-1
B.1.1	Number-To-Word Conversion	B-9
B.2	XSL Equivalents	B-10
B.3	Using FO Elements	B-11

C Designing Accessible Reports

C.1	Avoiding Nested Tables or Separated Tables.....	C-1
C.1.1	Examples	C-2
C.1.1.1	Nested Tables	C-2
C.1.2	Table Headers Must Not Be Separated from the Table Body	C-2
C.2	Defining a Document Title	C-3
C.3	Defining Alternative Text for an Image.....	C-3
C.4	Defining a Table Summary.....	C-3
C.5	Defining a Table Column Header.....	C-3
C.6	Defining a Table Row Header.....	C-4
C.7	Sample Supported Tables	C-4

D Supported XSL-FO Elements

D.1	Supported XSL-FO Elements.....	D-1
D.1.1	Property Groups Table.....	D-6

Index

Preface

Oracle Business Intelligence Publisher (BI 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.

Audience

This document is intended for report designers and developers using Oracle Business Intelligence Publisher to create reports, configure reports, add translations, and use other BI Publisher report design features.

This guide describes how to create report layouts and use BI Publisher's report editor to assemble the components of a report. See the guides that are listed in the following table for more information about using the product for other business roles.

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

Documentation Accessibility

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

Access to Oracle Support

Oracle customers have access to electronic support through My Oracle Support. For information, visit

<http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info> or visit <http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs> if you are hearing impaired.

Related Documentation and Other Resources

See the Oracle Business Intelligence documentation library for a list of related Oracle Business Intelligence documents.

In addition, go to the Oracle Learning Library for Oracle Business Intelligence-related online training resources.

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/technetwork/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/technetwork/software/products/ias/files/fusion_certification.html

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

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

Conventions

The following text conventions are used in this document:

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

New Features for Report Designers

This preface describes changes to Oracle BI Publisher report designing features for Oracle Business Intelligence Publisher 11g Release 1 (11.1.1).

This preface contains the following topics:

- [New Features for Oracle BI Publisher 11g Release 1 \(11.1.1.6\)](#)
- [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 (11.1.1.6)

New features for report designers in Oracle BI Publisher 11g Release 1 (11.1.1.6) include:

- [Improved Create Report Flow](#)
- [Enhancements to Parameter Display Options](#)
- [Support for HTML Formatting in Data](#)
- [Support for PDF/A and PDF/X Output](#)

Improved Create Report Flow

This release streamlines the report creation process introducing a process flow to better guide report designers. For more information, see [Section 2.3, "Creating a Report."](#)

Enhancements to Parameter Display Options

Enhancements to parameter display options include:

- Support for radio button and check box display of values
Parameters that are defined as a menu in the data model can now be configured to display the menu options as a list of radio buttons or check boxes. Parameters configured to support one value support the option to display as radio buttons. Parameters configured to support multiple values support the option to display as check boxes.
- New options for placement of parameters in the report viewer
Previously the report viewer always displayed the parameters in a horizontal region across the top of the viewer. In this release, the display of the parameter region can also be configured in one of the following ways:
 - In a vertical region along the right side of the viewer

- As a dialog
- As a separate full page
- New report viewer toolbar button to show or hide parameters

This release adds a toolbar button to the report viewer to enable users to hide or show the parameter region. This enhancement complements the new parameter display options. When report parameters are configured to display as a dialog or in a separate full page, the parameter display region is dismissed when the viewer displays the report. Use the **Parameters** button to redisplay the parameter region to make new selections.
- Search added to menus

All parameter menus having more than a specified number of options provide a Search option.
- Option to remove the **Apply** button

Reports can now be configured to remove the parameter **Apply** button. In these reports the action of selecting a new parameter value automatically reruns the report.

For more information about these enhancements, see [Section 2.7, "Configuring Parameter Settings for the Report."](#)

Support for HTML Formatting in Data

You can now convert stored XHTML to XSL-FO to display the HTML formatting from your data in your generated report. The XHTML data must be extracted wrapped in a CDATA section. Specific syntax must also be used in the RTF template to render it. See [Section 4.23, "Rendering HTML Formatted Data in a Report"](#) for more information.

Support for PDF/A and PDF/X Output

PDF/A is a variation of the PDF standard optimized for long term archival of documents. PDF/X is a variation of the PDF standard optimized for professional grade printing. BI Publisher now supports these two types of output. For more information, see [Section 2.9, "Generating PDF/A and PDF/X Output."](#)

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](#)
- [Excel Template Builder](#)
- [Interactive List Component for BI Publisher Layouts](#)

Excel Templates

An Excel template is a report layout that you design in Microsoft Excel for retrieving 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 the data in Excel output
- Split hierarchical data across multiple sheets and dynamically name the sheets
- Create sheets of data that have master-detail relationships

- Use native XSL functions in the data to manipulate it prior to rendering
- Use native Excel functionality

For more information, see [Chapter 6, "Creating Excel Templates."](#)

Excel Template Builder

BI Publisher provides a downloadable add-in to Excel that enables you to preview the Excel template with sample data. This facilitates design by enabling you to test and edit the 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 [Section 6.1.5, "Desktop Tools for Excel Templates."](#)

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 [Section 3.14, "About Lists."](#)

Redesigned Formula Dialog for BI Publisher

The formula dialog in the BI Publisher layout editor has been redesigned for better usability. For more information, see [Section 3.15, "Setting Predefined or Custom Formulas."](#)

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](#)
- [Shared BI Presentation Catalog](#)
- [Layout Editor Design Tool](#)
- [Interactive Viewer](#)
- [Sub Templates](#)
- [Style Templates](#)

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. For information about working in the new interface, see *Oracle Fusion Middleware User's Guide for Oracle Business Intelligence Publisher*.

Shared 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. For information

about the improved catalog, see *Oracle Fusion Middleware User's Guide for Oracle Business Intelligence Publisher*.

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. For a complete description, see [Chapter 3, "Creating BI Publisher Layout Templates."](#)

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. For more information, see [Section 3.1, "Overview of BI Publisher Layouts."](#)

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 RTF and XSL subtemplates can be saved and managed as objects in the BI Publisher catalog. For more information, see [Chapter 12, "Understanding Subtemplates."](#)

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. For more information, see [Chapter 11, "Creating and Implementing Style Templates."](#)

Zipped 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. For more information, see [Section A.2, "Generating Zipped PDF Output."](#)

Part I

Creating Reports and Layouts

This part describes how to create reports and layouts. It includes the following chapters:

- [Chapter 1, "Introduction to Designing Reports"](#)
- [Chapter 2, "Creating and Editing Reports"](#)
- [Chapter 3, "Creating BI Publisher Layout Templates"](#)
- [Chapter 4, "Creating RTF Templates"](#)
- [Chapter 5, "Creating RTF Templates Using the Template Builder for Word"](#)
- [Chapter 6, "Creating Excel Templates"](#)
- [Chapter 7, "Creating PDF Templates"](#)
- [Chapter 9, "Creating Flash Templates"](#)
- [Chapter 8, "Creating eText Templates"](#)
- [Chapter 10, "Setting Report Processing and Output Document Properties"](#)

Introduction to Designing Reports

This chapter introduces the components that comprise a BI Publisher report.

This chapter includes the following sections:

- [Section 1.1, "Overview for Report Designers"](#)
- [Section 1.2, "About the Layout Types"](#)
- [Section 1.3, "About Setting Run-Time Properties"](#)
- [Section 1.4, "About Translations"](#)
- [Section 1.5, "About Style Templates"](#)
- [Section 1.6, "About Sub Templates"](#)

1.1 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 designers 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 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.

1.2 About the Layout Types

BI Publisher offers several options for designing layouts for 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 plug-in 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 must 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 PDF templates using Adobe Acrobat Professional. PDF templates support only PDF output.
- Microsoft Excel (XLS)

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

If you must only view report data in Excel, then 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 the 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.

1.3 About Setting Run-Time Properties

BI Publisher provides a variety of user-controlled settings that are specified using an easily accessible Properties dialog. 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 [Chapter 10, "Setting Report Processing and Output Document Properties."](#) These settings are also configured at the system-level, but can be customized per report.

1.4 About Translations

BI Publisher provides the ability to create an XLIFF file from 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 an 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 runs automatically for users who select the corresponding locale. For more information, see [Chapter 15, "Translation Support Overview and Concepts."](#)

1.5 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 [Chapter 11, "Creating and Implementing Style Templates."](#)

1.6 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 [Chapter 12, "Understanding Subtemplates."](#)

Creating and Editing Reports

This chapter provides an overview of the BI Publisher report components and describes how to create and edit report definitions.

This chapter includes the following topics:

- [Section 2.1, "About Report Components"](#)
- [Section 2.2, "Creating a Report: Process Overview"](#)
- [Section 2.3, "Creating a Report"](#)
- [Section 2.4, "Editing Reports"](#)
- [Section 2.5, "Adding Layouts to the Report Definition"](#)
- [Section 2.6, "Configuring Layouts Using the List View"](#)
- [Section 2.7, "Configuring Parameter Settings for the Report"](#)
- [Section 2.8, "Configuring Report Properties"](#)
- [Section 2.9, "Generating PDF/A and PDF/X Output"](#)

2.1 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. For information on creating a data model, see *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 describes 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 sections that are listed in [Table 2-1](#).

Table 2-1 Information on Report Components

Topic	More Information
Creating a data model	See the <i>Oracle Fusion Middleware Data Modeling Guide for Oracle Business Intelligence Publisher</i>
Selecting a layout type	"Section 1.2, "About the Layout Types""
Creating specific layout template types	Chapter 3, "Creating BI Publisher Layout Templates" Chapter 4, "Creating RTF Templates" Chapter 5, "Creating RTF Templates Using the Template Builder for Word" Chapter 6, "Creating Excel Templates" Chapter 7, "Creating PDF Templates" Chapter 9, "Creating Flash Templates" Chapter 8, "Creating eText Templates" "Using the Analyzer for Excel" in the <i>Oracle Fusion Middleware User's Guide for Oracle Business Intelligence Publisher</i>
Translating reports	Chapter 15, "Translation Support Overview and Concepts"

2.2 Creating a Report: Process Overview

The create report process guides you through the steps to create the basic report:

1. Launch the Create Report guide and choose a data source.
See [Section 2.3, "Creating a Report."](#)
2. Create the report layout.
See [Section 2.5, "Adding Layouts to the Report Definition."](#)
3. Configure the properties for the layout.
See [Section 2.6, "Configuring Layouts Using the List View."](#)
4. Configure parameters for the report.
See [Section 2.7, "Configuring Parameter Settings for the Report."](#)
5. Configure report properties.
See [Section 2.8, "Configuring Report Properties."](#)
6. Add translations for the layouts. Complete this step if the report requires support for multiple languages.

2.3 Creating a Report

Launch the **Create Report** process 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**.
- From the catalog toolbar, click **New** and then click **Report**.

2.3.1 Choose Data Source

Choose one of the following options to begin building your report:

- Use Existing Data Model

Select an existing data model from the catalog. Locate the data model by navigating through the catalog or by using the **Search** option. (When using the shared BI Enterprise Edition catalog, **Search** must be configured. See "Configuring BI Publisher to Use the Oracle BI EE Catalog" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Business Intelligence Publisher* for more information.)

Click **Next** to proceed to **Choose Guide or Create**.

- Upload Spreadsheet

Upload a Microsoft Excel file (file type.xls). If the uploaded spreadsheet contains multiple sheets, select the sheet to use as the data source. You can include data from one sheet only.

Click **Next** to proceed to **Choose Guide or Create**.

- Create New Data Model

This option opens the data model editor for you to create a new data model.

For more information on creating data models, see "Using the Data Model Editor" in the *Oracle Fusion Middleware Data Modeling Guide for Oracle Business Intelligence Publisher*.

After creating the data model, launch the **Create Report** process again. This time choose **Use Existing Data Model**.

2.3.2 Choose Guide or Create

Table 2–2 describes the options on the Guide or Create dialog.

Table 2–2 Choosing Guide or Create

Option	Description
Guide Me	<p>This option guides you through selecting specific columns from your data source for this report and viewing a simple tabular layout. Choose this option to:</p> <ul style="list-style-type: none"> ■ Quickly view the data ■ Create a simple tabular view of the data ■ Be guided directly to the layout editor to create a layout for the selected data source <p>Proceed to Section 2.3.3, "Select Columns."</p>
Use Report Editor	<p>This option prompts you to save the report and then opens the report editor. Choose this option when you do not need to review the data and you want to proceed to configuring the report or creating a layout. Proceed to Section 2.4, "Editing Reports."</p>

2.3.3 Select Columns

Select the columns from the data source to include in the report. As you select columns, sample data for the column displays. The selected columns are displayed in the order selected.

Note the following about sample data:

- For data models: Sample data must be saved to the data model. If sample data is not attached to the data model, the selected column headings display without data.
- For uploaded spreadsheets: Sample data displays from the spreadsheet.

2.3.4 Choose View Report or Customize Layout

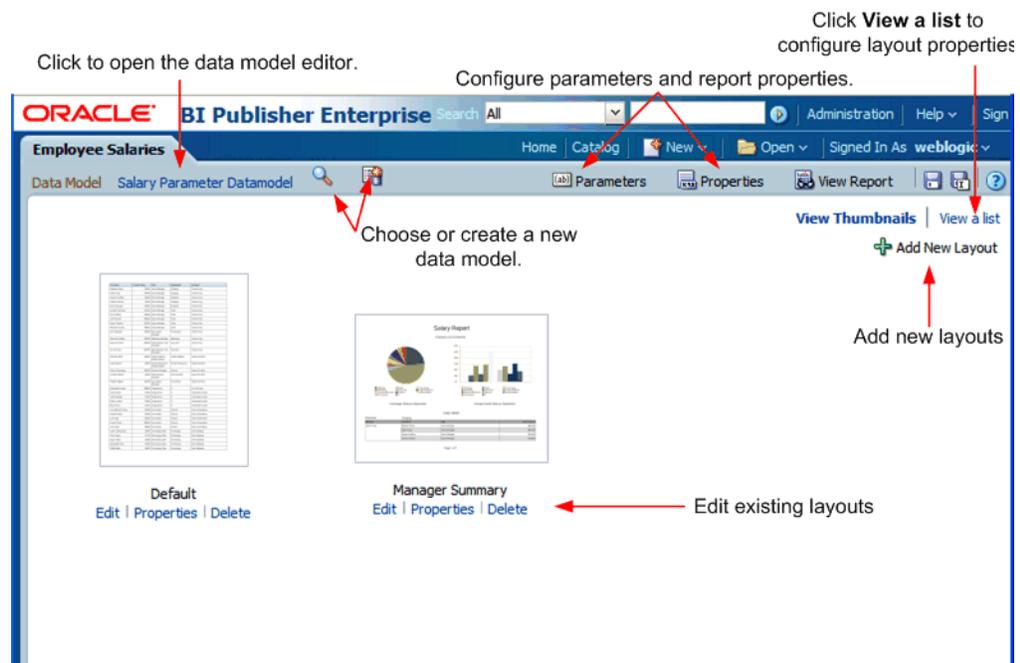
View Report: To view the report data in a simple tabular format, choose **View Report** and click **Finish**. The final dialog prompts you to save the report. After saving, BI Publisher runs and displays the report in the report viewer.

Customize Layout: To customize the layout, choose **Customize Layout** and click **Finish**. The final dialog prompts you to save the report. After saving, the report opens in the layout editor.

2.4 Editing Reports

The report editor is shown in [Figure 2–1](#). Navigate to the report editor through the Create Report process flow or by clicking a report's **Edit** link in the catalog.

Figure 2–1 Edit Report Page



Use the report editor to:

- Add layouts
- Configure layouts

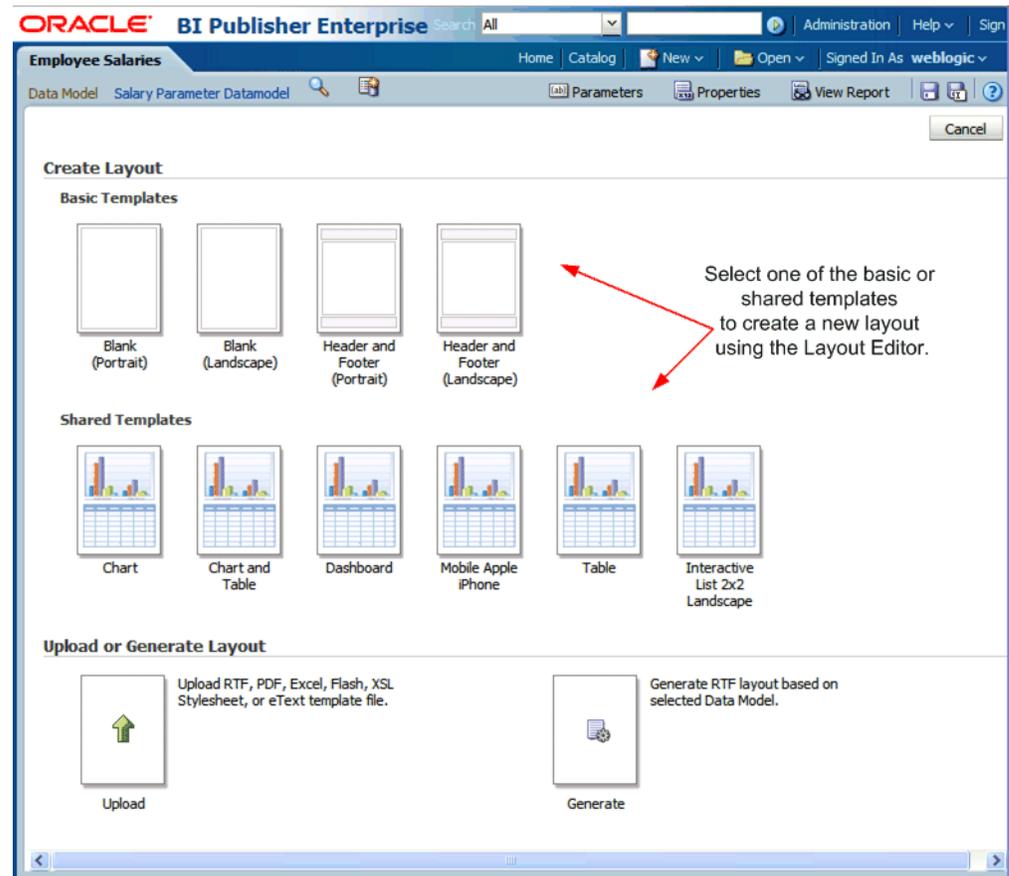
- Configure the parameters for this report
- Configure the report properties
- Update the data model associated with the report

This procedure addresses these options in the order listed.

2.5 Adding Layouts to the Report Definition

To add a new layout, from the report editor, click **Add New Layout**. The **Add Layout** page is shown in [Figure 2-2](#).

Figure 2-2 Add Layout



There are three options for adding a layout to a 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.

2.5.1 Adding a Layout Using the Layout Editor

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. The shared templates are preformatted layouts with common report components already inserted.
2. Design the template. For information on using the Layout Editor, see [Chapter 3, "Creating BI Publisher Layout Templates"](#).
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 [Section 2.6, "Configuring Layouts Using the List View."](#)

2.5.2 Adding a Layout by Uploading a Template File

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).

To add a layout by uploading a template file:

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 the 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.

For more information, see [Chapter 5, "Creating RTF Templates Using the Template Builder for Word"](#) or "Using the Analyzer for Excel" in *Oracle Fusion Middleware User's Guide for Oracle Business Intelligence Publisher*.

2.5.3 Adding a Layout by Generating a Template File

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**.

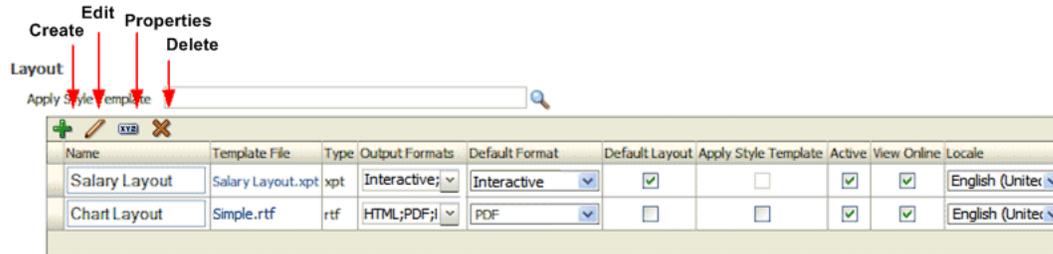
The autogenerate feature creates a simple table-based RTF layout that includes all the fields in the data model.

Note: 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.

2.6 Configuring Layouts Using the List View

After creating or uploading the layouts for the report, you can configure settings for the layout from the **List View**. [Figure 2–3](#) shows the **List View**.

Figure 2–3 The List View



2.6.1 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 [Chapter 11, "Creating and Implementing Style Templates."](#)

If you want to apply a style template to this layout, then 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.

2.6.2 About the Layouts Toolbar

The Layout toolbar buttons are described in [Table 2–3](#).

Table 2–3 Layout Toolbar Buttons

Toolbar Button	Description
Create	Launches the add layout page to upload or create a new layout.
Edit	Launches the Layout Editor for the selected layout. This button is enabled for BI Publisher layouts (.xpt) only.
Properties	Launches the Properties page to enable the upload of localized templates and XLIFF files to associate with this layout. This button is enabled for RTF (.rtf) and BI Publisher layouts (.xpt) only. For more information on localizing templates, see Chapter 15, "Translation Support Overview and Concepts."
Delete	Deletes the selected layout.

2.6.3 Configuring the Layout Settings Using the List View

The **List View** settings enable you to configure the following settings for the layout. The settings are described in [Table 2–4](#).

Table 2–4 List View Settings

Setting	Description
Name	Place the cursor in the text box to enter a new name for the layout.
Template File	Displays the name of the file that was saved to the report definition. Click the template file name to download it.

Table 2–4 (Cont.) List View Settings

Setting	Description
Type	Displays the template file type.
Output Formats	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. See Section 2.6.4, "Selecting Output Formats" for the complete list.
Default Format	Select the default output format for this layout when viewed or scheduled.
Default Layout	Select the layout that this report uses by default when viewed online or scheduled. Only one box in this column can be checked.
Apply Style Template	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 Section 2.6.1, "Applying a Style Template to the Layout" .
Active	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 does not display in the report viewer or the scheduler.
View Online	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.
Locale	Displays the locale selected when the layout was uploaded. This field is not updatable.

2.6.4 Selecting Output Formats

BI Publisher supports a wide range of output formats. Different layout types support different output types. [Table 2–5](#) lists all possible output types. [Table 2–6](#) lists valid output formats for each layout type.

Table 2–5 Output Formats

Output Format	Description
Data (CSV)	Enable this option to generate comma separated value output.
Data (XML)	Enable this option to generate XML output.
Excel	Enable this output type to view the report in Excel when report consumers have Microsoft Excel 2003. This option generates an MHTML format file that can be opened in Excel 2003 or later. This option supports embedded images, such as charts and logos.
Excel 2000	Enable this output type only when report consumers are running the earlier versions of Microsoft Excel 2000 or 2002. This option generates HTML that can be opened in Excel. It does not support embedded images.

Table 2–5 (Cont.) Output Formats

Output Format	Description
Excel 2007	<p>Enable this option to generate the report in Excel .xlsx (Excel XML format). If your report consumers have Excel 2007 or later installed, this option provides the best preservation of layout and formatting.</p> <p>Note: For this output format BI Publisher does not apply any formatting for number and date. BI Publisher saves the formatting mask and the actual value (date or number) into the XLSX output file. The formatting is handled by Microsoft Excel. For example:</p> <ul style="list-style-type: none"> ■ If the Microsoft Windows Region and Language of the client computer is set to English (United States), then the numbers and dates are formatted in en-US locale in the Excel 2007 output file. ■ If the Microsoft Windows Region and Language of the client computer is set to French (France), then the numbers and dates in the same Excel 2007 output file are formatted in fr-FR locale.
HTML	Enable HTML output for reports that require browser viewing.
Interactive	This output is only available for layouts designed using BI Publisher's Layout Editor. Interactive output enables pop-up chart value displays, scrollable and filterable tables, and other interactive features for a report.
MHTML	Enable Mime Hyper Text Markup Language to enable the report consumer to save a Web page and its resources as a single MHTML file (.mht), in which all images and linked files are saved as a single entity. A report consumer would use this option to send or save HTML output and retain the embedded images and stylesheet formatting.
PDF	Portable Document Format is commonly required for reports that require printing or sharing.
PDF/A	Use for reports that require long-term preservation or archiving. PDF/A is a specialized subset of the PDF standard that prohibits elements that may interfere with the preservation of the file as a self-contained document. For more information see Section 2.9.1, "Generating PDF/A Output."
PDF/X	Use for reports that require formatting for prepress graphics exchange. PDF/X is a specialized subset of the PDF standard that streamlines documents for high-quality print production output and restricts content that does not serve the print production, such as signatures, comments, and embedded multimedia. For more information see Section 2.9.2, "Generating PDF/X Output."
PowerPoint	Enable this output type to generate a PowerPoint file compatible with Microsoft PowerPoint 2003.
PowerPoint 2007	Enable this output type to generate a Microsoft PowerPoint 2007 file.
RTF	Rich Text Format. Enable this output for reports that must be opened for editing.
Text	This option generates text output for eText templates. Text output is available only for eText templates.

Table 2–5 (Cont.) Output Formats

Output Format	Description
Zipped PDF	BI Publisher can generate a zip file containing the report PDF output and index files. This option is only available for layouts that have been designed to enable zipped PDF output. For information on designing a report to generate zipped PDF, see "Generating Zipped PDF Output."

Table 2–6 Valid Output Formats for Each Layout Type

Layout Type	Valid Output Types
PDF	PDF, Zipped PDF, Data (CSV), Data (XML)
RTF	HTML, PDF, RTF, Excel, Excel2000, Excel 2007, PowerPoint, PowerPoint2007, MHTML, PDF/A, PDF/X, Data (CSV), FO Formatted XML, Data (XML), Zipped PDFs
XPT (created using the layout editor)	Interactive, HTML, PDF, PDFZ, RTF, Excel, Excel2000, PowerPoint, PowerPoint2007, MHTML, Data (CSV), FO Formatted XML, Data (XML),
XLS	Excel, Excel2000, Data (CSV), Data (XML),
Flash	Flash, PDF, MHTML, Data (CSV), Data (XML),
XSL Stylesheet (FO)	Same outputs as RTF
XSL Stylesheet (HTML XML/Text)	HTML, XML, Text, Data (CSV), Data (XML),
eText	Text, Data (CSV), Data (XML),

2.6.5 Editing a Layout

To edit a BI Publisher layout (.xpt file type):

1. Select the report from the list and click Edit.

To edit any other template type:

1. Click the File name link to download the layout to a local computer for editing.

2.7 Configuring Parameter Settings for the Report

Parameters are defined in the data model. 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 displayed as shown in [Figure 2–4](#).

Figure 2–4 The Parameters Dialog

Show	Name	Type	Multiple	Display Label	Default Value	Row Placement
<input checked="" type="checkbox"/>	P_MIN_SAL	Text	False	Minimum Salary		1
<input checked="" type="checkbox"/>	P_MAX_SAL	Text	False	Maximum Salary		1
<input checked="" type="checkbox"/>	P_YEAR	Radio Buttons	False	Year	Default	2
<input checked="" type="checkbox"/>	p_last_name	Menu	True	Employee Last Name(s)	Default	2
<input checked="" type="checkbox"/>	p_hire_date_from	Date	False	Hire Date From		3
<input checked="" type="checkbox"/>	p_hire_date_to	Date	False	Hire Date To		3

2. Customize the parameter settings for this report by making selections for the following display options:

Parameter Location

This property controls where the parameter region is displayed in the report viewer. The options are:

- Horizontal Region - displays the parameters horizontally across the top of the report viewer
- Vertical Region - displays the parameters vertically along the left side of the report viewer
- Full Page - displays the parameters on a separate page in the report viewer. After a user enters parameter values, the page is dismissed. To change parameter values, click the report viewer **Parameters** button to display the parameters page again.
- Dialog - invokes a dialog box to display the parameters. After a user enters parameter values, the dialog is dismissed. To change parameter values, click the report viewer **Parameters** button to display the dialog again.

Parameter Label Location

This property controls where the parameter labels are displayed. The options are:

- Place label on side - places the parameter label to the left side of the entry box.
- Place label on top - places parameter label on top of the entry box.

Show Apply Button

This property controls the display of the **Apply** button in the report viewer.

When set to True, reports with parameter options will display the **Apply** button in the report viewer. When a user changes the parameter values, he must click **Apply** to render the report with the new values.

When set to False, the report viewer does not display the **Apply** button. Instead, when a user enters a new parameter value, BI Publisher automatically renders the report after the new value is selected or entered without further action from the user.

This property is also set at the server level. To always use the server setting, choose the Default option. For information about configuring the server setting, see "Setting Report Viewer Properties" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Business Intelligence Publisher*.

When deciding whether to remove the **Apply** button consider the ability of the underlying data sources to quickly return data. Lists of values that are based on static lists or very fast data sources are ideally suited to turning off the **Apply** button. If the underlying data sources for the lists of values queries are slow, or if there are many parameter values to set and refine before rendering the report, then retaining the **Apply** button is recommended.

Show

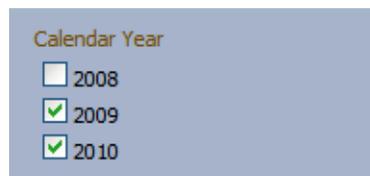
This property controls whether the parameter is displayed to the user. Disable the **Show** property if you do not want the user to see or change the parameter values that are passed to the data model.

Type

This property is customizable for menu type parameters only. For menu type parameters, the following display options are available:

- **Checkbox**
 If the parameter allows multiple selections, (Multiple = True) this option is available. The checkbox type displays all menu options in the parameter region of the report. Users can make multiple selections, as shown [Figure 2-5](#).

Figure 2-5 Menu Displayed as Checkboxes



- **Radio button**
 If the parameter allows only a single selection (Multiple = False), this option is available. The radio button type displays all menu options in the parameter region of the report. Users can make a single selection as shown in [Figure 2-6](#).

Figure 2-6 Menu Displayed as Radio Buttons



Tip: The checkbox and radio button options are best suited for menus when the list of values is small. These options also display well when the **Parameter Location** is the vertical region.

Multiple

This property is display only; it indicates whether multiple values may be selected for a menu parameter.

Display Label

Use this property to edit the display labels shown for each parameter. The default values are defined in the data model.

Default Value

Use this property to configure the default value for the parameter specifically for this report. Choose "Default" to pass the default value defined in the data model.

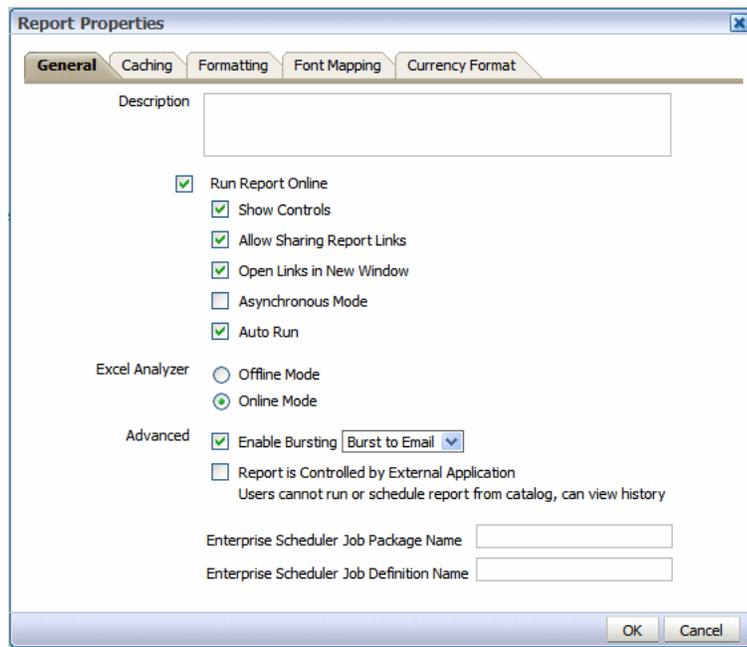
Row Placement

Use this property to configure the number of rows for displaying the parameters and in which row to place each parameter. For example, if your report has six parameters, you can assign each parameter to a separate row, 1 - 6, with one being the top row; or, you can assign two parameters each to rows 1, 2, 3. By default, all parameters are assigned to row 1.

Note that when the **Parameter Location** property is set to Vertical Region, only one parameter displays per row. You can use the Row Placement property to order the rows vertically.

2.8 Configuring Report Properties

Launch the **Report Properties** dialog by clicking **Properties** in the page header. [Figure 2-7](#) shows the **Report Properties** dialog.

Figure 2–7 The Report Properties Dialog

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

- **General** - Set general properties for the report.
- **Caching** - Specify caching options for this report.
- **Formatting** - Set the runtime configuration properties for the report. See [Chapter 10, "Setting Report Processing and Output Document Properties"](#) for information on setting these properties.
- **Font Mapping** - Create font mappings for this report.
- **Currency Format** - Define currency formats for this report.

2.8.1 Setting the General Properties

Set the properties on the General tab as follows:

2.8.1.1 Description

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

2.8.1.2 Run Report Online

Disable this property if you do not want users to view this report in the online Report Viewer. When disabled, users can Schedule the report only. For most reports you 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 properties that are described in [Table 2–7](#).

Table 2–7 Run Report Properties

Property	Description	Default
Show controls	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.	Enabled
Allow Sharing Report Links	The Actions menu of the Report Viewer includes the option Share Report Link , 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.	Enabled
Open Links in New Window	This property controls how links contained within a report are opened. By default links open in a new browser window. Disable this property to open links in the same browser window.	Enabled
Asynchronous Mode	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.	Not Enabled
Auto Run	When this property is enabled the report automatically runs when the user selects the Open link for the report. When Auto Run is disabled, selecting the Open link for the report displays the online viewer but does not run the report. The user must select an output type from the View Report menu to run the report.	Enabled

2.8.1.3 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:

- The 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 an 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 want to manipulate in Excel. Note that the Offline Mode also requires the data to be in <ROWSET><ROW>...</ROWS>...</ROWSET> format.

Table 2–8 details the differences between the online and offline modes.

Table 2–8 Differences Between Online and Offline Mode

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.
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 the Excel session. You can directly upload templates to the report, update the report parameters, and apply new templates from within the 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 the "Using the Analyzer for Excell" section in *Oracle Fusion Middleware User's Guide for Oracle Business Intelligence Publisher*.

2.8.1.4 Advanced Options

Table 2–9 describes the Advanced properties.

Table 2–9 Description of Report Properties Advanced Options

Property	Description
Enable Bursting	To enable bursting of the report output, select this box and then select the appropriate bursting definition from the list. When a user schedules the report, the selected bursting definition will be enabled in the Scheduler. The bursting definition is a component of the data model. For more information, see the "Adding Bursting Definitions" section in <i>Oracle Fusion Middleware Data Modeling Guide for Oracle Business Intelligence Publisher</i> .
Report is Controlled by External Application. Users cannot run or schedule report from catalog, can view history	Enable this option when 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. Reports run by BI Publisher are stored in the BI Publisher history tables and users can view completed reports from the Report Job History page.
Enterprise Scheduler Job Package Name	This property applies to Oracle Fusion Applications implementations only.
Enterprise Scheduler Job Definition Name	This property applies to Oracle Fusion Applications implementations only.

2.8.2 Setting the Caching Properties

Figure 2–8 shows the Caching tab.

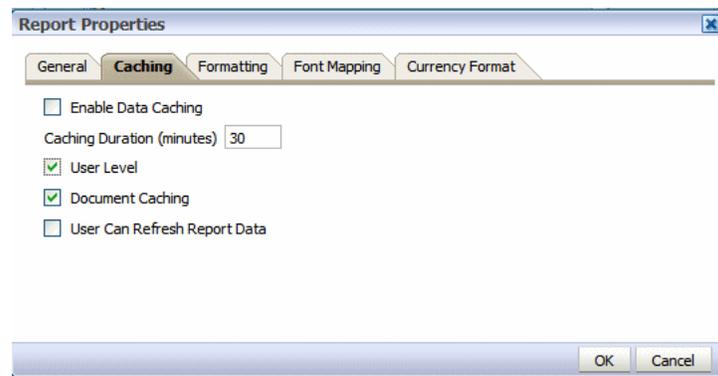
Figure 2–8 The Caching Tab

Table 2–10 describes the properties on the Caching tab.

Table 2–10 Caching Properties

Variable	Description	Default
Enable Data Caching	When this property is enabled, the data generated by the online submission of this report is stored in the cache. Subsequent requests to run this report with the same parameter selections display the report using the data that is 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 remains in the cache according to the time limit that is 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:	Not Enabled
Caching Duration (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 generates a fresh data set.	30 minutes
User Level	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.	Enabled
Document Caching	Enable this property to cache the report document. With document cache enabled, when a user views the report online, the document (data plus layout) is placed in the 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 is retrieved from the cache. The document remains in the cache according to the caching duration specified. Note that scheduled reports do not use document cache.	Enabled
User Can Refresh Report Data	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 generates a fresh data set for the report.	Not Enabled

2.8.3 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 takes precedence.

For a full description of each property, see [Chapter 10, "Setting Report Processing and Output Document Properties."](#)

2.8.4 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 are displayed. To change the settings for this report, edit the font mappings here.

For detailed information on font mapping, see [Section 10.16, "Defining Font Mappings."](#)

To create a font mapping:

1. Under RTF Templates or PDF Templates, select **Add Font Mapping**.
2. Enter the following on the **Add Font Mapping** page:
 - **Base Font:** Enter the font family to map 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, then 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 [Section 10.16.4, "BI Publisher's Predefined Fonts."](#)

2.8.5 Configuring Currency Formats

The Currency Formats tab enables you to map a number format mask to a specific currency so that 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 are displayed. To change the settings for this report, edit the currency formats here.

To apply these currency formats in an RTF template, you must use the format-currency function. See [Section 4.15.14, "Formatting Currencies"](#) 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

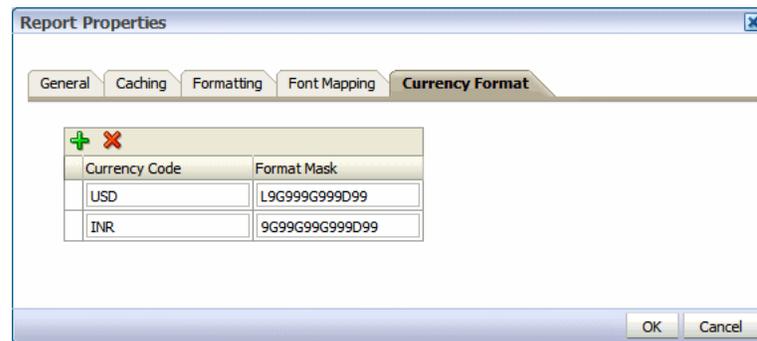
D represents the decimal separator

0 represents an explicitly displayed number regardless of incoming data

For more information about these format mask components, see [Section 4.15.6, "Using the Oracle Format Mask."](#)

Figure 2–9 shows sample currency formats.

Figure 2–9 Sample Currency Formats



2.9 Generating PDF/A and PDF/X Output

This section describes requirements for generating PDF/A and PDF/X output. It contains the following topics:

- [Section 2.9.1, "Generating PDF/A Output"](#)
- [Section 2.9.2, "Generating PDF/X Output"](#)

2.9.1 Generating PDF/A Output

PDF/A is a variation of PDF file format designed for the long-term archiving of electronic documents. Some governments and standards organizations require PDF/A to ensure preservation of documents. A PDF/A file is a PDF file viewable by PDF viewers such as Adobe Reader, but the PDF/A file must follow additional requirements specified in the ISO standard. These requirements specify both required objects and features not supported for long-term archiving. BI Publisher generates the PDF/A-1b variation of the PDF/A standard.

2.9.1.1 Requirements and Limitations

The following lists limitations of and requirements for generating PDF/A output:

- **Supported template types:** The following template types support the generation of PDF/A: RTF, FO, and XSL. There are no additional template requirements to generate PDF/A.
- **Font requirements:** By default, all fonts in the template are replaced with Albany fonts in the output. To use a different font in the output, specify the font mappings in the report configuration. If Albany fonts are not available in the JVM font directory and you have not specified a font mapping, the output will not become a valid PDF/A file. In this case, Helvetica font is used.
- **PDF features not supported in PDF/A documents:**
 - All audio and video content, including Flash embedding
 - Transparency (transparent colors render as opaque)
 - Encryption
- **Reprocessing by BI Publisher utilities is not supported:** Reprocessing of the PDF/A file using the BI Publisher utilities PDFBookBinder, PDFDocMerger, or PDFSignature (digital signature) is not supported. The reprocessed PDF file may lose conformance to the PDF/A standard.
- **Required report configuration properties:** The report run-time properties must be set as shown in [Table 2–11](#).

Table 2–11 Required Report Property Settings for PDF/A

Property	Required Setting
pdf-version	Must be no higher than 1.4
Enable PDF Security (pdf-security)	Must be set to false
Encryption Level (pdf-encryption-level)	Must be set to 0
pdf-font-embedding	Must be set to true

Formatting properties specific to PDF/A output can be set in the **Report Properties** dialog. For more information, see [Section 10–4, "PDF/A Output Properties."](#)

2.9.1.2 Additional Resources

For more information about the PDF/A standard, see the following resources:

- <http://www.adobe.com/enterprise/standards/pdfa/>
- ISO 190005-1 Document management — Electronic document file format for long-term preservation: Part 1: Use of PDF 1.4 (PDF/A-1); Addendum Cor 1:2007

2.9.2 Generating PDF/X Output

PDF/X is a collection of ISO standards that defines methods for the exchange of digital graphic data using PDF to ensure predictable and consistent printing in a professional print environment. A PDF/X document is a PDF file viewable by PDF readers such as Adobe Reader, but it follows an additional set of rules defined by the ISO specifications. These rules specify both required objects and features not supported for graphics exchange. The PDF/X standard follows strict rules in color management. BI Publisher supports the PDF/X-1a:2003 variation of the PDF/X standard.

2.9.2.1 Prerequisites

The generation of PDF/X output requires that you obtain the International Color Consortium (ICC) profile data file and place it under `<bi_publisher_repository>/Admin/Configuration`.

The ICC profile is a binary file describing the color characteristics of the intended output device. For production environments, the color profile may be provided by your print vendor or by the printing company that prints the generated PDF/X file. An example of an ICC profile data file is: CoatedFOGRA27.icc.

Profile data is also available from Adobe (<http://www.adobe.com/support>) or colormangement.org (<http://www.colormangement.org/>).

2.9.2.2 Requirements and Limitations

The following lists limitations of and requirements for generating PDF/X output:

- **Supported template types:** The following template types support the generation of PDF/X: RTF, FO, and XSL. There are no additional template requirements to generate PDF/X.
- **Color requirements:** The color data in the template (text color, images, and SVG) is stored as RGB data, but at the time the PDF/X file is generated, the color data is converted to CMYK using an ICC profile that you must provide to BI Publisher. Specify the ICC profile using the PDF/X ICC Profile Data property. See [Table 2–12](#).
- **PDF features not supported in PDF/X documents:**
 - Transparency (transparent colors render as opaque)
 - Encryption
- **Font requirements:** By default, all fonts are replaced with Albany fonts. To use a different font in the output, specify the font mappings in the report configuration. If Albany fonts are not available in the JVM font directory and you have not specified a font mapping (that is, there is no embeddable font available), the output will not become a valid PDF/X file. In this case, Helvetica font is used.
- **Reprocessing by BI Publisher utilities is not supported:** Reprocessing of the PDF/X file using the BI Publisher utilities PDFBookBinder, PDFDocMerger, or PDFSignature is not supported. The reprocessed file becomes a regular PDF file and may lose conformance to the PDF/X standard.
- **Required report configuration properties:** The report run-time properties must be set as shown in [Table 2–11](#).

Table 2–12 Required PDF Report Property Settings for PDF/X

Property	Required Setting
pdf-version	Must be no higher than 1.4.
Enable PDF Security (pdf-security)	Must be set to false.
Encryption Level (pdf-encryption-level)	Must be set to 0.
pdf-font-embedding	Must be set to true.

Formatting properties specific to PDF/X output can be set in the **Report Properties** dialog. Of the formatting properties, the following two are required:

Table 2–13 Required PDF/X Formatting Property Settings

Property	Description	Valid Values	Default Value
PDF/X ICC Profile Data	The name of the ICC profile data file placed under <BI Publisher repository>/Admin/Configuration.	ICC profile data file name, for example: CoatedFOGRA27.icc	None
PDF/X output condition identifier	The name of one of the standard printing conditions registered with ICC. The list of standard CMYK printing conditions to use with PDF/X-1a is provided on the following ICC website: http://www.color.org/chartata/drsection1.xalter .	A valid "Reference name," for example: FOGRA43	None

For more information, see [Section 10–5, "PDF/X Output Properties."](#)

2.9.2.3 Additional Resources

For more information about the PDF/X standard, see the following resources:

- Application Notes for PDF/X Standards
- ISO 15930-4:2003 Graphic technology -- Prepress digital data exchange using PDF -- Part 4: Complete exchange of CMYK and spot color printing data using PDF 1.4 (PDF/X-1a)

Creating BI Publisher Layout Templates

This chapter describes creating BI Publisher (.xpt) layout templates using the layout editor.

This chapter includes the following topics:

- [Section 3.1, "Overview of BI Publisher Layouts"](#)
- [Section 3.2, "Launching the Layout Editor"](#)
- [Section 3.3, "About the Layout Editor Interface"](#)
- [Section 3.4, "Page Layout Tab"](#)
- [Section 3.5, "Inserting Layout Components"](#)
- [Section 3.6, "About Layout Grids"](#)
- [Section 3.7, "About Repeating Sections"](#)
- [Section 3.8, "About Data Tables"](#)
- [Section 3.9, "About Charts"](#)
- [Section 3.10, "About Gauge Charts"](#)
- [Section 3.11, "About Pivot Tables"](#)
- [Section 3.12, "About Text Items"](#)
- [Section 3.13, "About Images"](#)
- [Section 3.14, "About Lists"](#)
- [Section 3.15, "Setting Predefined or Custom Formulas"](#)
- [Section 3.16, "Saving a Layout"](#)

3.1 Overview of BI Publisher Layouts

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 Figure 3-1.

Figure 3-1 Example of Interactive Output



Notice the following features:

- Pop-up chart details - Hover 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.

3.1.1 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).

3.1.2 Prerequisites, Recommendations, and Limitations

- 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 the data model before you create a new layout. For information on adding sample data to the data model, see the "Testing Data Models and Generating Sample Data" section in *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.
- The layout editor does not support namespaces or attributes in the XML data.

3.2 Launching the Layout Editor

Launch the layout editor in one of the following ways:

- [Section 3.2.1, "When Creating a New Report"](#)
- [Section 3.2.2, "When Editing a Report"](#)
- [Section 3.2.3, "When Viewing a Report"](#)

3.2.1 When Creating a New Report

To launch the Layout Editor when creating a new report:

1. Selecting the data model for the new report.
The Report Editor displays the Add Layout page.
2. From the Create Layout region, click a predefined template to launch the Layout Editor.

3.2.2 When Editing a Report

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.

- From the **Create Layout** region, click a predefined template to use to launch the Layout Editor.

3.2.3 When Viewing a Report

To launch the Layout Editor when viewing a report:

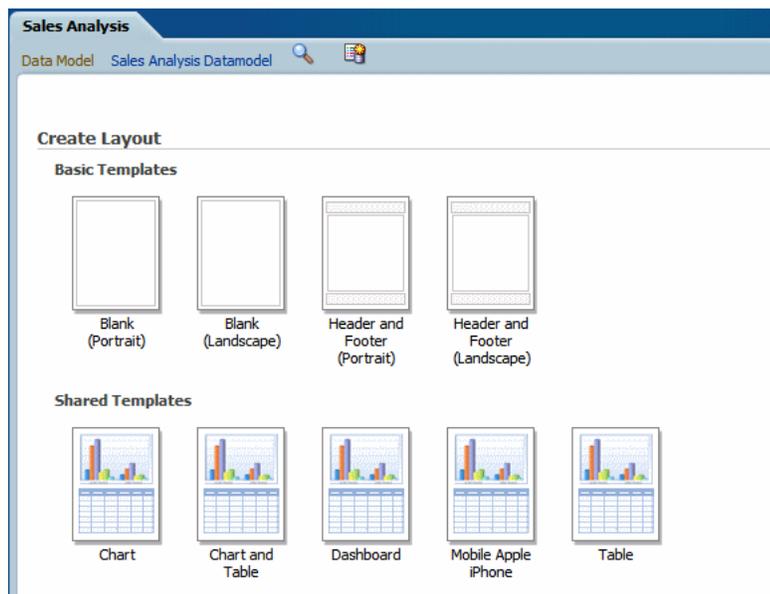
- Click **Actions** and then click **Edit Layout**.

The layout must have been created in the layout editor.

3.2.4 Selecting a Predefined Layout

When you create a new layout, you are given the option of selecting a predefined layout to help you get started. [Figure 3–2](#) shows the predefined layouts offered by the Basic and Shared Templates.

Figure 3–2 Predefined Layouts



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, then you can add predefined layouts for your own use, or your Administrator can add more for all users.

3.2.4.1 Adding Shared Templates for All Users

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

- Log in with Administrator privileges and navigate to the Catalog.
- In the **Shared Folders** directory, open the **Components** folder.
- Locate the **Boilerplates** report and click **Edit**.
- Click **Add New Layout**.
- 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 the boilerplate a name. This layout is now displayed 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 are displayed to all users as a Shared Template.

3.2.4.2 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 does 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 the boilerplate a name.

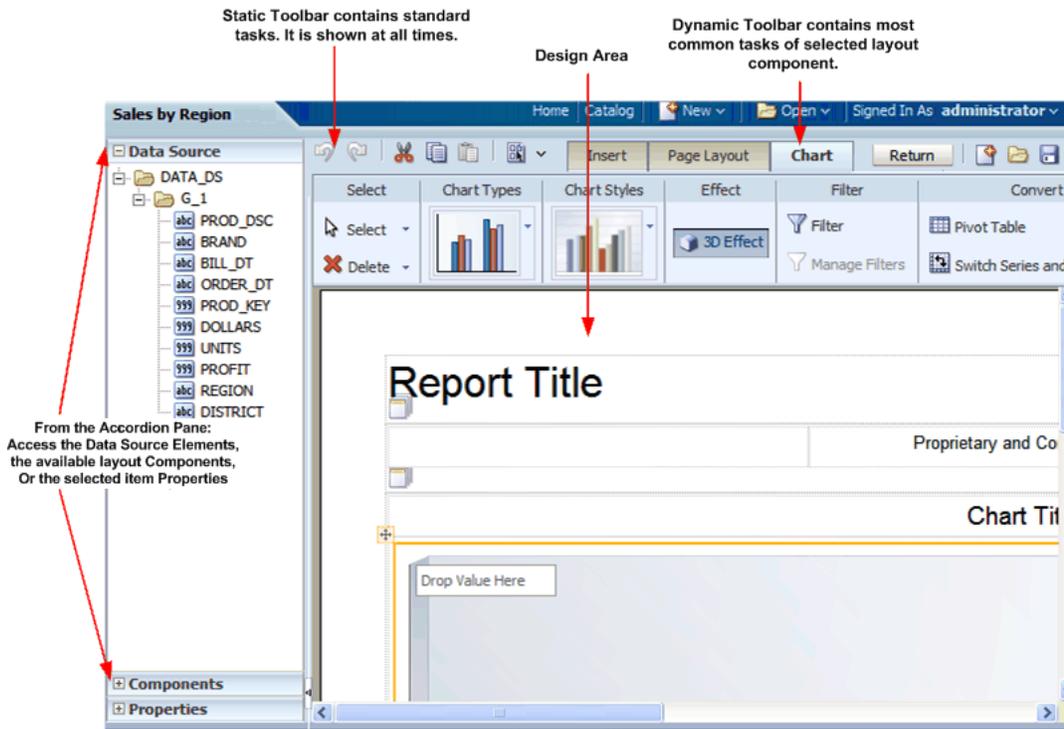
To upload a layout: Click **Upload** to upload a predefined BI Publisher Template (.xpt file).

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

3.3 About the Layout Editor Interface

Figure 3–3 shows the Layout Editor.

Figure 3–3 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 [Section 3.3.4, "About the Static Toolbar."](#)
 - 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 the design area. See [Section 3.3.5, "About the Tabbed Toolbar."](#)
- 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 the layout.

3.3.1 About the Data Source Pane

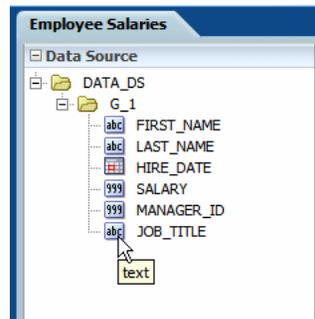
The Data Source pane displays the structure of the data model and the data elements that are available to insert into the layout.

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.

Figure 3–4 shows the data source pane. The icon beside each element indicates the data type.

Figure 3–4 The Data Source Pane



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.

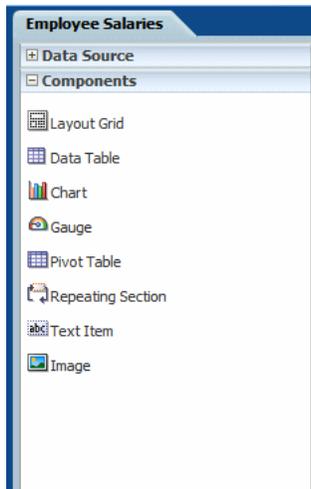
3.3.2 About the Components Pane

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 the layout.

Figure 3–5 shows the Components pane.

Figure 3–5 The Components Pane



3.3.3 About the Properties Pane

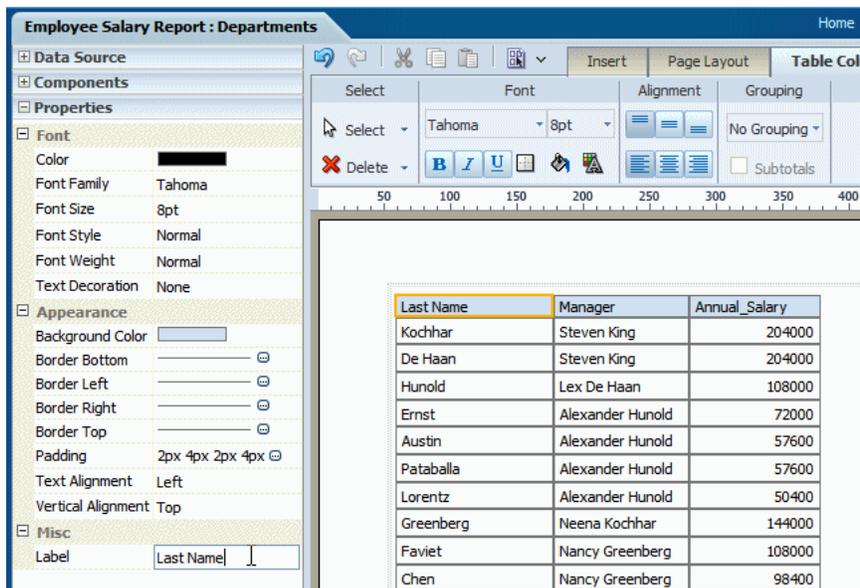
The Properties pane displays the properties for the selected component. The properties displayed are determined by the selected component. Some of the properties available in the Properties pane are also editable in the dynamic tab for the component.

Click a property value to edit it. The change is applied to the component when you move the 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. If a property field is blank, then the default is used.

Figure 3–6 shows a sample Properties pane for a table column header.

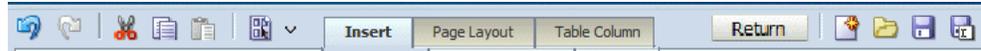
Figure 3–6 Sample Properties Pane for a Table Column Header



3.3.4 About the Static Toolbar

The Static toolbar extends on either side of the tabbed toolbar and is shown in [Figure 3-7](#).

Figure 3-7 The Static Toolbar



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.

3.3.5 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 [Section 3.5, "Inserting Layout Components."](#)
- The **Page Layout** tab provides common page-level tools and commands. See [Section 3.4, "Page Layout Tab."](#)
- 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.

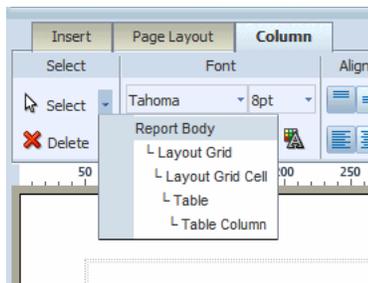
To set or control more properties for the selected component, open the Properties pane in the accordion pane, as described in [Section 3.3.3, "About the Properties Pane"](#)

3.3.5.1 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.

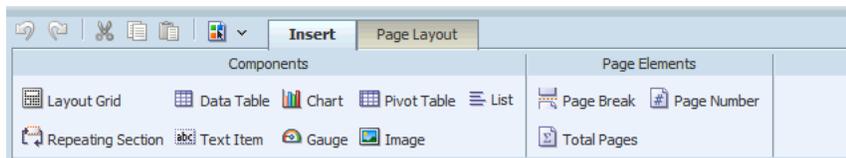
[Figure 3-8](#) shows the **Select** tool.

Figure 3–8 The Select Tool

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

3.3.5.2 About the Insert Tab

Use the **Insert** tab to insert report components and page elements. [Figure 3–9](#) shows the **Insert** tab.

Figure 3–9 The Insert Tab

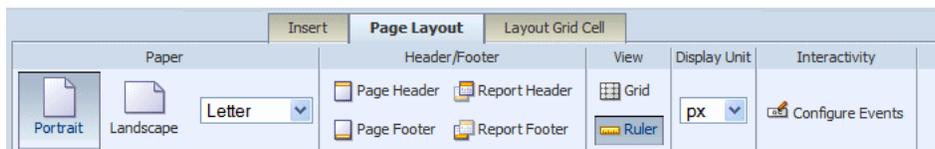
The **Components** group displays the report components that you can insert into the layout. To insert a component, select and drag the item to the desired location in the design area. For more information about each component, see its corresponding section in this chapter.

The **Page Elements** group contains page-level elements for the 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.

3.4 Page Layout Tab

[Figure 3–10](#) shows the Page Layout tab.

Figure 3–10 The Page Layout Tab

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

3.4.1 Paper Options

Table 3–1 describes paper options.

Table 3–1 Paper Options

Option	Description
Orientation	Choose Portrait or Landscape .
Paper Size	Select from the following paper size options: Letter, Legal, A4, A3, Executive, B5, Com-10, Monarch DL, or C5. The paper size determines the dimensions of the layout area.

3.4.2 Header/Footer Options

Table 3–2 describes header and footer options.

Table 3–2 Header and Footer Options

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

3.4.2.1 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.

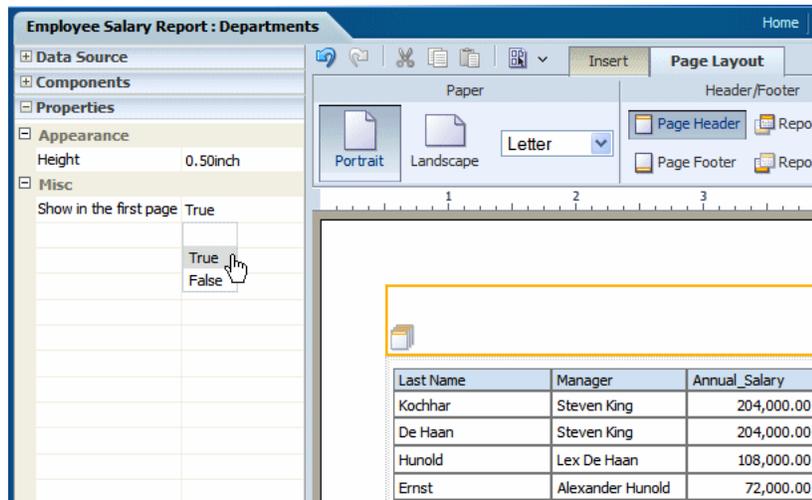
For all report and page headers and footers:

- **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.

Figure 3–11 shows the **Properties** for a report header.

Figure 3–11 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.

3.4.3 View Options

Table 3–3 describes view options.

Table 3–3 View Options

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

3.4.4 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).

3.4.5 Interactivity: Event Configuration

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

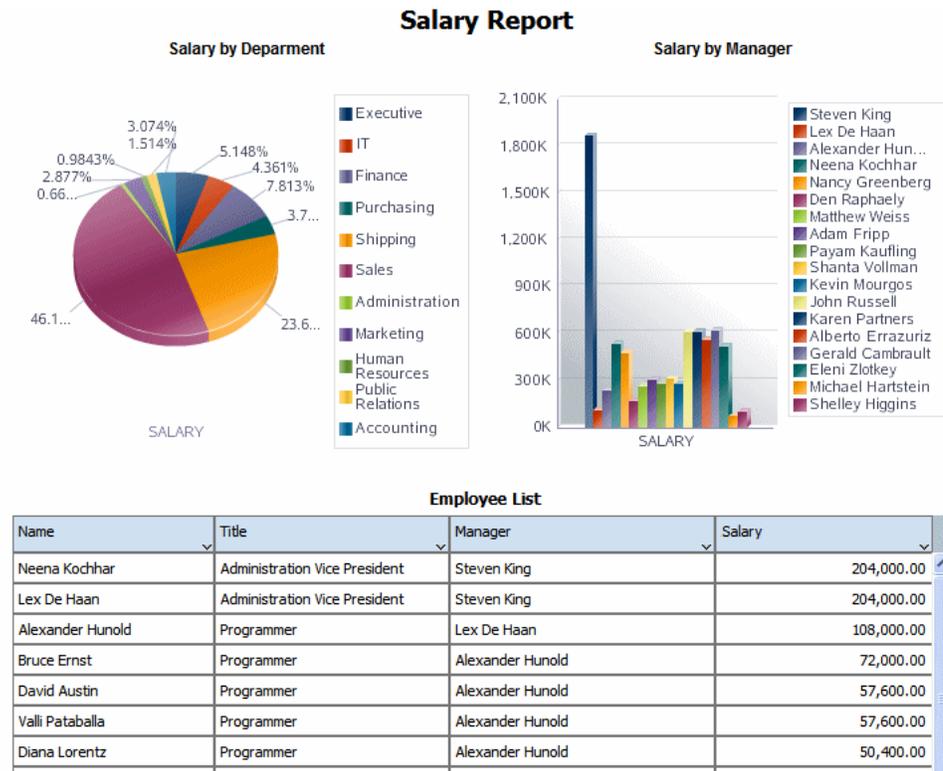
The two types of events are:

- **Filter** - If you click an element in a list, chart, or pivot table, that element is 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) shows the results for the selected element only. This action does not affect other components of the report.

3.4.5.1 Example of Filter Event Configuration

Figure 3–12 shows an example of filter event configuration. 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.

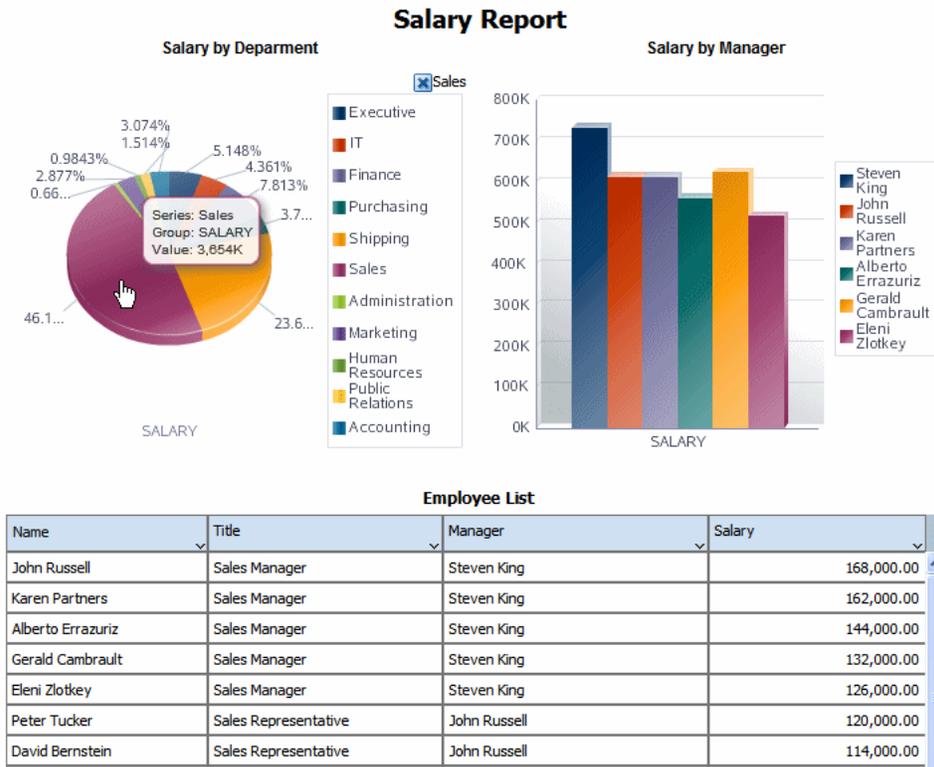
Figure 3–12 Example of Filter Event Configuration



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.

Figure 3–13 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.

Figure 3–13 Example of Automatic Filtering



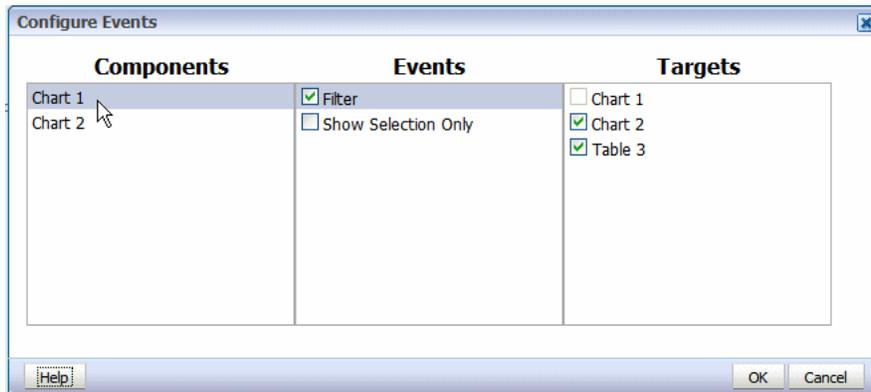
3.4.5.2 Configuring Automatic Filtering

To configure automatic filtering:

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

Figure 3–14 shows the **Configure Events** dialog.

Figure 3–14 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.

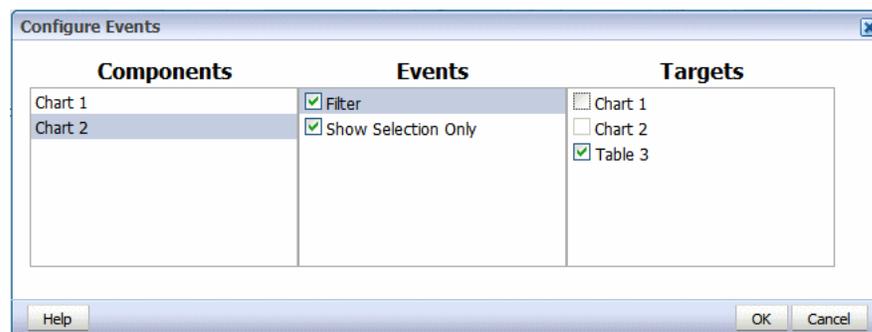
The **Show Selection Only** option is not enabled for Chart 1. That means that Chart 1 continues to display all values when one its elements is selected.

3.4.5.3 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).

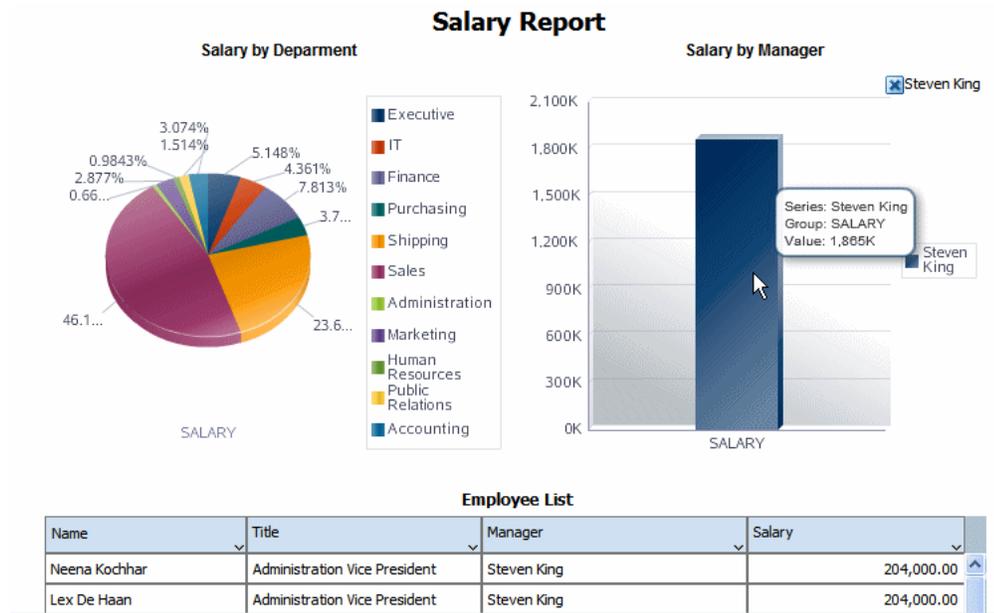
In the example in [Figure 3–15](#), Chart 2 is configured with **Show Selection Only** enabled and **Filter** enabled with Table 3 as the Target.

Figure 3–15 Example of Show Selection Only



This configuration results in the output shown in [Figure 3–16](#). When the user clicks on Chart 2, only the selected value is shown in Chart 2. Because the Filter event is enabled for Table 3, the selection is applied as a filter to Table 3.

Figure 3–16 Example of Output from the Show Selection Only Configuration

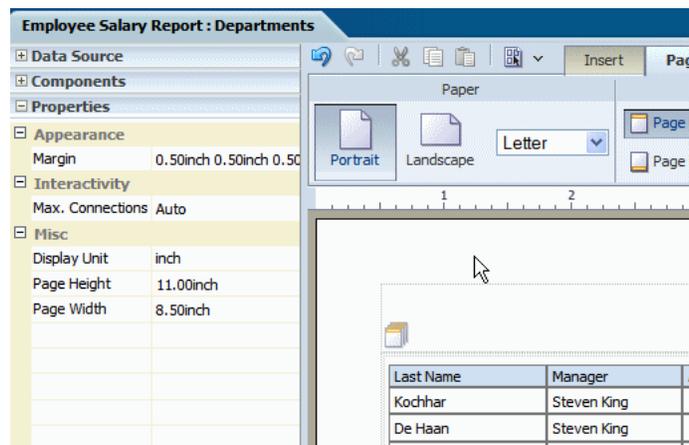


3.4.6 Setting Page Margins

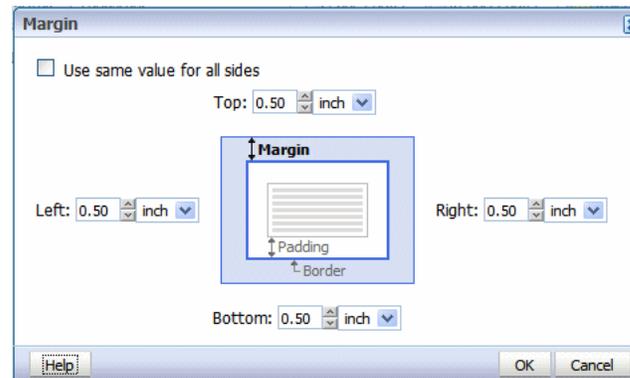
To set the page margins for the report:

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. [Figure 3–17](#) shows the Properties for the page.

Figure 3–17 The Properties for the Page



3. Click the value shown for **Margin** to launch the **Margin** dialog. [Figure 3–18](#) shows the **Margin** dialog.

Figure 3–18 The Margin Dialog

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

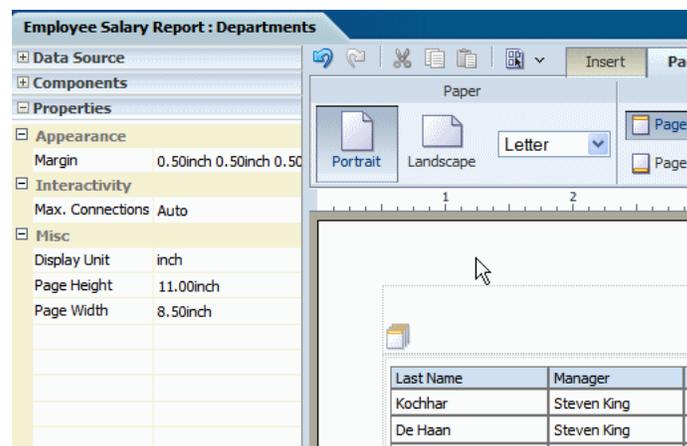
To automatically set the same value for all sides, select the box: **Use same value for all sides**. This action disables all but the Top margin entry. Enter the value in the Top to apply to all sides.

3.4.7 Setting Maximum Connections for an Interactive Report

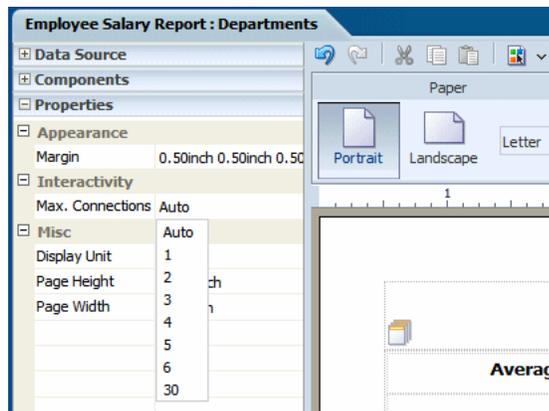
You can limit the connections from the browser to the server for the interactive viewer. More connections are faster but increase server load. The default is six connections. Reduce the number to reduce the load on the server for large reports.

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. [Figure 3–19](#) shows the Properties for the page.

Figure 3–19 The Properties for the Page

3. Click the value shown for **Max. Connections** and select the desired value from the list, as shown in [Figure 3–20](#).

Figure 3–20 Example of Max. Connections

3.5 Inserting Layout Components

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

- [Section 3.6, "About Layout Grids"](#)
- [Section 3.7, "About Repeating Sections"](#)
- [Section 3.8, "About Data Tables"](#)
- [Section 3.9, "About Charts"](#)
- [Section 3.10, "About Gauge Charts"](#)
- [Section 3.11, "About Pivot Tables"](#)
- [Section 3.12, "About Text Items"](#)
- [Section 3.13, "About Images"](#)
- [Section 3.14, "About Lists"](#)

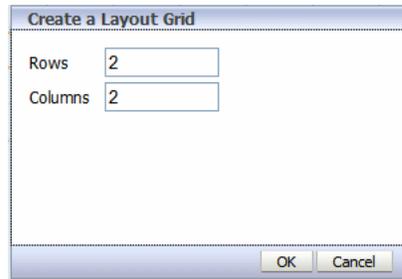
3.6 About Layout Grids

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.

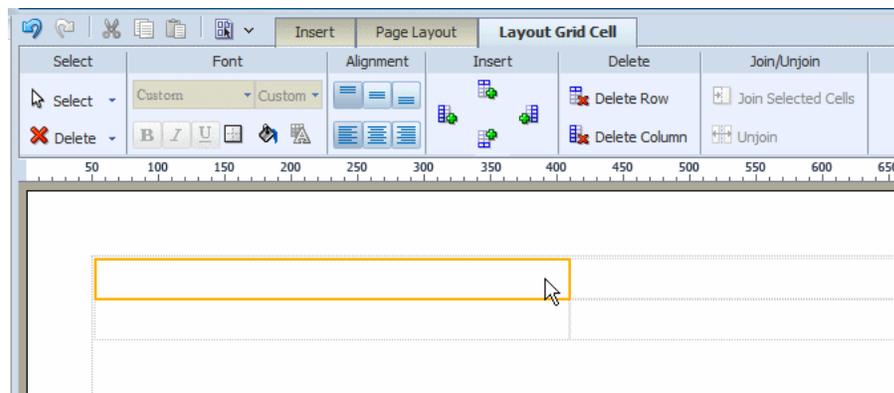
To create a layout grid:

1. Select and drag the Layout Grid component to the design area.

[Figure 3–21](#) shows the Create a Layout Grid dialog.

Figure 3–21 Create a Layout Grid Dialog

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 [Figure 3–22](#).

Figure 3–22 Example of a Grid Inserted in the Design Area

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.

3.6.1 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 want to display the gridlines in the finished report, then select the grid cell and click the **Set Border** command button to launch the **Border** dialog.

To add a background color to a cell:

1. Click the Background Color command button to launch the Color Picker.

3.6.2 About the Insert Options

When 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

3.6.3 About the Join and Unjoin Options

To join cells horizontally or vertically:

1. Select multiple adjacent cells by holding down the Ctrl key and clicking each grid cell.
2. Click the **Join** command button.

To unjoin cells that have been joined:

1. Select the joined cell and click the **Unjoin** button.

3.6.4 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 the 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. [Figure 3–23](#) shows this option on the Properties pane.

Figure 3–23 The Interactive: Expand/Collapse Property

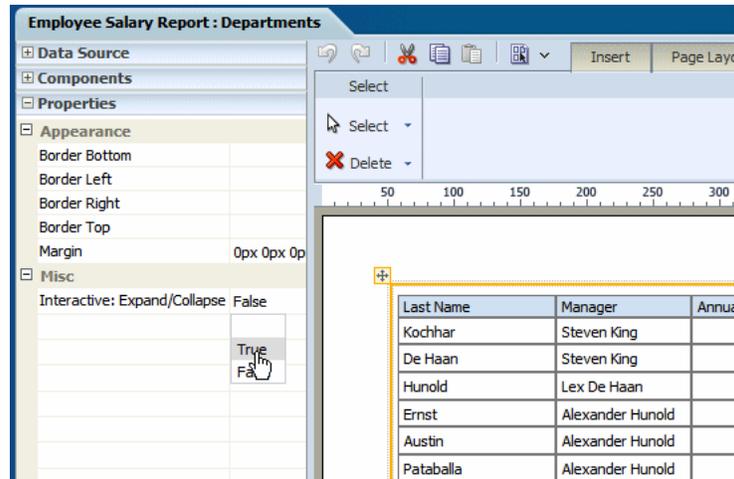
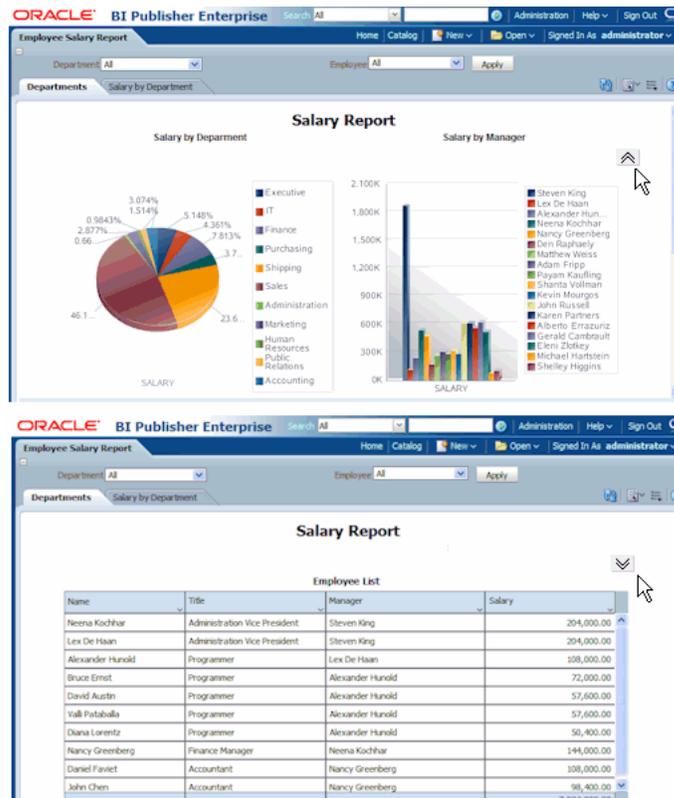


Figure 3–24 demonstrates the expand and collapse behavior when the report is viewed in interactive mode. The top of the figure shows the collapse icon in the upper right area of the report. Click the icon to collapse the grid. The bottom of the figure shows the report with the region collapsed.

Figure 3–24 Example of the Expand and Collapse Behavior



3.7 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

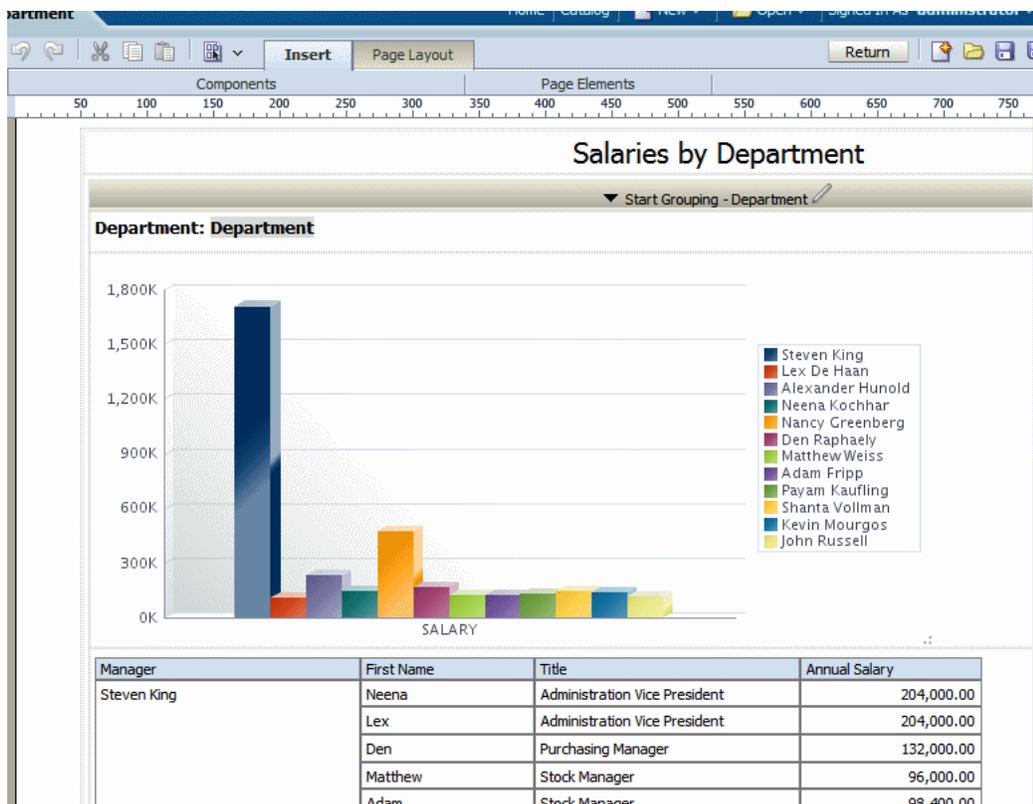
banded reports, as well as repeating pages or sections for different data elements (such as Group Above/Outline).

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 the 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.

Figure 3–25 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 are repeated.

Figure 3–25 An Example of a Layout that has a Repeating Section Defined for the Element Department



3.7.1 Setting Page Break Options for a Repeating Section

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

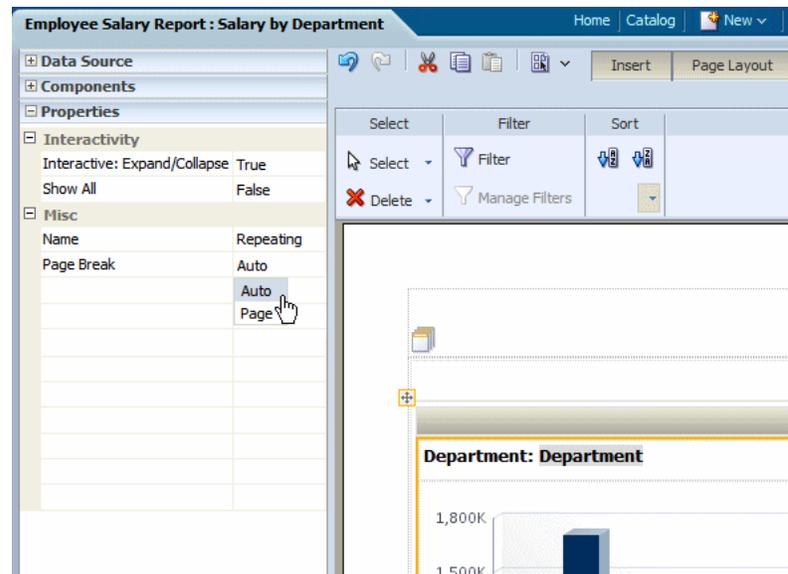
Using the preceding example, it is desirable for the PDF output of this report to break after each department.

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**.

Figure 3–26 shows the Properties for a repeating section.

Figure 3–26 The Properties for a Repeating Section

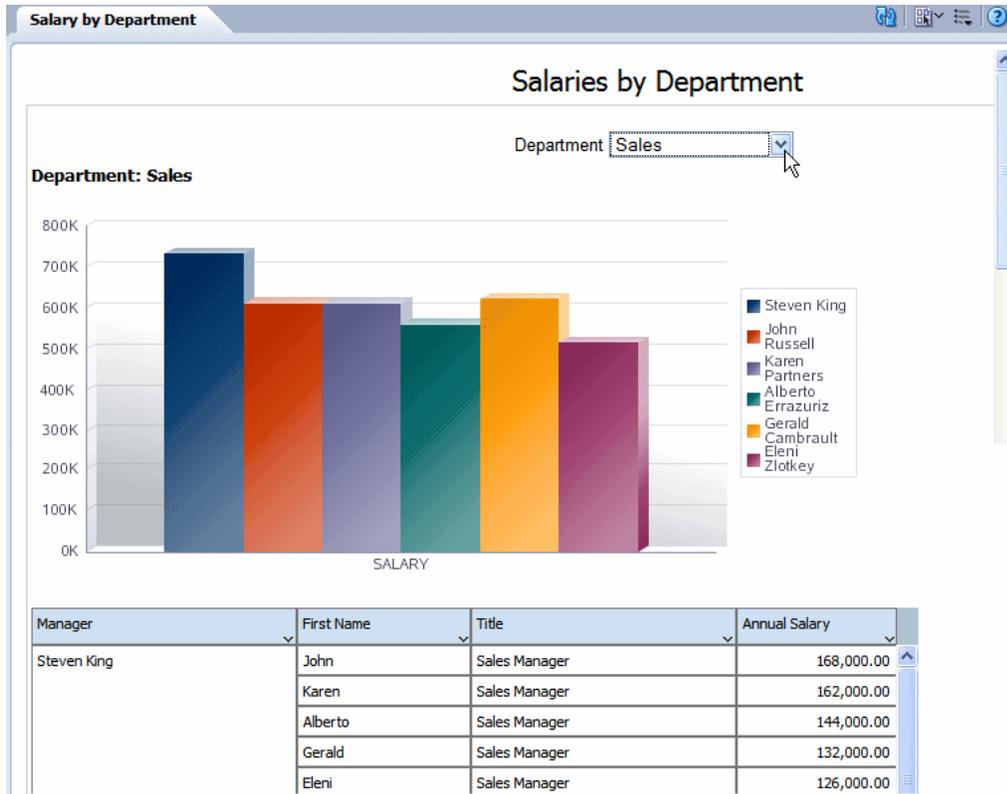


3.7.2 How Repeating Sections Display in Interactive Mode

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.

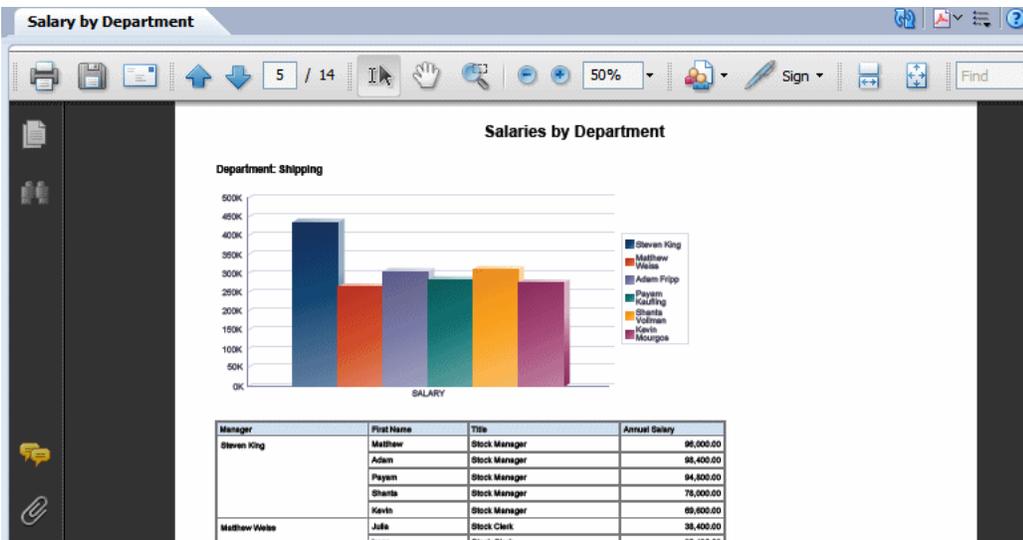
Figure 3–27 shows the repeat by element Department displayed in a list of values.

Figure 3–27 The Repeat by Element Department Displayed in a List of Values



By contrast, [Figure 3–28](#) shows 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.

Figure 3–28 Repeat by Element Department Layout Displayed in PDF



3.7.3 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.

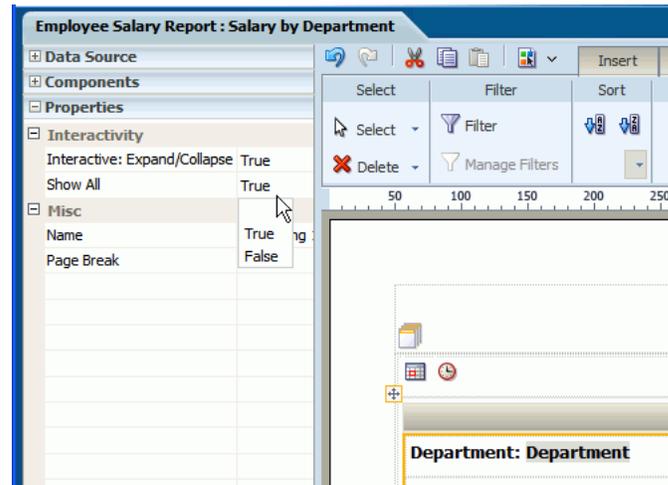
To enable a report consumer to view the results in the repeating section for all values of the element, the Repeating Section component provides the property: Show All. When this property is set to true, the value "All" is added to the list to enable the display of results for all values.

To enable Show All:

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

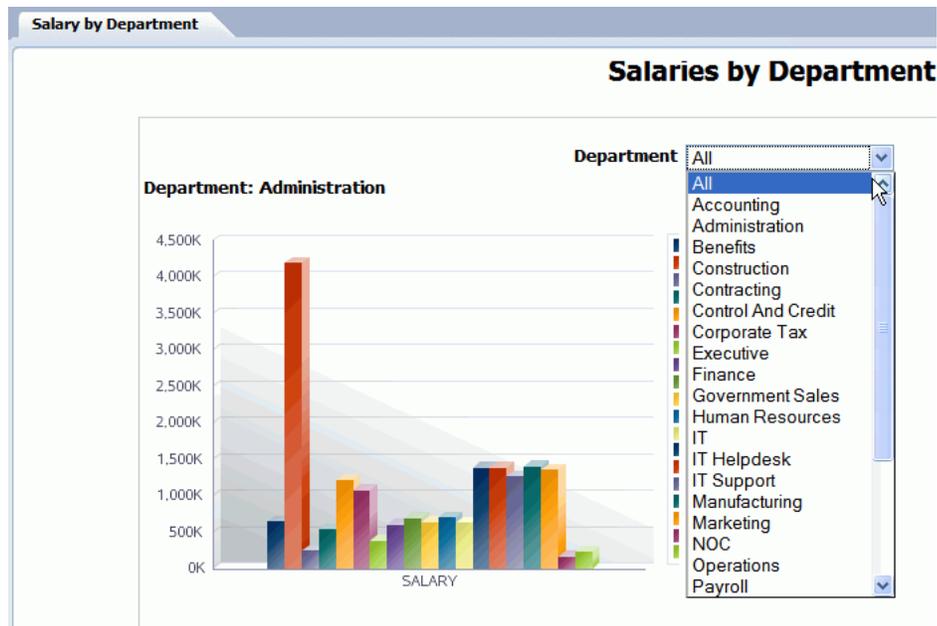
Figure 3–29 shows the **Show All** setting in the **Properties** pane.

Figure 3–29 The Show All Setting in the Properties Pane



When you view the report, the option All is added to the menu of values, as shown in Figure 3–30.

Figure 3–30 The All Option Added to the Menu of Values



3.8 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

This section contains the following topics about working with tables:

- [Section 3.8.1, "Inserting a Data Table"](#)
- [Section 3.8.2, "Setting Alternating Row Colors"](#)
- [Section 3.8.3, "About the Table Tab"](#)
- [Section 3.8.4, "About the Table Column Header Tab"](#)
- [Section 3.8.5, "About the Column Tab"](#)
- [Section 3.8.6, "About the Total Cell Tab"](#)
- [Section 3.8.7, "Inserting Dynamic Hyperlinks"](#)

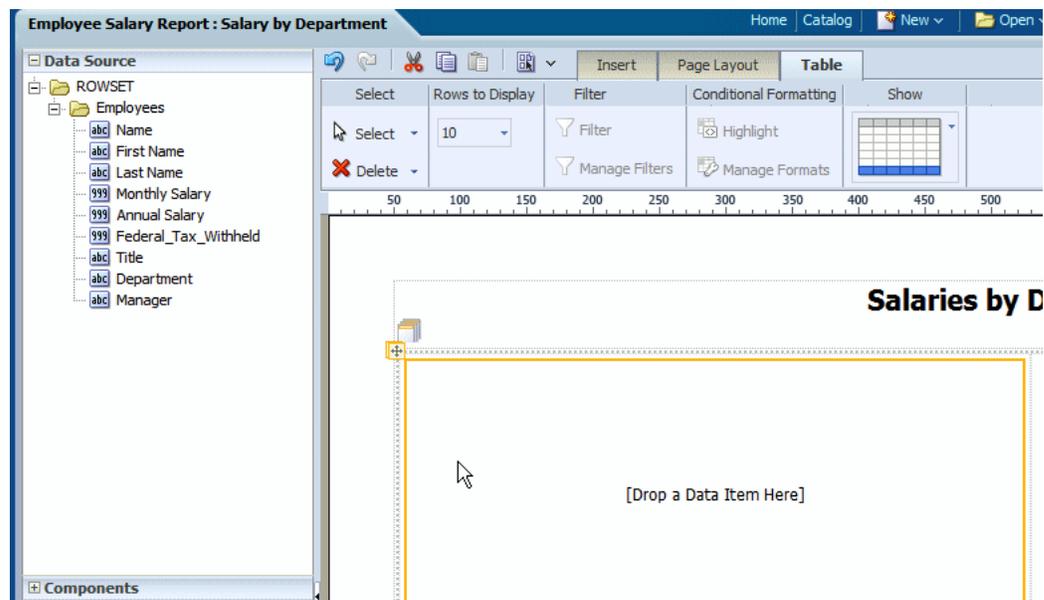
3.8.1 Inserting a Data Table

To insert a data table:

1. From the **Insert** tab, select and drag the **Data Table** component to the design area.

Figure 3–31 shows an inserted, empty data table. Notice that the **Table** tab is now displayed.

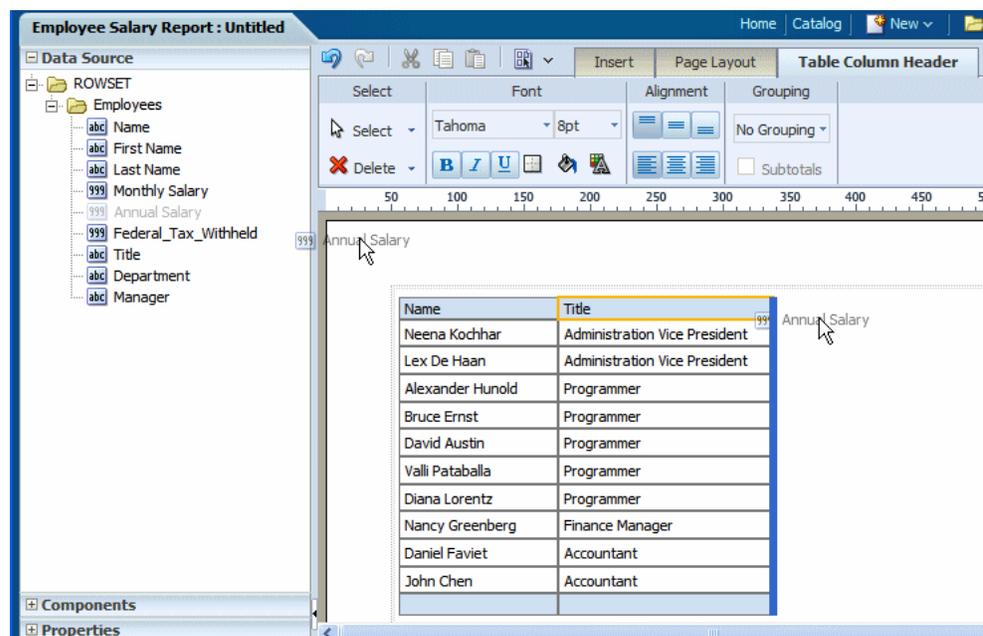
Figure 3–31 Example of an Inserted, Empty Data Table



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

Figure 3–32 shows the columns being added to the table. Notice that when you drop a column on the table the sample data is immediately displayed.

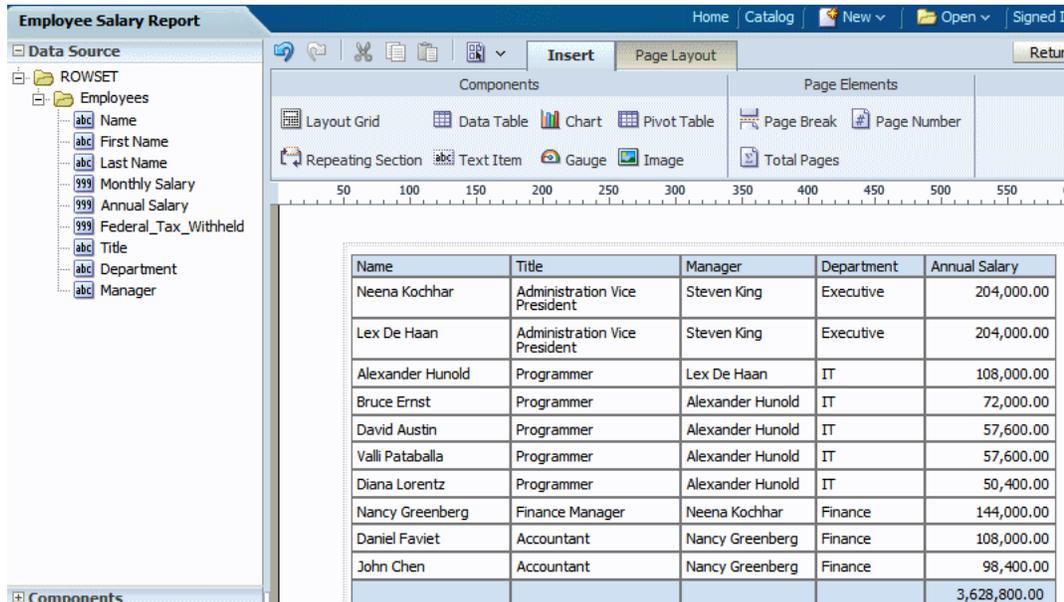
Figure 3–32 Example of Columns Added to a Table



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

Figure 3–33 shows a completed data table.

Figure 3–33 Example of 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](#).
- Default date formatting is applied. To change the default formatting, see [About the Column Tab](#).
- Default number formatting and alignment is applied. To change the default formatting, see [Section 3.8.5, "About the Column Tab."](#)

3.8.2 Setting Alternating Row Colors

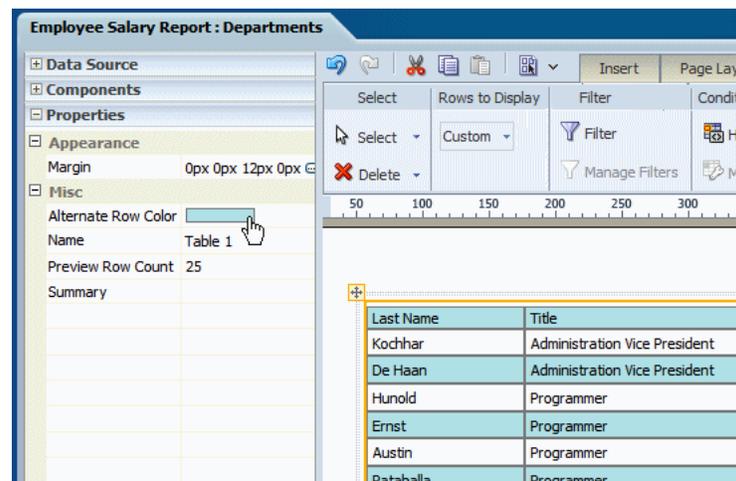
Some data tables are easier to read when the rows display alternating colors, as shown in [Figure 3–34](#).

Figure 3–34 Example of Rows Displayed in Alternating Colors

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
Ernst	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
Scarm	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. [Figure 3–35](#) shows the **Alternate Row Color** option.

Figure 3–35 The Alternate Row Color Option

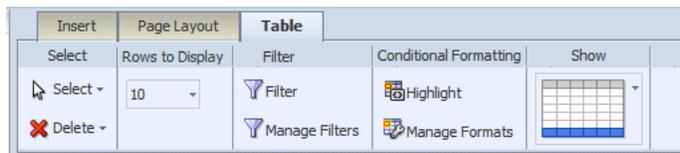
4. Choose a color and click **OK**.

3.8.3 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

[Figure 3–36](#) shows the Table tab.

Figure 3–36 The Table Tab

3.8.3.1 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 are 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 can impact the performance of the Layout Editor.

3.8.3.2 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 the table without having to perform additional coding. For example, you could add a filter to meet some of the following report conditions:

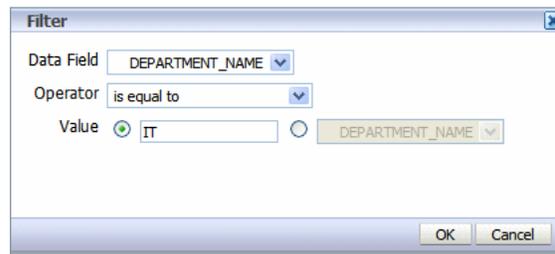
- Display only the top 10 salaries
- Display only the bottom 25 store sales
- Display only employees in the IT department
- Display only sales that are between \$10,000 and \$20,000 and in the Southern region

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

3.8.3.3 Setting Filters for a Table

To set a filter:

1. Click the **Filter** toolbar button. This launches the **Filter** dialog, as shown in [Figure 3–37](#).

Figure 3–37 The Filter Dialog

2. Enter the fields to define a filter, as described in [Table 3–4](#).

Table 3–4 Fields to Define Filters

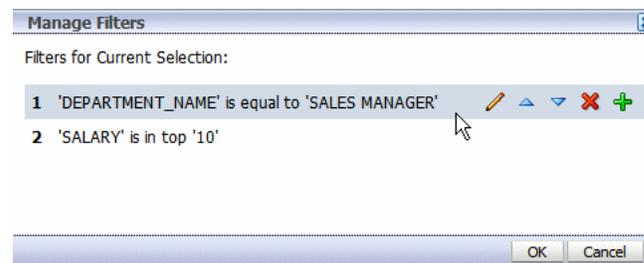
Field	Description
Data Field	Select the data field to filter the table data by. All elements are available regardless of whether they are included as table columns.
Operator	Select from the following operators: is equal to is not equal to is less than is greater than is less than or equal to is greater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

3.8.3.4 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, as shown in [Figure 3–38](#).

Figure 3–38 The Manage Filters Dialog

2. Hover the 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.

3.8.3.5 About Conditional Formats

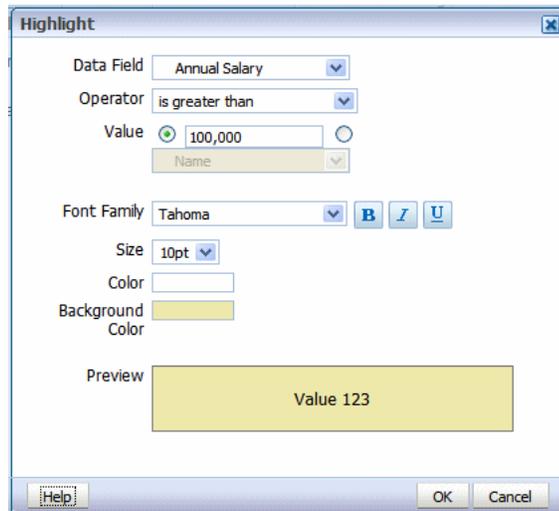
A conditional format changes the formatting of an element in the table based on a condition. This feature is extremely useful for highlighting target ranges of values in the table. For example, you could create a set of conditional formats for the table that display rows in different colors depending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

3.8.3.6 Applying Conditional Formats to a Table

To apply a conditional format:

1. Click the **Highlight** button. This launches the **Highlight** dialog, as shown in [Figure 3–39](#).

Figure 3–39 The Highlight Dialog



2. Enter the fields to define a condition and format to apply, as described in [Table 3–5](#).

Table 3–5 Fields to Define Conditions and Formats

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

[Figure 3–40](#) shows the table in the layout with the condition applied.

Figure 3–40 Example of Conditional Formatting

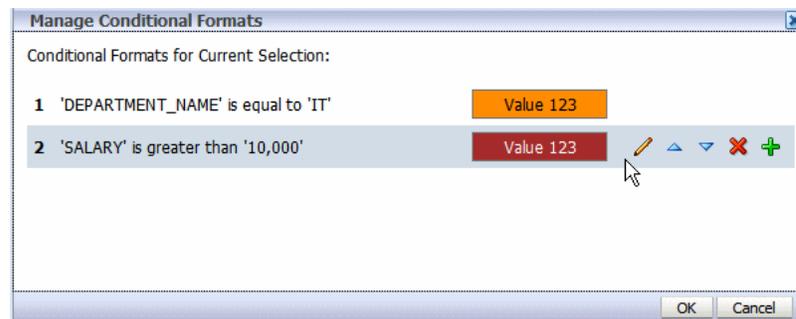
Name	Title	Manager	Department	Annual Salary
Neena Kochhar	Administration Vice President	Steven King	Executive	204,000.00
Lex De Haan	Administration Vice President	Steven King	Executive	204,000.00
Alexander Hunold	Programmer	Lex De Haan	IT	108,000.00
Bruce Ernst	Programmer	Alexander Hunold	IT	72,000.00
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

3.8.3.7 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, as shown in [Figure 3–41](#).

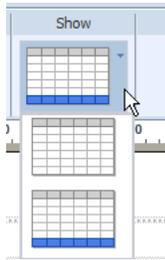
Figure 3–41 The Manage Conditional Formats Dialog

2. Hover the 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. The order of the conditions is important because only the first condition that is met is applied.

3.8.3.8 Controlling the Display of the Total Row

By default, the layout editor inserts a total row in a table that sums numeric columns. To remove the total row, click the **Show** menu and select the table view without the highlighted total row. [Figure 3–42](#) shows the **Show** menu options.

Figure 3–42 The Show Menu Options

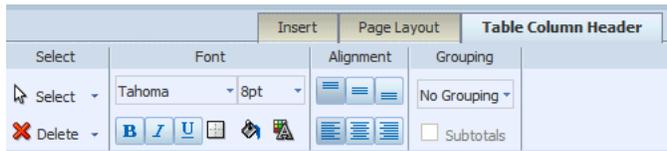


The total row can be further customized using the **Total Cell** tab and the **Properties** pane. For more information see [Section 3.8.6, "About the Total Cell Tab."](#)

3.8.4 About the Table Column Header Tab

[Figure 3–43](#) shows the **Table Column Header** tab.

Figure 3–43 The Table Column Header Tab



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

3.8.4.1 About Grouping

"Grouping" groups together elements in the data of the same value. In a table, applying grouping can make the 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. [Figure 3–44](#) shows a table that has been grouped by Manager using **Group Left**.

Figure 3–44 Example of a Table 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. [Figure 3–45](#) shows a table that has been grouped by Manager using **Group Above**.

Figure 3–45 Example of a Table 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

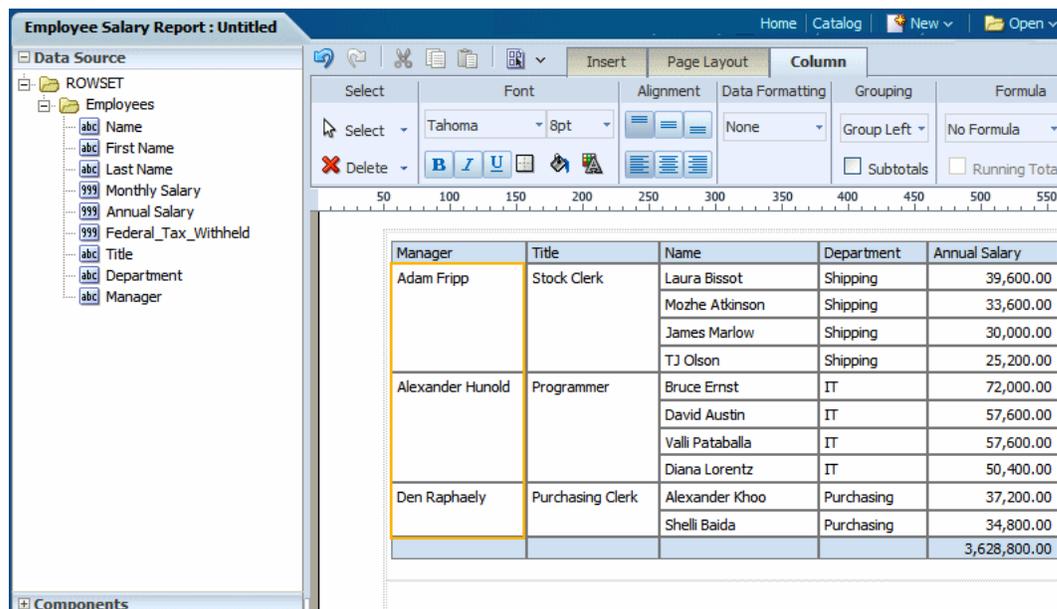
Manager Weiss			
Employee	Title	Hire Date	SALARY
Fleur	Shipping Clerk	Feb 22, 1998	3,100.00
Geoni	Shipping Clerk	Feb 02, 2000	2,800.00
Landry	Stock Clerk	Jan 13, 1999	2,400.00
Markle	Stock Clerk	Mar 07, 2000	2,200.00
Mikkilineni	Stock Clerk	Sep 27, 1998	2,700.00
Nayer	Stock Clerk	Jul 15, 1997	3,200.00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

3.8.4.2 Example: Group Left

In [Figure 3–46](#), 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.

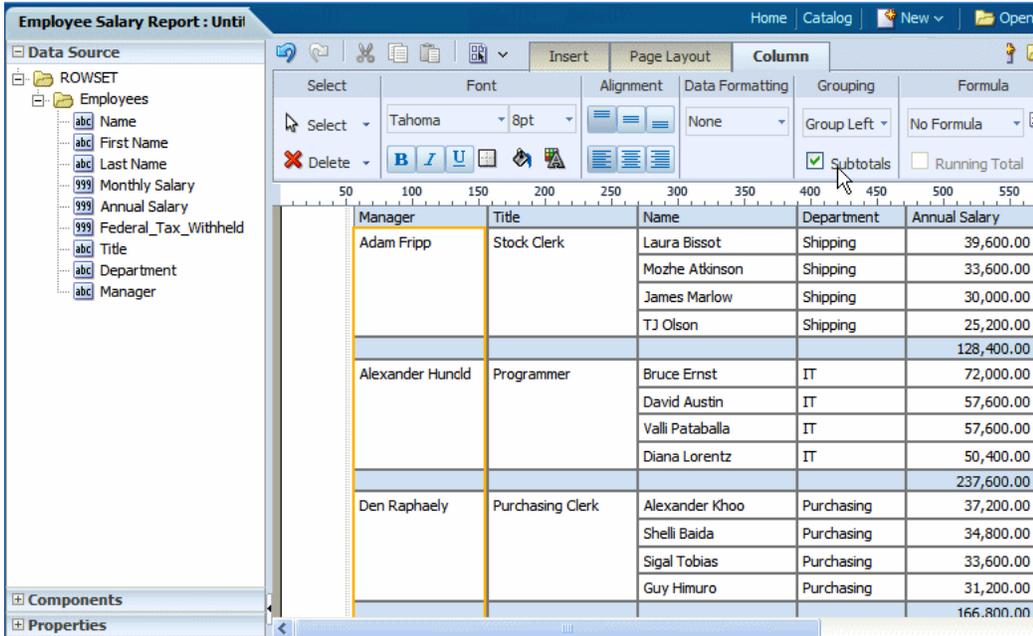
Figure 3–46 Group Left Example



3.8.4.3 Applying Subtotals

To further enhance a table, you can add a subtotal row to display for each grouped occurrence of the element. [Figure 3–47](#) shows the same table with the **Subtotals** box checked. Notice that for each manager a subtotal row has been inserted.

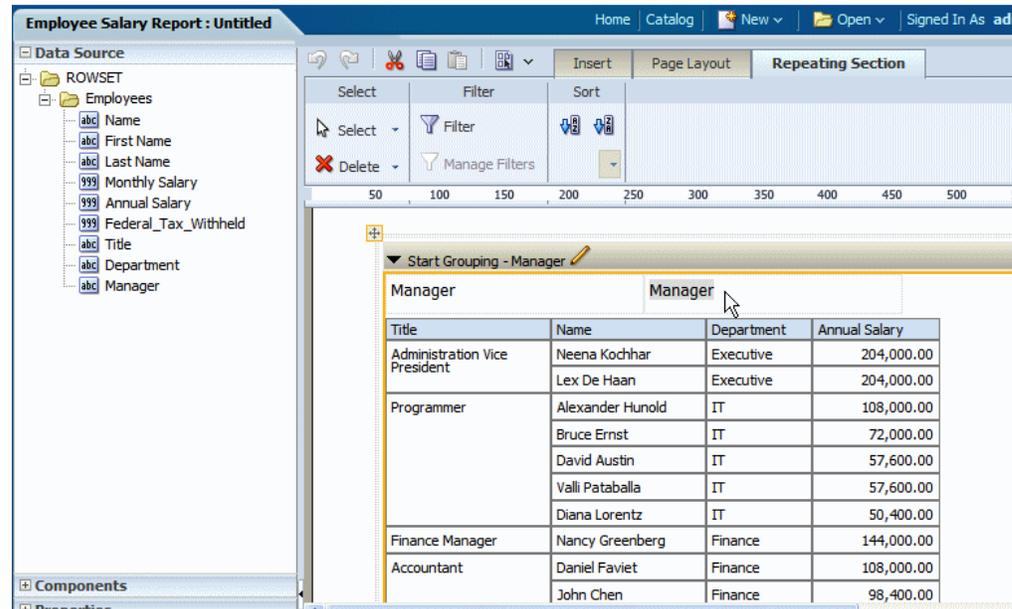
Figure 3–47 Example of Subtotals



3.8.4.4 Example: Group Above

In [Figure 3–48](#), 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.

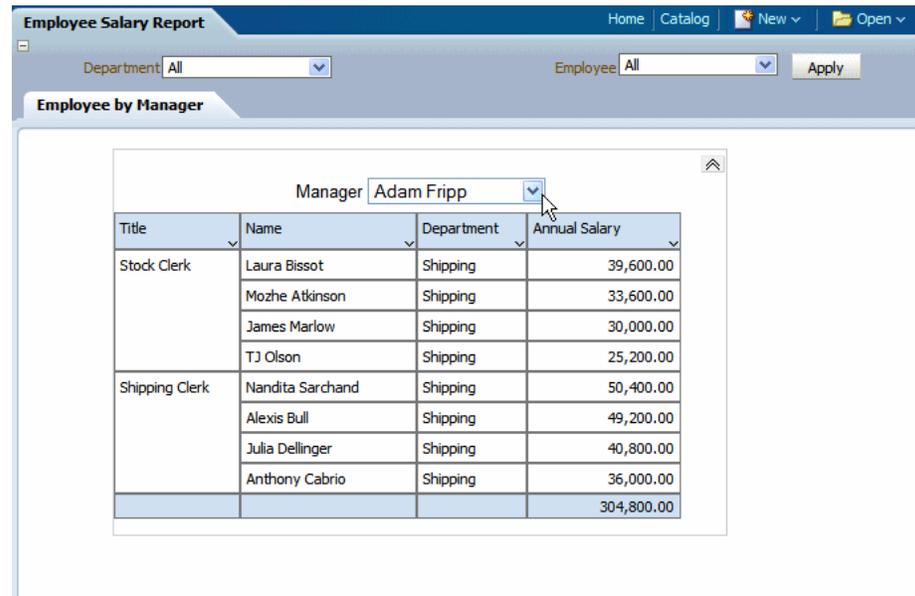
Figure 3–48 Group Above Example



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

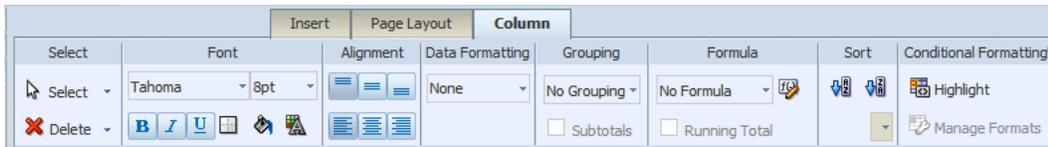
When you run the report, a separate table is created for each occurrence of the grouping element. In Interactive output mode, the grouping element displayed at the top of the table is displayed as a filter. Select the value that you want to view from the list, as shown in [Figure 3–49](#).

Figure 3–49 Grouping Element Displayed as a Filter



3.8.5 About the Column Tab

The Column tab is enabled when you select a specific column in a table. [Figure 3–50](#) shows the Column tab.

Figure 3–50 The Column Tab

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

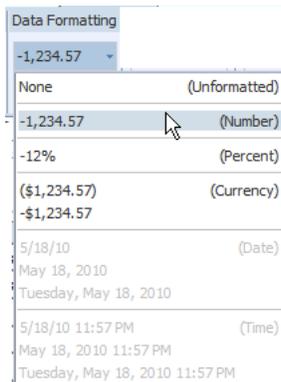
3.8.5.1 About the Data Formatting Options for Columns

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.

3.8.5.2 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 [Figure 3–51](#).

Figure 3–51 Number, Percent, and Currency Formats

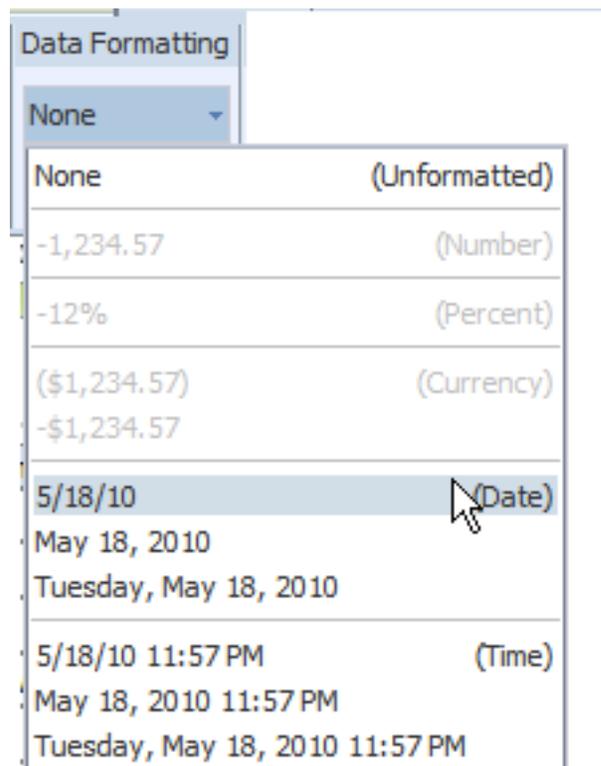
- **Decimal position** - Click the **Move Left** or **Move Right** to increase or decrease the decimal positions displayed.
- **Show/Hide Grouping Separator** - Click this button to hide the grouping separator (for example, 1,234.00 displays as 1234.00). To show the grouping separator, click the button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

3.8.5.3 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 [Figure 3–52](#).

Figure 3–52 Date and Time Formats



3.8.5.4 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 [Section 3.15, "Setting Predefined or Custom Formulas."](#)

3.8.5.5 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 employee salary table shown in [Figure 3–53](#), assume you want to sort ascending first by Title then sort descending by Annual Salary:

Figure 3–53 Employee Salary Table

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.

Figure 3–54 shows the **Priority** list.

Figure 3–54 Priority List



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 Figure 3–55.

Figure 3–55 Example of a Sorted Table

Name	Hire Date	Title	Annual Salary
Daniel Faviet	8/16/94	Accountant	\$108,000.00
John Chen	9/28/97	Accountant	\$98,400.00
Jose Manuel Urman	3/7/98	Accountant	\$93,600.00
Ismael Sciarra	9/30/97	Accountant	\$92,400.00
Luis Popp	12/7/99	Accountant	\$82,800.00
Jennifer Whalen	9/17/87	Administration Assistant	\$52,800.00
Neena Kochhar	9/21/89	Administration Vice President	\$204,000.00
Lex De Haan	1/13/93	Administration Vice President	\$204,000.00
Nancy Greenberg	8/17/94	Finance Manager	\$144,000.00
Michael Hartstein	2/17/96	Marketing Manager	\$156,000.00
			7,411,200.00

3.8.5.6 Removing a Sort Order

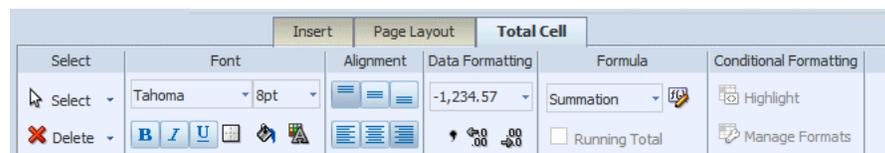
To remove a sort order applied to a column:

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.

3.8.6 About the Total Cell Tab

The Layout Editor automatically inserts a grand total row when you insert a data table to the layout. As shown in the section on grouping, you can also insert subtotal rows within the 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 [Figure 3–56](#).

Figure 3–56 The Total Cell Tab



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

3.8.6.1 Applying Data Formatting to a Total Cell

See [Section 3.8.5.1, "About the Data Formatting Options for Columns."](#)

3.8.6.2 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 [Section 3.15, "Setting Predefined or Custom Formulas."](#)

3.8.7 Inserting Dynamic Hyperlinks

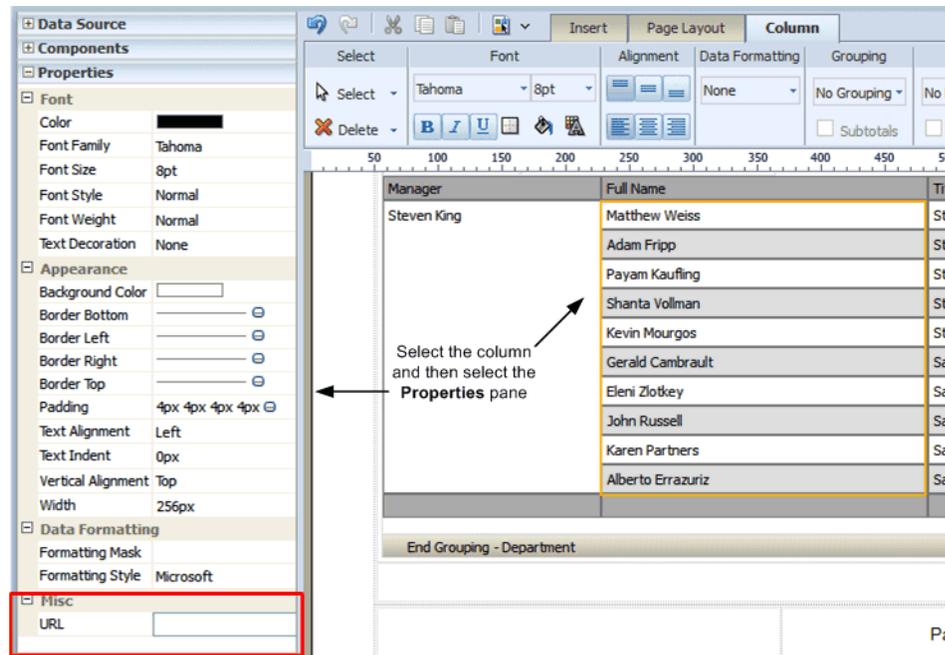
The layout editor supports dynamic hyperlinks in tables.

To insert a dynamic hyperlink:

1. Select the table column.

- Click **Properties**. The column properties include an option for URL, as shown in [Figure 3-57](#)

Figure 3-57 URL Property Option for the Selected Column



- In the URL field, enter the static portion of the URL and embed the absolute path to the element that provides the dynamic portion of the URL within curly braces {}. For example:

```
http://example.com/show_page?id={/DATA/GROUP1/ELEMENT_NAME}
```

where `http://example.com/show_page?id=` is the static portion of the URL and `{/DATA/GROUP1/ELEMENT_NAME}` is the absolute path to the element in the data that supplies the dynamic portion.

For example, in the employee salary report, suppose each employee name should render as a hyperlink to the employee's person record. Assume the static portion of the URL to each person record is

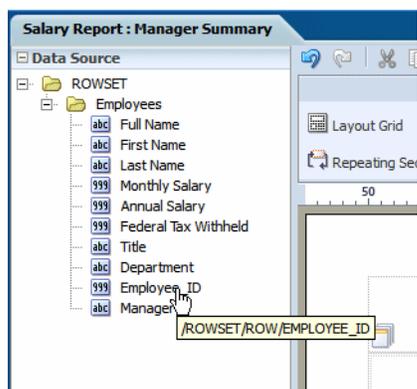
```
https://people.hrserver.com/records/show_page?id=
```

The dynamic portion comes from the data element `EMPLOYEE_ID`. For this example, append the full path to the `EMPLOYEE_ID` element within curly braces and enter this in the URL field as follows:

```
https://people.hrserver.com/records/show_
page?id={/ROWSET/ROW/EMPLOYEE_ID}
```

If you are unsure of the correct element names in the absolute path, hover your mouse over the data element on the **Data Source** pane to display the path in the hover text, as shown in [Figure 3-58](#).

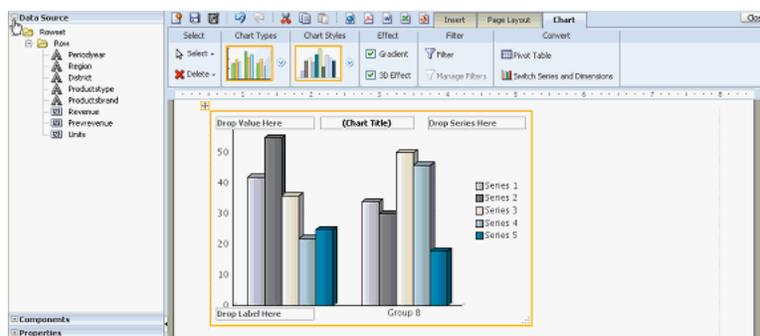
Figure 3–58 *Hover Text Displaying Absolute Path to Element*



3.9 About Charts

The layout editor supports a variety of chart types and styles to graphically present data in the layout. [Figure 3–59](#) shows side-by-side vertical bar and pie charts in the layout editor.

Figure 3–59 *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 following Chart Label properties apply to Scatter and Bubble chart types only: Title Font, Title Horizontal Align, Title Text, and Title Visible.

- Chart Values

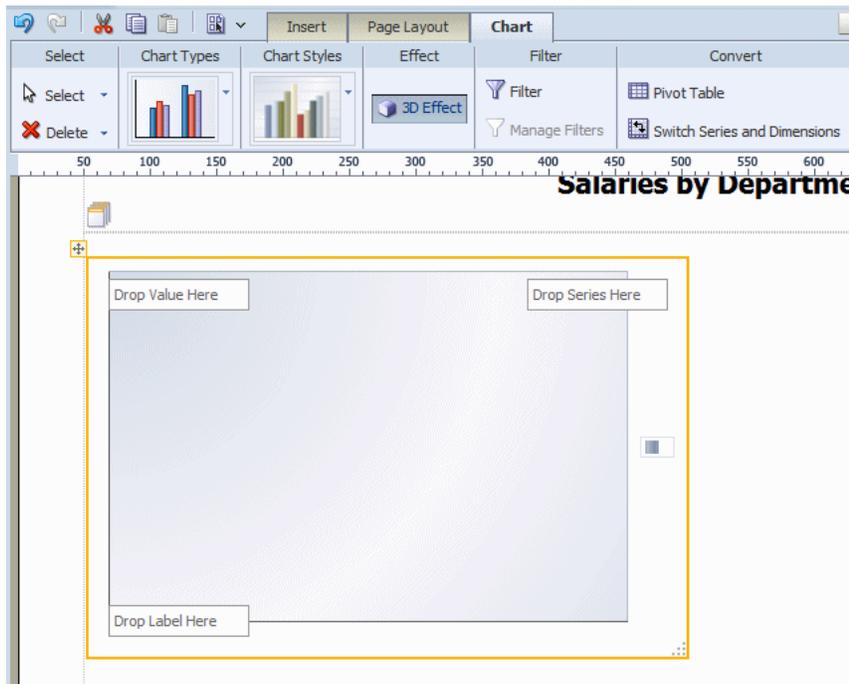
Note: Some font effects such as underline, italic, and bold might not render in PDF output.

3.9.1 Inserting a Chart

To insert a chart:

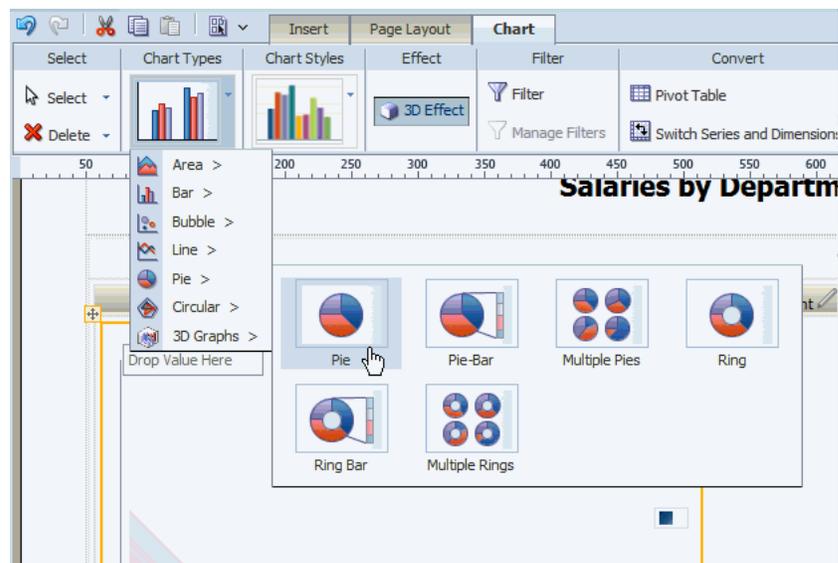
1. From the **Insert** menu, select and drag the Chart component to the layout.
By default an empty vertical bar chart is inserted and the **Chart** dynamic tab is displayed, as shown in [Figure 3–60](#).

Figure 3–60 Chart Dynamic Tab



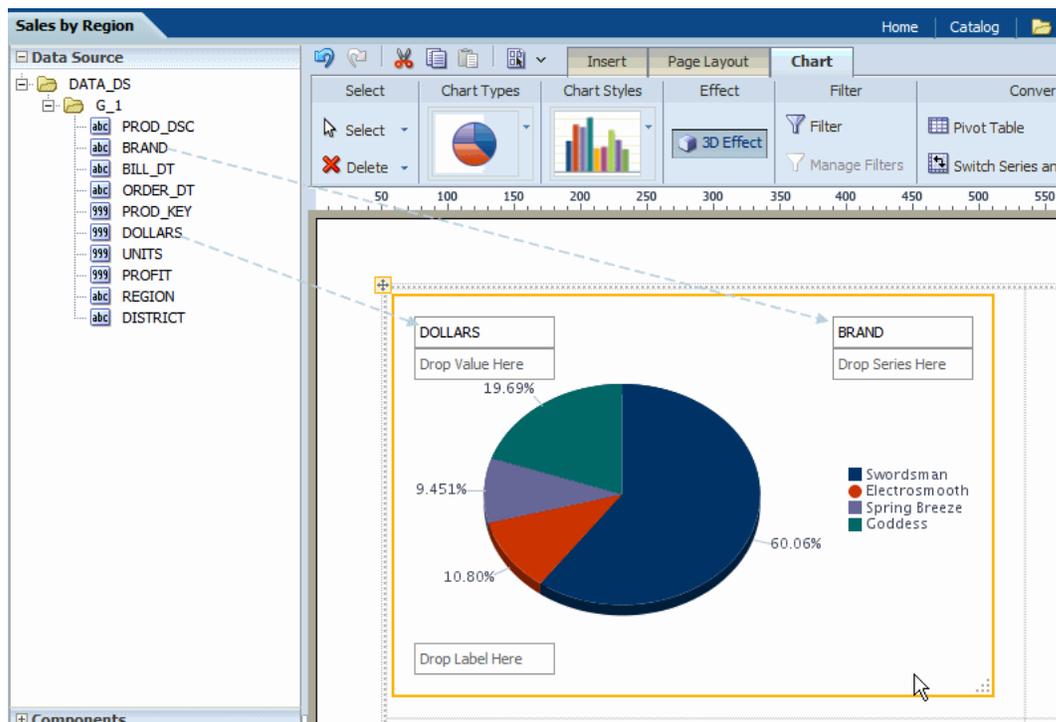
2. To change the chart type, click the Chart Type list to select a different type. In [Figure 3–61](#) the chart type is changed to Pie.

Figure 3-61 The Pie Chart Type

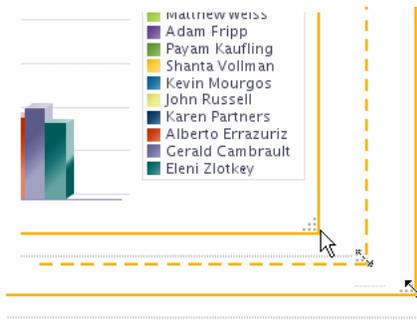


3. Select and drag the data fields from the Data Source pane to the appropriate areas in the chart. The chart immediately updates with the preview data, as shown in Figure 3-62.

Figure 3-62 Dragging and Dropping Data Fields to a Chart



4. To resize the chart, drag and drop the resize handler on the lower right corner of the chart, as shown in Figure 3-63.

Figure 3–63 Chart Resizing

3.9.1.1 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

3.9.1.2 Applying and Managing Filters

See [Section 3.8.3.2, "About Filters"](#) for information on how to apply and manage filters.

3.9.1.3 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 converts the label, series, and value elements of the chart into the appropriate rows, columns, and data elements of a pivot table.

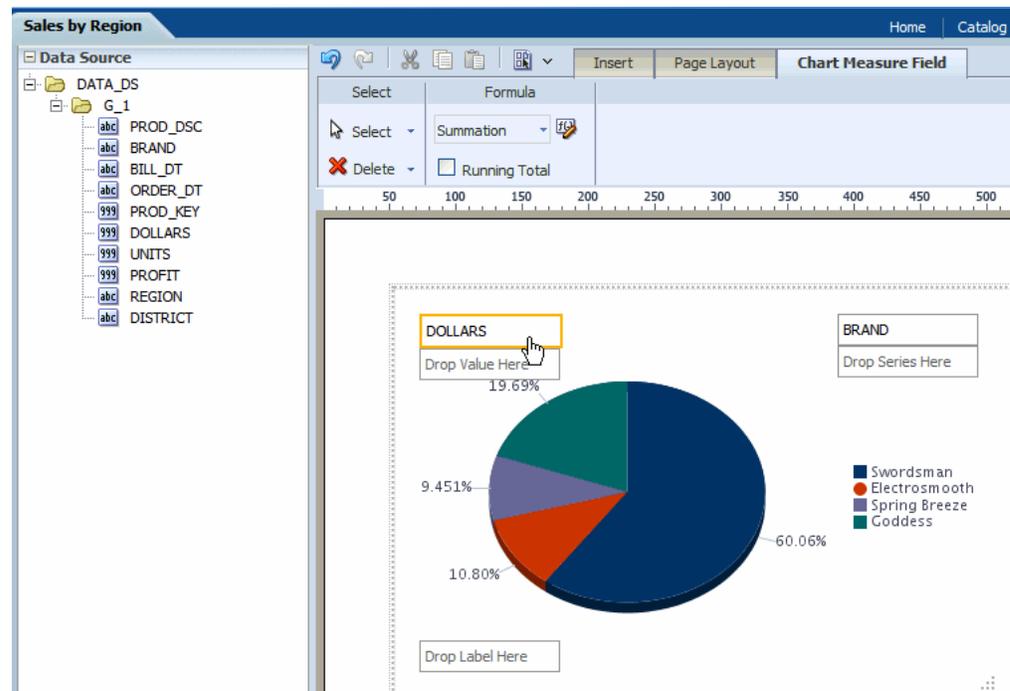
3.9.2 Changing the Formula Applied to a Chart Measure Field

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.

To change the formula:

1. Select the measure field in the chart. This displays the **Chart Measure Field** tab, as shown in [Figure 3–64](#).

Figure 3–64 Chart Measure Field Tab



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

3.9.3 Sorting a Chart Field

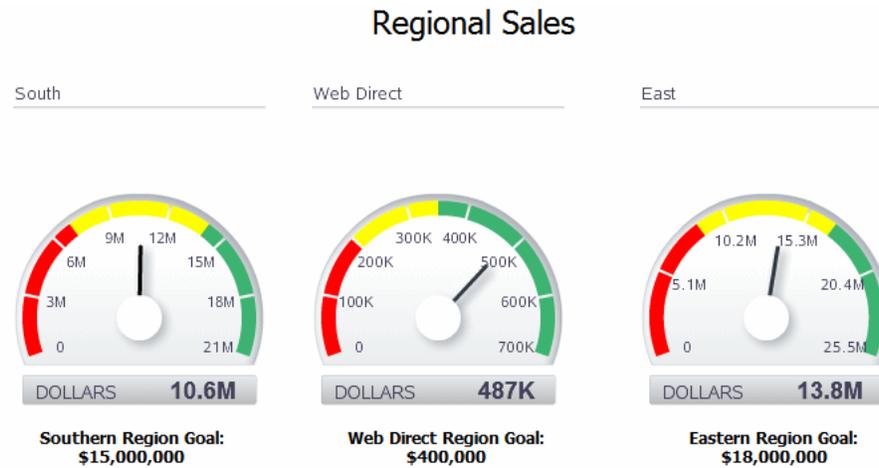
To sort a field in the chart:

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.

3.10 About Gauge Charts

A gauge chart is a useful way to illustrate progress or goals. For example, [Figure 3–65](#) shows a report with three gauges to indicate the status of regional sales goals:

Figure 3–65 Gauges Showing the Status of Regional Sales Goals

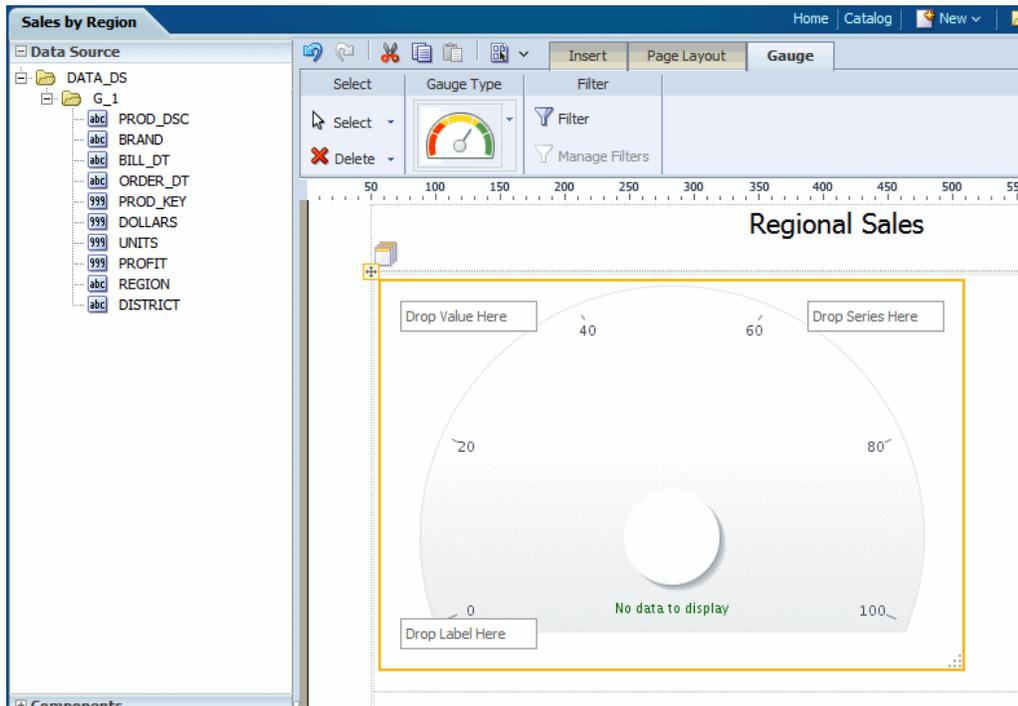


3.10.1 Inserting a Gauge Chart

To insert a gauge chart in the layout:

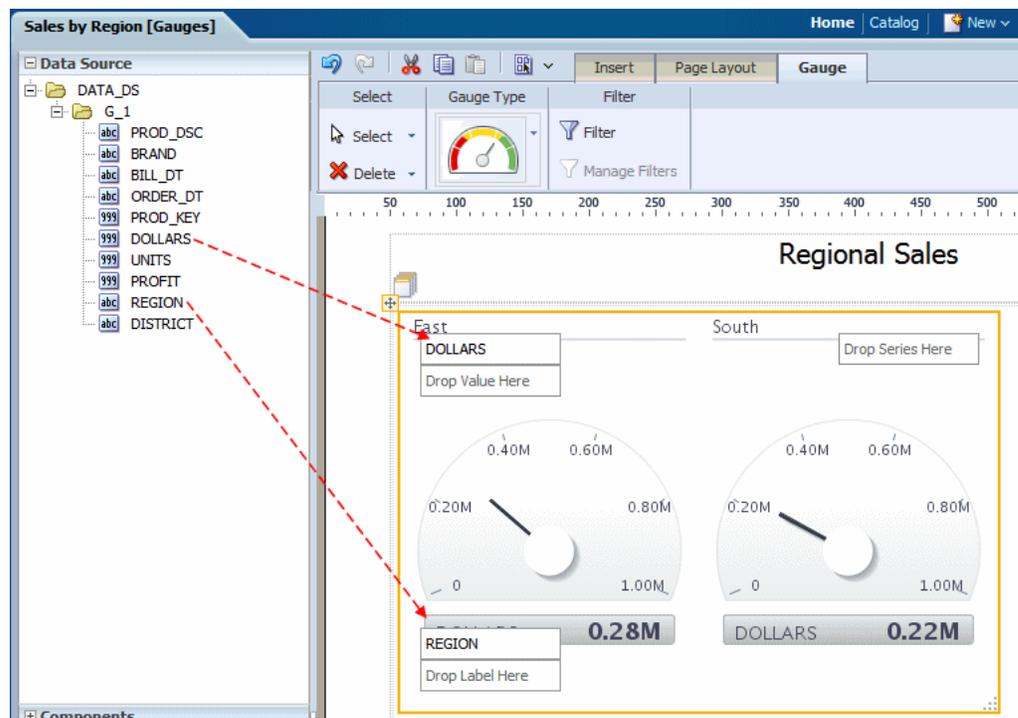
1. From the **Insert** menu, select and drag the **Gauge** component to the layout. This inserts an empty gauge chart, as shown in [Figure 3–66](#).

Figure 3–66 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 immediately updates with the preview data.

[Figure 3–67](#) shows REGION being dragged to the Label area and DOLLARS being dragged to the Value area:

Figure 3–67 Dragging and Dropping REGION and DOLLARS to a Chart

Note the following:

- A separate gauge is created for each occurrence of the Label (that is, each REGION). One set of properties applies 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 [Section 3.9.2, "Changing the Formula Applied to a Chart Measure Field."](#)
- You can apply a sort to the other gauge chart fields.

3.10.2 Setting the Properties for a Gauge Chart

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

3.10.3 Applying and Managing Filters

See [Section 3.8.3.2, "About Filters"](#) for information on how to apply and manage filters.

3.11 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. [Figure 3–68](#) shows a pivot table.

Figure 3–68 A Pivot Table

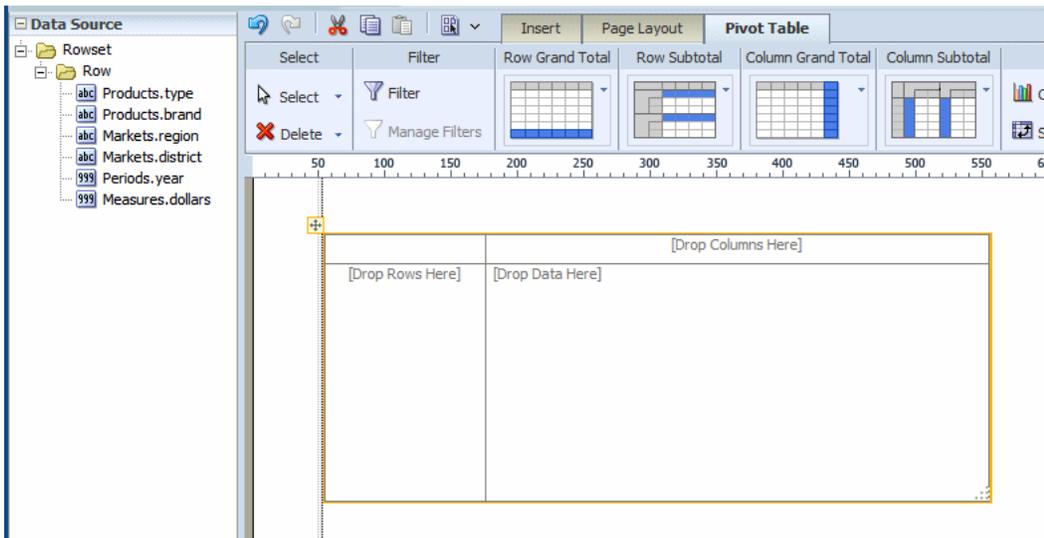
		Magicolor		Subtotal		Total	
		PrevRevenue	Revenue	PrevRevenue	Revenue	PrevRevenue	Revenue
CENTRAL REGION	CHICAGO DISTRICT	5,587.00	3,741.00	5,587.00	3,741.00	5,587.00	3,741.00
	CINCINNATI DISTRICT	9,875.00	38,054.00	9,875.00	38,054.00	9,875.00	38,054.00
	DETROIT DISTRICT	5,706.00	2,149.00	5,706.00	2,149.00	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
Subtotal		30,775.00	99,694.00	30,775.00	99,694.00	30,775.00	99,694.00
Total		30,775.00	99,694.00	30,775.00	99,694.00	30,775.00	99,694.00

3.11.1 Inserting a Pivot Table

To insert a pivot table:

1. From the **Insert** tab, select and drag the **Pivot Table** component to the layout. [Figure 3–69](#) shows the empty pivot table structure.

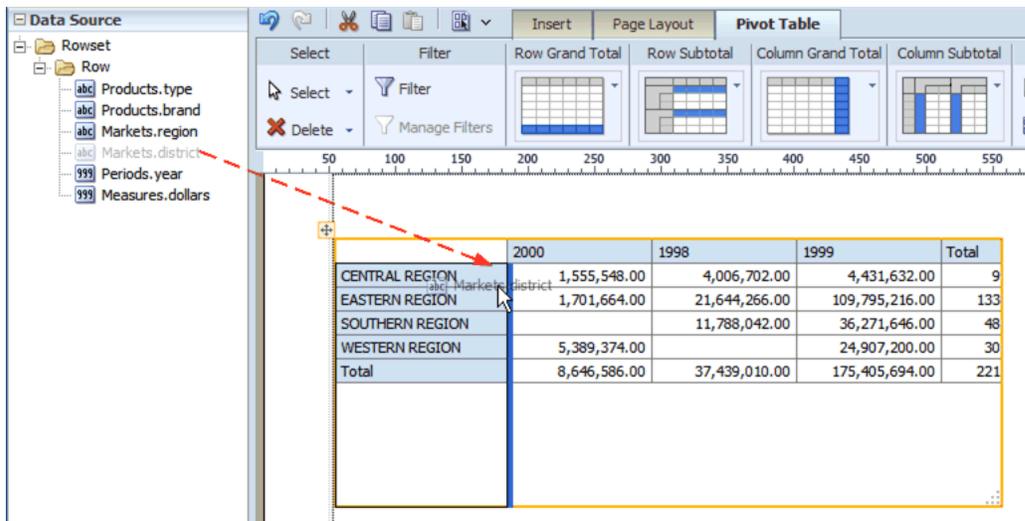
Figure 3–69 The Empty Pivot Table Structure



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

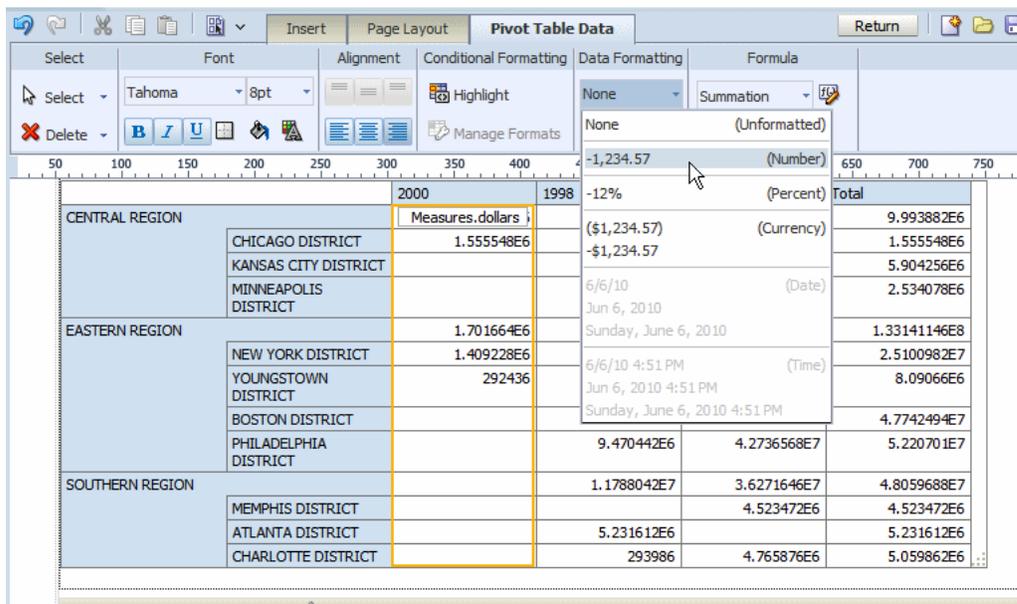
Drag multiple fields to the pivot table and place them precisely to structure the pivot table, as shown in [Figure 3–70](#).

Figure 3–70 Dragging and Dropping Data Fields to a Pivot Table



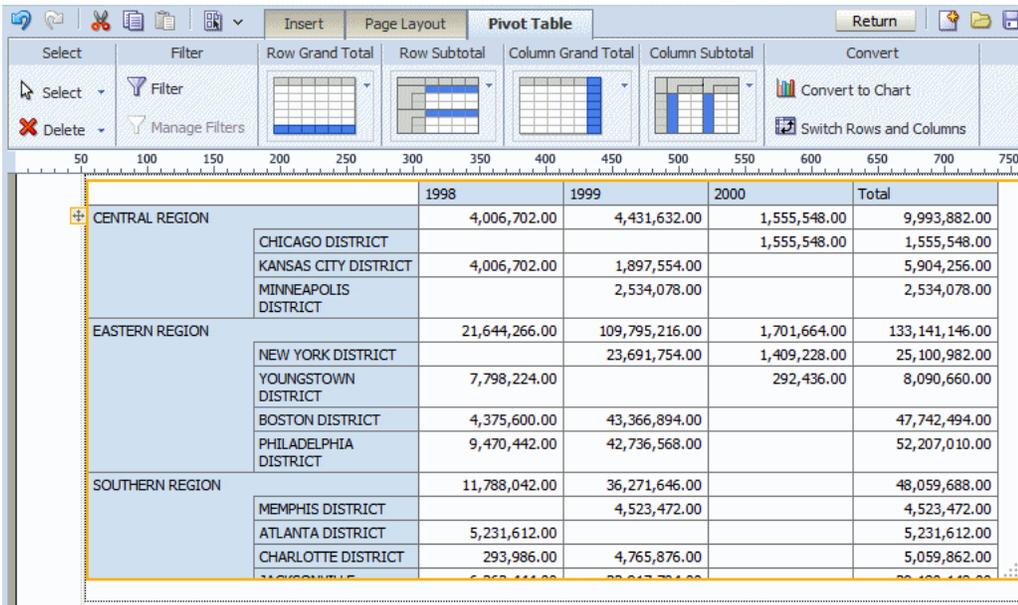
- By default the pivot table is inserted with no data formatting applied. To apply a format to the data, click the first column of data to enable the Pivot Table Data toolbar. On the Data Formatting group, select the appropriate format as shown in Figure 3–71.

Figure 3–71 Selecting a Format



- Optionally resize the pivot table by clicking and dragging the handler in the lower right corner of the pivot table, as shown in Figure 3–72.

Figure 3–72 Resizing a Pivot Table



3.11.2 Customizing a Pivot Table Menu

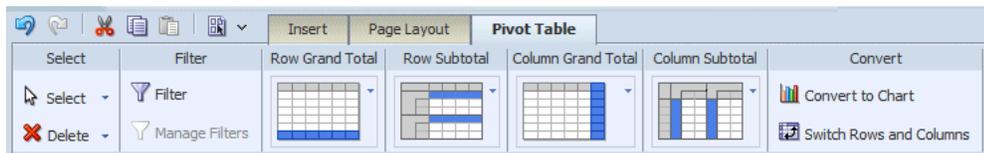
After you insert a pivot table customize the appearance and layout using the following dynamic tabs:

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

3.11.3 About the Pivot Table Tab

Figure 3–73 shows the Pivot Table tab.

Figure 3–73 The Pivot Table Tab



3.11.3.1 Applying Filters

See [Section 3.8.3.2, "About Filters"](#) for a description of the **Filter** and **Manage Filters** features.

3.11.3.2 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 subgroup, with no header

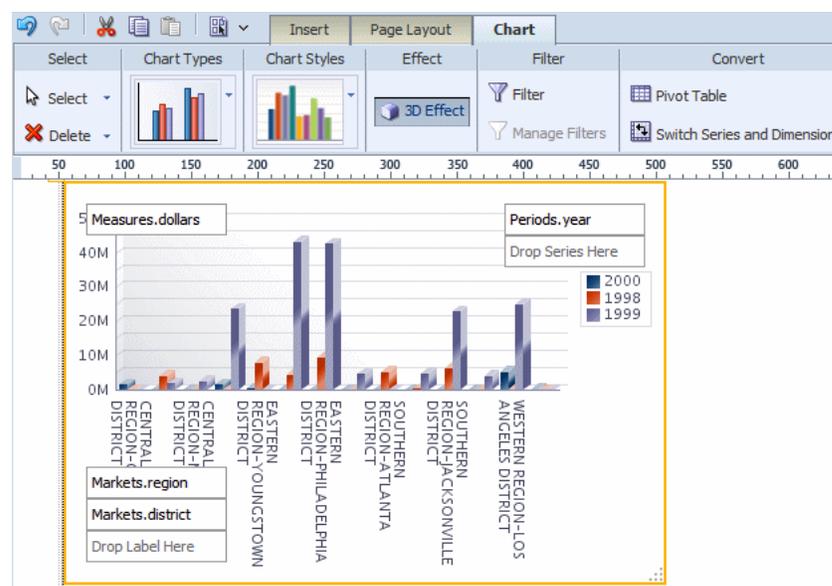
Change the positioning and display of totals and subtotals by clicking the appropriate group in the tab and selecting the desired layout pattern from the menu.

3.11.3.3 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 [Section 3.9, "About Charts."](#)

[Figure 3-74](#) shows the pivot table created in the preceding step converted to a vertical bar chart.

Figure 3-74 A Pivot Table Converted to a Vertical Bar Chart

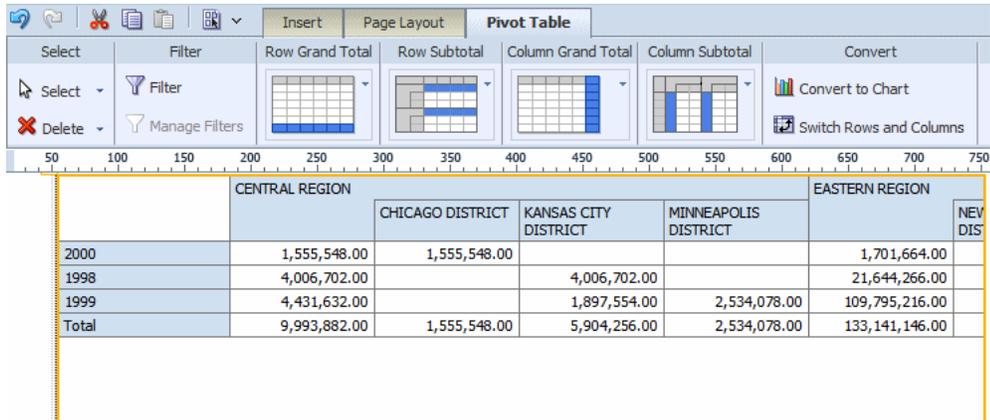


3.11.3.4 Switching Rows and Columns

Use the **Switch Rows and Columns** command to see a different view of the same data.

[Figure 3-75](#) shows the pivot table created in the previous step with rows and columns switched.

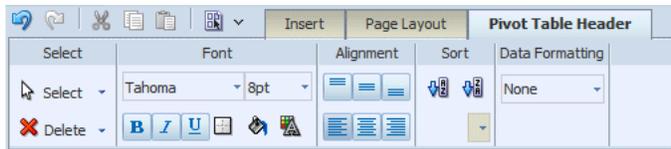
Figure 3–75 A Pivot Table with Rows and Columns Switched



3.11.4 Customizing the Pivot Table Headers

The **Pivot Table Header** tab is shown in [Figure 3–76](#).

Figure 3–76 The Pivot Table Header Tab



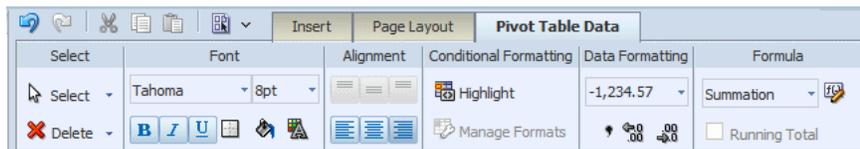
Select the column or row header of the pivot table and use the **Pivot Table Header** tab to perform the following:

- Customize the fonts, colors, alignment and other display features of the header
- Apply a sort order (for more information see [Section 3.8.5.5, "About the Sort Option"](#))
- Apply data formatting (if the data type is number or date)

3.11.5 Customizing the Pivot Table Data

The **Pivot Table Data** tab is shown in [Figure 3–77](#).

Figure 3–77 The Pivot Table Data Tab



Select the data area of the pivot table and use the **Pivot Table Data** tab to perform the following actions. 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 [Section 3.8.3.5, "About Conditional Formats"](#))
- Apply data formatting (see [Section 3.8.5.1, "About the Data Formatting Options for Columns"](#))
- Apply a formula (see [Section 3.8.6.2, "Applying a Formula"](#))

3.12 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.

3.12.1 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 places the data field beneath the text field as shown in [Figure 3–78](#).

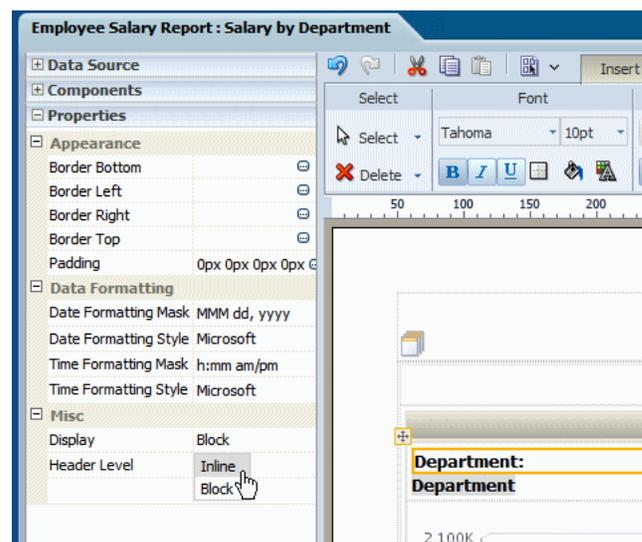
Figure 3–78 A Data Field Beneath a Text Item



To display the data field inline with the text item:

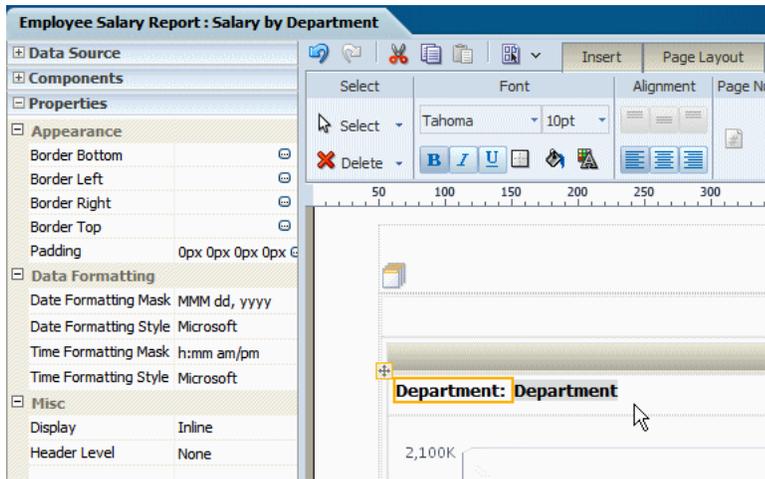
1. Set the Display property to **Inline** in the **Properties** pane, as shown in [Figure 3–79](#).

Figure 3–79 Setting the Display Property to Inline



This setting enables the positioning of text items and data fields into a single line as shown in [Figure 3–80](#).

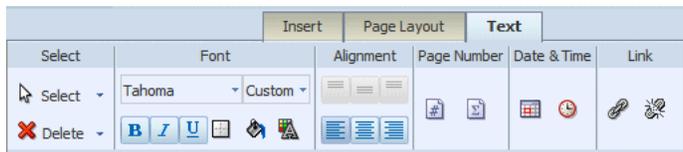
Figure 3–80 Text Items and Data Fields Positioned in a Single Line



3.12.2 About the Text Toolbar

The Text tab is shown in [Figure 3–81](#).

Figure 3–81 The Text tab



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

3.12.2.1 Editing Font Properties

Use the Font group of commands to set the following:

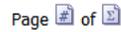
- Select a font style
- Select a font size
- Apply emphasis (bold, italic, or underline)
- Insert a border around the text item
- Apply a background color
- Apply a font color

3.12.2.2 Inserting Page Numbers

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

Figure 3–82 shows the Page # of *N* construction.

Figure 3–82 Page # of *N* Construction



Page # of N

To create the Page # of *N* construction:

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.

3.12.2.3 Inserting the Date and Time

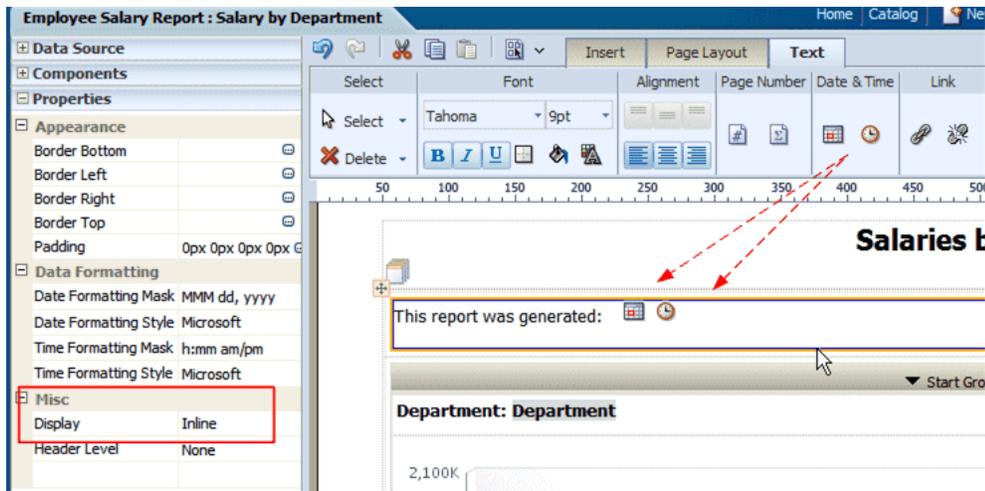
To insert the date and time in a report:

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, set the Text Item property to "Inline".

Figure 3–83 shows the insertion of the date and time icons.

Figure 3–83 Inserting the Date and Time Icons



When this report is viewed, the date and time are displayed according to the server time zone if viewed online, or for scheduled reports, the time zone selected for the schedule job. [Figure 3–84](#) shows the date and time displayed in a report.

Figure 3–84 The Date and Time Displayed in a Report



3.12.2.4 Inserting a Hyperlink

To insert a hyperlink in a 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.

3.13 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 the data. The value of the element is evaluated at runtime enabling dynamic insertion of images.

To insert an image:

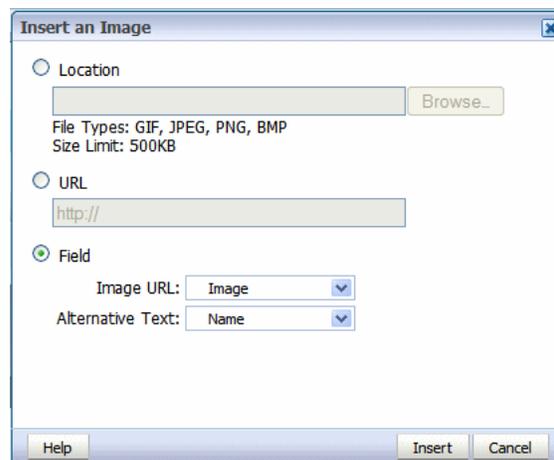
1. Drag and drop the image component to the layout.
2. In the **Insert an Image** dialog, specify one of the following sources for the image:
 - **Location:** Click **Browse** to specify the file name and directory of the image on a local or mapped drive to upload the image.
 - **URL:** Enter the URL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

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

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

Figure 3–85 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 is used as alternative text.

Figure 3–85 *Insert an Image Dialog*

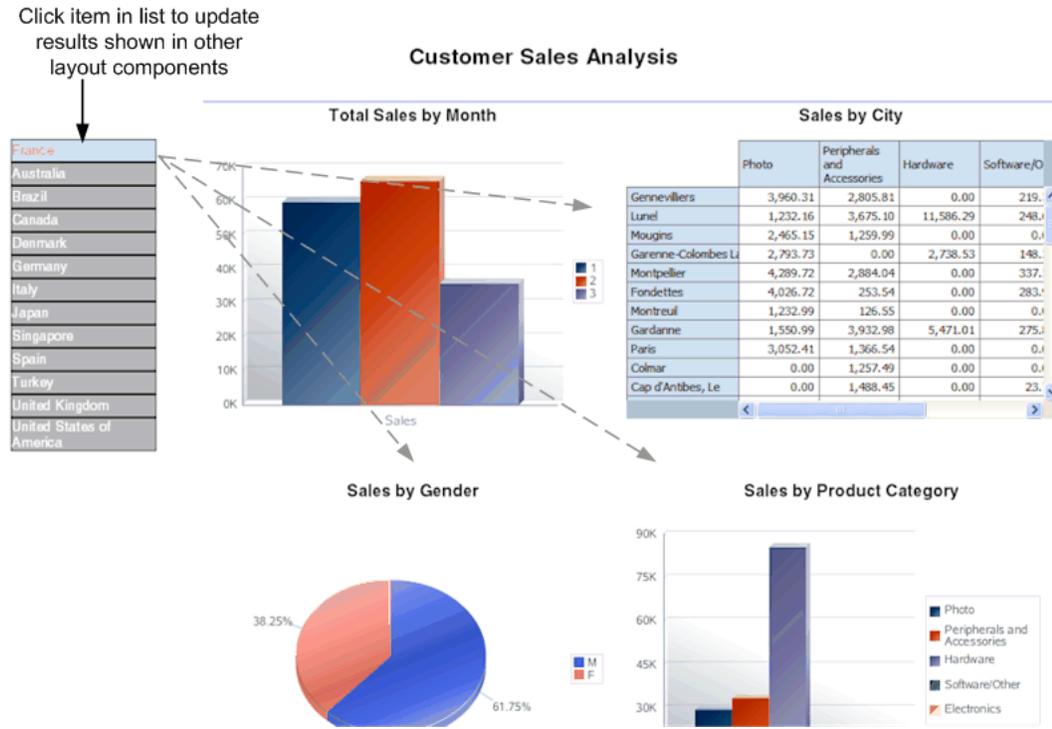


3. Optionally resize the image in one of these ways:
 - Drag the right bottom corner of the image. To preserve the aspect ratio when resizing an image, press and hold the Shift key before starting to drag the corner.
 - Modify the width and height in the Properties pane.

3.14 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. [Figure 3–86](#) 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.

Figure 3–86 Using a List Component to Update Results



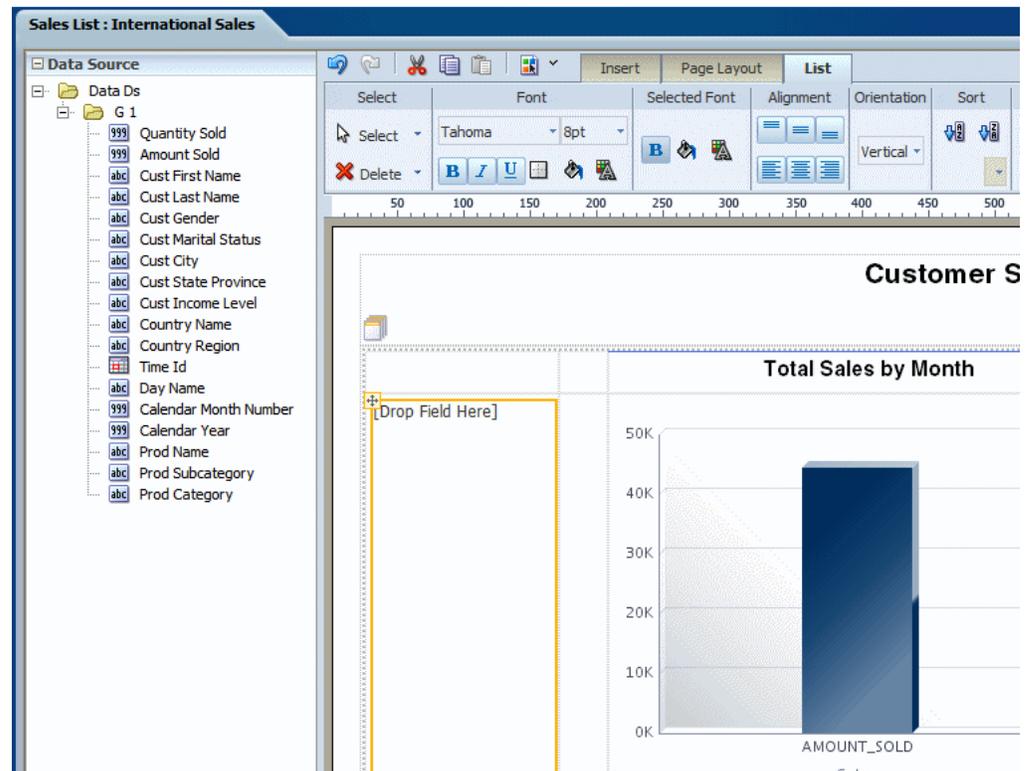
3.14.1 Inserting a List

To insert a list:

1. From the **Insert** tab, select and drag the **List** component to the design area.

[Figure 3–87](#) shows an inserted, empty list.

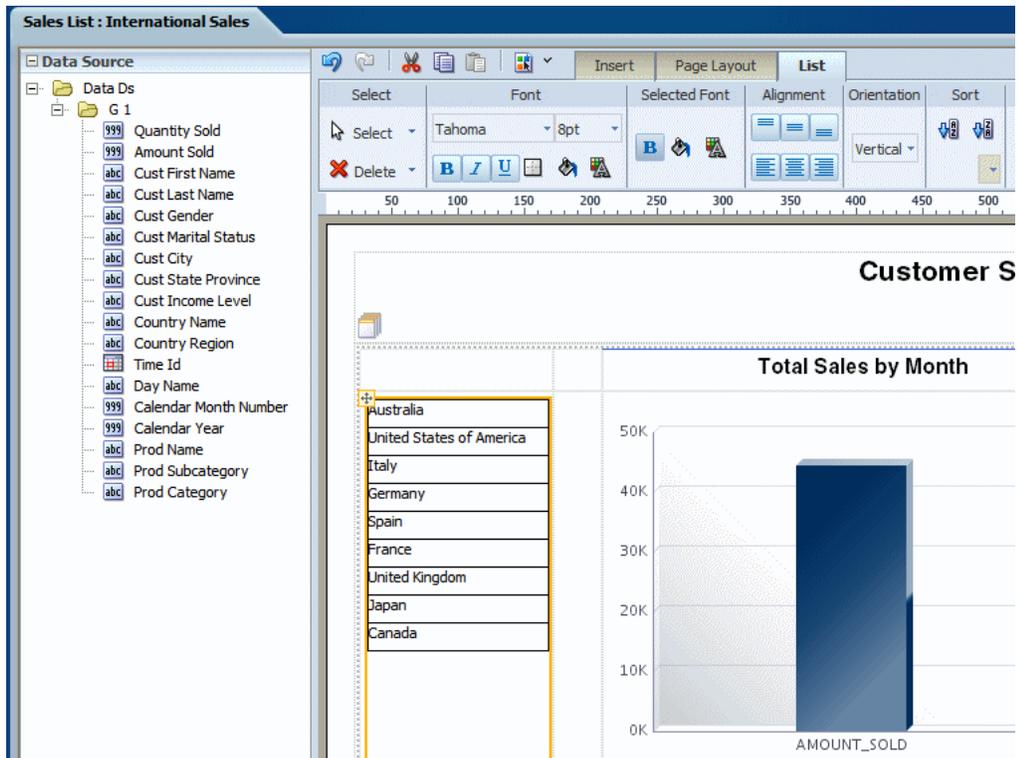
Figure 3–87 An Inserted, Empty List



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

Figure 3–88 shows the list component after dragging the element Country Name to it.

Figure 3–88 A List Component Showing Country Names



3. Customize the appearance of the list. See [Section 3.14.2, "Customizing a List."](#)
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 [Section 3.4.5, "Interactivity: Event Configuration"](#)

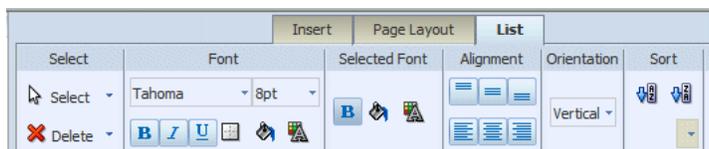
3.14.2 Customizing a List

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

[Figure 3–89](#) shows the **List** tab.

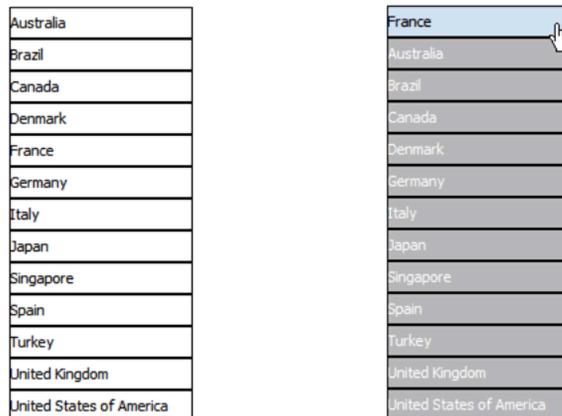
Figure 3–89 The List Tab



3.14.2.1 Customizing the Font Style and the Selected Font Style Commands

In [Figure 3–90](#), the list on the left shows the default format of the list. The list on the right shows the **Selected Font** default format:

Figure 3–90 *Default Formats*



Default format of list with
no items selected

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.

3.14.2.2 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 [Figure 3–91](#).

Figure 3–91 The Hide Excluded Property

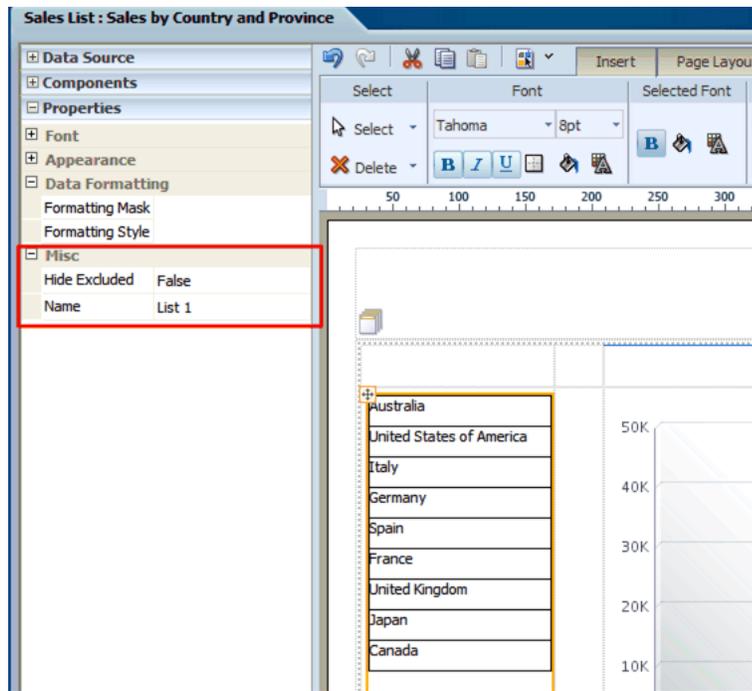


Figure 3–92 shows the difference in the display depending on the setting of the property.

Figure 3–92 Display Differences of the Hide Excluded Property Settings

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

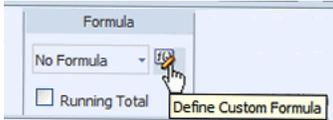
Australia
Brazil
Canada
Denmark
France
Germany
Italy
Japan
Singapore
Spain
Turkey
United Kingdom
United States of America

Selected item not moved; other items unchanged

3.15 Setting Predefined or Custom Formulas

Figure 3–93 shows the Define Custom Formula icon.

Figure 3–93 Define Custom Formula Icon



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.

3.15.1 About the Predefined Formulas

The menu provides the predefined formulas that are described in Table 3–6.

Table 3–6 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:

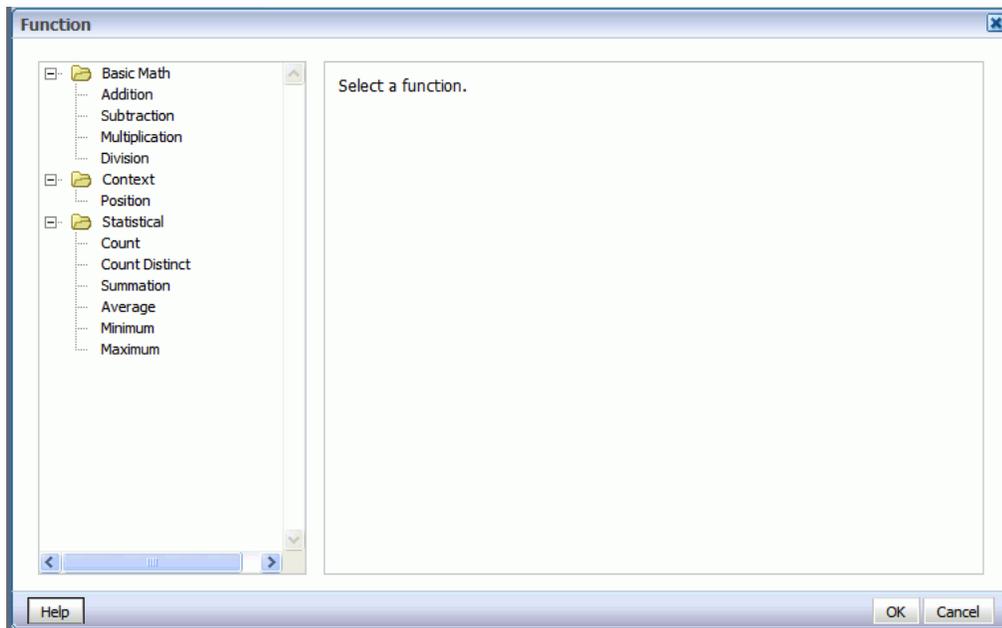
- Blank Text
- Count
- Count Distinct

3.15.2 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 the layout.

Figure 3–94 shows the **Function** dialog.

Figure 3–94 The Function Dialog

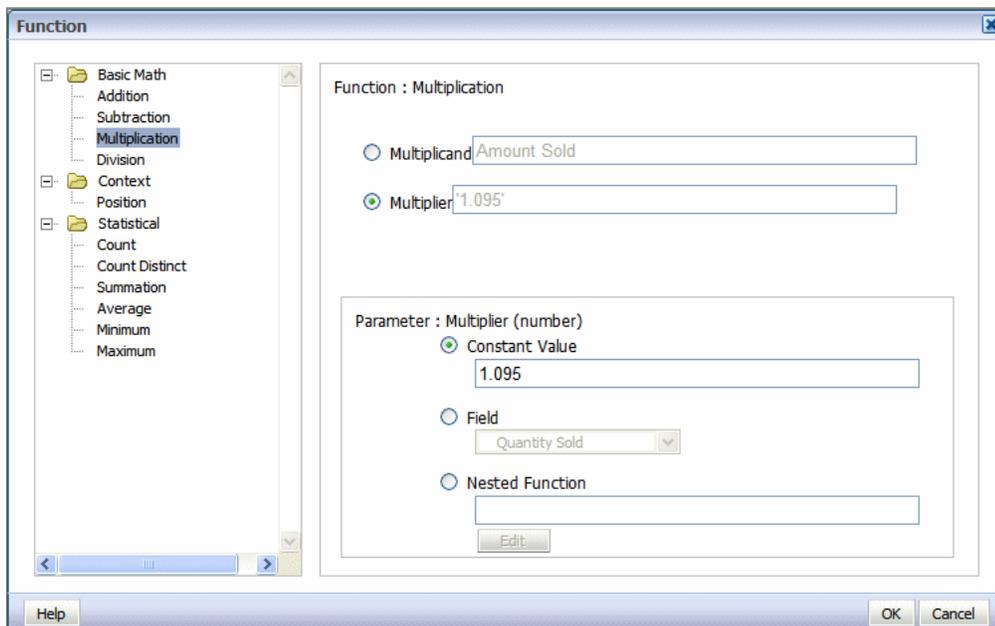


3.15.2.1 About the Basic Math Functions

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 [Figure 3–95](#), 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.

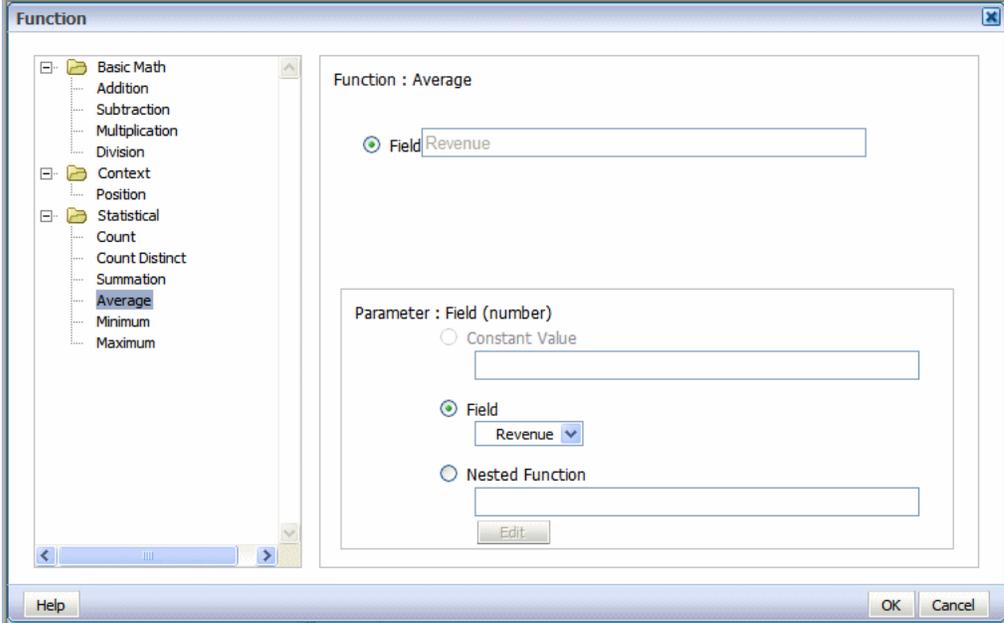
Figure 3–95 Example of the Multiplication Function



3.15.2.2 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 [Figure 3-96](#), clicking the Average function displays prompts for you to specify the source of the values for which to calculate the average.

Figure 3-96 The Average Function



3.15.3 Applying a Custom Formula: Examples

Example 1: Subtraction

[Figure 3-97](#) shows data for Revenue and Cost for each Office:

Figure 3-97 A Table Showing Revenue and Cost Data 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).

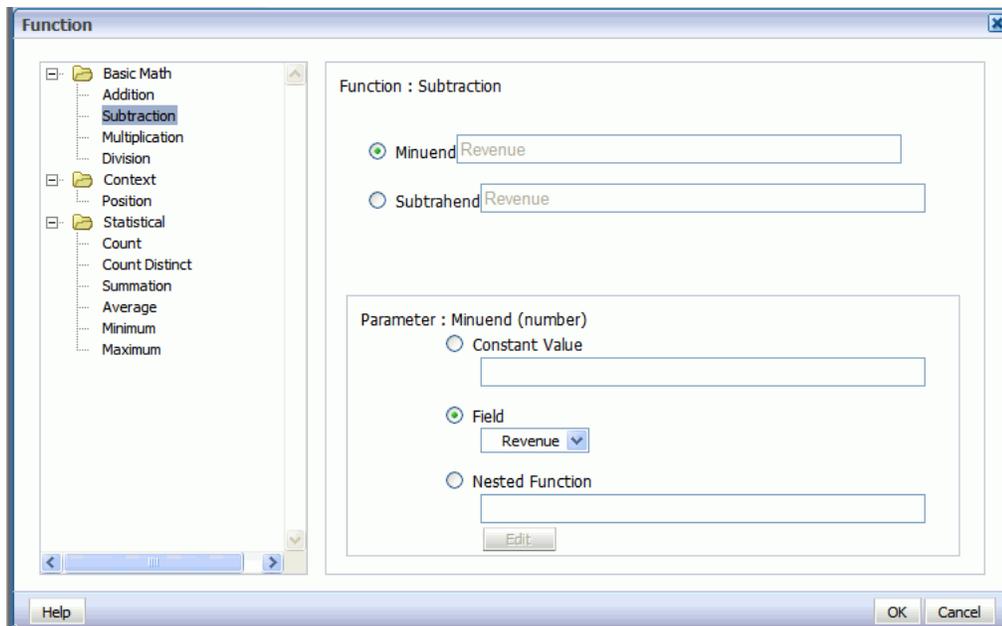
1. Add another numeric data column to the table. For example, drag another instance of Revenue to the table, as shown in [Figure 3-98](#).

Figure 3–98 A Table With Two Instances of Revenue

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

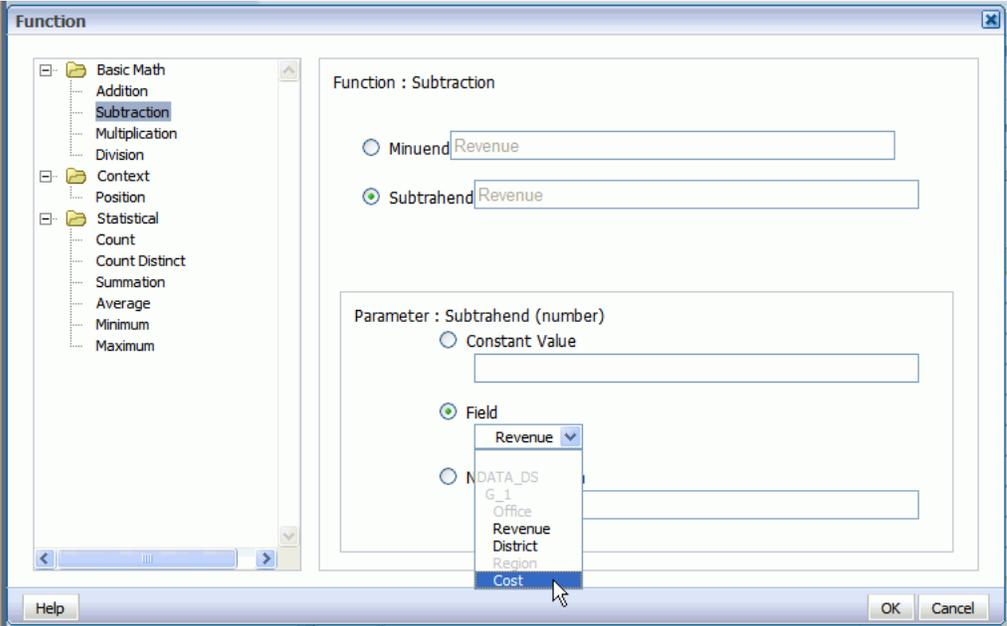
2. With the table column selected, click **Define Custom Formula**.
3. In the **Function** dialog select **Subtraction** from the list, as shown in [Figure 3–99](#). Because the source data for the column is Revenue, by default the Minuend and the Subtrahend both show the Revenue element.

Figure 3–99 The Subtraction Function



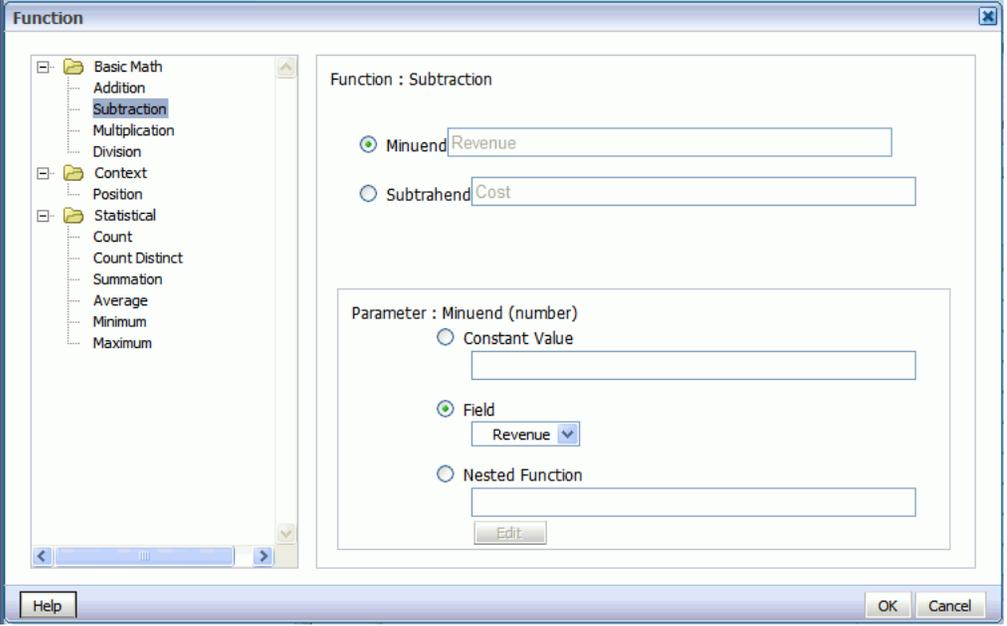
4. Select Subtrahend, then in the Parameter region, select **Field** and choose the Cost element, as shown in [Figure 3–100](#).

Figure 3-100 Subtraction Function with the Cost Element Selected



The dialog is updated to show that the formula is now Revenue minus Cost, as shown in Figure 3-101.

Figure 3-101 Updated Subtraction Function Showing a Formula of 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 the table has a Profit column, as shown in Figure 3-102.

Figure 3–102 A Table Showing the Custom Formula Column Titled Profit

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 uses a nested function to create a column that shows Revenue less taxes.

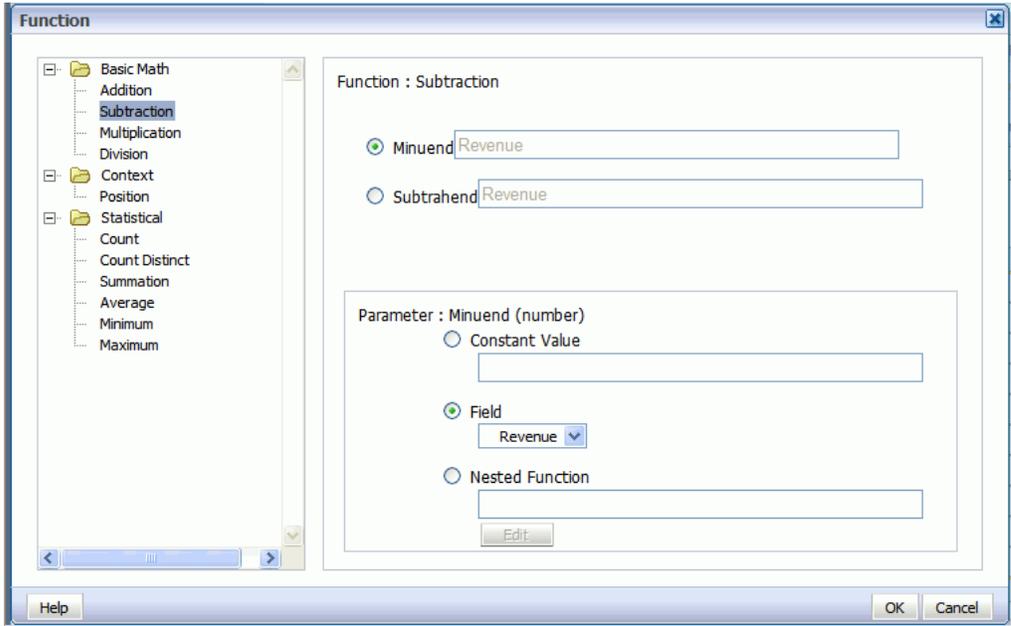
1. Add another numeric data column to the table. For example, drag another instance of Revenue to the table, as shown in [Figure 3–103](#).

Figure 3–103 A Table With Two Instances of Revenue

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

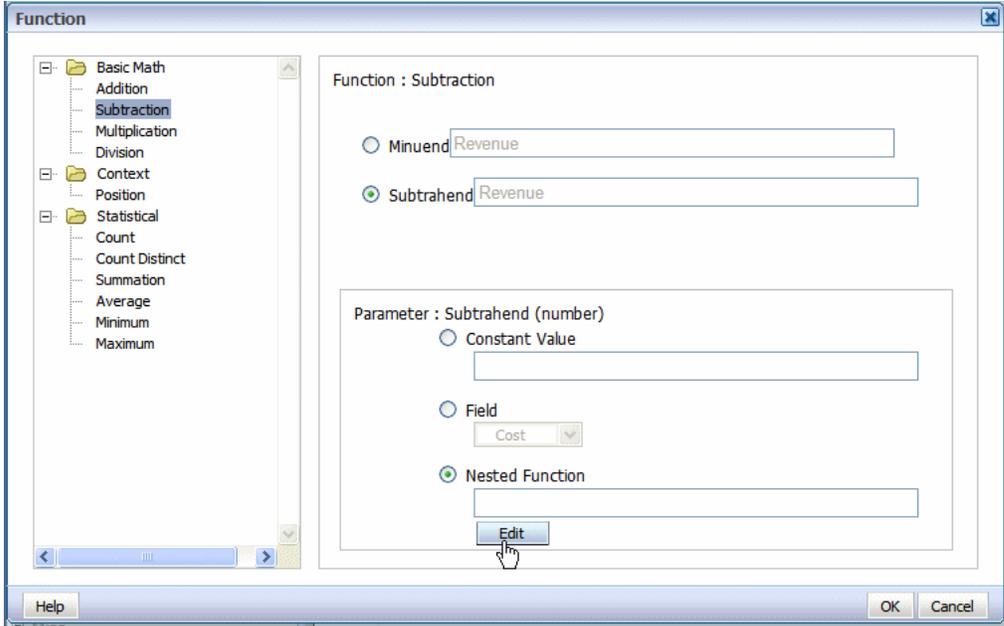
2. With the table column selected, click **Define Custom Formula**.
3. In the **Function** dialog select **Subtraction** from the list. Because the source data for the column is Revenue, by default the Minuend and the Subtrahend both show the Revenue element, as shown in [Figure 3–104](#).

Figure 3-104 Subtraction Function with Minuend and Subtrahend Showing the Revenue Element



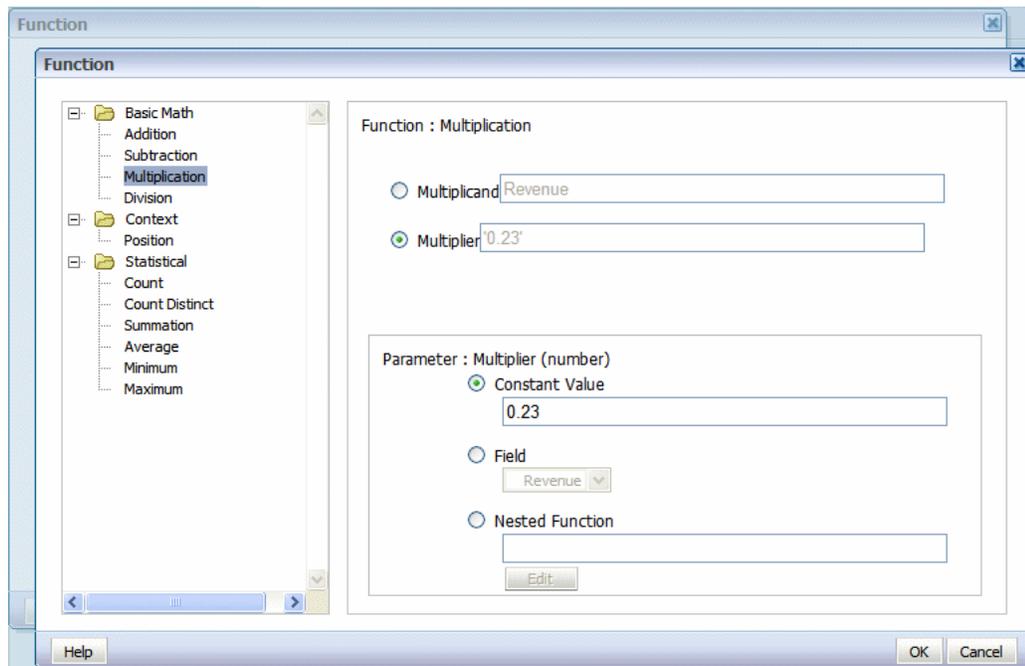
- 4. Select Subtrahend, then in the Parameter region, select **Nested Function** and click **Edit**, as shown in [Figure 3-105](#).

Figure 3-105 Subtraction Function with the Nested Function Selected



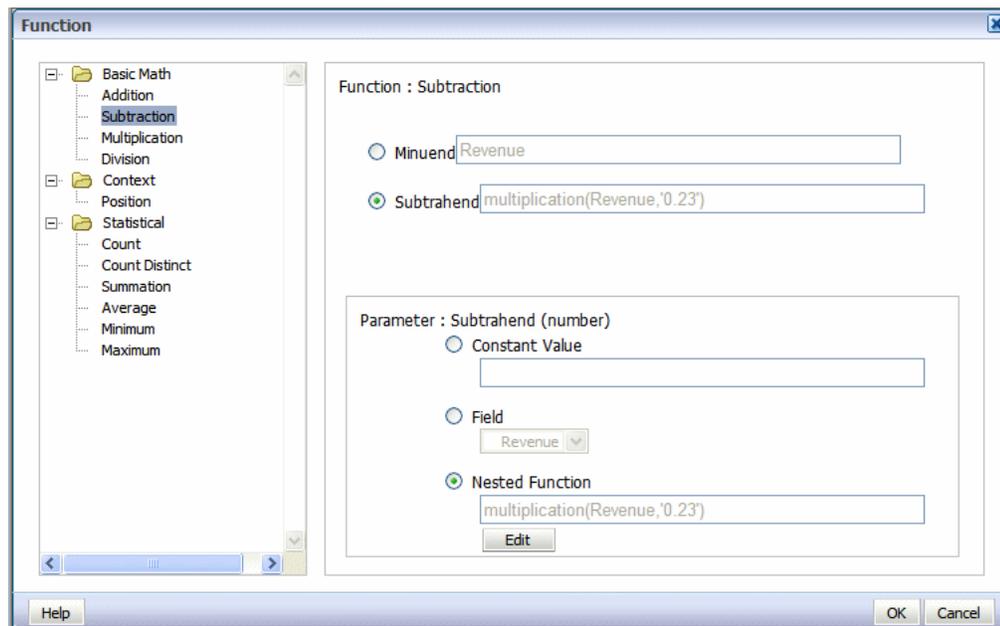
A second Function dialog is displayed to enable you to define the nested function. In this case the nested function is Revenue times a constant value (tax rate of .23), as shown in [Figure 3-106](#).

Figure 3–106 The Function Dialog Showing the Nested Function Revenue Times a Constant Value



5. Click **OK** to close the dialog. The primary Function dialog now shows the nested function as the source of the subtrahend, as shown in [Figure 3–107](#).

Figure 3–107 The Function Dialog Showing a Nested Function as the Source of the Subtrahend



6. Click **OK** to close the Function dialog. The table column displays the custom formula. Edit the table column header label, and now the table displays the custom function, as shown in [Figure 3–108](#).

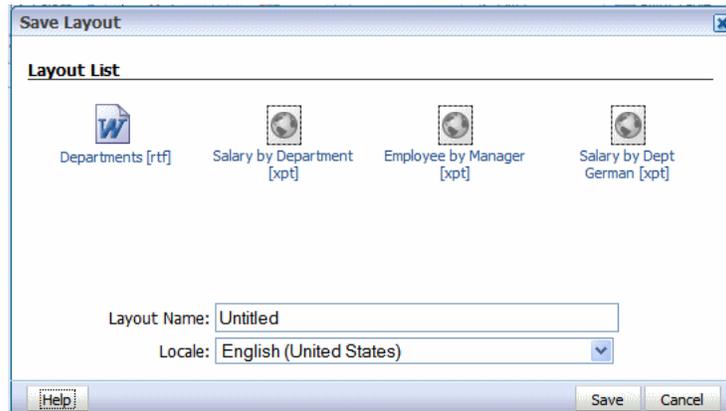
Figure 3–108 A Table Showing the Custom Function Revenue less tax (23%)

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

3.16 Saving a Layout

To save the layout to the report definition:

1. 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:

Figure 3–109 The Save Layout Dialog

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

Note: When you have saved the layout, the Locale cannot be updated.

Creating RTF Templates

This chapter describes the concepts of associating XML data to layout elements in an RTF report template. It describes basic and advanced techniques for creating complex and highly conditionalized report formats.

This chapter includes the following topics:

- Section 4.1, "Getting Started"
- Section 4.2, "Associating the XML Data to the Template Layout"
- Section 4.3, "Adding Markup to the Template Layout"
- Section 4.4, "Defining Groups"
- Section 4.5, "Defining Headers and Footers"
- Section 4.6, "Inserting Images and Charts"
- Section 4.7, "Adding Drawings, Shapes, and Clip Art"
- Section 4.8, "Supported Formatting Features of Microsoft Word"
- Section 4.9, "Template Features"
- Section 4.10, "Using Conditional Formatting"
- Section 4.11, "Inserting Page-Level Calculations"
- Section 4.12, "Handling Data"
- Section 4.13, "Setting Variables, Parameters, and Properties"
- Section 4.14, "Using Advanced Report Layouts"
- Section 4.15, "Formatting Numbers, Dates, and Currencies"
- Section 4.16, "Supporting Calendars and Time Zones"
- Section 4.17, "Using External Fonts"
- Section 4.18, "Controlling the Placement of Instructions Using the Context Commands"
- Section 4.19, "Using XPath Commands"
- Section 4.20, "Declaring Namespaces"
- Section 4.21, "Using FO Elements and XSL Elements"
- Section 4.22, "Guidelines for Designing RTF Templates for Microsoft PowerPoint Output"
- Section 4.23, "Rendering HTML Formatted Data in a Report"

4.1 Getting Started

This chapter describes the concepts of associating XML data to layout elements in a 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, then see [Chapter 5, "Creating RTF Templates Using the Template Builder for Word"](#) before reading this chapter. The demos and samples provided in the Template Builder installation can 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 in a template using the Template Builder.

This section covers the following topics:

- [Section 4.1.1, "What Are RTF Templates?"](#)
- [Section 4.1.2, "Prerequisites for Designing Templates"](#)
- [Section 4.1.3, "What is XSLT Compatibility?"](#)
- [Section 4.1.4, "Key Concepts"](#)
- [Section 4.1.5, "Designing the Template Layout"](#)
- [Section 4.1.6, "About Adding BI Publisher Code"](#)

4.1.1 What Are RTF Templates?

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 processor features.

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

In addition to the word processor'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 want to include code directly in the template, then you can include *any* XSL element, many FO elements, and a set of SQL expressions that BI Publisher extends.

4.1.2 Prerequisites for Designing Templates

Before you design a template, you must:

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

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

- Be familiar with the formatting features of Microsoft Word.

4.1.3 What is 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*.

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

XSLT compatibility is set as a Build Option in the Template Builder for Word. See [Section 5.9.1, "Setting UI Options."](#)

4.1.4 Key Concepts

When you design the 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.

4.1.5 Designing the Template Layout

Use the word processor'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 [Section 4.5, "Defining Headers and Footers."](#)

For a detailed list of supported formatting features in Microsoft Word, see [Section 4.8, "Supported Formatting Features of Microsoft Word."](#)

4.1.6 About Adding BI Publisher Code

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

- Basic RTF Method
Use any word processor 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 the template.

Note: If you use XSL or XSL:FO code rather than the simplified syntax, then 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 [Chapter 5, "Creating RTF Templates Using the Template Builder for Word."](#)

4.2 Associating the XML Data to the Template Layout

Figure 4–1 shows a sample layout for a Payables Invoice Register.

Figure 4–1 Sample Layout for Payables Invoice Register

The figure shows a sample layout for a Payables Invoice Register. It includes the Oracle E-Business Suite logo, the report title 'Payables Invoice Register', and page information 'Page 1 of 1' and '25th July 2003'. Below the header, there are filter controls: 'Group: Suppliers' and 'Sort by Supplier'. The main data is presented in a table with columns: Invoice Num, Invoice Date, GL Date, Curr, Entered Amt, and Accounted Amt. The table contains one data row and a summary row. The summary row shows 'Total for Supplier: Supplier1' with 'Entered Amt' of 1000.00 and 'Accounted Amt' of 1000.00. The layout ends with 'End:Suppliers'.

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
Total for Supplier: Supplier1				1000.00	1000.00

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 are repeated when the report is run.
For example, all the fields on the template are repeated for each Supplier that is reported. Each row of the invoice table is repeated for each invoice that is reported.

4.2.1 Using an XML Input File

Following is the XML file that is used as input to the Payables Invoice Register report template. To simplify the example, the XML output shown 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.

4.2.2 Identifying Placeholders and Groups

The template content and layout must correspond to the content and hierarchy of the input XML file. Each data field in the template must map to an element in the XML file. Each group of repeating elements in the 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 the report requires grouping that does not follow the hierarchy of the incoming XML data. For information on using this feature, see [Section 4.12.3, "Regrouping the XML Data."](#)

4.2.3 Using Placeholders

Each data field in the report template must correspond to an element in the XML file. When you mark up the 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 the template, you create a placeholder for `VENDOR_NAME` in the position of the Supplier field. At runtime, this placeholder is replaced by the value of the element from the XML file (the value in the sample file is `COMPANY A`).

4.2.4 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 loops through the occurrences of the element and displays the fields each time.

4.3 Adding Markup to the Template Layout

BI Publisher converts the formatting that you apply in the word processor to XSL-FO. You add markup to create the mapping between the layout and the XML file and to include features that cannot be represented directly in the 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 the template. For the XSL equivalents of the BI Publisher tags, see [Section B.2, "XSL Equivalents."](#)

4.3.1 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 the document using the following syntax:

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

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

There are two ways to insert placeholders in the document, as described in the following sections:

- [Section 4.3.1.1, "Using the Basic RTF Method"](#): Insert the placeholder syntax directly into the template document.
- [Section 4.3.1.2, "Using the 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 the template.

For more information, see [Section 5.4.1, "Inserting a Field."](#)

4.3.1.1 Using the Basic RTF Method

Enter the placeholder syntax in the 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 the document, enter:

```
<?VENDOR_NAME?>
```

The entry in the template is shown in [Figure 4-2](#).

Figure 4-2 *Template Entry*

Supplier: <?VENDOR_NAME?>

Invoice Num	Invoice

Total for Supplier:

4.3.1.2 Using the Form Field Method

To use Microsoft Word's Form Field method to insert the placeholder tags:

1. Enable the **Forms** toolbar in the Microsoft Word application.
2. Position the cursor in the location where you want to create a placeholder.

3. Select the **Text Form Field** toolbar icon. This action inserts a form field area in the 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 populates 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?>`

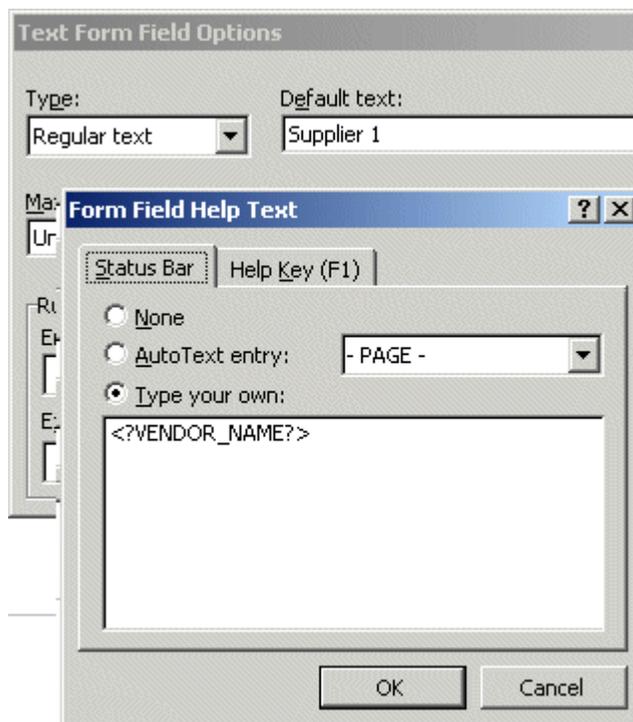
You can enter multiple element tag names in the text entry field.

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

`<?VENDOR_NAME?>`

Figure 4-3 shows the **Text Form Field Options** dialog and the **Form Field Help Text** dialog with the appropriate entries for the Supplier field.

Figure 4-3 Dialogs for 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. Click **OK** to apply.

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

Figure 4–4 shows the Supplier field from the template with the added form field markup.

Figure 4–4 Supplier Field with Added Markup

Supplier: Supplier1

Invoice Num	Inv
Tota	

4.3.1.3 Completing the Example

Table 4–1 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.

Table 4–1 Entries to Complete the Example

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?>

Figure 4–5 shows the Payables Invoice Register with the completed form field placeholder markup. See Figure 4–6 for the completed basic RTF markup.

Figure 4–5 Payables Invoice Register with Form Field Markup

ORACLE® Payables Invoice Register Page 1 of 1
E-Business Suite 25th January 2004

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

4.4 Defining Groups

By defining a group, you notify BI Publisher that *for each* occurrence of an element, you want the included fields displayed. At runtime, BI Publisher loops through the occurrences of the element and displays the fields each time.

In the example, for each occurrence of G_VENDOR_NAME in the XML file, you 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), you 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?>
```

Note: For more information, see [Section 5.4.6, "Inserting a Repeating Group."](#)

4.4.1 Grouping Scenarios

When grouping, 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, then the text and formatting elements between the group tags are repeated.
- If you insert the tags around a table, then the table is repeated.
- If you insert the tags around text in a table cell, then the text in the table cell between the tags is repeated.

- If you insert the tags around two different table cells, but in the same table row, then the single row is repeated.
- If you insert the tags around two different table rows, then the rows between the tags are repeated (this does not include the row that contains the "end group" tag).

4.4.2 Using the Basic RTF Method

Enter the tags in the 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.

Figure 4–6 shows the Payables Invoice Register with the basic RTF grouping and placeholder markup.

Figure 4–6 Payables Invoice Register with Basic RTF Grouping Markup

ORACLE
E-Business Suite

Payables Invoice Register

Page 1 of 1
25th January 2004

<?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

4.4.3 Using the Form Field Method

To use Microsoft Word's Form Field method to defining a group:

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.

Figure 4-7 shows the template after the markup to designate the Suppliers group was added.

Figure 4-7 *Template with Suppliers Group Added*

Group: Suppliers

Supplier: Supplier 1

Invoice Num	Invoice
1234566	1-Jan-

End:Suppliers

4.4.4 Completing 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 repeats only the row. If you place the tags around the table, then for each new invoice the entire table with headings is 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.

Figure 4-8 shows the completed example using the form field method.

Figure 4–8 Payables Invoice Register with Groups


Page 1 of 1
25th July 2003

Payables Invoice Register

Group: Suppliers Sort by Supplier

Supplier: **Supplier 1**

Invoice Num	Invoice Date	GL Date	Curr	Entered Amt	Accounted Amt
Group:Invoices 1234568	1-Jan-2004	1-Jan-2004	USD	1000.00	1000.00
				End:Invoices	
Total for Supplier: Supplier1				1000.00	1000.00

End:Suppliers

Company Confidential

4.5 Defining Headers and Footers

You can define headers and footers as part of the template. This section covers the following topics:

- [Section 4.5.1, "Native Support for Headers and Footers"](#)
- [Section 4.5.2, "Inserting Placeholders in the Headers and Footers"](#)
- [Section 4.5.3, "Creating Multiple or Complex Headers and Footers"](#)
- [Section 4.5.4, "Defining Different First Page and Different Odd and Even Pages"](#)

4.5.1 Native Support for Headers and Footers

BI Publisher supports the use of the native RTF header and footer feature. To create a header or footer, use the word processor'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 the report.

4.5.2 Inserting Placeholders in the Headers and Footers

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.

4.5.3 Creating Multiple or Complex Headers and Footers

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

Use the following tags to enclose the body area of the 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.

To define the body area:

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?>

Figure 4–9 shows the Payables Invoice Register with the start body/end body tags inserted.

Figure 4–9 Payables Invoice Register with Body Tags

ORACLE
E-Business Suite

Payables Invoice Register

Page 1 of 1
25 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				1000.00	1000.00

End:Suppliers
<?end body?>

Company Confidential

4.5.4 Defining Different First Page and Different Odd and Even Pages

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

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

To define different page setups:

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 the headers and footers into the template as desired.

At runtime the generated report exhibits the defined header and footer behavior.

4.6 Inserting Images and Charts

BI Publisher supports several methods for including images in the published document, as described in the following sections:

- [Section 4.6.1, "Directly Inserting Images"](#)
- [Section 4.6.2, "Inserting Images with URL References"](#)
- [Section 4.6.3, "Inserting Images with an Element Reference from an XML File"](#)
- [Section 4.6.4, "Rendering an Image Retrieved from BLOB Data"](#)
- [Section 4.6.5, "Adding Charts to Templates"](#)

4.6.1 Directly Inserting Images

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

4.6.2 Inserting Images with URL References

To insert images with URL references:

1. Insert a dummy image in the 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.example.com/images/ora_log.gif'}
```

4.6.3 Inserting Images with an Element Reference from an XML File

To insert images with element references:

1. Insert a dummy image in the 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 the 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 the 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) }
```

4.6.4 Rendering an Image Retrieved from BLOB Data

If results XML contains image data that had been stored as a BLOB in the database, then use the following syntax in a form field inserted in the 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 the XML data.

Note that you can specify height and width attributes for the image to set its size in the published report. BI Publisher scales 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%">
...
```

4.6.5 Adding Charts to Templates

The following list summarizes the steps to add a chart to the template. These steps are discussed in detail in this section through the use of an example.

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

Note: For more information, see [Section 5.4.4, "Inserting a Chart."](#)

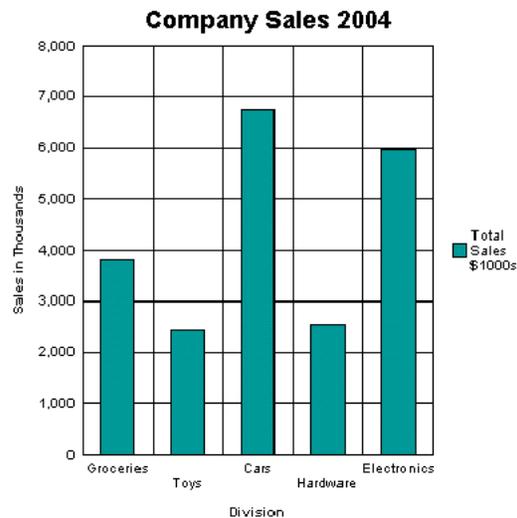
4.6.5.1 Adding a Sample Chart

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 describes how to insert a chart into the template to display it as a vertical bar chart, as shown in [Figure 4-10](#).

Figure 4-10 Bar Chart of Sales Data



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.

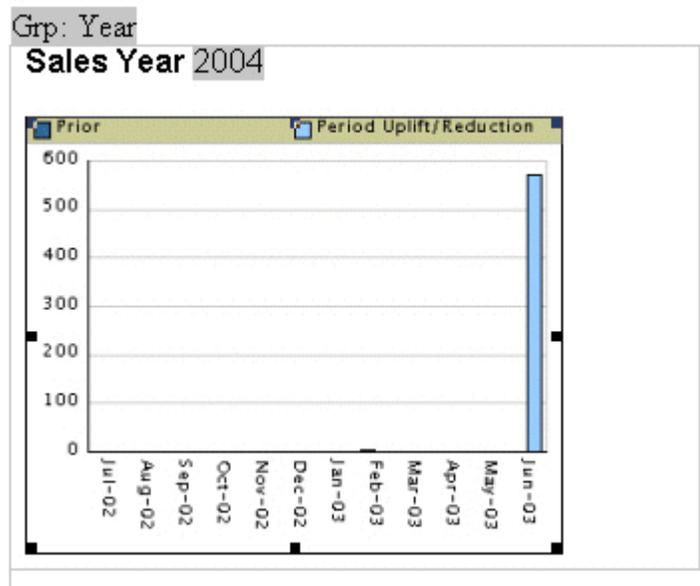
4.6.5.2 Inserting the Dummy Image

The first step is to add a dummy image to the template in the position you want the chart to appear. The image size defines how large the chart image is in the final document.

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

Figure 4–11 shows an example of a dummy image.

Figure 4–11 Sample Dummy Image



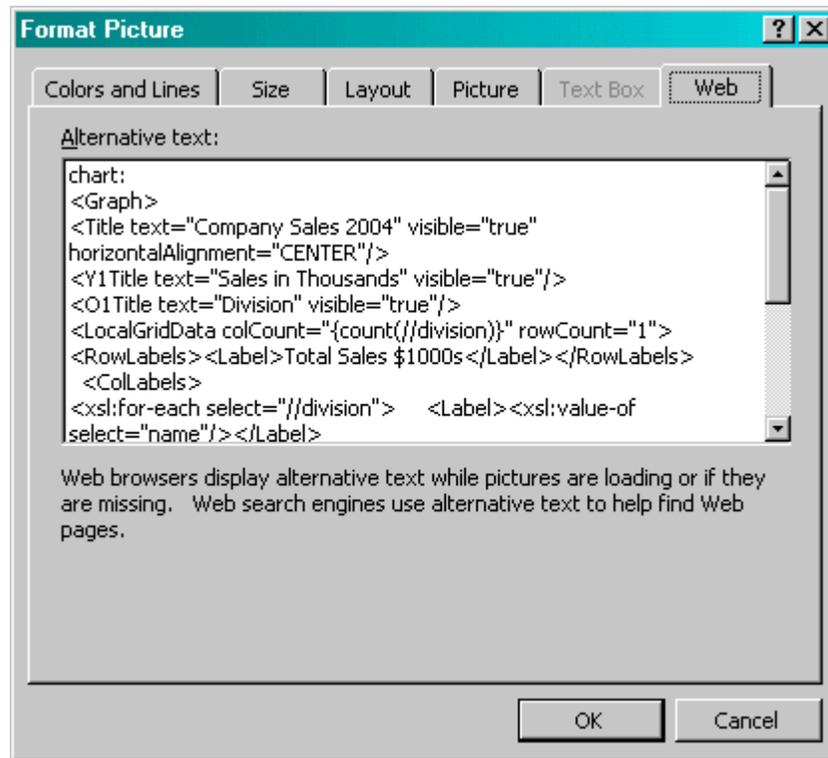
The image can be embedded inside a for-each loop like any other form field if you want the chart to be repeated in the output based on the repeating data. In this example, the chart is defined within the sales year group so that a chart is generated for each year of data present in the XML file.

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.

4.6.5.3 Adding Code to the Alternative Text Box

Figure 4–12 shows an example of the BI Publisher code in the **Format Picture Alternative text** box.

Figure 4–12 Alternate Text Box



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

```

chart:
<Graph graphType = "BAR_VERT_CLUSTER">
<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 the 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"/>
<Y1Title text="Sales in Thousands" visible="true"/>
<O1Title 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 are 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 is 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 code generates 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>
```

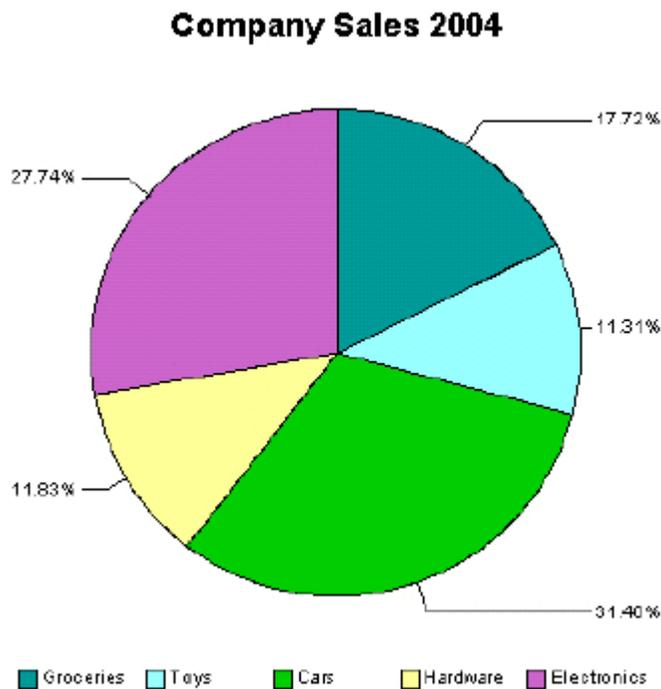
Similar to the labels section, the code loops through the data to build the XML that is passed to the BI Beans rendering engine. This code generates the following XML:

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

4.6.5.4 Additional Chart Samples

You can also display this data in a pie chart as shown in [Figure 4-13](#).

Figure 4–13 Sample Pie Chart



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

```

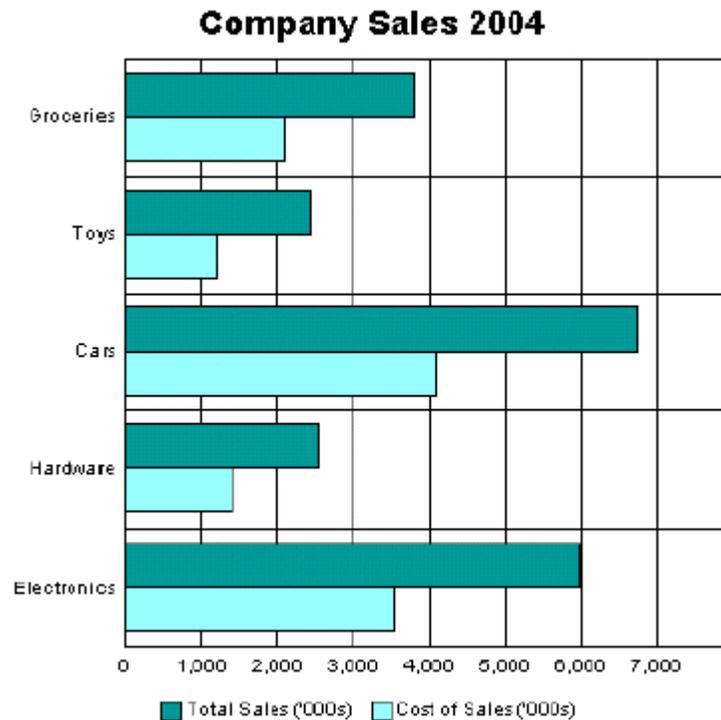
chart:
<Graph graphType="PIE">
  <Title text="Company Sales 2004" visible="true"
    horizontalAlignment="CENTER"/>
  <LocalGridData rowCount="{count(//division)}" colCount="1">
    <RowLabels>
    <xsl:for-each select="//division">
    <Label>
    <xsl:value-of select="name"/>
    </Label>
    </xsl:for-each>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

```

4.6.5.5 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, as shown in [Figure 4–14](#).

Figure 4-14 Sample Horizontal Bar 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>

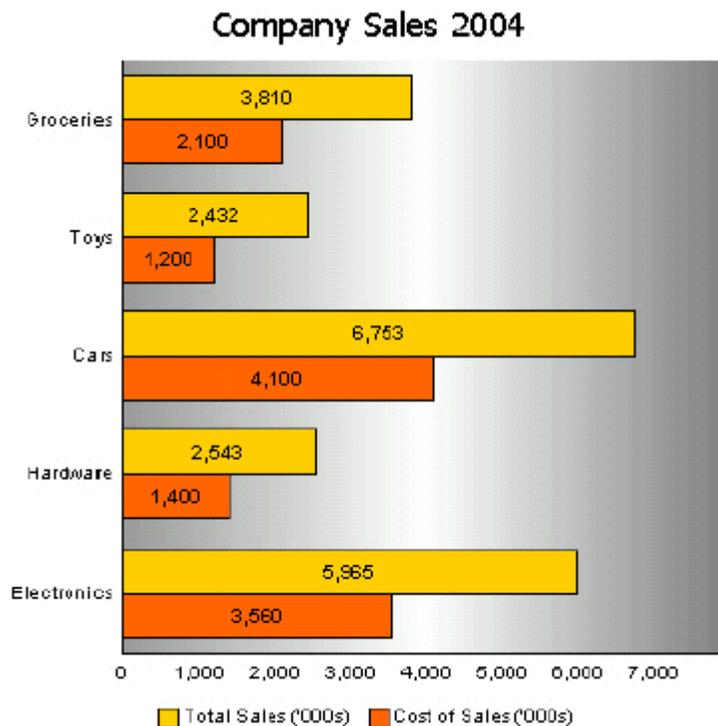
```

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.

4.6.5.6 Changing the Appearance of the Chart

There are many attributes available from the BI Beans graph DTD that you can manipulate to change the look and feel of the 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 [Figure 4–15](#).

Figure 4–15 *Changed Bar Chart*



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>

```

4.7 Adding Drawings, Shapes, and Clip Art

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

The following AutoShape categories are supported:

- Lines - Straight, arrows, 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 the templates using the Microsoft Clip Art libraries

4.7.1 Adding Freehand Drawings

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

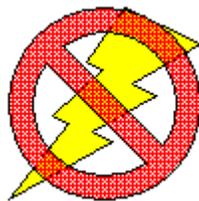
4.7.2 Adding Hyperlinks

You can add hyperlinks to shapes. See [Section 4.9.5, "Inserting Hyperlinks."](#)

4.7.3 Layering Shapes

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. [Figure 4-16](#) shows an example of layered shapes.

Figure 4-16 Layered Shapes



4.7.4 Using 3-D Effects

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

4.7.5 Adding Microsoft Equations

Use the equation editor to generate equations in the output. [Figure 4-17](#) shows an example of an equation.

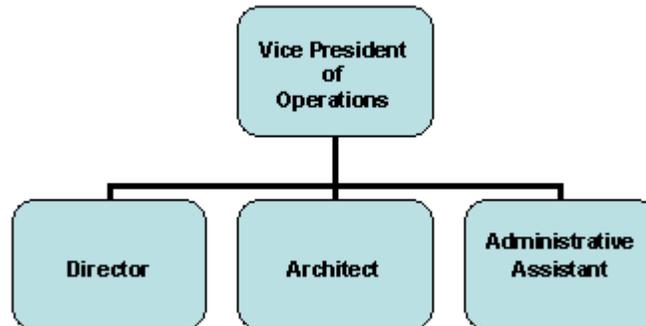
Figure 4-17 Sample Equation

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

4.7.6 Adding Organization Charts

Use the organization chart functionality in the templates and the chart that is rendered in the output. [Figure 4–18](#) shows an example of an organization chart.

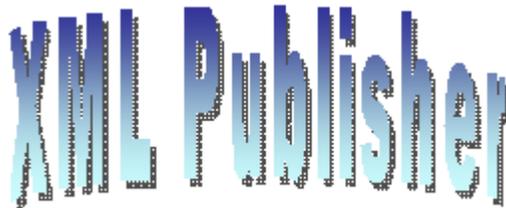
Figure 4–18 Sample Organization Chart



4.7.7 Adding WordArt

You can use Microsoft Word's WordArt functionality in the templates. [Figure 4–19](#) shows a WordArt example.

Figure 4–19 Sample WordArt Example



Note: Some Microsoft WordArt uses a bitmap operation that currently cannot be converted to SVG. To use the unsupported WordArt in the 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.

4.7.8 Adding Data-Driven Shapes

In addition to supporting the static shapes and features in the 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

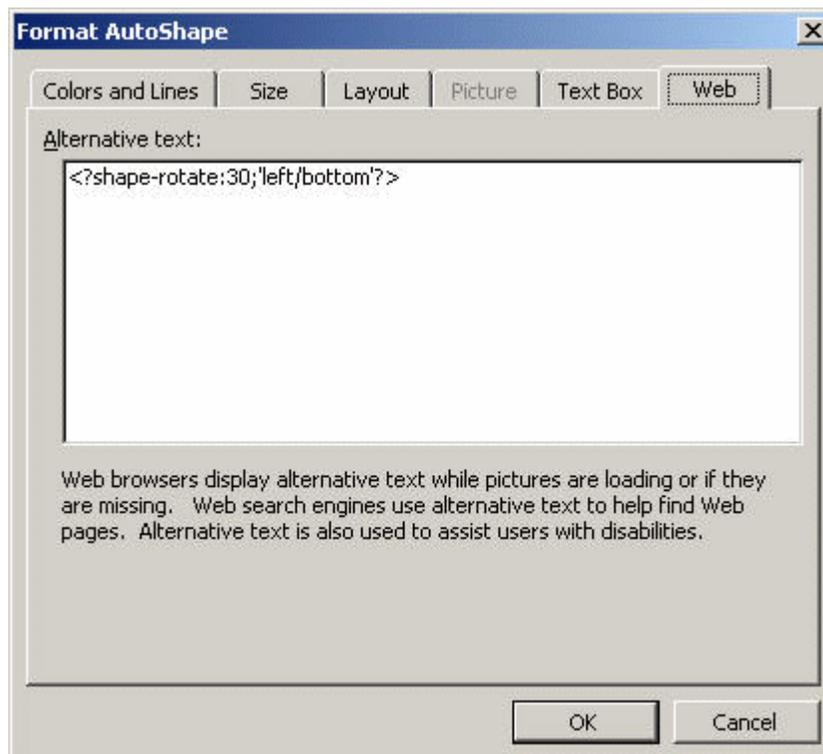
- Skew
- Rotate

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.

4.7.9 Including Manipulation Commands

Enter manipulation commands for a shape in the Web tab of the shape's properties dialog as shown in [Figure 4-20](#).

Figure 4-20 *Format AutoShape Dialog*



4.7.10 Replicating Shapes

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 is 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.

4.7.11 Adding Text to Shapes

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 is inserted into the shape.

4.7.12 Adding 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 is inserted above and along the line.

4.7.13 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?>
```

4.7.14 Rotating a Shape

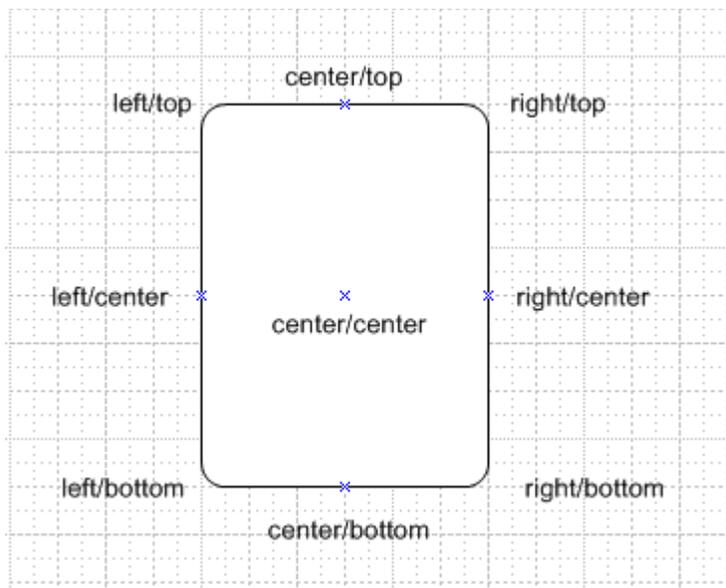
To rotate a shape about a specified axis based on the incoming data, use the following command:

```
<?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. [Figure 4-21](#) shows these valid values.

Figure 4–21 Shape Rotation 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.

4.7.15 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 [Figure 4–21](#). 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'?>
```

4.7.16 Changing the Size of Shapes

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.

4.7.17 Combining Commands

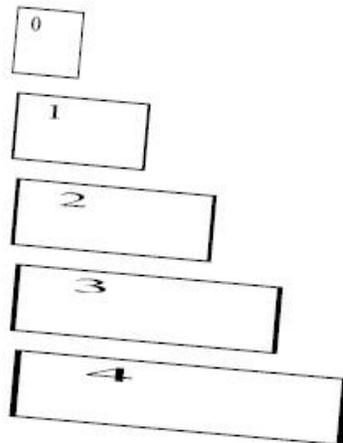
You can also combine these commands to carry out multiple transformations on a shape at one time. For example, you can replicate a shape and for each replication, rotate it by some angle and change the size at the same time.

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?>
```

These commands generate the output shown in [Figure 4-22](#).

Figure 4-22 *Shape Transformation*



4.7.18 CD Ratings Example

This example demonstrates how to set up a template that generates 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, you 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 [Figure 4-23](#).

Figure 4-23 Visual Representation of Ratings

Title	Artist	Rating
F TITLE	ARTIST	E 

The values for the fields are shown in [Table 4-2](#).

Table 4-2 Values for Fields

Field	Form Field Entry
F	<?for-each:CD?>
TITLE	<?TITLE?>
ARTIST	<?ARTIST?>
E	<?end for-each?>

Table 4–2 (Cont.) Values for Fields

Field	Form Field Entry
(star shape)	Web Tab Entry: <pre><?for-each@shape:xdoxslt:foreach_number(\$_XDOCTX,1,USER_RATING,1)?> <?shape-offset-x:(position()-1)*25?> <?end for-each?></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 you can use a command to create a "for...next loop" construct. 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, it creates an inner loop to repeat a star shape for every USER_RATING value (that is, a value of 4 generates 4 stars). The output from this template and the XML sample is shown in [Figure 4–24](#).

Figure 4–24 Ratings Data from XML File

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

4.7.19 Grouped Shape Example

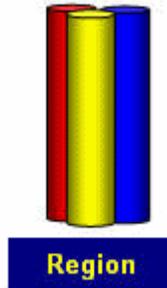
This example shows how to combine shapes into a group and have them react to the incoming data both individually and as a group. Assume the following XML data:

```
<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 want to manipulate. [Figure 4–25](#) shows a composite shape comprised of four components.

Figure 4–25 Composite Shape



The shape consists of three cylinders: red, yellow, and blue. These 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 generates as twice its current height.

The text-enabled rectangle contains the following command in its Web tab:

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

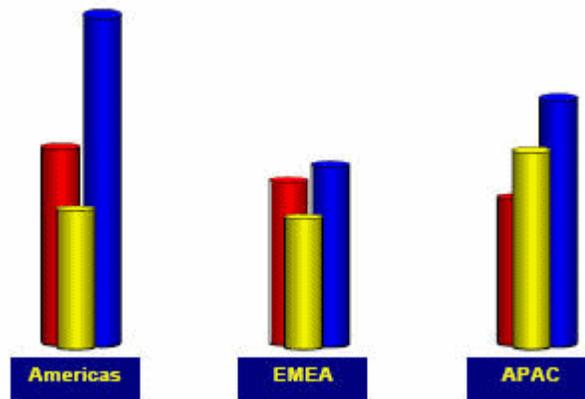
At runtime the value of the `REGION` element is displayed in the rectangle.

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?>
```

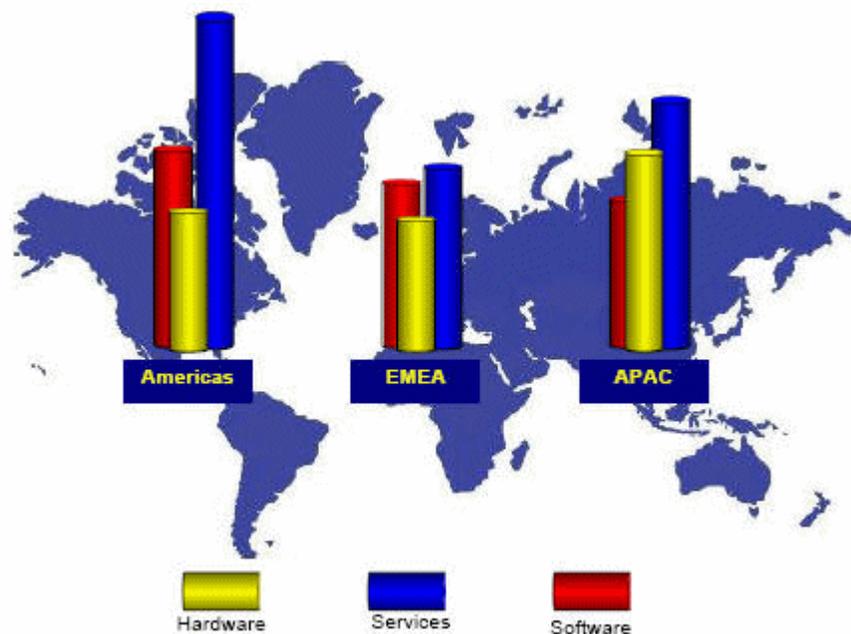
In this set of commands, the `for-each@shape` loops over the `SALE` group. The `shape-offset` command moves the next shape in the loop to the right by a specific number of pixels. The expression `(position()-1)` sets the position of the object. The `position()` function returns a record counter while 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 are rendered across the page, as shown in [Figure 4–26](#).

Figure 4–26 *Rendering Sets of Composite Shapes*

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 autoshapes". Using this option removes the need to do the final grouping of the map and shapes. You can now generate a visually appealing output for the report as seen in [Figure 4-27](#).

Figure 4–27 *Shapes on World Map*

4.8 Supported Formatting Features of Microsoft Word

In addition to the features already listed, BI Publisher supports the features of Microsoft Word that are described in the following sections:

- [Section 4.8.1, "General Features of Microsoft Word"](#)
- [Section 4.8.2, "Aligning Objects"](#)
- [Section 4.8.3, "Inserting Tables"](#)
- [Section 4.8.4, "Inserting Date Fields"](#)
- [Section 4.8.5, "Inserting Multiple Columns on Pages"](#)
- [Section 4.8.6, "Inserting Backgrounds and Watermarks"](#)
- [Section 4.8.7, "Microsoft Word Features That Are Not Supported"](#)

4.8.1 General Features of Microsoft Word

- Large blocks of text
- Page breaks

(Not supported for HTML output) To insert a page break, press Ctrl+Enter right 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 starts on a new page.

Using this Microsoft Word native feature causes a single blank page to print at the end of the report output. To avoid this single blank page, use BI Publisher's page break alias. See [Section 4.9.1, "Inserting Page Breaks."](#)

- Page numbering

Insert page numbers into the final report by using the page numbering methods of the word processor. 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 are 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 is maintained in RTF output reports.

4.8.2 Aligning Objects

Use the word processor's alignment features to align text, graphics, objects, and tables. Bidirectional languages are handled automatically using the word processor's left/right alignment controls.

4.8.3 Inserting Tables

The following Microsoft Word features are supported in BI Publisher:

- Nested Tables
- Cell Alignment

You can align any object in the template using the word processor's alignment tools. This alignment is reflected in the final report output.

- Row spanning and column spanning

To span both columns and rows in the template:

1. Select the cells that you want 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 appear merged.

- Table Autoformatting

BI Publisher recognizes the table autoformats available in Microsoft Word.

To autoformat tables:

1. Select the table that you want to format.
2. From the **Table** menu, select **Autoformat**.
3. Select the desired table format.

At runtime, the table is formatted using your selection.

- Cell patterns and colors

To 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 the data is displayed in a table and you expect the table to extend across multiple pages, then you can define the header rows that you want to repeat at the start of each page.

To repeat header rows:

1. Select the row(s) that you want 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**.

To keep a row's contents together on one page:

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 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.
 - Text truncation

By default, if the text within a table cell does not fit within the cell, then the text is wrapped. To truncate the text instead, use the table properties dialog.

Note that table text truncation is supported for PDF and PPT outputs only.

To truncate the text within a table cell:

 1. Place the cursor in the cell in which you want the text truncated.
 2. Right-click 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.

An example of truncation is shown in [Figure 4–28](#).

Figure 4–28 *Truncated Text*

Wrap Text checked	Wrap Text unchecked
The quick brown fox jumped over the lazy river.	The quick brown fox

Note: When using multibyte characters (for example, simplified Chinese) in tables, ensure that the column widths are large enough to contain the width of the largest character plus the cell's left and right margins to avoid unexpected character display in your final output.

4.8.4 Inserting Date Fields

Insert dates using the date feature of the word processor. Note that this date corresponds to the publishing date, not to the request run date.

4.8.5 Inserting Multiple Columns on Pages

BI Publisher supports Microsoft Word's Columns function to enable you to publish the output in multiple columns on a page. (Note that this is not supported for HTML output.)

Select **Format**, then **Columns** to display the **Columns** dialog to define the number of columns for the template.

To generate address labels in a two-column format:

1. Divide the 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 [Figure 4–29](#).

Figure 4–29 Repeatable Groups in First Column



Name	CUSTOMER_NAME
Number	CUSTOMER_NUMBER
City	CITY
State	STATE
Zip Code	Z I P _ C O D 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 [Section 4.8.3, "Inserting Tables."](#)

This template produces the multicolumn output that is shown in [Figure 4–30](#).

Figure 4–30 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																					

4.8.6 Inserting Backgrounds and Watermarks

BI Publisher supports the "Background" feature in Microsoft Word. You can specify a single, graduated color or an image background for the 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 the template, use the Format > Background menu option.

4.8.6.1 Adding 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.

From this dialog, select one of the following supported options:

- Gradient: This can be either one or two colors
- Texture: Select 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

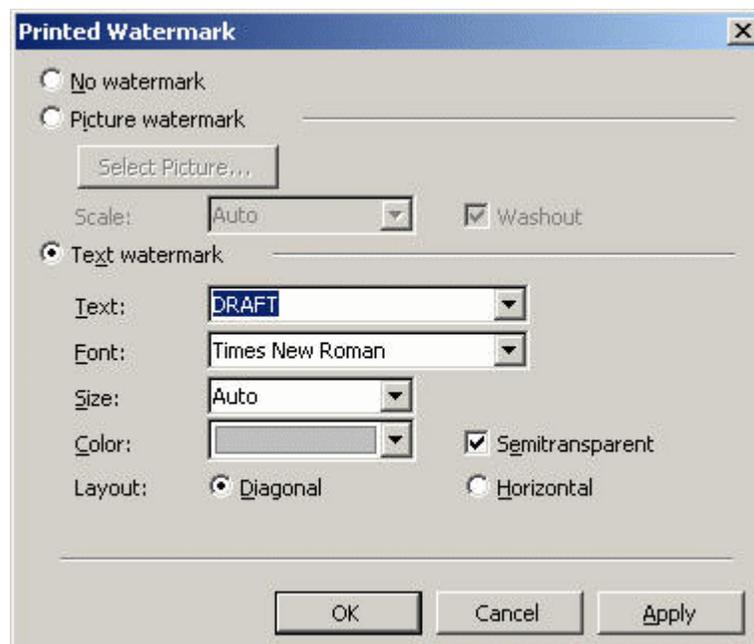
4.8.6.2 Adding a Text or Image Watermark Using Microsoft Word 2002 or Later

These versions of Microsoft Word allow you to add either a text or image watermark.

From the Format menu, select **Background**, then **Printed Watermark** to select the following in the dialog:

- 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.

[Figure 4–31](#) shows the Printed Watermark dialog completed to display a text watermark:

Figure 4–31 Printed Watermark Dialog

4.8.7 Microsoft Word Features That Are Not Supported

Do not use soft returns in your RTF template to achieve specific text placement. Instead use hard carriage returns. A soft return may have unexpected results in your generated output.

4.9 Template Features

BI Publisher supports the template features that are described in the following sections:

- [Section 4.9.1, "Inserting Page Breaks"](#)
- [Section 4.9.2, "Inserting an Initial Page Number"](#)
- [Section 4.9.3, "Specifying Last Page Only Content"](#)
- [Section 4.9.4, "Ending on Even or Odd Pages"](#)
- [Section 4.9.5, "Inserting Hyperlinks"](#)
- [Section 4.9.6, "Including a Table of Contents"](#)
- [Section 4.9.7, "Generating Bookmarks in PDF Output"](#)
- [Section 4.9.8, "Inserting Check Boxes"](#)
- [Section 4.9.9, "Inserting Drop-Down Lists"](#)

4.9.1 Inserting 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 causes 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:?>
```

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 [Figure 4-32](#), the field called PageBreak contains the split-by-page-break syntax:

Figure 4-32 Inserting Page Breaks

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 ensures that 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.

4.9.2 Inserting an 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 the 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 the report requires that the page numbering continue into the next section, use the following command:

```
<?initial-page-number:'auto'?>
```

This command allows the continuation of the page numbering from the previous section.

Example 1 - Set page number from XML data element

If the XML data contains an element to carry the initial page number, for example:

```
<REPORT>
  <PAGESTART>200<\PAGESTART>
  . . .
</REPORT>
```

Enter the following in the template:

```
<?initial-page-number:PAGESTART?>
```

The initial page number is 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, then you can pass the initial value by calling the parameter.

Enter the following in the template:

```
<?initial-page-number:$PAGESTART?>
```

Note: You must first declare the parameter in the template. See [Section 4.13.2, "Setting Parameters."](#)

4.9.3 Specifying 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 recognizes the settings that 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 specify last page only content:

1. Create a section break in the 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 is displayed 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.

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>
  </VENDOR>
  <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>
</INVOICELIST>
```

```

<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 [Figure 4-33](#).

Figure 4-33 *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

```

<<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 [Figure 4-34](#).

Figure 4-34 *Last Page Only Layout*

```

Last Page Placeholder
    
```

Tax Summary

Tax Code	Entered Amount	Accounted Amount
F VAT 18.5	100	100E

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 is displayed 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 the 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, see [Section 4.5, "Defining Headers and Footers."](#)

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 is reset. To continue the page numbering from the previous section, use the following command:

```
<?initial-page-number:'auto'?>
```

This command allows 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 is used. If the 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 results in the last page layout for reports that are only one page long.

4.9.4 Ending on Even or Odd Pages

Note: This feature is supported for PDF and PDF output only. It is not supported for RTF and HTML output.

If the report has different odd and even page layouts, then you might want to force the 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 the report using the different odd/even footer functionality (see [Section 4.5.4, "Defining Different First Page and Different Odd and Even Pages"](#)) and you want to ensure that the terms and conditions are printed on the final page.

Or, you may have binding requirements to have the report end on an even page, without specific layout.

To end on an even page with layout:

1. Insert the following syntax in a form field in the template:

```
<?section:force-page-count;'end-on-even-layout'?>
```

To end on an odd page layout:

1. Insert the following syntax in a form field in the template:

```
<?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, then use the following syntax:

```
<?section:force-page-count;'end-on-even'??>
```

or

```
<?section:force-page-count;'end-on-odd'??>
```

4.9.5 Inserting Hyperlinks

Note: Hyperlinks are supported for PDF, RTE, 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.

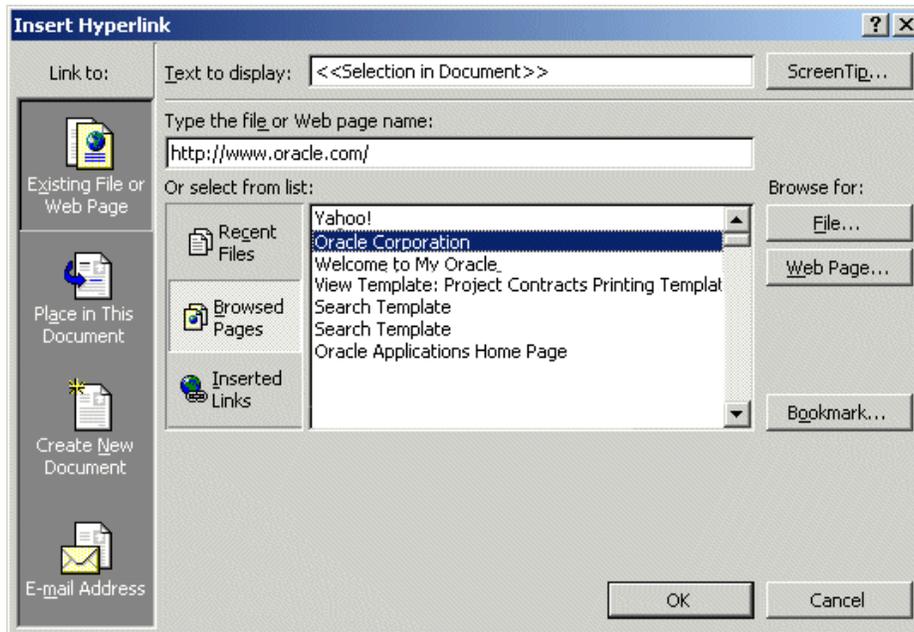
- To insert static hyperlinks to either text or a shape, use the word processor's insert hyperlink feature.

To insert a static hyperlink to a text or a shape:

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.

[Figure 4-35](#) shows the insertion of a static hyperlink using Microsoft Word's **Insert Hyperlink** dialog.

Figure 4–35 Static Hyperlink



- If the input XML data includes an element that contains a hyperlink or part of one, then you can create dynamic hyperlinks at runtime. In the **Type the file or Web page name** field of the **Insert Hyperlink** dialog, 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 the XML data file to construct the URL, enter the following syntax:

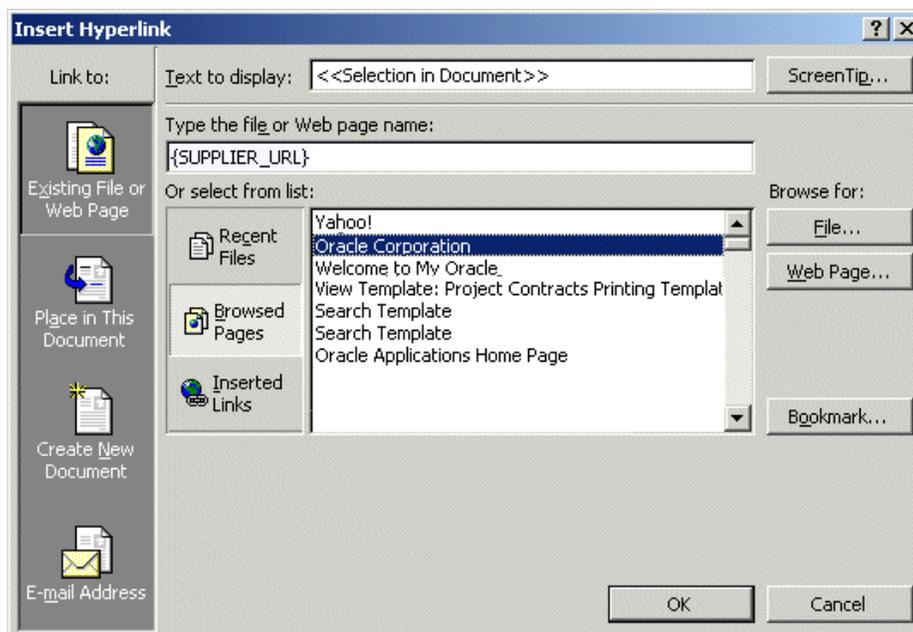
http://www.example.com?product={PRODUCT_NAME}

where PRODUCT_NAME is the incoming data element name.

In both these cases, at runtime the dynamic URL is constructed.

Figure 4–36 shows the insertion of a dynamic hyperlink using Microsoft Word's **Insert Hyperlink** dialog. The data element SUPPLIER_URL from the incoming XML file contains the hyperlink that is inserted into the report at runtime.

Figure 4–36 Dynamic Hyperlink



- 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/CustomerReport/cstid=1234
```

To add the target attribute to a URL, add the following to the URL string:

```
??target=_target_value
```

For example:

```
http://www.example.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.example.com/index.html??target={myTarget}
```

where `myTarget` is the name of the parameter that holds the value.

4.9.5.1 Inserting Internal Links

Insert internal links into the template using Microsoft Word's Bookmark feature.

To insert internal links:

1. Position the cursor in the desired destination in the document.
2. From the **Insert** menu, select **Bookmark**.
3. In the Bookmark dialog, enter a name for this bookmark, and select **Add**.
4. Select the text or shape in the 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. Select the bookmark that you created from the list.

At runtime, the link is maintained in the generated report.

4.9.6 Including a 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 the word processor's procedures for inserting a table of contents.

BI Publisher also provides the ability to create dynamic section headings in the 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 the word processor's style feature. You cannot use form fields for this functionality.

For example, you want the report to display a heading for each company reported. The XML data element tag name is <COMPANY_NAME>. In the 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 the word processor's table of contents feature.

At runtime, the TOC placeholders and heading text are substituted.

4.9.7 Generating Bookmarks in PDF Output

If you have defined a table of contents in the RTF template, then you can use the table of contents definition to generate links in the Bookmarks tab in the navigation pane of the output PDF. The bookmarks can be either static or dynamically generated.

Note: 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 [Section 4.9.6, "Including a Table of Contents."](#)

- To create links for a static table of contents:

Enter the syntax:

```
<?copy-to-bookmark:??>
```

directly above the 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 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 the 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.

4.9.8 Inserting Check Boxes

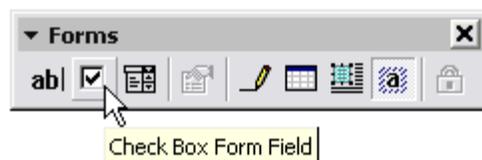
Note: Check boxes are supported in PDF output only.

You can include a check box in the 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 the template:

1. Position the cursor in the template where you want the check box to display, and select the Check Box Form Field from the Forms tool bar, as shown in [Figure 4-37](#).

Figure 4-37 Check Box Form Field



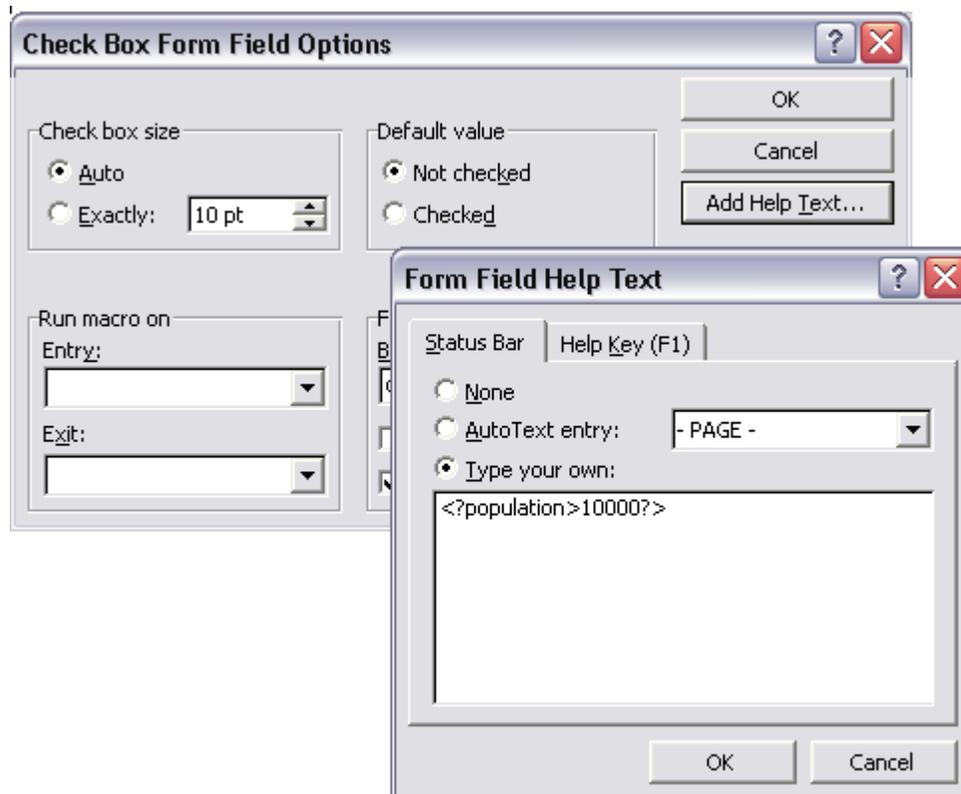
2. Right-click the field to open the **Check Box Form Field Options** dialog.
3. Specify the **Default value** as either Checked or Not Checked.
4. In the Form Field Help Text dialog, enter the criteria for how the box should behave. This must be a boolean expression (that is, one that returns a true or false result).

For example, suppose the 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?>
```

The help text coding is shown in [Figure 4-38](#).

Figure 4–38 Sample Help Text Coding



Note that you do not have to construct an "if" statement. The expression is treated as an "if" statement.

See [Section 4.9.9, "Inserting Drop-Down Lists"](#) for a sample template using a check box.

4.9.9 Inserting Drop-Down Lists

BI Publisher allows you to use the drop-down form field to create a cross-reference in the template from the 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>
```

```

. . .
</countries>

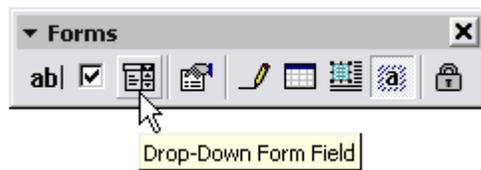
```

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 the template that cross-references the <continentindex> value to the actual continent name. You can then display the name in the published report.

To create the index for the continent example:

1. Position the cursor in the 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, as shown in [Figure 4–39](#).

Figure 4–39 Drop-Down Form Field



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 are indexed starting from one for the first, and so on. For example, the list of continents is stored as shown in [Table 4–3](#).

Table 4–3 Indexed Values for Continents

Index	Value
1	Asia
2	North America
3	South America
4	Europe
5	Africa
6	Australia

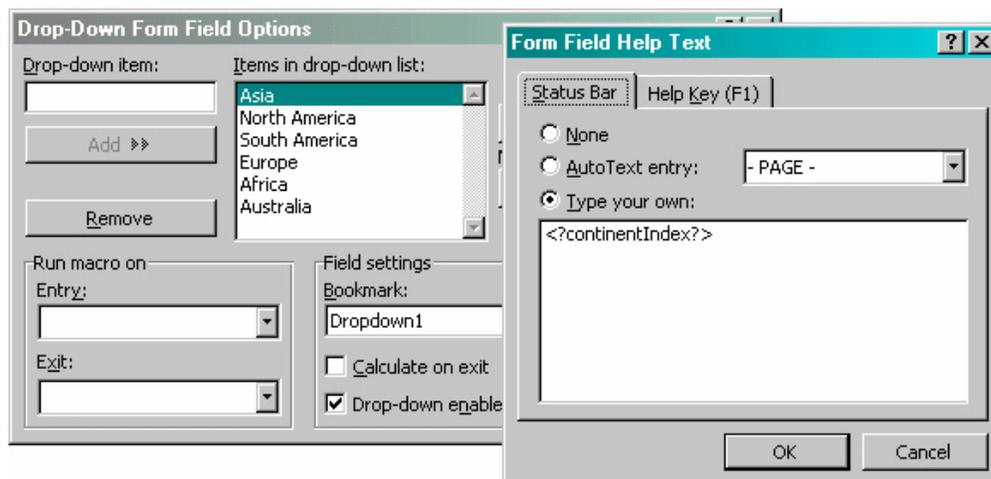
4. Now use the Help Text box to enter the XML element name that holds the index for the drop-down field values.

For this example, enter

```
<?continentIndex?>
```

[Figure 4–40](#) shows the **Drop-Down Form Field Options** dialogs for this example:

Figure 4–40 Drop-Down Form Field Options



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 [Figure 4–41](#).

Figure 4–41 Report of Population Data with Check Boxes

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 [Figure 4–42](#) and the fields have the values shown in [Table 4–4](#).

Figure 4–42 Template for Population Data Report

Country	Population	more than 10M?	Continent
FE China	1,000,000	<input type="checkbox"/>	Asia EFE

Table 4–4 Fields for Drop-Down List

Field	Form Field Entry	Description
FE	<?for-each:country?>	Begins the country repeating group.
China	<?name?>	Placeholder for the name element.
1,000,000	<?population?>	Placeholder for the population element.
(check box)	<?population>1000000?>	Establishes the condition for the check box. If the value for the population element is greater than 1,000,000, the check box is displayed as checked.
Asia	<?continentIndex?>	The drop-down form field for the continentIndex 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.

Table 4–4 (Cont.) Fields for Drop-Down List

Field	Form Field Entry	Description
EFE	<?end for-each?>	Ends the country group.

4.10 Using Conditional Formatting

Note: For information about using the Template Builder to insert conditional regions and conditional formatting, see [Section 5.4.8, "Inserting and Editing Conditional Regions"](#) and [Section 5.4.9, "Inserting Conditional Formatting."](#)

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 are displayed shaded in red. Or, you can use this feature to hide table columns or rows depending on the incoming XML data.

This section covers the following topics of conditional formatting:

- [Section 4.10.1, "Using If Statements"](#)
- [Section 4.10.2, "Using If Statements in Boilerplate Text"](#)
- [Section 4.10.3, "Using If-Then-Else Statements"](#)
- [Section 4.10.4, "Inserting Choose Statements"](#)
- [Section 4.10.5, "Formatting Columns"](#)
- [Section 4.10.6, "Formatting Rows"](#)
- [Section 4.10.7, "Highlighting Cells"](#)

4.10.1 Using If Statements

Use an if statement to define a simple condition; for example, if a data field is a specific value.

To use an if statement:

1. Insert the following syntax to designate the beginning of the conditional area.

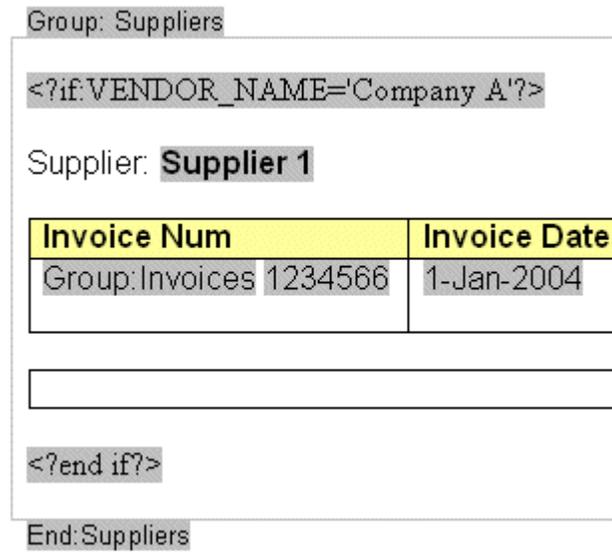
```
<?if:condition?>
```

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 [Figure 4–43](#). Note that you can insert the syntax in form fields, or directly into the template.

Figure 4–43 Sample IF Statement

4.10.2 Using If Statements in Boilerplate Text

Assume that you want to incorporate an "if" statement into the following free-form text:

The program was (not) successful.

You want the "not" to display only if the value of an XML tag called <SUCCESS> equals "N".

To achieve this requirement, 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 [Section 4.18, "Controlling the Placement of Instructions Using the Context Commands."](#)

For example, if you construct the code as follows:

The program was <?if:SUCCESS='N'?>not<?end if?> successful.

The following undesirable result occurs:

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 results in the following display:

The program was successful.

If SUCCESS does not equal 'N';

or

The program was not successful.

If SUCCESS equals 'N'.

Note: If you use `@inlines` with `if` syntax, any other `if` syntax inside the statement must use the context command `@inline`.

If you use `@inlines` with `FOR-EACH` syntax any other `if` or `FOR-EACH` syntax inside the statement must use the context command `@inline`.

4.10.3 Using If-Then-Else Statements

BI Publisher supports the common programming construct "if-then-else". This is extremely useful when you must 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 the RTF template:

```
<?xdofx: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":

```
<?xdofx:if AMOUNT > 1000 then 'Higher'
else
  if AMOUNT < 1000 then 'Lower'
  else
    'Equal'
end if?>
```

4.10.4 Inserting 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 are 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?>

4.10.4.1 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 renders light gray. When POSTED_FLAG equals "*" the row of data renders shaded dark gray. Otherwise, the row of data renders with no shading.

In [Figure 4-44](#), the form field default text is displayed. The form field help text entries are shown in [Table 4-5](#).

Figure 4-44 Conditional Formatting Example

Column Legend:						
	'Not Posted'					
	'Reduces Available Exemption Limit'					

	Tax Code	Taxable Recoverable	Taxable Non-Recoverable	Recoverable Tax	Tax Non-Recoverable	Total
<Grp:VAT						
<Choose						
<When EXEMPT_FLAG='^'	VAT 15%	1000	1000	1000	1000	1000
End When>						
<When POSTED_FLAG='*'	VAT 15%	1000	1000	1000	1000	1000
End When>						
Otherwise	VAT 15%	1000	1000	1000	1000	1000
End Otherwise>						
End Choose>						
End VAT>						

Table 4-5 Form Field Default Text

Default Text Entry in Example Form Field	Help Text Entry in Form Field
<Grp:VAT	<?for-each:G_VAT?> starts the G_VAT group
<Choose	<?choose:?> opens the choose statement
<When EXEMPT_FLAG='^'	<?when: EXEMPT_FLAG='^'?> tests the EXEMPT_FLAG element, if true, use the first table shown
End When>	<?end when?> ends the EXEMPT_FLAG test
<When POSTED_FLAG='*'	<?when:POSTED_FLAG='*'?> tests the POSTED_FLAG element, if true, use the table following
End When>	<?end when?> ends the POSTED_FLAG test
Otherwise	<?otherwise:?> If none of above are true then use the following table
End Otherwise>	<?end otherwise?> ends the otherwise statement

Table 4–5 (Cont.) Form Field Default Text

Default Text Entry in Example Form Field	Help Text Entry in Form Field
End Choose>	<?end choose?> ends the choose statement
End Vat>	<?end for-each?> ends the G_VAT group

4.10.5 Formatting Columns

You can conditionally show and hide columns of data in the 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 shows 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, the quantity column should not be shown in the output, but you want to develop only one template for both versions based on the list type.

Figure 4–45 contains a simple template that conditionally shows or hides the quantity column.

Figure 4–45 Showing or Hiding the Quantity Column

Name	IFQuantityend-if	Price
grp:ItemPlasma TV	IF 20 end-if	1,000.00end grp

Table 4–6 shows the entries made in the template that is shown in Figure 4–45.

Table 4–6 Template Entries

Default Text	Form Field Entry	Description
grp:Item	<?for-each:item?>	Holds the opening for-each loop for the item element.

Table 4–6 (Cont.) Template Entries

Default Text	Form Field Entry	Description
Plasma TV	<?name?>	The placeholder for the name 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 templates, see Section 4.19, "Using XPath Commands."
Quantity	N/A	Boilerplate heading
end-if	<?end if?>	Ends the if statement.
IF	<?if@cell:/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 price element.
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 the report as a result of the if statement, but the column still would because you had drawn it in the template.

Note: The @column clause is an example of a context command. For more information, see [Section 4.18, "Controlling the Placement of Instructions Using the Context Commands."](#)

The example renders the output that is shown in [Figure 4–46](#).

Figure 4–46 Sample Price Report

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", then the output that is shown in [Figure 4–47](#) is rendered from the same template.

Figure 4–47 Sample Quantity and Price Report

Name	Quantity	Price
Plasma TV	10	4,000.00
DVD Player	3	300.00
VCR	20	200.00
Receiver	22	350.00

4.10.6 Formatting Rows

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. These tags are demonstrated in the sample template that is shown in [Figure 4–48](#).

Figure 4–48 Template for Conditionally Displaying Rows

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>

[Table 4–7](#) describes the fields from the template in [Figure 4–48](#).

Table 4–7 Form Fields for Conditionally Displaying Rows

Default Text Entry	Form Field Help Text	Description
for-each SALE	<code><?for-each:SALE?></code>	Opens the for-each loop to repeat the data belonging to the SALE group.
if big	<code><?if:SALES>5000?></code>	If statement to display the row only if the element SALES has a value greater than 5000.
INDUSTRY	<code><?INDUSTRY?></code>	Data field
YEAR	<code><?YEAR?></code>	Data field
MONTH	<code><?MONTH?></code>	Data field
SALES end if	<code><?end if?></code>	Closes the if statement.
end SALE	<code><?end for-each?></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 [Figure 4–49](#).

Figure 4–49 Template for Conditionally Highlighting Rows

Industry	Year	Month	Sales
for-each SALE format; INDUSTRY	YEAR	MONTH	SALES end SALE

Table 4–8 shows values of the form fields from the template in Figure 4–49.

Table 4–8 Form Fields to Highlight Rows

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 Table 4–8, 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 is set to light gray. This setting results in the output that is shown in Figure 4–50.

Figure 4–50 Industry Sales Report

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 [Section 4.18, "Controlling the Placement of Instructions Using the Context Commands."](#)

4.10.7 Highlighting Cells

The following example demonstrates how to conditionally highlight a cell based on a value in the XML file.

This example uses the following XML code:

```
<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. The final report will highlight in red any cell whose value is greater than 1000. The template for this is shown in figure [Figure 4-51](#):

Figure 4-51 Sample Template to Conditionally Highlight Cells

Account	Debit	Credit
FE:Account 1-232-4444	CH1 100.00	CH2 100.00 EFE

The field definitions for the template are shown in [Table 4-9](#).

Table 4-9 Form Fields for Highlighting Cells

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.
CH2	<?if:credit>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 credit amount is greater than 1000.
100.00	<?credit?>	The placeholder for the credit element.
EFE	<?end for-each?>	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).

This template results in the output that is shown in [Figure 4-52](#).

Figure 4-52 Debit and Credit Report

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

4.11 Inserting Page-Level Calculations

BI Publisher supports the page-level calculations that are described in the following sections for PDF and PPT outputs only:

- [Section 4.11.1, "Displaying Page Totals"](#)
- [Section 4.11.2, "Inserting Brought Forward and Carried Forward Totals"](#)
- [Section 4.11.3, "Inserting Running Totals"](#)

4.11.1 Displaying Page Totals

BI Publisher allows you to display calculated page totals in the 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. This page totaling function works only if the source XML code 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 the 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 the 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 want to use to for the display, using the Oracle format mask (for example: 'C9G999D00'). For the list of Oracle format mask symbols, see [Section 4.15.12, "Oracle Abstract Format Masks."](#)

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 code:

```
<balance_sheet>
  <transaction>
    <debit>100</debit>
    <credit>90</credit>
  </transaction>
  <transaction>
    <debit>110</debit>
    <credit>80</credit>
  </transaction>
  ...
</balance_sheet>
```

[Figure 4–53](#) shows the table to insert in the template to hold the values.

Figure 4–53 Table for Inserting Values

Debit	Credit
FE 100.00	90.00 Net EFE

[Table 4–10](#) shows the form field entries made in the template whose table is shown in [Figure 4–53](#).

Table 4–10 Fields for Displaying Totals

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. 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. 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" you perform 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 the 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 [Figure 4–54](#).

Figure 4–54 Debit and Credit Report Totals

	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

4.11.2 Inserting Brought Forward and 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 and 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 such as HTML, RTF, and Excel.

An example is displayed in the [Figure 4–55](#).

Figure 4–55 Report of Forward Totals

Page 1			Page 2			Page 3		
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			Brought Forward: 300 Carried Forward: 600			Brought 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 [Displaying Page Totals](#) feature. The following example walks through the syntax and setup required to display the brought forward and carried forward totals in the published report.

Assume that you have the following XML code:

```
<?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 sample template that is shown in [Figure 4-56](#) creates the invoice table and declares a placeholder that holds the page total.

Figure 4-56 *Template for Totals*

Init PTs

Invoice	Date	Amount
FE 132342	10-May-2005	1,000.00 InvAmt EG

End PTs

[Table 4-11](#) shows the fields in the template that is shown in [Figure 4-56](#).

Table 4-11 *Fields for Totals*

Field	Form Field Help Text Entry	Description
Init PTs	<?init-page-total: InvAmt?>	Declares "InvAmt" as the placeholder that holds 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;INV AMT?>	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 list describes the elements that comprise the brought forward syntax:

- **inline-total** - This element has two properties:
 - name - Specifies the name of the variable you declared for the field.
 - display-condition - Sets the display condition. This is an optional property that takes one of the following values:
 - * first - Contents are displayed only on the first page.
 - * last - Contents are displayed only on the last page.
 - * exceptfirst - Contents are displayed on all pages except first.
 - * exceptlast - Contents are displayed on all pages except last.
 - * everytime - (Default) Contents are displayed on every page.

In this example, display-condition is set to "exceptfirst" to prevent the value from appearing on the first page where the value would be zero.
- **Brought Forward:** - This string is optional and is displayed as the field name on the report.
- **show-brought-forward** - Shows the value on the page. It has the following properties:
 - name - The name of the field to show. In this case, "InvAmt". This property is mandatory.
 - format - 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 [Section 4.15.12, "Oracle Abstract Format Masks."](#)
 - number-separators="{\$_XDONFSEPARATORS}" - This attribute is required to apply the grouping separator and number separator for the format mask you defined.

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, then 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 is. BI Publisher recognizes 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 [Section 4.5.3, "Creating Multiple or Complex Headers and Footers"](#) for details.

Place the carried forward object at the bottom of the 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 is displayed 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 the template, you can create multiple brought forward/carried forward objects in the template pointing to various numeric elements in the data.

Note: Ensure that you do not include the commands `<?init-page-total:invAmt?>` and `<?end-page-total:InvAmt?>` as shown in the preceding example. The display-condition logic computation depends on these commands to function correctly.

4.11.3 Inserting Running Totals

The variable functionality (see [Section 4.13.1, "Setting Variables"](#)) can be used to add a running total to the 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>
```

You can use this XML code to create a report that contains running totals as shown in [Figure 4-57](#).

Figure 4-57 Report of Running Totals

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 [Figure 4-58](#).

Figure 4–58 Template for Running Total Field

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 that is shown in [Figure 4–58](#) are described in [Table 4–12](#).

Table 4–12 Fields for Running Totals

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
100.00	<?xdoxslt:set_variable(\$_XDOCTX, 'RTotalVar', xdoxslt:get_variable(\$_XDOCTX, 'RTotalVar') + INVAMT)?> <?xdoxslt:get_variable(\$_XDOCTX, 'RTotalVar')?>	Sets the value of RTotalVar to the current value plus the new Invoice Amount. Retrieves the RTotalVar value for display.
EFE	<?end for-each?>	Ends the INVOICE group.

4.12 Handling Data

The following sections describe methods for handling data in templates:

- [Section 4.12.1, "Sorting Data"](#)
- [Section 4.12.2, "Checking for Null Values"](#)
- [Section 4.12.3, "Regrouping the XML Data"](#)

4.12.1 Sorting Data

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'?>
```

4.12.2 Checking for Null Values

Within the 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 the 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?>
```

4.12.3 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, the template does not have to follow the hierarchy of the source XML file. You are therefore no longer limited by the structure of the data source.

4.12.3.1 XML Sample

To demonstrate the for-each-group standard, the following XML data sample of a CD catalog listing is 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.

4.12.3.2 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 the 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 loops through the occurrences of the new groupings, displaying the fields that you defined in the 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>

```

4.12.3.3 Template Example

Figure 4–59 shows a template that displays the CDs by Country, then Year, and lists the details for each CD.

Figure 4–59 Template for Regrouping

Group by Country		
Country:USA		
Group by Year		
Year:2000		
Title	Artist	Price
Group:DetailsMy CD	John Doe	1.00End Group
End Group by Year		
End Group by Country		

Table 4–13 shows the BI Publisher syntax entries made in the form fields of the template that is shown in Figure 4–59.

Table 4–13 Syntax Entries for Regrouping XML Data

Default Text Entry	Form Field Help Text Entry	Description
Group by Country	<?for-each-group:CD;COUNTRY Y?>	The <?for-each-group:CD;COUNTRY ?> tag declares the new group. It regroupes the existing CD group by the COUNTRY element.
USA	<?COUNTRY?>	Placeholder to display the data value of the COUNTRY tag.
Group by Year	<?for-each-group:current-group ();YEAR?>	The <?for-each-group:current-group();YEAR?> tag regroupes the current group (that is, COUNTRY), by the YEAR element.
2000	<?YEAR?>	Placeholder to display the data value of the YEAR tag.

Table 4–13 (Cont.) Syntax Entries for Regrouping XML Data

Default Text Entry	Form Field Help Text Entry	Description
Group: Details	<?for-each:current-group()?>	Once the data is grouped by COUNTRY and then by YEAR, the <?for-each:current-group()?> 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	<?TITLE?>	Placeholder to display the data value of the TITLE tag.
John Doe	<?ARTIST?>	Placeholder to display the data value of the ARTIST tag.
1.00	<?PRICE?>	Placeholder to display the data value of the PRICE tag.
End Group	<?end for-each?>	Closes out the <?for-each:current-group()?> tag.
End Group by Year	<?end for-each-group?>	Closes out the <?for-each-group:current-group();YEAR?> tag.
End Group by Country	<?end for-each-group?>	Closes out the <?for-each-group:CD;COUNTRY ?> tag.

This template produces the report that is shown in [Figure 4–60](#) when merged with the XML file.

Figure 4–60 Report of Regrouping

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

4.12.3.4 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?>
```

To demonstrate this feature, an XML data sample that simply contains average temperatures per month is used as input to a template that calculates the number of months having an average temperature within a certain range.

The following XML code 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 shown in [Figure 4–61](#).

Figure 4–61 Report of Annual Temperature

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 enables you to group the temperatures by increments of 10 degrees. You can then display a count of the members of each grouping, which is the number of months having an average temperature that falls within each range.

[Figure 4–62](#) shows the template to create the report that is shown in [Figure 4–61](#).

Figure 4–62 Template for Annual Temperature

Annual Temperature Summary

Range	Number of Months
Group by TmpRng	Months Month(s)End TmpRng
Range	

[Table 4–14](#) shows the form field entries made in the template that is shown in [Figure 4–62](#).

Table 4–14 Form Fields for Regrouping Data

Default Text Entry	Form Field Help Text Entry
Group by TmpRng	<?for-each-group:temp;floor(degree div 10)?> <?sort:floor(degree div 10)?>
Range	<?concat(floor(degree div 10)*10,' F to ',floor(degree div 10)*10+10,' F')?>
Months	<?count(current-group())?>
End TmpRng	<?end for-each-group?>

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 generates 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 are created: 0, 1, 2, and 3.

- The `<?concat(floor(degree div 10)*10,'F to ', floor(degree div 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.

4.13 Setting Variables, Parameters, and Properties

This section covers the following topics:

- [Section 4.13.1, "Setting Variables"](#)
- [Section 4.13.2, "Setting Parameters"](#)
- [Section 4.13.3, "Setting Properties"](#)

4.13.1 Setting Variables

Updatable variables differ from standard XSL variables `<xsl:variable>` in that they are updatable during the template application to the XML data. This allows you to create many new features in the templates that require updatable 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 [Section 4.11.3, "Inserting Running Totals"](#) for an example of the usage of updatable variables.

4.13.2 Setting Parameters

You can pass runtime parameter values into the 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 want to utilize in the template using the syntax described below.

To use 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 FOProcessor 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 the template that filters the 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>

```

Figure 4–63 shows a template that accepts a parameter value to limit the invoices displayed in the final document based on the parameter value.

Figure 4–63 *Template for Accepting Parameter Values*

InvThresh Declaration

Invoice Number	Invoice Amount
FE IF 13222-2	\$100.00 EI EFE

Table 4–15 describes the fields for defining parameters as shown in the template in Figure 4–63.

Table 4–15 *Fields for Defining Parameters*

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 are displayed. If you pass in a parameter value of 1,000, then the report that is shown in Figure 4–64 is produced.

Figure 4–64 *Report of Invoices*

Invoice Number	Invoice Amount
981110	1100
981112	8343

Notice the second invoice does not display because its amount was less than the parameter value.

4.13.3 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 the appendix "Configuration File Reference" in *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 the template, then rather than specify the font location in the configuration file, you can embed the font property inside the template. If you must secure the generated PDF output, then 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}

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 the 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 looks for the font in the /tmp/fonts directory. Note that if the template is deployed in multiple locations, then you must ensure that the path is valid for each location.

For more information about setting font properties, see the section "Font Definitions" in *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 persists 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 [Section 4.13.2, "Setting Parameters."](#)

4.14 Using Advanced Report Layouts

This section describes the following tasks for advanced report layouts:

- [Section 4.14.1, "Creating Batch Reports"](#)
- [Section 4.14.2, "Handling No Data Found Conditions"](#)
- [Section 4.14.3, "Inserting Pivot Tables"](#)
- [Section 4.14.4, "Constructing Dynamic Data Columns"](#)

4.14.1 Creating 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 requires 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), then these must be reset as well.

BI Publisher supports this requirement through the use of a context command. This command allows you to define elements of the 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 code 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 is as follows:

```
<?for-each@section:G_INVOICE?>
```

The closing <?end for-each?> tag is not changed.

Figure 4–65 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 are reset by the command.

Figure 4–65 Template for Batch Reports

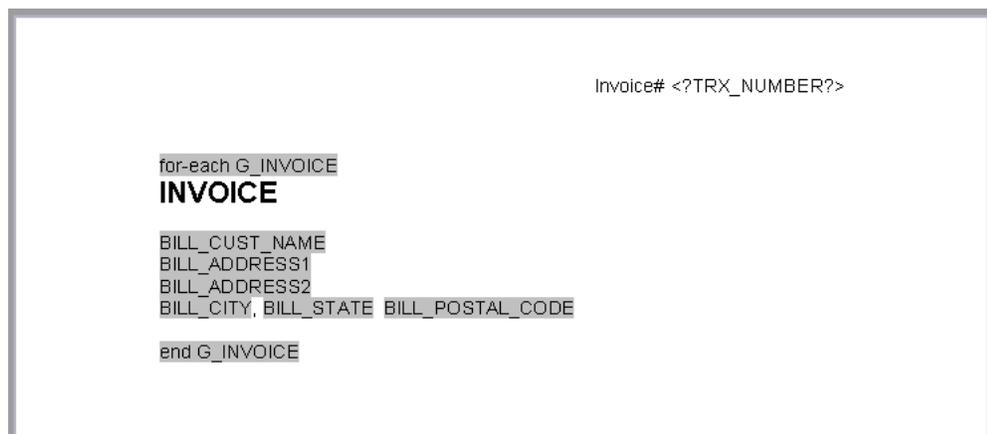


Table 4–16 shows the values of the form fields from the template in Figure 4–65.

Table 4–16 Fields for Creating Batch Reports

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 is 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 begins. The page numbers restart, and if header or footer information is derived from the data, it is reset as well.

4.14.2 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, edit the RTF template.

To handle no data found conditions:

1. At the end of the 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 the for-each@section loop.

Now if no data exists for ELEMENT_NAME, a valid PDF is generated with the text "No Data Found".

4.14.3 Inserting Pivot Tables

Note: For more information, see [Section 5.4.5, "Inserting a Pivot Table."](#)

The columns of a pivot table are data dependent. At design time you do not know how many columns are reported, or what the appropriate column headings are. Moreover, if the columns should break onto a second page, you must 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 code, a report is generated that shows each industry and totals the sales by year as shown in [Figure 4–66](#).

Figure 4–66 Report of Yearly Totals

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

[Figure 4–67](#) shows the template to generate the report that is shown in [Figure 4–66](#).

Figure 4–67 Template for Yearly Totals

Industry header column	for: YEAR end
for: INDUSTRY	for: sum(SALES) end end

The form fields in the template that is shown in [Figure 4–67](#) have the values that are described in [Table 4–17](#).

Table 4–17 Fields for Inserting Tables

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 Section 4.14.4.1, "Defining Columns to Repeat Across Pages."
for:	<?for-each-group@column:RESULTS;YEAR?>	Uses the regrouping syntax (see Section 4.12.3, "Regrouping the XML Data") 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 Section 4.18, "Controlling the Placement of Instructions Using the Context Commands."
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 Section 4.12.3, "Regrouping the XML Data") 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 must use the @cell context to create the table cells for the column. (For more information about context commands, see [Section 4.18, "Controlling the Placement of Instructions Using the Context Commands."](#))

4.14.4 Constructing 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 correctly renders a table when the number of columns that is 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 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 the column widths are defined in character cells, then you must 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 the 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, then the widths of the columns are calculated as a percentage of the total width of the table. The width calculation is illustrated in [Table 4–18](#).

Table 4–18 Calculating Column Widths

Width Definition	Column 1 (Width = 10)	Column 2 (Width = 12)	Column 3 (Width = 14)
Multiplier not present -% width	10/10+12+14*100 28%	%Width = 33%	%Width =39%
Multiplier = 6 - width	60 pts	72 pts	84 pts

4.14.4.1 Defining Columns to Repeat Across Pages

If the 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.

4.14.4.2 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 [Table 4-19](#).

Table 4-19 *Dynamic Data Columns*

Test Score	Test Score Range 1	Test Score Range 2	Test Score Range 3	...Test Score Range n
Test Category	# students in Range 1	# students in Range 2	# students in Range 3	# of students in Range n

However, you do not know how many Test Score Ranges are 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>` 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 [Figure 4–68](#). The first column, "Test Score" is static. The second column, "Column Header and Splitting" is the dynamic column. At runtime this column is split according to the data, and the header for each column is appropriately populated. The Default Text entry and Form Field Help entry for each field are listed in [Table 4–20](#). (See [Section 4.3.1.2, "Using the Form Field Method"](#) for more information on using form fields).

Figure 4–68 Dynamic Data Columns

Test Score	Column Header and Splitting
Group:TestScores Test Category	Content and Splitting end:TestScores

Table 4–20 Fields for Dynamic Data Columns

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?>
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 is also the row heading.
- The second column is the one to be split dynamically. The width you specify is divided by the number of columns of data. In this case, there are 5 data columns.
- The second column contains the dynamic "range" data. The width of the column is divided according to the split column width. Because this example does not contain the unit value tag (<?split-column-width-unit:value?>), the column is split on a percentage basis. Wrapping of the data occurs if required.

If the tag (<?split-column-width-unit:value?>) were present, then the columns have a specific width in points. If the total column widths were wider than the allotted space on the page, then the table breaks 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 renders the output that is shown in [Figure 4–69](#).

Figure 4–69 Report of Test Scores

Test Score	0-20	21-40	41-60	61-80	81-100
Mathematics	30	45	50	102	22

4.15 Formatting Numbers, Dates, and Currencies

This section provides details for formatting numbers, dates, and currencies.

4.15.1 Formatting Numbers

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. For information, see [Section 4.21.2.8, "Formatting Native XSL Numbers."](#)

Use only one of these methods. If the number format mask is specified using both methods, then the data is formatted twice, causing unexpected behavior.

The group separator and the number separator are set at runtime based on the template locale. If you are working in a locale other than en-US, or the templates require translation, use the Oracle format masks.

4.15.2 Data Source Requirements

To use the Oracle format mask or the Microsoft format mask, the numbers in the 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) then the format does not work.

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.

4.15.3 Localization Considerations

If you are working in a locale other than en-US, or the templates require translation, then 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 fixes 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.

4.15.4 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.

To apply a number format to a form field:

1. Open the **Form Field Options** dialog for the placeholder field.
2. Set the **Type** to Number.
3. Select the appropriate **Number format** from the list of options.

4.15.5 Supported Microsoft Format Mask Definitions

Table 4–21 lists the supported Microsoft format mask definitions.

Table 4–21 Supported Microsoft Format Mask Definitions

Symbol	Location	Meaning
0	Number	Digit. Each explicitly set 0 appears, 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
.	Number	Determines the position of the decimal separator. The decimal separator symbol used is determined at runtime based on template locale. Example: Format mask: ###0.00 Data: 1234.56 Display for English locale: 1,234.56 Display for German locale: 1.234,56
-	Number	Determines placement of minus sign for negative numbers.
,	Number	Determines the placement of the grouping separator. The grouping separator symbol used is determined at runtime based on template locale. Example: Format mask: ###0.00 Data: 1234.56 Display for English locale: 1,234.56 Display for German locale: 1.234,56
E	Number	Separates mantissa and exponent in a scientific notation. Example: 0.###E+0 plus sign always shown for positive numbers 0.###E-0 plus sign not shown for positive numbers
;	Subpattern boundary	Separates positive and negative subpatterns. See the Note that follows the table.
%	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#)".

4.15.6 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.

Figure 4–70 shows an example Form Field Help Text dialog entry for the data element "empno".

Figure 4–70 Form Field Help Text Entry

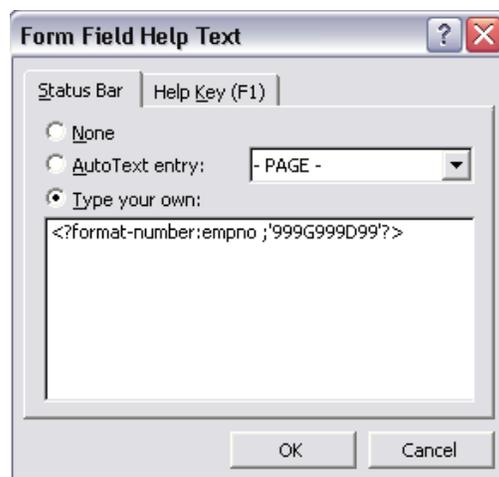


Table 4–22 lists the supported Oracle number format mask symbols and their definitions.

Table 4–22 Supported Oracle Number Format Mask Symbols

Symbol	Meaning
0	Digit. Each explicitly set 0 appears, if no other number occupies the position. Example: Format mask: 00.0000 Data: 1.234 Display: 01.2340
9	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. Example: Format mask: 99.9999 Data: 1.234 Display: 1.234
C	Returns the ISO currency symbol in the specified position.
D	Determines the placement of the decimal separator. The decimal separator symbol used is determined at runtime based on template locale. For example: Format mask: 9G999D99 Data: 1234.56 Display for English locale: 1,234.56 Display for German locale: 1.234,56
EEEE	Returns a value in scientific notation.
G	Determines the placement of the grouping (thousands) separator. The grouping separator symbol used is determined at runtime based on template locale. For example: Format mask: 9G999D99 Data: 1234.56 Display for English locale: 1,234.56 Display for German locale: 1.234,56
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 "-"

4.15.7 Formatting Dates

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 is formatted twice, which causes unexpected behavior.

4.15.8 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.

Note: If the time component and time zone offset are not included in the XML source date, then BI Publisher assumes it represents 12:00 AM UTC (that is, yyyy-mm-ddT00:00:00-00:00).

4.15.9 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, then the abstract format mask "MEDIUM" is used as default. See [Section 4.15.12, "Oracle Abstract Format Masks"](#) for the description.

[Table 4–23](#) lists the supported Microsoft date format mask components.

Table 4–23 Supported Microsoft Date Format Mask Components

Symbol	Meaning
d	The day of the month. Single-digit days do not have a leading zero.
dd	The day of the month. Single-digit days 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 do not have a leading zero.
MM	The numeric month. Single-digit months 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.
h	The hour in a 12-hour clock. Single-digit hours do not have a leading zero.

Table 4–23 (Cont.) Supported Microsoft Date Format Mask Components

Symbol	Meaning
hh	The hour in a 12-hour clock. Single-digit hours have a leading zero.
H	The hour in a 24-hour clock. Single-digit hours do not have a leading zero.
HH	The hour in a 24-hour clock. Single-digit hours have a leading zero.
m	The minute. Single-digit minutes do not have a leading zero.
mm	The minute. Single-digit minutes have a leading zero.
s	The second. Single-digit seconds do not have a leading zero.
ss	The second. Single-digit seconds do 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.
:	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.

4.15.10 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, then the abstract format mask "MEDIUM" is used as the default.

Example form field help text entry:

```
<?format-date:hiredate;'YYYY-MM-DD'?>
```

Table 4–24 lists the supported Oracle format mask components.

Table 4–24 Supported Oracle Format Mask Components

Symbol	Meaning
- / , . ; : "text"	Punctuation and quoted text are reproduced in the result.
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.
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 P.M.	Meridian indicator with or without periods.
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.

Table 4–24 (Cont.) Supported Oracle Format Mask Components

Symbol	Meaning
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.)
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 Y	Last 2, or 1 digit(s) of year.

4.15.11 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 [Section 4.15.12, "Oracle Abstract Format Masks"](#) 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?>
```

4.15.12 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 depends on the locale that is 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'??>
```

Table 4–25 lists the abstract format masks and the sample output that would be generated for the US locale.

Table 4–25 Abstract Format Masks and Sample Output

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

4.15.13 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';?>
```

4.15.14 Formatting Currencies

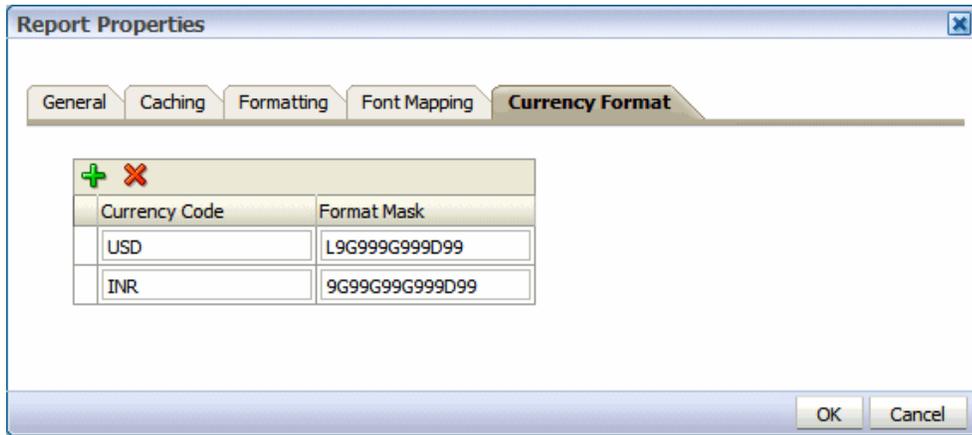
BI Publisher enables you to define specific currency format masks to apply to the published data at runtime.

To utilize currency formatting in the RTF template:

1. Set up the 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 Figure 4–71.

Figure 4–71 Sample Currency Codes



See [Section 2.8.5, "Configuring Currency Formats"](#) for more information.

2. Enter the format-currency command in the RTF template to apply the format to the field at runtime, as described in [Section 4.15.15, "Applying a Currency Format to a Field."](#)

4.15.15 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 the data.

CurrencyCode can either be set to a static value or it can be set dynamically. If the value is static for the report, then 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 that contains the currency code must be present in the data.

At runtime, the Amount_Field is 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 is displayed in the report based on the value for CurrencyCode. If you do not want the currency symbol to be displayed, then you can either enter 'false' or simply do not specify the parameter.

4.15.16 Example: Displaying Multiple Currency Formats in a Report

[Table 4–26](#) provides an example that assumes you have set up the various currency formats in the report properties.

Table 4–26 display multiple currency formats

Currency Code	Format Mask
USD	9G999D99

Table 4–26 (Cont.) display multiple currency formats

Currency Code	Format Mask
INR	9G99G99G999D99

In this example, you need not set the currency code dynamically. You have the following elements in the XML data:

```
<TOTAL_SALES>
  <US_SALES>8596526459.56</US_SALES>
  <INDIA_SALES>60000000</INDIA_SALES>
</TOTAL_SALES>
```

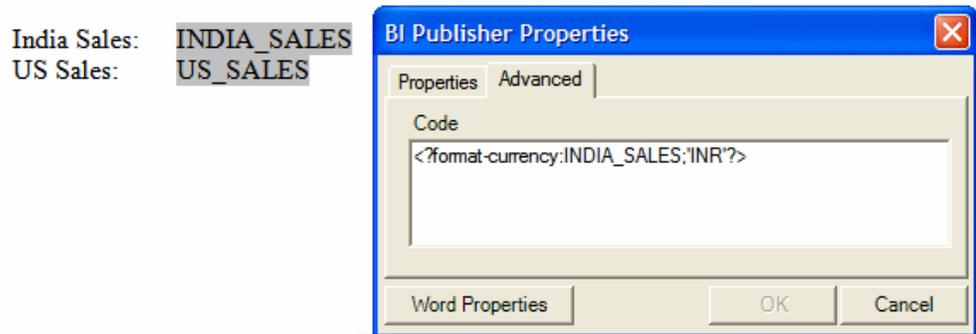
You want to display these two total fields in the template.

For US_SALES, the syntax in the BI Publisher properties field is as follows:

```
<?format-currency:US_SALES;'USD' ?>
```

Figure 4–72 shows the two fields in a template with the BI Publisher Properties dialog displaying the entry for INDIA_SALES;

Figure 4–72 Properties to Display Sales



At runtime, the fields are displayed as shown in Figure 4–73.

Figure 4–73 Displaying Sales Fields

```
India Sales: 6,00,00,000.00
US Sales: 8,596,526,459.56
```

4.15.17 Example: Displaying Multiple Currency Codes in a Single Report

The following simple XML code includes an element that contains the Amount (Trans_amount) and an element that contains 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 the template for the field in which you want the amounts to display:

```
<?format-currency:Trans_Amount;Cur_Code;'true' ?>
```

Figure 4–74 shows the report that is generated.

Figure 4–74 Multiple Currency Report

USD Amount:	\$123.00
GBP Amount:	-£456.00
EUR Amount:	€ 748.00
JPY Amount:	-¥987

4.16 Supporting Calendars and Time Zones

This section describes support for calendars and time zones.

4.16.1 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';?>`

Figure 4–75 shows the output generated using this definition with locale set to zh-TW and time zone set to Asia/Taipei:

Figure 4–75 Output with Locale Set to zh-TW

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 that is specified in the template overrides the calendar type set in the profile option.

4.16.2 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, then 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 the template:

```
<?format-date:hiredate;'LONG_TIME_TZ';'Asia/Shanghai'?>
```

4.17 Using External Fonts

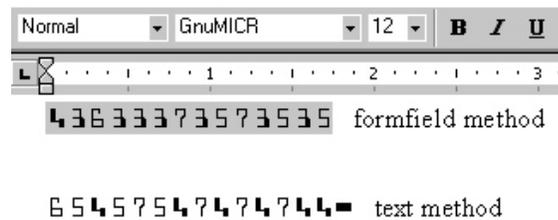
BI Publisher enables you to use external fonts in the output that are not normally available on the server. To set up a new font for the report output, use the font to design the template on your client machine, then make it available on the server, and configure BI Publisher to access the font at runtime.

Note: External fonts are supported for PDF output only.

To use external fonts:

1. Use the font in the template.
 - a. Copy the font to the <WINDOWS_HOME>/fonts directory.
 - b. Open Microsoft Word and build the template.
 - c. Insert the font in the 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.

[Figure 4–76](#) shows an example of the form field method and the text method.

Figure 4–76 Form Field and Text Methods

- 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, then the fonts are located in `$JAVA_HOME/jre/lib/fonts`.

- 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:

- 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 [Section 2.8, "Configuring Report Properties."](#)

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 the appendix "Configuration File Reference" in *Oracle Fusion Middleware Administrator's Guide for Oracle Business Intelligence Publisher* for more information.

Now you can run the report and BI Publisher uses 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.

4.17.1 Using the Barcode Fonts Shipped with BI Publisher

[Table 4–27](#) describes the barcodes that are included with BI Publisher.

Table 4–27 Barcodes Included 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 executes the preprocessing on the 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 applies 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 the template. For information on font mapping, see [Section 2.8, "Configuring Report Properties."](#)

If you want to use the font directly in Microsoft Word, then add the appropriate .TTF file to the C:\WINDOWS\Fonts directory. To use the Template Builder Preview function, map the font in the Template Builder configuration file. See [Section 4.22.5, "Configuring Fonts for the BI Publisher Template Builder."](#)

To use the barcode fonts in the report output:

1. Insert a field in the template where the barcode is to display in the report output.
2. In the form field, enter the following command:

```
<?format-barcode:data;'barcode_type' ?>
```

where

data is the element from the 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 the template, for example, Bookman.
4. Configure the font in the Font Mapping page. For more information about the Font Mapping page, see [Section 2.8, "Configuring Report Properties."](#)

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 is treated as UPC-A message with a check digit, 11 is without a check digit.
- The upce algorithm accepts only UPC-E message strings and encodes into UPC-E barcode.
- A string of 8 characters is treated as a UPC-E message with both a front and end guard bar; a string of 6 characters is without guard bars.

4.17.2 Implementing Custom Barcode Formats

If you choose to use a custom barcode instead, use this procedure to implement a custom barcode.

BI Publisher offers the ability to execute preprocessing on the data prior to applying a barcode font to the data in the output document. For example, you might 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 apply the formatting in the template. For information, see the section "Advanced Barcode Font Formatting" in *Oracle Fusion Middleware Developer's Guide for Oracle Business Intelligence Publisher*.

To enable the formatting feature in the template, you must use two commands in the 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.

4.17.2.1 Registering the Barcode Encoding Class

Use the following syntax in a form field in the template to register the barcode encoding class:

```
<?register-barcode-vendor:java_class_name;barcode_vendor_id?>
```

This command requires a Java class name (this carries 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.

4.17.2.2 Encoding the Data

To format the data, use the following syntax in a form field in the template:

```
<?format-barcode:data;'barcode_type';'barcode_vendor_id'?>
```

where

data is the element from the 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 is then be applied to the data in the final output.

4.18 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 the template, is converted to XSL-FO at runtime. The placement of these instructions within the converted stylesheet determines the behavior of the template.

BI Publisher's RTF processor places these instructions within the XSL-FO stylesheet according to the most common context. However, sometimes you must 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 want to restart the page numbering, then 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 the tags to specify the context for the insertion of the instructions. The value of the context determines where the 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 context types that are described in [Table 4-28](#).

Table 4–28 Supported Context Types

Context	Description
section	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. Note that you can retain continuous page numbering across sections by using the <code><?initial-page-number:'auto'?></code> command. See Section 4.14.1, "Creating Batch Reports" for an example of this usage.
column	The statement affects the whole column of a table. This context is typically used to show and hide table columns depending on the data. See Section 4.10.5, "Formatting Columns" for an example.
cell	The statement affects the cell of a table. This is often used together with <code>@column</code> in pivot tables to create a dynamic number of columns. See Section 4.14.3, "Inserting Pivot Tables" for an example.
block	The statement affects multiple complete fo:blocks (RTF paragraphs). This context is typically used for if and for-each statements. It can also be used to apply formatting to a paragraph or a table cell. See Section 4.10.7, "Highlighting Cells" for an example.
inline	The context becomes the single statement inside an fo:inline block. This context is used for variables.
incontext	The statement is inserted immediately after the surrounding statement. This is the default for <code><?sort?></code> statements that must follow the surrounding for-each as the first element.
inblock	The statement becomes a single statement inside an fo:block (RTF paragraph). This is typically not useful for control statements (such as if and for-each) but is useful for statements that generate text, such as call-template.
inlines	The statement affects multiple complete inline sections. An inline section is text that uses the same formatting, such as a group of words rendered as bold. See Section 4.10.2, "Using If Statements in Boilerplate Text." If you use <code>@inlines</code> with if syntax, any other if syntax inside the statement must use the context command <code>@inline</code> . If you use <code>@inlines</code> with FOR-EACH syntax any other if or FOR-EACH syntax inside the statement must use the context command <code>@inline</code> .
begin	The statement is placed at the beginning of the XSL stylesheet. This is required for global variables. See Section 4.13.2, "Setting Parameters."
end	The statement is placed at the end of the XSL stylesheet.

Table 4–29 shows the default context for the BI Publisher commands.

Table 4–29 Default Context for BI Publisher Commands

Command	Context
apply-template	inline
attribute	inline
call-template	inblock
choose	block
for-each	block
if	block

Table 4–29 (Cont.) Default Context for BI Publisher Commands

Command	Context
import	begin
param	begin
sort	incontext
template	end
value-of	inline
variable	end

4.19 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 might 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.

4.19.1 Locating Data

Locate information in an XML document using location-path expressions.

A node is the most common search element that you 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 are returned.

To retrieve the individual TITLE elements, use the following command:

```
/CATALOG/CD/TITLE
```

This example returns the following XML:

```
<CATALOG>
  <CD cattype=Folk>
    <TITLE>Empire Burlesque</TITLE>
  </CD>
  <CD cattype=Rock>
    <TITLE>Hide Your Heart</TITLE>
  </CD>
</CATALOG>
```

Further limit the 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 the 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, you can find all CDs released in 1985 or later using the following expression:

```
/CATALOG/CD[YEAR >=1985]
```

4.19.2 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.

4.19.3 Specifying Context and Parents

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 two periods (..) are 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, then you might get erroneous results.

XPath is an extremely powerful standard when combined with RTF templates allowing you to use conditional formatting and filtering in the template.

4.20 Declaring Namespaces

If the 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.example.com/fsg/2002-30-20/?>
```

Once declared, you can use the namespace in the placeholder markup, for example:

```
<?fsg:ReportName?>
```

4.21 Using FO Elements and XSL Elements

This section describes how to use FO elements and XSL elements.

4.21.1 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 that BI Publisher supports is in [Appendix D, "Supported XSL-FO Elements."](#)

4.21.2 Using XSL Elements

You can use any XSL element in the 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 the 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.

4.21.2.1 Applying 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.

4.21.2.2 Copying 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?>

4.21.2.3 Calling a Named 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?>

4.21.2.4 Declaring a Template

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?>

4.21.2.5 Declaring a Variable

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.

4.21.2.6 Importing a Style Sheet

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?>

4.21.2.7 Defining the Root Element of the Style Sheet

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 [Section 4.20, "Declaring Namespaces."](#)

4.21.2.8 Formatting Native XSL Numbers

The native XSL format-number function takes the basic format:

`format-number (number, format, [decimalformat])`

The following list describes the parameters:

- number - Required. Specifies the number to be formatted.
- format - Required. Specifies the format pattern. Use the following characters to specify the pattern:
 - # (Denotes a digit. Example: #####)
 - 0 (Denotes leading and following zeros. Example: 0000.00)
 - . (The position of the decimal point Example: ###.##)
 - , (The group separator for thousands. Example: ###,###.##)
 - % (Displays the number as a percentage. Example: ##%)
 - ; (Pattern separator. The first pattern is used for positive numbers and the second for negative numbers)
- decimalformat - Optional. For more information on the decimal format, consult any basic XSLT manual.

4.22 Guidelines for Designing RTF Templates for Microsoft PowerPoint Output

BI Publisher can generate the RTF template as PowerPoint output enabling you to get report data into key business presentations. Currently, the PowerPoint document generated is a simple export of the formatted data and charts to PowerPoint.

4.22.1 Limitations

Following are limitations when working with PowerPoint as an output:

- When designing tables for a 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 want to use a font that is not installed, then ensure that you have configured it with BI Publisher.

4.22.2 Usage Guidelines

Following are guidelines to help you when designing an RTF template intended for PowerPoint output:

- PowerPoint output preserves the page orientation (portrait or landscape) defined in the RTF template. Most presentations are oriented in landscape so this is the recommended orientation of an RTF template.
- A page break in an RTF template generates a new slide.
- The background color of the slides are always generated as white. If you prefer a different background color, then 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, ensure that you specify the font for the whole tag "<?XXXXXX?>", including the "<?" and "?>" in the template.
- When using multibyte characters (for example, simplified Chinese) in tables, ensure that the column widths are large enough to contain the width of the largest character plus the cell's left and right margins to avoid unexpected character display.

4.22.3 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) is 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 [Chapter 10, "Setting Report Processing and Output Document Properties"](#) for more information.

Note that when **Enable PPTX native chart support** is set to false, all charts are rendered as images in PowerPoint2007 output. Therefore, set this option to false only when a report includes the non-native chart types.

4.22.4 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 the template, then you must configure the BI Publisher server for each font used in the RTF template for generating PowerPoint output. You must copy these fonts to the BI Publisher Server and define the Font Mappings for RTF templates. This can be done for the entire system or for individual reports. See [Section 10.16, "Defining Font Mappings"](#) for more details.

4.22.5 Configuring Fonts for the BI Publisher Template Builder

When using the BI Publisher Template Builder to design a report, to correctly preview PPT output that uses non-English or non-standard fonts, you must 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 might 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 encounter issues such as characters overlays and wrapping that does not work.

To configure fonts for the BI Publisher Template Builder:

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 the section "Font Definitions" in *Oracle Fusion Middleware Administrator's Guide for Oracle Business Intelligence Publisher*.
3. Save the xdo.cfg in UTF-8 format.

[Figure 4-77](#) shows a sample xdo.cfg file.

Figure 4–77 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>
</config>

```

4.23 Rendering HTML Formatted Data in a Report

This section describes how to preserve HTML formatting from a data source in your final output report. It contains the following topics:

- [Supported HTML Features](#)
- [Data Model Requirements](#)
- [RTF Template Requirements](#)
- [Example](#)

4.23.1 Supported HTML Features

Supported HTML features are:

- Hyperlink
- List
 - Bulleted list
 - Ordered list
- Paragraph
- Font style (bold, italic, plain, underline, subscript, superscript, strike-through)
- Font size
- Font family
- Background color
- Foreground color
- Paragraph alignment (center, left, right, and justify)
- Paragraph indent

The following HTML features are not supported:

- Nested list (list with indent)
- Any HTML tags or attributes manually inserted by modifying the HTML source code; for example, inserted tables or images.

4.23.2 Data Model Requirements

The XML data used as input to the report must wrap the HTML portion of the data in a CDATA section. See "Handling XHTML Data Stored in a CLOB Column" in the *Oracle Fusion Middleware Data Modeling Guide for Oracle Business Intelligence Publisher* for information about retrieving the HTML data for display in the report.

4.23.3 RTF Template Requirements

To render the HTML in your report, use the following tag in the RTF template:

```
<?html2fo: elementname?>
```

where *elementname* is the XML element name that contains the HTML data.

4.23.4 Example

This example uses the following XML data with embedded HTML data:

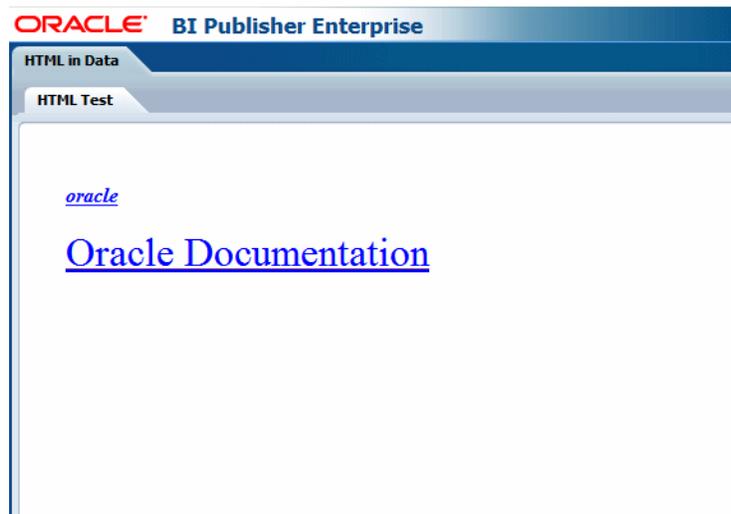
```
<?xml version="1.0" encoding="UTF-8"?>
<RTECODE>
<![CDATA[
<p><font style="font-style: italic; font-weight: bold;" size="3">
<a href="http://www.oracle.com">oracle</a></font> </p>
<p><font size="6"><a href="http://docs.oracle.com/">Oracle Documentation</a>
</font></p>
]]>
</RTECODE>
```

To render this sample as formatted HTML a report, enter the following in your RTF template:

```
<?html2fo: RTECODE?>
```

Figure 4–78 shows how the sample will render in the output report:

Figure 4–78 HTML Formatting from Data Rendered in Report



Creating RTF Templates Using the Template Builder for Word

This chapter describes creating RTF templates in BI Publisher using the Template Builder for Word add-in.

This chapter includes the following sections:

- [Section 5.1, "Overview"](#)
- [Section 5.2, "Getting Started"](#)
- [Section 5.3, "Accessing Data for Building Templates"](#)
- [Section 5.4, "Inserting Components to the Template"](#)
- [Section 5.5, "Previewing a Template"](#)
- [Section 5.6, "Template Editing Tools"](#)
- [Section 5.7, "Uploading a Template to the BI Publisher Server"](#)
- [Section 5.8, "Using the Template Builder Translation Tools"](#)
- [Section 5.9, "Setting Options for the Template Builder"](#)
- [Section 5.10, "Setting Up a Configuration File"](#)
- [Section 5.11, "BI Publisher Menu Reference"](#)

5.1 Overview

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 increase 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 the template with sample XML data
- Browse and update the content of form fields
- Extract boilerplate text into an XLIFF translation file and test translations

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 [Chapter 4, "Creating RTF Templates."](#)

5.1.1 Before You Get Started

The Template Builder installation provides samples and demo files to help you get started. The demos can be accessed from the Windows Start menu as follows:

Select **Start, Programs, Oracle BI Publisher Desktop, and Demos.**

You can also access the demos 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:

- **TemplateBuilderDemo.exe** - 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.
- **TemplateBuilderInvoice.exe** - 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.
- **LocalizationDemo.exe** - 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 the subfolders:

- eText templates
- Excel templates
- PDF templates
- RTF templates

The eText, PDF, and Excel 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.

5.1.2 Prerequisites and Limitations

The following list describes prerequisites and limitation for this feature:

- **Prerequisites:**
 - The report data model has been created and runs successfully. For information on creating Data Models, see the *Oracle Fusion Middleware Data Modeling Guide for Oracle Business Intelligence Publisher*.
 - Supported versions of Microsoft Word and Microsoft Windows are installed on the client.

Note: See [System Requirements and Certification](#) for the most up-to-date information on supported hardware and software.

- The BI Publisher Template Builder has been downloaded and installed on the 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.

5.2 Getting Started

This section describes how to get started. It contains the following topics:

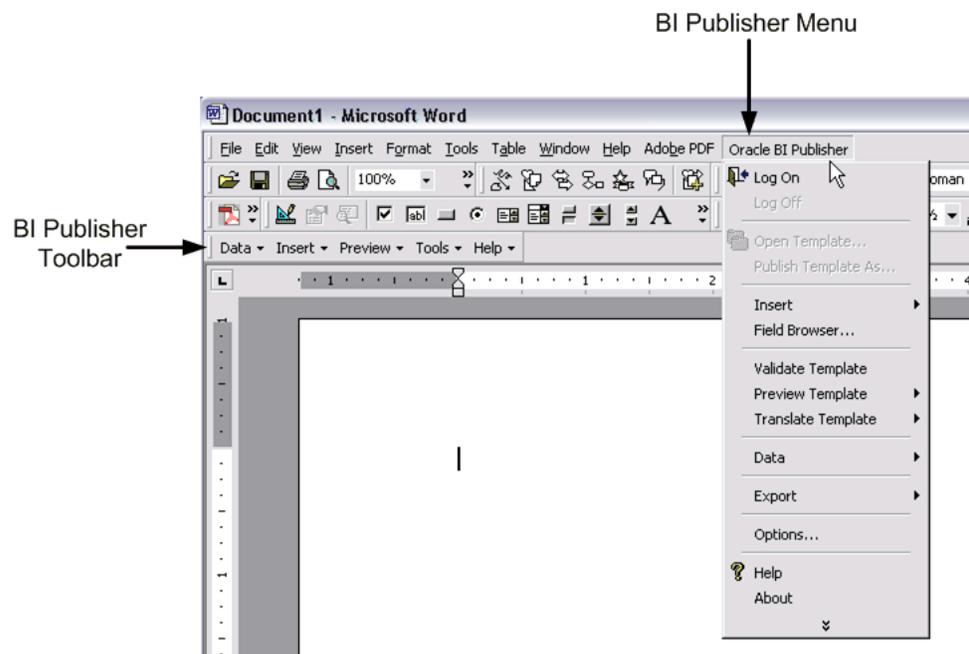
- [Section 5.2.1, "Features of the Oracle BI Publisher Template Builder for Word"](#)
- [Section 5.2.2, "Building and Uploading a Template"](#)
- [Section 5.2.2.1, "Working in Connected Mode"](#)

5.2.1 Features of the Oracle BI Publisher Template Builder for Word

When you open Microsoft Word after installing the Template Builder, you see the Oracle BI Publisher menu.

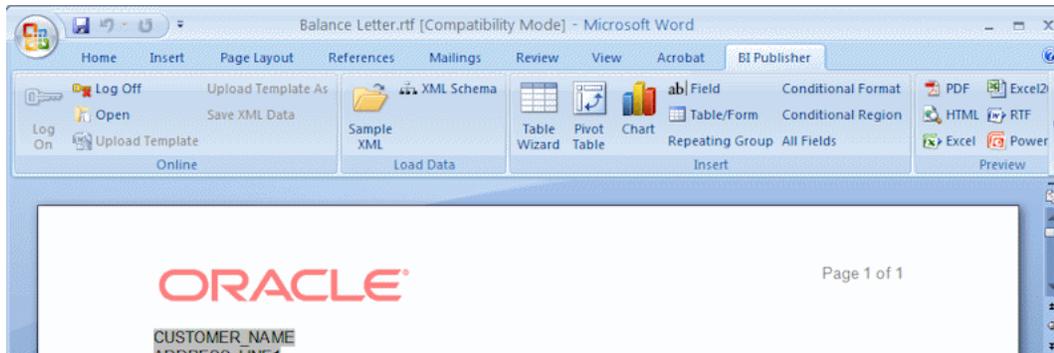
For versions of Microsoft Word prior to 2007, the menu and toolbar are displayed as shown in [Figure 5–1](#).

Figure 5–1 *Display of the Menu and Toolbar in Versions of Microsoft Word Prior to 2007*



For Microsoft Word 2007 users, the BI Publisher commands are displayed in the ribbon format, as shown in [Figure 5–2](#).

Figure 5–2 BI Publisher Commands Displayed in the Ribbon Format



Use the menu (or toolbar) to perform the following:

- Insert data fields into age RTF templates
- Insert tables, forms, charts, and pivot tables
- Preview the template in multiple outputs
- Browse and update the content of form fields
- Validate the template
- Perform calculations on fields within the template
- Connect to the Oracle BI Publisher catalog to retrieve data to build the template
- Upload the template to the Oracle BI Publisher server
- Extract boilerplate text into an XLIFF translation file and test translations

5.2.2 Building and Uploading a Template

You can build and upload the template using a direct connection with the BI Publisher server, or you can build and upload the template in disconnected mode.

5.2.2.1 Working in Connected Mode

To work 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 that you upload the template to.

Enter a Report Name and select the folder in which to save it.

Click **Save**.

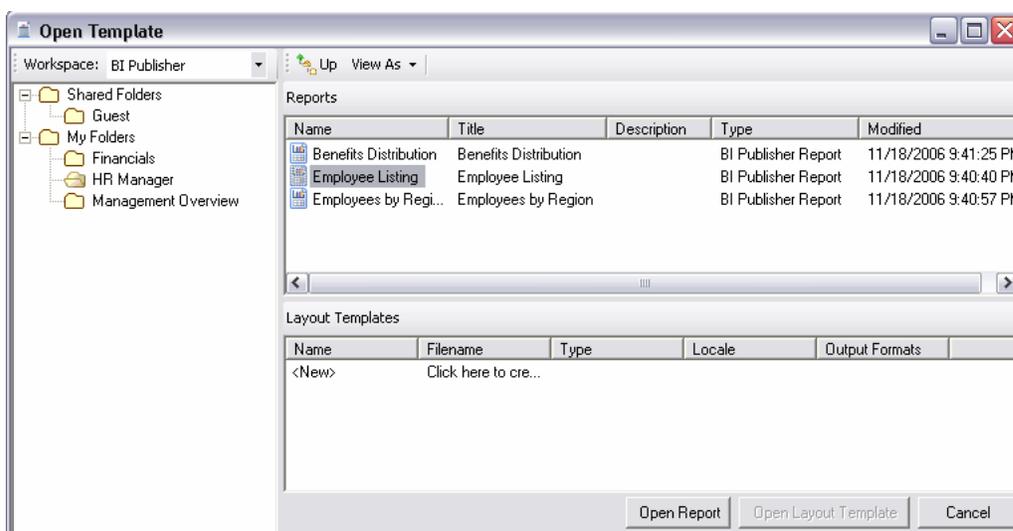
The sample data from the data model is loaded to the Template Builder.

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, as shown in [Figure 5-3](#).

Any existing templates are listed in the **Layout Templates** pane.

Figure 5-3 The Layout Templates Pane



6. Follow the guidelines in this chapter to insert data fields and design the template using features such as tables, charts, and graphics. Use Microsoft Word to apply formatting to fonts and other objects in the template.

For more advanced template options, use the guidelines in [Chapter 4, "Creating RTF Templates."](#)

7. To upload the template file to the BI Publisher server and add it to the report definition, select **Upload Template As** from the Oracle BI Publisher menu.

If you have not saved the template, then you are prompted to save it in Rich Text Format.

8. Enter a name and select a locale in the **Upload as New** dialog. This is the name that is displayed under **Layouts** in the Report Editor. This is also the layout name that is 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 [Section 2.6.3, "Configuring the Layout Settings Using the List View"](#) for more information.

5.2.2.2 Working in Disconnected Mode

To work in disconnected mode, you must have a sample data file available in the local work environment.

To work in disconnected mode:

1. Save a sample data file to your local computer. See [Section 5.3, "Accessing Data for Building Templates."](#)
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 the sample data file in the 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 the report also contains no data.

4. Follow the guidelines in this chapter to insert data fields and design the template using features such as tables, charts, graphics, and other layout components. Use Microsoft Word to apply formatting to fonts and other objects in the template.

For more advanced template options, use the guidelines in [Chapter 4, "Creating RTF Templates."](#)

5. Upload the layout template file.

In the BI Publisher catalog, open the report in the Report Editor. Click **Add New Layout**.

Complete the fields in the dialog and then select **Upload**. The template now appears as a layout for the report.

6. Configure properties for this layout.

See [Section 2.6, "Configuring Layouts Using the List View"](#) for more information.

5.3 Accessing Data for Building Templates

The data model defines the XML format that is merged with the RTF template. The Template Builder requires sample data to build the 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 procedure in [Section 5.3.1, "Loading XML Data from a Local File."](#) If you are connected, then use the procedure in [Section 5.3.2, "Loading Data from the BI Publisher Catalog."](#)

5.3.1 Loading XML Data from a Local File

One method of loading data to the Template Builder is to save a sample of the report data to a local directory.

For information on saving sample data from a report data model, see the "Testing Data Models and Generating Sample Data" section in *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, then 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 are 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** - Enables you to load a sample XML file that contains all fields that you want to insert into the template as a data source. If you are not connected to the BI Publisher server, then use this method to load the data.
- **XML Schema** - Enables you to load an XML Schema file (.xsd) that contains the fields available in the report XML data. The XML schema has the advantage of being complete (a sample XML file might 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.

5.3.2 Loading Data from the BI Publisher Catalog

You can connect directly to the BI Publisher Server to load the 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 [Section 5.2.2.1, "Working in Connected Mode."](#)
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 are prompted to create a report in the catalog.

5.4 Inserting Components to the Template

This section includes the following topics:

- [Section 5.4.1, "Inserting a Field"](#)
- [Section 5.4.2, "Inserting a Table Using the Table Wizard"](#)
- [Section 5.4.3, "Inserting a Table or Form Using the Insert Table/Form Dialog"](#)
- [Section 5.4.4, "Inserting a Chart"](#)
- [Section 5.4.5, "Inserting a Pivot Table"](#)
- [Section 5.4.6, "Inserting a Repeating Group"](#)
- [Section 5.4.7, "Creating Grouping Fields Around an Existing Block"](#)
- [Section 5.4.8, "Inserting and Editing Conditional Regions"](#)

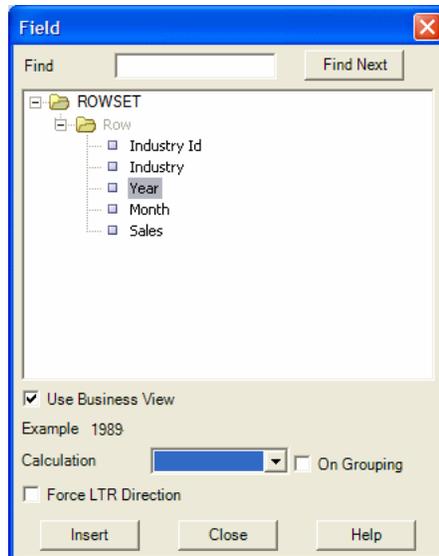
- [Section 5.4.9, "Inserting Conditional Formatting"](#)

5.4.1 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 the loaded data source in a tree view, as shown in [Figure 5-4](#).

Figure 5-4 The Field Dialog



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 the 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.

5.4.1.1 About the Insert Field Dialog

The Insert Field dialog fields are described in the following sections:

5.4.1.2 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 the search expression is selected. Click the **Find Next** button again to see the next occurrence.

5.4.1.3 Example

When you select a field name in the tree view, an example value for this field is shown.

5.4.1.4 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.

5.4.1.5 Calculation

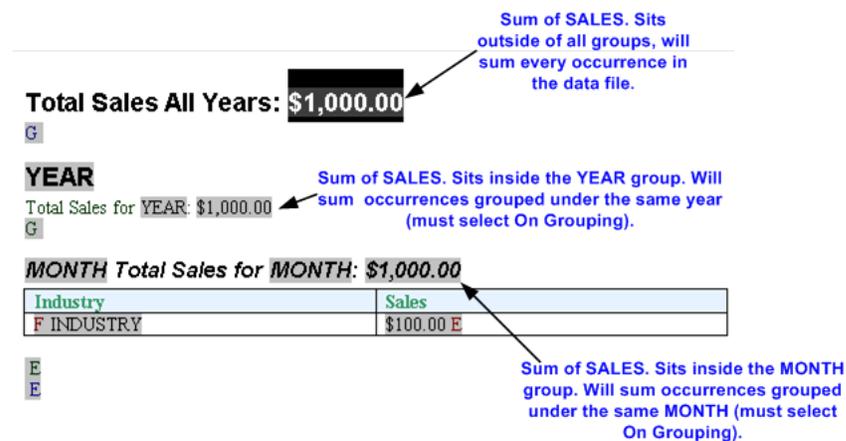
This feature enables you to perform aggregation functions on data fields, such as sum, average, count, minimum, and maximum.

For example, if you select sum for a data field, then the field shows 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.

Figure 5–5 shows an example.

Figure 5–5 Grouping Context 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 a template using the Template Builder, see [Section 5.4.6, "Inserting a Repeating Group"](#) and [Section 4.4, "Defining Groups."](#)

5.4.2 Inserting a Table Using the Table Wizard

The **Insert Table Wizard** enables you to create standard reports. On the **Insert** menu select **Table Wizard** and complete the following steps:

1. [Section 5.4.2.1, "Step 1: Selecting Report Format"](#)
2. [Section 5.4.2.2, "Step 2: Selecting Table Data"](#)
3. [Section 5.4.2.3, "Step 3: Selecting Data Fields"](#)
4. [Section 5.4.2.4, "Step 4: Grouping the Table"](#)
5. [Section 5.4.2.5, "Step 5: Inserting a Break for the Group"](#)
6. [Section 5.4.2.6, "Step 6: Sorting the Table"](#)

- 7. [Section 5.4.2.7, "Step 7: Clicking Finish"](#)
- 8. [Section 5.4.2.8, "Step 8: Customizing the Table Using Microsoft Word Functionality"](#)

5.4.2.1 Step 1: Selecting Report Format

Start by selecting the basic report format. Choose from **Table**, **Form**, or **Free Form**. [Figure 5–6](#) shows examples of each format.

Figure 5–6 Examples of Table, Form, and Free Form Formats

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	<p>John</p> <p>Doe</p> <p>(123) 456-7890</p>						

5.4.2.2 Step 2: Selecting 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 that is required for the table.

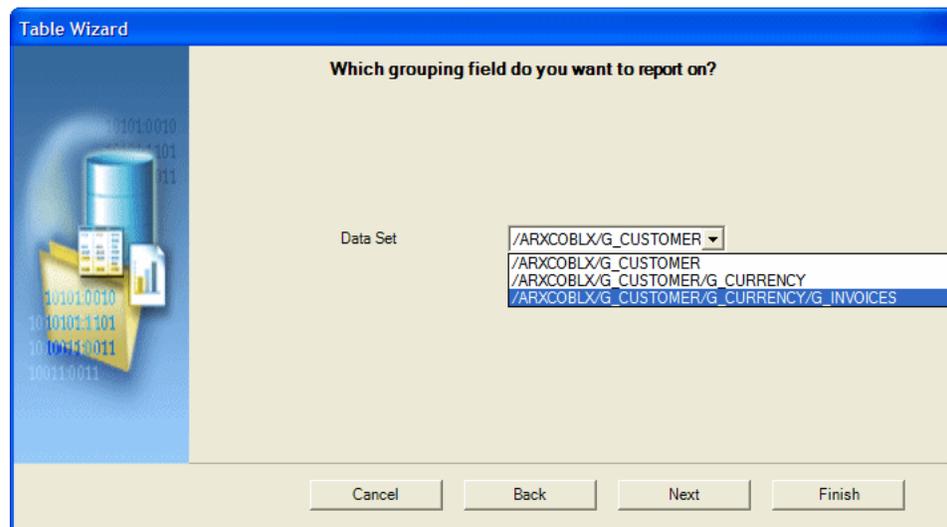
For example, in the Balance Letter sample RTF template (found in the Template Builder installed files under Oracle\BI Publisher\BI Publisher Desktop\samples\RTF Templates), the sample XML file contains three data groups as follows:

- 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 the XML data file. Select the group that contains the data fields for the table.

[Figure 5–7](#) shows the Table Wizard Step 2: Selecting Table Data.

Figure 5–7 The Table Wizard Step 2: Selecting Table Data



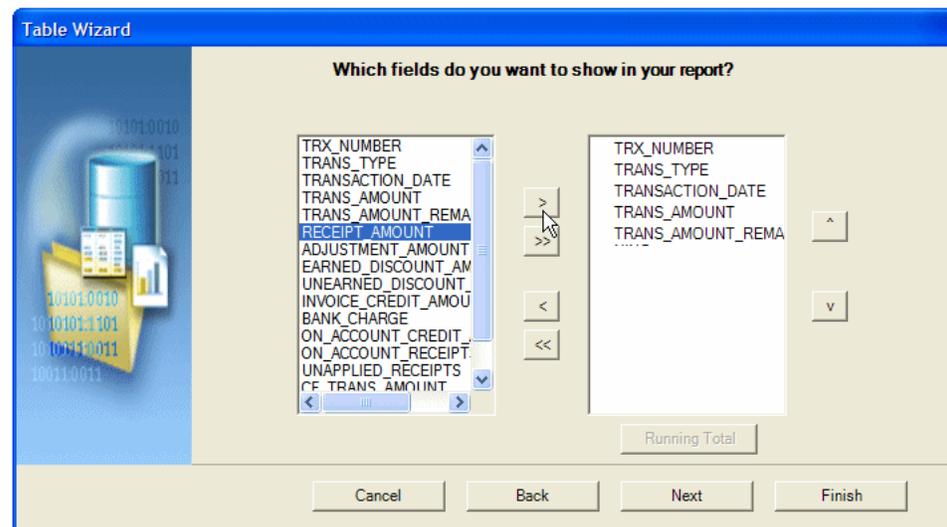
To build a table to list the invoices contained in the data, select: ARXCOBLX/G_CUSTOMER/G_CURRENCY/G_INVOICES as the data set.

5.4.2.3 Step 3: Selecting Data Fields

The Table Wizard presents the data fields from the selected data set.

Figure 5–8 shows the Table Wizard Step 3: Selecting Data Fields.

Figure 5–8 The Table Wizard Step 3: Selecting Data Fields



Use the shuttle buttons to select the data fields to show in the table. Use the up and down arrows to reorder the fields after selecting them.

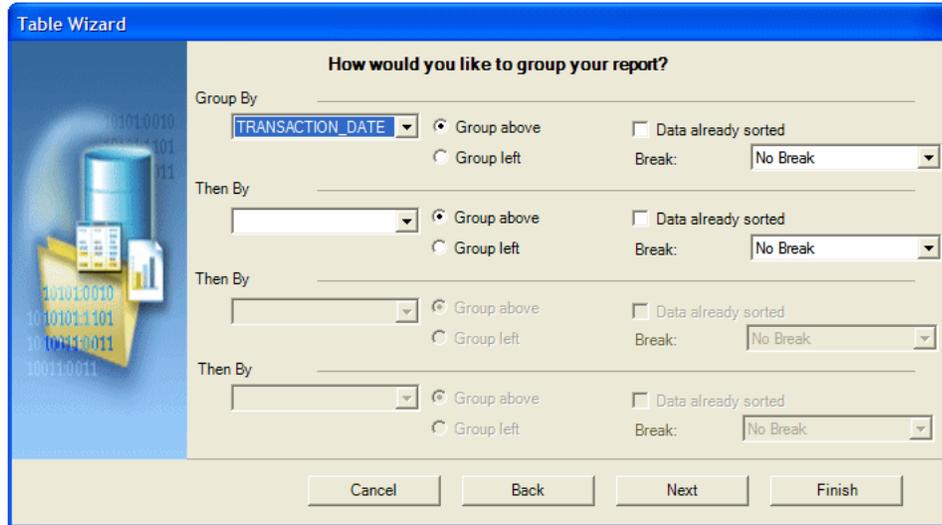
5.4.2.4 Step 4: Grouping the Table

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.

Figure 5–9 shows the Table Wizard Step 4: Grouping the Table.

Figure 5–9 The Table Wizard Step 4: Grouping the Table



There are two options for grouping: Group Left or Group Above. Group Left creates a nested table. The Group By field displays to the left in the outer table. Group Above creates a new table for each new value of the 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 in Figure 5–10.

Figure 5–10 Results of the Group Left Option

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, as shown in Figure 5–11.

Figure 5–11 Results of the Group Above Option**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, then select the **Data already sorted** check box. This selection improves performance.

5.4.2.5 Step 5: Inserting a Break for the Group

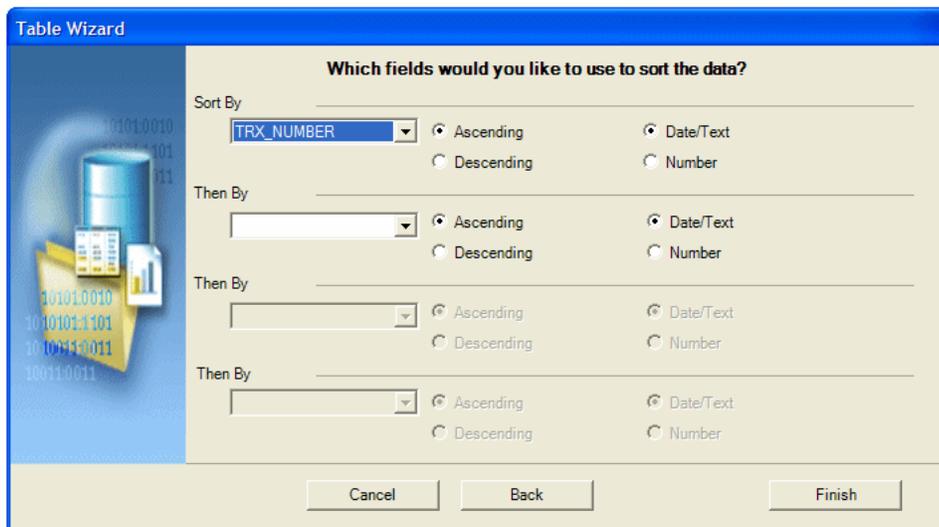
Use the **Break** option to insert either a **Page** break or **Section** break after each occurrence of this group. Note that a **Section** break can only be created on the top-level group. The subsequent grouping options only display the **Page** break option.

A page break starts the next group on a new page; a section break starts 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.

5.4.2.6 Step 6: Sorting 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" comes before "2" (alphanumerical order). If number is selected, "2" comes before "12".

Figure 5–12 shows the Table Wizard Step 6: Sorting the Table.

Figure 5–12 The Table Wizard Step 6: Sorting the Table

5.4.2.7 Step 7: Clicking Finish

Click **Finish** to create the table and insert it to the Microsoft Word document.

5.4.2.8 Step 8: Customizing 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.

5.4.3 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 is 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 are copied to the template when you click the Insert button.

5.4.3.1 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 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, then the node and all children are copied. However, if you use the right mouse button for dragging, a dialog is displayed 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.

5.4.3.2 Defining the Layout

When you select an element in the right Template pane, you see its properties as well as a preview of how the node is 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.

5.4.3.3 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.

5.4.3.4 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 might want to group all invoices by customer. You can select a child element of the selected element as a grouping criterion. For more information see [Section 5.4.3.6, "Grouping."](#)

- **Show Grouping Value**

This property is shown only if you have selected a node created by the Grouping functionality. By default, the field you have selected to group the data by is displayed in the report. If you do not want the grouping data field displayed, then 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 is 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 is inserted between each element after the first occurrence.

Tip: 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, then a section break is 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 typically use this option if you want to print multiple documents (for example invoices or purchase orders) to a single PDF file.

5.4.3.5 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 the document.

5.4.3.6 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 makes this node (e.g. quarter) the parent of the other child nodes (Customer and Amount).

The new structure looks like the following:

```

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.

5.4.3.7 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

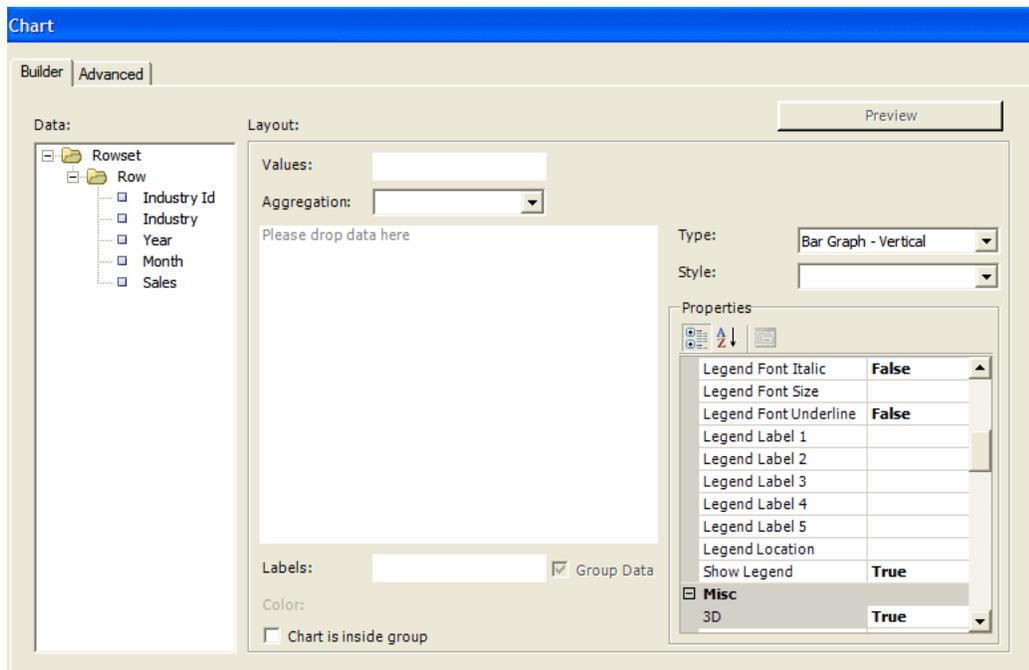
Form fields representing data elements are replaced with the data when the template is processed. Form fields indicating repeating sections are shown as **for-each** and **end for-each** in the document.

Note: If you have selected the Abbreviated form field display option, then the for-each and end for-each form fields are displayed 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.

5.4.4 Inserting a Chart

Use the Chart dialog to insert a chart into a template.

Figure 5–13 shows the Chart dialog.

Figure 5–13 The Chart Dialog

5.4.4.1 Chart Type

BI Publisher supports a large variety of chart types. Expand the **Type** list to select the chart type for this template.

5.4.4.2 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).

The Values field changes depending on the Chart Type that you select:

- Combination Graph - Enables three fields for the Value selections.
- Scatter Graph - Compares pairs of values. Drag and drop the X and Y data elements to compare.
- Bubble Graph - Compares sets of three values. Similar to the scatter graph, 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.

5.4.4.3 Aggregation

You can choose to aggregate the Values data as a sum, a count, or an average.

5.4.4.4 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, then selecting Group Data accumulates the values for Year, so that only one occurrence of each year is displayed in the chart. If you do not select Group Data, then the value for every occurrence of Year in the data is plotted separately.

5.4.4.5 Color

If you want to add a series element to the chart, then drag and drop the element to display as a series. Each value is displayed as a new color in the graph.

5.4.4.6 Chart is Inside Group

Select this box if the chart is inside a grouping and you want the chart to display data only for the occurrences of the data elements within the group.

5.4.4.7 Style

Select a color scheme and style for the chart.

5.4.4.8 Properties

The properties region enables you to change value and label display names, select color, font, and other display options for the chart. The properties list changes depending on the chart selection.

5.4.4.9 Preview

Click **Preview** to display the chart with the sample data.

5.4.4.10 Group Data

By default the data is grouped by the Value element and aggregated by sum.

If you deselect the Group Data check box, the each occurrence of the value element is charted and aggregation functions are not available.

5.4.4.11 Editing an Inserted Chart

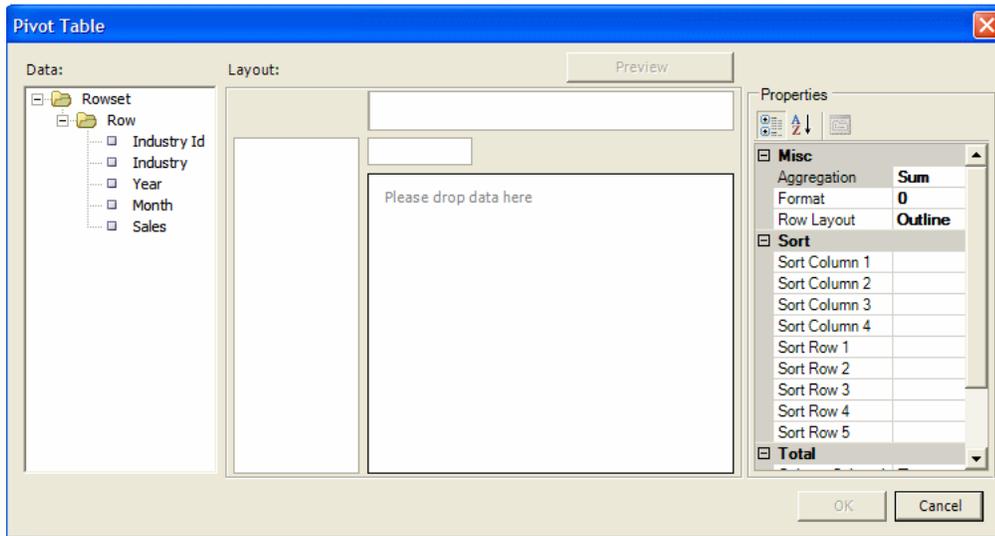
To edit a chart that you have already inserted into the template, right-click the chart and select BI Publisher Chart from the menu. This invokes the chart dialog to enable you to edit the chart.

5.4.5 Inserting a Pivot Table

To insert a pivot table:

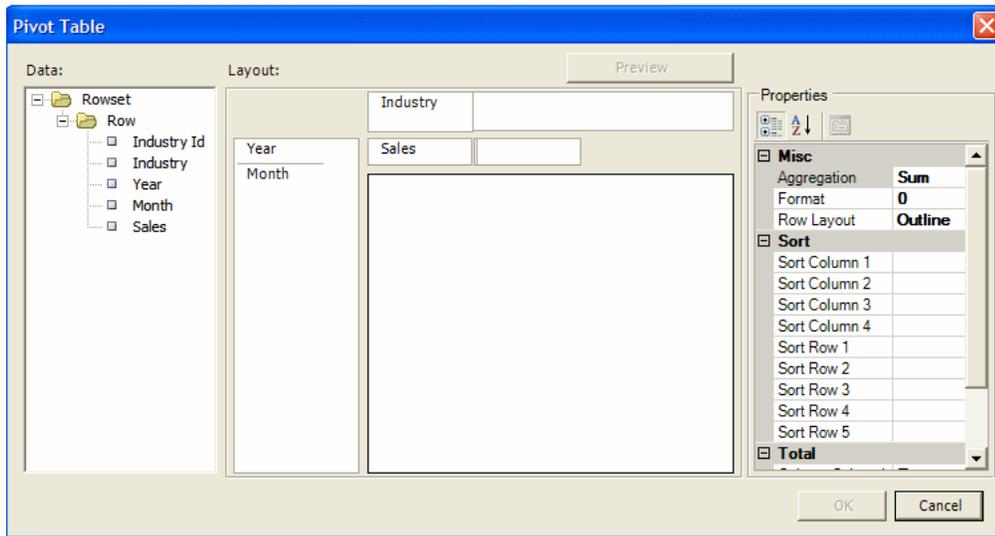
1. On the BI Publisher menu on the **Insert** group, click **Pivot Table**. The **Pivot Table** dialog presents the data in the left pane with empty **Layout** panes on the right for you to drag and drop data elements. [Figure 5-14](#) shows the Pivot Table dialog.

Figure 5–14 The Pivot Table Dialog



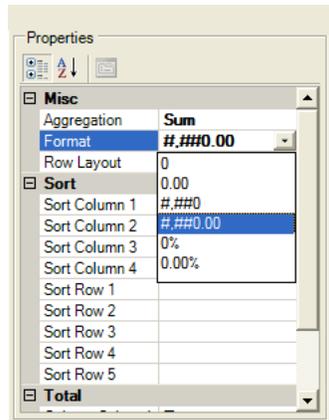
2. Drag and drop the elements from the Data pane to the Layout pane to build the pivot table structure. In [Figure 5–15](#), the layout shows Sales by Industry accumulated by Year and by Month.

Figure 5–15 A Layout Showing 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**, as shown in [Figure 5–16](#).

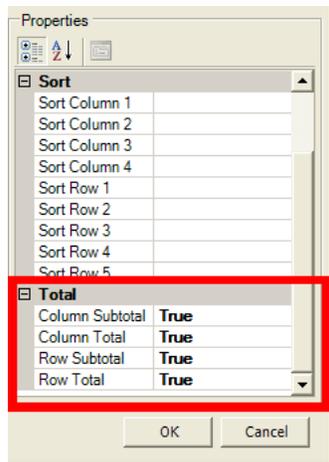
Figure 5–16 The Format Option



4. By default subtotals for rows and columns are displayed. You can choose not to display the subtotals by setting the properties to False.

Figure 5–17 shows the properties for setting totals and subtotals.

Figure 5–17 Properties for Setting Totals and Subtotals



5. Click **Preview** to see how the pivot table is displayed before you insert it into the template. Click **OK** to insert the pivot table into the template. Figure 5–18 shows how the pivot table is displayed in the template.

Figure 5–18 Example of a Pivot Table in a Template

CH	G	INDUSTRIYE	Total
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 is generated as shown in Figure 5–19.

Figure 5–19 A Pivot Table at Runtime

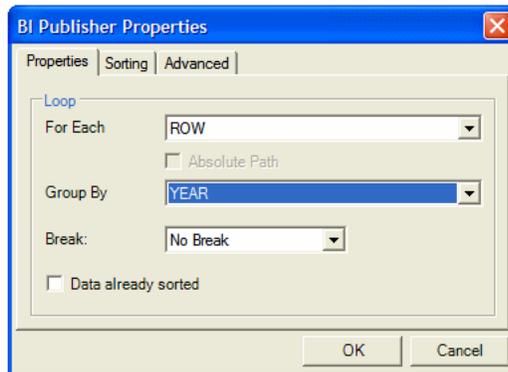
		Building material & garden eq. & supplies dealers	Clothing & clothing accessories stores	Electronics & appliance stores	Food services & drinking places	Food & beverage stores	Furniture & home furnishings stores
2006		48,531.00	31,528.00	15,472.00	61,967.00	81,303.00	17,073.00
	February	23,146.00	15,611.00	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

5.4.6 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, as shown in Figure 5–20.

Figure 5–20 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 that the 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 updatable because it is already defined by the preexisting outer loop. The For Each field is displayed as "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 the 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 actually regroups 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 want to insert a break after each occurrence of this group.

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 is not available.

Note also that when you insert a section break, the page numbering is reset, headers and footers are reset, and any running calculations are 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.

If you are sorting by four criteria and the XML data element names are long, then you might exceed the character length limitation (393 characters) of the Microsoft Word form field.

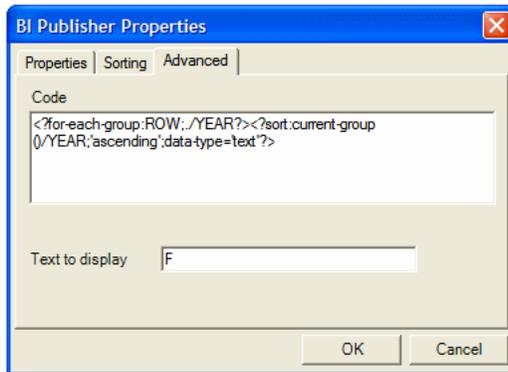
5. The Advanced tab enables you to edit the code directly and to enter Text to Display for the field.

The Code region displays the code and processing instructions that the Template Builder has inserted for the field. You can edit this if you want to change the processing instructions for this field.

The Text to Display field shows how this field displays in the 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 are less intrusive to the look and feel of the template.

Note: You can set the default display text as Descriptive or Abbreviated using the Options tab.

Figure 5–21 shows the Advanced tab of the BI Publisher Properties dialog.

Figure 5–21 The BI Publisher Properties Dialog: Advanced Tab

- When you have completed the dialog options, click OK. This inserts the form fields in the template. By default, the beginning for-each form field displays the text "F" and is inserted at the beginning of the selected template section. At the end of the selection, an "E" form field is inserted to denote the end of the repeating group.

5.4.7 Creating Grouping Fields Around an Existing Block

To create a group around an existing block of text or elements in a template:

- Select the block of text. For example, a table row.

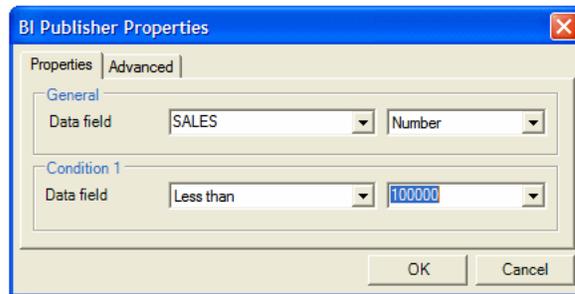
If any preexisting BI Publisher tags are included in the block, then you must include the beginning and ending tags. For example, if the block contains any opening for-each, if, or for-each-group tags, then you must include the end for-each, end-if, and end for-each-group tags in the selection.
- On the Oracle BI Publisher menu, on the Insert group, click **Repeating Group**.
- In the BI Publisher Properties dialog, enter the fields to define the group as described in [Section 5.4.6, "Inserting a Repeating Group."](#)
- Click **OK** to insert the grouping fields around the block. For example, if the block is a table row, then the begin field is inserted at the beginning of the first cell and the end field is inserted at the end of the last field.

5.4.8 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, the data contains sales information. The report contains a table that displays sales by industry. You want this table in the 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.

- 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, then select the region that contains the table. Note that the region must be inside a loop.
- On the Oracle BI Publisher menu, on the Insert group, click **Conditional Region**. [Figure 5–22](#) shows the BI Publisher Properties dialog for a Conditional Region.

Figure 5–22 The BI Publisher Properties 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 can 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 that contain the conditional logic are inserted around the region. The beginning form field displays the text "C" and the form field closing the region displays 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**.

5.4.9 Inserting Conditional Formatting

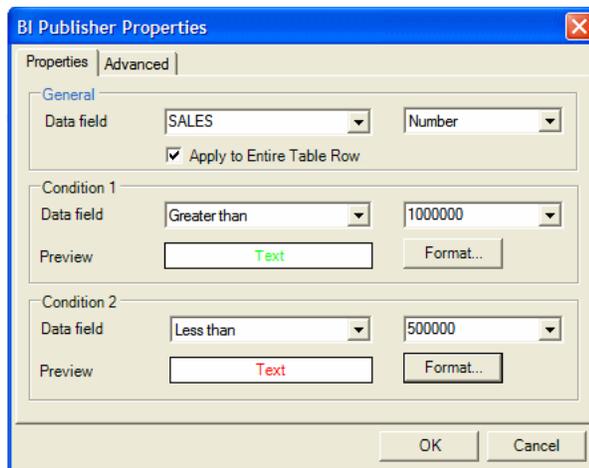
Using the Conditional Format feature you can insert simple conditional formats to apply to table rows or cells. The dialog provides several common options that you can select and the Template Builder inserts the code automatically. The Conditional Format dialog supports two conditions per field.

Note: 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**. [Figure 5–23](#) shows the BI Publisher Properties dialog for a Conditional Format.

Figure 5–23 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, then select this box.

Condition 1) Data field - Select the comparison operator.

Select the value to meet the condition. You can 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 displays the format chosen.

Figure 5–24 shows the Format dialog.

Figure 5–24 The Format Dialog



5. Define a second condition if desired.
6. Click **OK**. The conditional format field is 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**.

5.5 Previewing a Template

The Preview menu group enables you to preview the RTF template with sample XML data.

Note: If you have not already done so, then you must load sample data to the Template Builder to preview the report. See [Section 5.3, "Accessing Data for Building Templates."](#)

From the Preview group, select the output format. If you have not yet saved the template as an RTF file, then you are 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 the 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.

5.6 Template Editing Tools

This section describes additional tools provided with the Template Builder to help you validate and edit the template. This section includes:

- [Section 5.6.1, "Editing and Viewing Field Properties"](#)
- [Section 5.6.2, "Validating a Template"](#)
- [Section 5.6.3, "Using the Field Browser"](#)
- [Section 5.6.4, "Checking Accessibility"](#)

5.6.1 Editing and Viewing Field Properties

Once you have inserted a data field (see [Section 5.4.1, "Inserting a Field"](#)) you can view or edit the field properties in the BI Publisher Properties dialog.

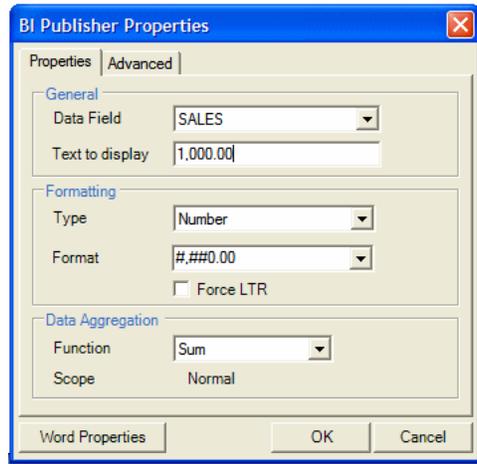
To invoke the BI Publisher Properties dialog, perform one of the following:

- Double-click the field

- Right-click the field, from the menu select BI Publisher, then Properties

Figure 5–25 shows the BI Publisher Properties dialog: note the Properties tab, the Advanced tab, and the Word Properties button. Some fields might display only the Advanced tab.

Figure 5–25 The BI Publisher Properties Dialog



5.6.1.1 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 is 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 determines 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, then the field shows the sum of all occurring values for this data field depending on the scope (see below). See also [Section 5.4.1, "Inserting a Field"](#) for information on aggregation functions.

Scope (informational only) - This field has two possible values:

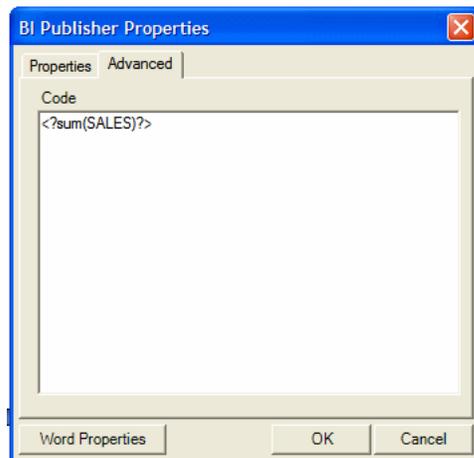
- Group Item** - Indicates that the data field is inside a group. If you choose to perform a function on the field, then only the occurrences of the field within the current group are included in the aggregation.
- Normal** - Indicates that the field is not inside a group. Aggregation functions are performed on all occurrences of the field in the data.

5.6.1.2 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), then the BI Publisher Properties dialog displays this tab only.

Use this tab to edit or add code to the form field manually. Click **OK** to update the template. [Figure 5–26](#) shows the Advanced tab.

Figure 5–26 The BI Publisher Properties Dialog: Advanced Tab



5.6.1.3 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.

5.6.2 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 the template:

1. On the BI Publisher menu, on the Tools group, click **Validate Template**.

If there are no validation errors, then a "No Error found" message is returned. If an error is found, then an error message is displayed. You can use the Field Browser to help locate the error.

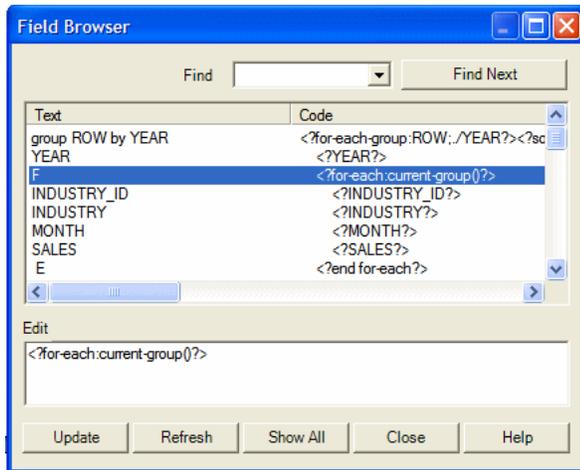
5.6.3 Using the Field Browser

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**.

[Figure 5–27](#) shows the Field Browser dialog.

Figure 5–27 The Field Browser Dialog



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 is selected in the Microsoft Word document.

If you select some part of the text before opening the Field Browser, then the dialog shows only the fields in the selection. If no text is selected, then the field browser shows all fields in the document.

The options in the Field Browser are described in the [Table 5–1](#).

Table 5–1 Field Browser Options

Option	Description
Edit	You can update processing instructions directly from the Field Browser dialog. 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 .
Refresh	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.
Show All	If you opened the browser with a part of the document selected, then you see only the form fields in the selected area. Click Show All to see all the form fields in the document.
Close	Click Close to close the field property browser. The only button does not automatically update any changes in the edit field, therefore ensure that you select Update if you want to save edits.

5.6.4 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 generates a report that indicates 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 cannot determine if the accessibility feature is present and generates a warning. The report designer can then verify that the accessibility features are present.

For information on how to add these features to the template, see [Appendix C, "Designing Accessible Reports."](#)

5.7 Uploading a Template to the BI Publisher Server

If you used the Open Template dialog to connect to BI Publisher, and load the data to the Template Builder, or if you downloaded an existing template from the BI Publisher catalog, then you can upload the new or updated layout back to the report definition on the server. See [Section 5.2.2.1, "Working in Connected Mode."](#)

If you downloaded an existing template and want to upload the modifications to the template, then select **Upload Template** from the Oracle BI Publisher menu.

If this is a new template for the report definition, then 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.

5.8 Using the Template Builder Translation Tools

The Template Builder provides tools to help you create and test translations for templates.

5.8.1 About Translations

There are two options for adding translated templates to a 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, see [Chapter 15, "Translation Support Overview and Concepts."](#)

To use the Template Builder translation tools to create 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 the Template Builder installation (located in the BI Publisher\BI Publisher Desktop\demos folder where you installed BI Publisher Desktop).

5.8.2 Extracting Text to an XLIFF File for Translation

This menu item allows you to create a standard XLIFF translation file that contains the boilerplate text from the 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.

To extract text to an XLIFF file for translation:

1. From the BI Publisher menu, select **Tools**, then **Translate Template**, then **Extract Text**.
2. You are 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 want to translate the template manually, 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 [Section 15.2, "Working with Translation Files."](#)
4. When done, you can preview the translation. Then upload the file to the BI Publisher report definition.

5.8.3 Previewing the Template and Translation File

To preview the template with the translated XLIFF file applied:

1. From the BI Publisher, in the Tools group, click **Translation**, then **Preview Translation**.
2. You are prompted to select the saved XLIFF file. Locate the file, and click **Open**.

The Template Builder merges the sample data, the translation file, and the RTF template to generate a PDF for preview.

5.8.4 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 a template into an XLIFF file that can be translated and then applied at runtime, if the reports for additional languages only require the translation of these text strings, then you only need to supply translated XLIFF files to accompany the 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 invokes a warning message that localizing the template overwrites the template. Click **OK**.

2. You are prompted to select the XLIFF translation file. Locate the appropriate file and click **Open**.
The translated XLIFF file is applied to the template that you currently have open in Microsoft Word.
3. Save the localized template.
4. Upload the template file to the appropriate report definition in the BI Publisher catalog. Select the appropriate locale in the upload dialog.

5.9 Setting Options for the Template Builder

Access the **Options** dialog as follows: In the **Options** group, click **Options**.

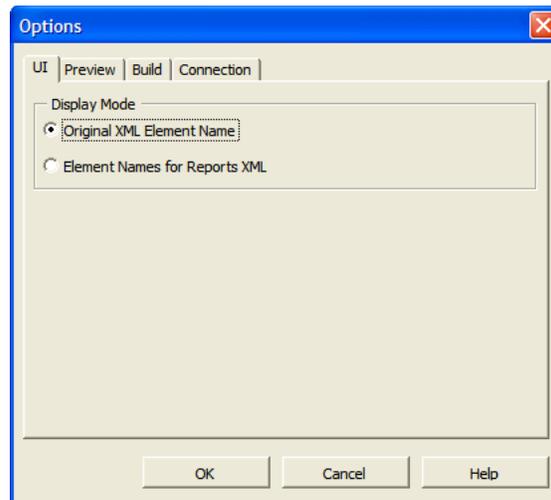
The Options dialog contains four tabs: UI, Preview, Build, and Connection, as described in the following sections.

5.9.1 Setting UI Options

Use the Options dialog: UI tab to set options that influence the look and feel of the Template Builder.

Figure 5–28 shows the Options dialog: UI tab.

Figure 5–28 Options Dialog: UI Tab



The tree view that shows 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 ("_").

5.9.2 Setting Preview Options

The Options dialog: Preview tab allows you to specify options that influence the Preview functionality of the Template Builder.

Figure 5–29 shows the Options dialog: Preview tab.

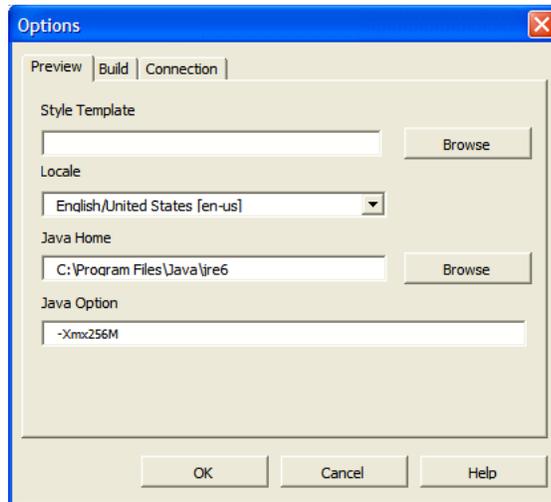
Figure 5–29 The Options Dialog: Preview Tab

Table 5–2 describes the options available from the Preview tab.

Table 5–2 Preview Options

Option	Description
Style Template	If you have a BI Publisher Style Template available locally, then 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, Chapter 11, "Creating and Implementing Style Templates."
Locale	You can choose the language and territory used for previewing the template. While this change does 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.
Java Home	The Preview (and export functionality) requires Java code. You can change the path to the 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 the environment variables of Windows.
Java Option	Specify the memory to reserve for the Template Builder to process the template. The default value is -Xmx256M.

5.9.3 Setting Build Options

Use the Options dialog: Build tab to specify options that influence how the Template Builder generates tables and forms.

[Figure 5–30](#) shows the Options dialog: Build tab.

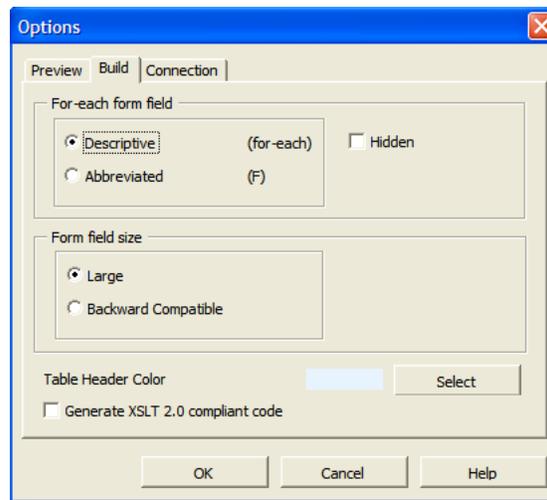
Figure 5–30 The Options Dialog: Build Tab

Table 5–3 describes the options available from the Build tab.

Table 5–3 Build Tab Options

Option	Description
For-each form field	<p>Select how the Template Builder creates the form fields for processing instructions in the Insert Table/Form dialog.</p> <p>The Descriptive option (for example: for-each Invoice) 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 Abbreviated option (for example: F) provides a one letter abbreviation for each instruction.</p> <p>Select the Hidden 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>
Form Field Size	<p>Large - 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 the template is intended for use with the 10g version of BI Publisher, use the Backward Compatibility setting.</p> <p>Backward Compatible - 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>
Table Header Color	<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 the templates.</p>

Table 5–3 (Cont.) Build Tab Options

Option	Description
Generate XSLT 2.0 compliant code	<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</i>.</p> <p>By default, BI Publisher is compatible with XSLT 1.0. If you want to use XSLT and XPath 2.0 features in the template, then enable this option. This configuration is performed at the template level. The template-level setting overrides the server setting.</p>

5.9.4 Setting Connection Options

Options on this tab are reserved for a future release.

5.10 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 the templates
- Use security settings for PDF files

See the appendix "Configuration File Reference" in *Oracle Fusion Middleware Administrator's Guide for Oracle Business Intelligence Publisher* for the syntax of the configuration file.

5.11 BI Publisher Menu Reference

When you install the Template Builder the next time you open Microsoft Word, you see the Oracle BI Publisher menu.

Note: If you are using Microsoft Word 2007, then you might have to modify the Add-In settings. Click the Office Button, click Word Options, then click Add-Ins.

5.11.1 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 [Section 5.2.2.1, "Working in Connected Mode."](#)

Figure 5–31 shows the Online group of commands.

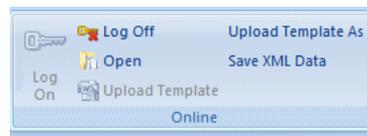
Figure 5–31 The Online Group of Commands

Table 5–4 describes the commands available for the Online group.

Table 5–4 Online Group Commands

Command	Description
Log on	Enables you to log in to BI Publisher. Enter your user name 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 Open Template dialog is displayed. Note: You must log in directly to the BI Publisher server. For example: http://www.example.com:7001/xmlpserver .
Open	After you log on, this command becomes available to enable you to open a report in the BI Publisher catalog.
Upload Template	If you used the Open Template 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.
Upload Template As	If you used the Open Template 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.
Save XML Data	If you are working in connected mode, then use this command to save the data to a local directory if you also need access to the data in disconnected mode.

5.11.2 About the Load Data Group

The Load 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 [Section 5.3, "Accessing Data for Building Templates"](#) for more options for loading data to the Template Builder.

Figure 5–32 shows the Load Data group of commands.

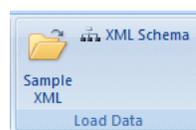
Figure 5–32 The Load Data Group of Commands

Table 5–5 describes the commands available for the Load Data group.

Table 5–5 Load Data Group Commands

Command	Description
Sample XML	This command enables you to load a previously saved sample XML file from the report data source. If you are not connected to the BI Publisher server, use this method to load the data.

Table 5–5 (Cont.) Load Data Group Commands

Command	Description
XML Schema	This command enables you to load an XML Schema file (.xsd) that contains the fields available in the 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.

5.11.3 About the Insert Group

Use the Insert group of commands to insert the layout components to the template. For more information, see [Section 5.4, "Inserting Components to the Template."](#)

[Figure 5–33](#) shows the Insert group of commands.

Figure 5–33 The Insert Group of Commands

[Table 5–6](#) describes the commands available for the Insert group.

Table 5–6 Insert Group Commands

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.
Field	This function allows you to select fields from the data source and insert them into the template. Note: As a beginner, you should use Insert Fields only for data fields that are unique - none repeating - in the document. See Section 5.4.2, "Inserting a Table Using the Table Wizard" for additional information on how to insert repetitive fields.
Table/Form	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.
Repeating Group	Enables you to select or define a group of elements that you want repeated for each occurrence of an element in the data.
Conditional Format	Enables you to define simple conditional formats to apply to table rows or cells.
Conditional Region	Enables you to insert a conditional statement around a region of the template.

Table 5–6 (Cont.) Insert Group Commands

Command	Description
All Fields	<p>This function inserts all fields found in the XML data into the document. It also inserts processing instructions into the document that repeats a section - such as a table row - when the associated XML element is repeated.</p> <p>Note: 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>

5.11.4 About the Preview Group

The Preview group of commands enables you to preview the RTF template with the sample XML data. The preview menu offers PDF, HTML, RTF, PowerPoint, Excel (MHTML format) and EXCEL2000 as output formats. When you select any of these output formats, the Template Builder merges the data into the template and creates the output document.

Figure 5–34 shows the Preview group of commands.

Figure 5–34 The Preview Group of Commands

Note: You must have Adobe Acrobat Reader version 5.0 or higher installed to preview documents in PDF format.

5.11.5 About the Tools Group

For more information about using the commands in the Tools group refer to [Section 5.6, "Template Editing Tools"](#) and [Section 5.8, "Using the Template Builder Translation Tools."](#)

Figure 5–35 shows the Tools group of commands.

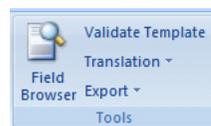
Figure 5–35 The Tools Group of Commands

Table 5–7 describes the commands available for the Tools group.

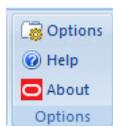
Table 5–7 Tools Group Commands

Command	Description
Field Browser	The field browser is a tool for advanced users who must change the BI Publisher commands that are 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.
Validate Template	The validation function checks the template for incorrect use of BI Publisher commands and unsupported elements in the Word file.
Translation	Includes the following subcommands: <ul style="list-style-type: none"> ▪ Extract Text - Enables you to create a standard XLIFF translation file that contains the boilerplate text from the 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 Section 15.2, "Working with Translation Files." ▪ Preview Translation - Enables you to preview the template as a PDF file using a specified XLIFF translation file. This functionality enables you to test translation files. ▪ Localize Template - Applies a translation file to an RTF template. This means that in the current RTF template all boilerplate text is translated. The main function of this feature is to create a language-specific version of a template.
Export	Includes the following functions: <ul style="list-style-type: none"> ▪ XSL-FO Stylesheet - Allows you to convert the RTF template into an enhanced XSL-FO stylesheet. This function can be used to generate XSL-FO for debugging or further customization. ▪ Formatted XML - Enables 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. ▪ PDF - Converts the Word document to PDF.

5.11.6 About the Options Group

The Options group of commands allows you to define preferences and options for using BI Publisher and access online help.

[Figure 5–36](#) shows the Options group of commands.

Figure 5–36 The Options Group of Commands

See [Section 5.9, "Setting Options for the Template Builder."](#)

Creating Excel Templates

This chapter describes creating report templates for BI Publisher in Microsoft Excel.

This chapter includes the following sections:

- [Section 6.1, "Overview"](#)
- [Section 6.2, "Concepts"](#)
- [Section 6.3, "Building a Simple Template"](#)
- [Section 6.4, "Formatting Dates"](#)
- [Section 6.5, "Defining BI Publisher Functions"](#)
- [Section 6.6, "Preprocessing the Data Using an XSL Transformation \(XSLT\) File"](#)
- [Section 6.7, "Using the Template Viewer to Debug a Template"](#)

6.1 Overview

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

6.1.1 Features of Excel Templates

With Excel templates you can:

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

6.1.2 Limitations of Excel Templates

The following are limitations of Excel templates:

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

- 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

6.1.3 Prerequisites

Following are prerequisites for designing Excel templates:

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

6.1.4 Supported Output

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

6.1.5 Desktop Tools for Excel Templates

BI Publisher provides a downloadable add-in to Excel that enables you to preview a template with sample data. This facilitates design by enabling you to test and edit a 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**.

6.1.6 Sample Excel Templates

The Template Builder includes sample Excel templates.

To access the samples from a Windows desktop:

1. Click **Start, Programs, Oracle BI Publisher Desktop, Samples**, then **Excel**.

This action launches the folder that contains the Excel sample templates.

6.2 Concepts

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

6.2.1 Identifying Data Field Placeholders and Groups

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

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

Note: See [Section 6.6, "Preprocessing the Data Using an XSL Transformation \(XSLT\) File"](#) and [Section 6.5.3, "Grouping Functions"](#) for more information about these options.

6.2.2 Use of Excel Defined Names

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

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

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

6.2.3 About the XDO_ Defined Names

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

6.2.4 Using Native Excel Functions

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

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

6.2.5 About the XDO_METADATA Sheet

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

provide specific report features. Other formatting and calculations can be expressed in XSLT.

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

Note: For more information see [Section 6.3.5.1, "Format of the XDO_METADATA Sheet"](#) and [Section 6.5, "Defining BI Publisher Functions."](#)

6.3 Building a Simple Template

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

Step 1: [Obtain Sample XML Data from the Data Model](#)

Step 2: [Open the BlankExcelTemplate.xls File and Save As the Template Name](#)

Step 3: [Design the Layout in Excel](#)

Step 4: [Assign the BI Publisher Defined Names](#)

Step 5: [Prepare the XDO_METADATA Sheet](#)

Step 6: [Test the Template](#)

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

You need sample data in order to know the field names and the hierarchical relationships to properly mark up the template. For information on saving sample data from the report data model, see the "Testing Data Models and Generating Sample Data" section in *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, then 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 are 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>

```

6.3.2 Step 2: Open the BlankExcelTemplate.xls File and Save As the Template Name

Note: It is recommended that you install the Template Builder for Excel. For information on downloading the tool, see [Section 6.1.5, "Desktop Tools for Excel Templates."](#)

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 the Excel workbook.

Tip: If you are building a new template from an existing template, then be sure to clear any existing defined names in the template sheet. See [Section 6.3.4.4, "Deleting Defined Names and Groups."](#)

To open the BlankExcelTemplate.xls file:

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 the selected name in the Microsoft Excel 97-2003 Workbook format (*.xls).

6.3.3 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 [Figure 6-1](#).

Figure 6–1 A Sample Design

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

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

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

For example: XDO_*?EMPLOYEE_ID?*

- Data groups: XDO_GROUP_*?group_name?*

where

XDO_GROUP_ is the required prefix and

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

For example: XDO_GROUP_*?DEPT?*

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

6.3.4.1 Applying a Defined Name to a Cell

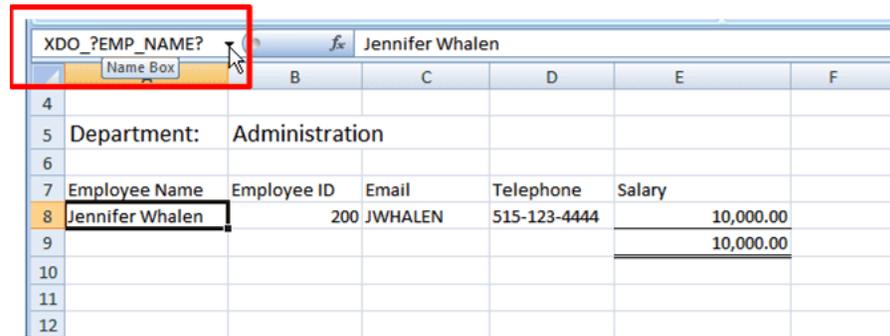
To apply a defined name to a cell:

1. Click the cell in the Excel worksheet.
2. Click the Name box at the left end of the formula bar. The default name is displayed in the Name box. By default, all cells are named according to position, for example: A8.

3. In the Name box, enter the name using the XDO_ prefix and the tag name from the data. For example: XDO_?EMP_NAME?
4. Press Enter.

Figure 6–2 shows the defined name for the Employee Name field entered in the Name box.

Figure 6–2 Defined Name for the Employee Name Field Entered in the Name Box



	B	C	D	E	F
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					
12					

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

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

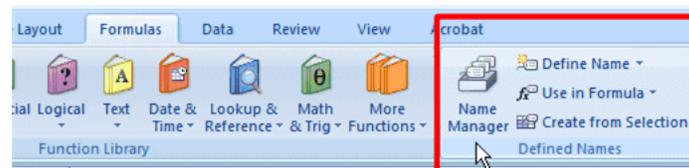
You cannot edit the Name box while you are editing the cell contents.

The name cannot be more than 255 characters in length.

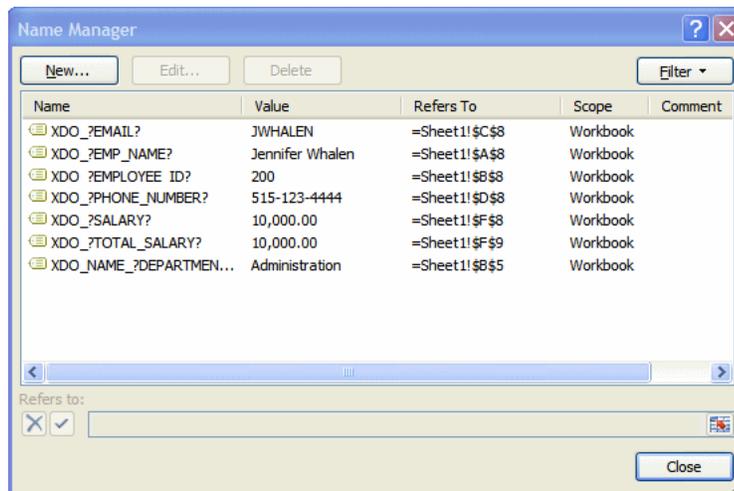
6. For the total salary field, a calculation is mapped to that cell. For now, name that cell XDO_?TOTAL_SALARY?. The calculation is added later.

After you have entered all the fields, you can review the names and make any corrections or edits using the Name Manager feature of Excel. Access the Name Manager from the Formulas tab in Excel, as shown in Figure 6–3.

Figure 6–3 The Name Manager



After you have named all the cells for this example, the Name Manager dialog contains the entries as shown in Figure 6–4.

Figure 6–4 The Name Manager Dialog

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

6.3.4.2 Understanding Groups

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

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

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

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

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

establishes the grouping desired for the report.

6.3.4.3 Creating Groups in the Template

To create groups in the template:

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

Figure 6–5 shows the XDO_GROUP_ defined name entered for the Employees group. Note that just the row of employee data is highlighted. Do not highlight the headers. Note also that the total cell XDO_?TOTAL_SALARY? is not highlighted.

Figure 6-5 The XDO_GROUP_?EMPS? Defined Name Entered for the Employees Group

	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 [Figure 6-6](#).

Figure 6-6 The Department Name Cell and All Employee Fields

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

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.

6.3.4.4 Deleting Defined Names and Groups

To delete defined names or groups from your template, use the Name Manager. Access the Name Manager dialog from the Formulas tab in Excel. Select the item to delete and click **Delete**.

6.3.5 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.

6.3.5.1 Format of the XDO_METADATA Sheet

The XDO_METADATA sheet must have the format shown in [Figure 6-7](#).

Figure 6–7 Format of the XDO_METADATA Sheet

	A	B	C
1	Version		
2	ARU-dbdrv		
3	Extractor Version		
4	Template Code		
5	Template Type	TYPE_EXCEL_TEMPLATE	
6	Preprocess XSL Template		
7	Last Modified Date		
8	Last Modified By		
9			
10	Data Constraints:		
11			
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 [Section 6.5, "Defining BI Publisher Functions"](#)

6.3.5.2 Creating the XDO_METADATA Sheet

If you copied the XDO_METADATA sheet in Step 2, then skip this section and proceed to [Section 6.3.5.3, "Adding the Calculation for the XDO_?TOTAL_SALARY? Field."](#)

To set up the hidden sheet:

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

6.3.5.3 Adding the Calculation for the XDO_?TOTAL_SALARY? Field

Earlier in this procedure, you assigned the defined name XDO_?TOTAL_SALARY? to the cell that is to display the total salaries listed in the SALARY column. In this step,

you add the calculation to the Data Constraints section of the XDO_METADATA sheet and map the calculation to the XDO_?TOTAL_SALARY? field.

To add the calculation to the field:

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

The completed XDO_METADATA sheet is shown in [Figure 6–8](#).

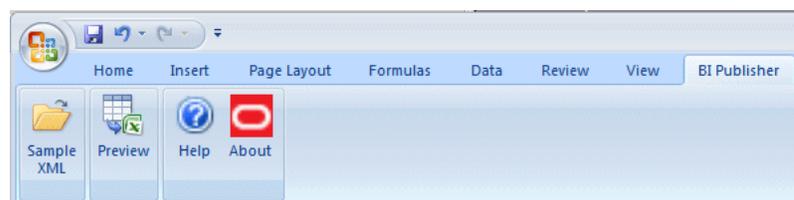
Figure 6–8 A Completed XDO_METADATA Sheet

	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_?TOTAL_SALARY?	<?sum(./SALARY)?>
12		
13		
14		
15		

6.3.6 Step 6: Test the Template

If you have installed the Template Builder for Excel, then the BI Publisher tab is displayed on the ribbon menu as shown in [Figure 6–9](#).

Figure 6–9 The BI Publisher Tab



To preview the report using sample data:

1. Click **Sample XML**. You are prompted to select the sample data file.
2. Click **Preview**.

The sample data is applied to the template and the output document is opened in a new workbook. [Figure 6–10](#) shows the preview of the template with the sample data.

Figure 6–10 A Preview of a Template with Sample Data

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

6.4 Formatting Dates

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

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

Example: Formatting a Canonical Date in Excel

Using the Employee by Department template and data from the first example, assume you want to add the HIRE_DATE element to the layout to and display the date as shown in Column E of [Figure 6–11](#).

Figure 6–11 The Employee by Department Template Showing the Hire Date

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

To format the date:

1. Copy and paste a sample value for HIRE_DATE from the XML data into the cell that is to display the HIRE_DATE field. For example:
 Copy and paste
 1996-02-03T00:00:00.000-07:00
 into the E8 cell.
2. Assign the cell the defined name XDO_?HIRE_DATE? to map it to the HIRE_DATE element in the data, as shown in [Figure 6–12](#).

Figure 6–12 Assigning the Defined Name XDO_?HIRE_DATE?

	A	B	C	D	E	F
1	Employees by Department Report					
2						
3						
4						
5	Department: Admin					
6						
7	Employee Name	Employee ID	Email	Telephone	Hire Date	Salary
8	John Thomas	100001	john@oracle.com	111-333-3333	1996-02-03T00:00:00.000-07:00	10,000
9						10,000
10						

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

3. Insert a new Hire Date column. This is now column F, as shown in [Figure 6–13](#).

Figure 6–13 The New Hire Date Column in Column F

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

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

- To retain the full date and timestamp, enter:
`=--REPLACE(SUBSTITUTE(E8,"T"," "),LEN(E8)-6,6,"")`
- To retain only the date portion (YYY-MM-DD), enter:
`DATEVALUE(LEFT(E8,10))`

After you enter the function, it populates the F8 cell as shown in [Figure 6–14](#).

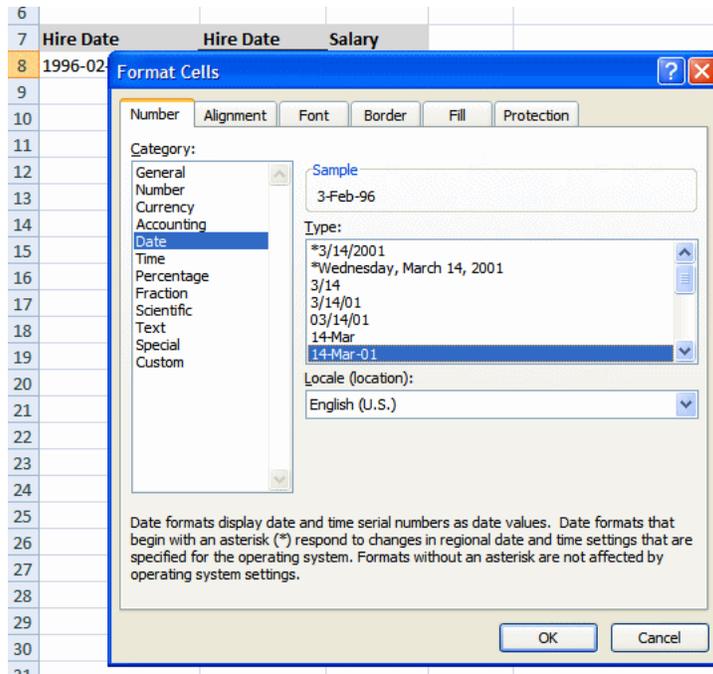
Figure 6–14 Hire Date Cell (F8) Populated

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

5. Apply formatting to the cell.

Right-click the F8 cell. From the menu, select Format Cells. In the Format Cells dialog, select Date and the desired format, as shown in [Figure 6–15](#).

Figure 6–15 Applying the Format for the Date in the Format Cells Dialog



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

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

Figure 6–16 shows column E hidden.

Figure 6–16 Column E Hidden

	A	B	C	D	F	G	H
1	Employees by Department Report						
2							
3							
4							
5	Department:	Admin					
6							
7	Employee Name	Employee ID	Email	Telephone	Hire Date	Salary	
8	John Thomas	100001	john@oracle.com	111-333-3333	3-Feb-96	10,000	
9						10,000	
10							
11							
12							
13							

6.5 Defining BI Publisher Functions

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

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

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

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

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

The functions are described in the following three sections:

- [Section 6.5.1, "Reporting Functions"](#)
- [Section 6.5.2, "Formatting Functions That Rely on Specific Data Attribute Values"](#)
- [Section 6.5.3, "Grouping Functions"](#)

6.5.1 Reporting Functions

Table 6–1 lists functions that you can add to a template using the commands shown and a combination of BI Publisher syntax and XSL. A summary list of the commands is shown in Table 6–1. See the corresponding section for details on usage.

Table 6–1 Reporting Functions

Function	Commands
Section 6.5.1.1, "Splitting the Report into Multiple Sheets"	XDO_SHEET_? with XDO_SHEET_NAME_?
Section 6.5.1.2, "Declaring and Passing Parameters"	XDO_PARAM_?n?
Section 6.5.1.3, "Defining a Link"	XDO_LINK_?link object name?
Section 6.5.1.4, "Importing and Calling a Subtemplate"	XDO_SUBTEMPLATE_?n?
Section 6.5.1.5, "Referencing Java Extension Libraries"	XDO_EXT_?n?

6.5.1.1 Splitting the Report into Multiple Sheets

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

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

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

Table 6–2 describes the column entries.

Table 6–2 Column Entries

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

Table 6–2 (Cont.) Column Entries

Column A Entry	Column B Entry	Column C Entry
XDO_SHEET_NAME_?	<p><?xsl_expression to name the sheet?></p> <p>Example:</p> <p><?concat(../DEPARTMENT_NAME,'-',count(../EMP_NAME))?></p>	<p>(Optional)</p> <p><?original sheet name?></p> <p>Example:</p> <p><?Sheet3?></p>

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

If the data is flat, then you cannot use this command unless you first preprocess the data to create the desired hierarchy. To preprocess the data, define the transformation in an XSLT file, then specify this file in the Preprocess XSLT File field of the header section of the XDO_METADATA sheet. For more information, see [Section 6.6, "Preprocessing the Data Using an XSL Transformation \(XSLT\) File."](#)

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

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

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

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

Example: Splitting the data into multiple sheets

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

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

To split the data into sheets:

1. Enter the defined names for each cell of employee data and create the group for the repeating employee data, as shown in [Figure 6–17](#).

Figure 6–17 Defining Employee Data and the Group for 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 is split by department.

- Enter the values that are described in [Table 6-3](#) in the Data Constraints section of the XDO_METADATA sheet.

Table 6-3 Data Constraints Values

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

The entries are shown in [Figure 6-18](#).

Figure 6-18 Entries for Data Constraints

A	B
1	Version
2	ARU-dbdrv
3	Extractor Version
4	Template Code
5	Template Type TYPE_EXCEL_TEMPLATE
6	Preprocess XSL Template
7	Last Modified Date
8	Last Modified By
9	
10	Data Constraints:
11	XDO_SHEET_? <?//DEPT?>
12	XDO_SHEET_NAME_? <?concat(//DEPARTMENT_NAME,'-',count(//EMP_NAME))?>
13	XDO_?TOTAL_SALARY? <?sum(//SALARY)?>
14	
15	
16	

[Figure 6-19](#) shows the generated report. Each department data now displays on its own sheet, which shows the naming convention specified.

Figure 6-19 Example of a Generated Report

Employee Name	Employee ID	Email	Telephone	Hire Date	Salary
Sheili Baida	116	SBAIDA	515.127.4563	24-Dec-97	2,900
Sigal Tobias	117	STOBIAS	515.127.4564	24-Jul-97	2,800
Karen Colmenare	119	KCOLMENA	515.127.4566	10-Aug-99	2,500
Alexander Khoo	115	AKHOO	515.127.4562	18-May-95	3,100
Guy Himuro	118	GHIMURO	515.127.4565	15-Nov-98	2,600
Den Raphaely	114	DRAPHEAL	515.127.4561	7-Dec-94	11,000
					24,900

6.5.1.2 Declaring and Passing Parameters

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

To declare the parameter, use the command that is described in [Table 6-4](#).

Table 6-4 Command for Declaring Parameters

Column A Entry	Column B Entry
XDO_PARAM_?n? where <i>n</i> is unique identifier for the parameter	<?param@begin:parameter_name;parameter_value?> 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: <?param@begin:Country;US?>

To use the value of the parameter directly in a cell, refer to the parameter as \$parameter_name in the definition for the XDO_ defined name, as described in [Table 6-5](#).

Table 6-5 Using a Parameter Directly

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

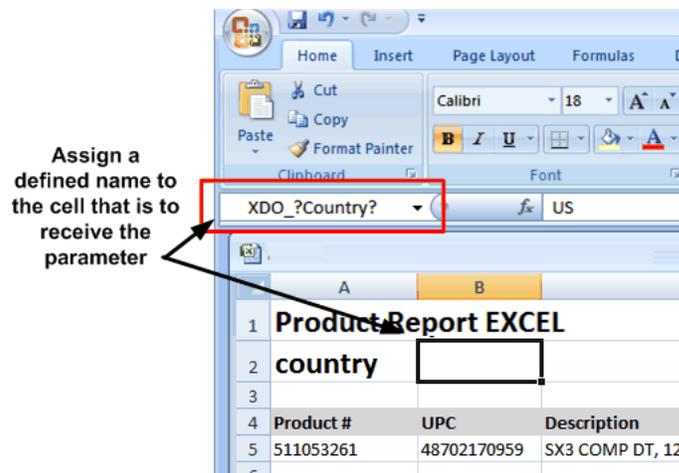
You can also refer to the parameter in other logic or calculations in the XDO_METADATA sheet using \$parameter_name.

Example: Defining and passing a parameter

To declare and reference a parameter named Country:

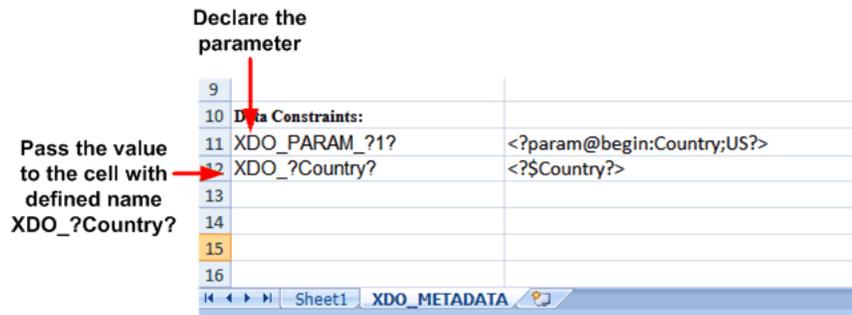
1. In the template sheet, mark the cell with a defined name. In the figure below, the cell has been marked with the defined name XDO_?Country?

Figure 6-20 Cell Marked with the Defined Name XDO_?Country?



- In the hidden sheet assign that cell the parameter value, as shown in [Figure 6-21](#).

Figure 6-21 Assigning a Parameter Value to the XDO_?Country?Cell



6.5.1.3 Defining a Link

Use the XDO_LINK_? command to define a hyperlink for any data cell, as described in [Table 6-6](#).

Example: Defining a Link

Table 6-6 Defining a Link

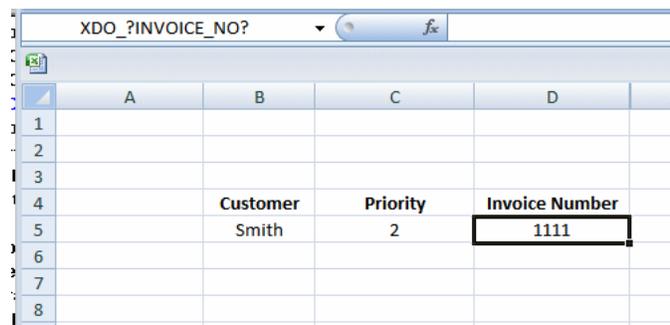
Column A Entry	Column B Entry
XDO_LINK_?cell object name?	<xsl statement to build the dynamic URL>
For example: XDO_LINK_?INVOICE_NO?	For example: <xsl:value-of select="concat('https://server.company.com/document s/invoice_print.show?c_rptno=',./INVOICE_NO)"/>

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

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

- In the template sheet, assign the cell that is to display the INVOICE_NO the XDO defined name: XDO_?INVOICE_NO?, as shown in [Figure 6-22](#).

Figure 6-22 Assigning the Defined Name XDO_?INVOICE_NO?



- In the XDO_METADATA sheet, enter the appropriate values, as described in [Table 6-7](#):

Table 6–7 Generating a Report with a Link

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

The entries in Excel are shown in [Figure 6–23](#).

Figure 6–23 XDO_LINK?_?INVOICE_NO? Entries in Excel

10	Data Constraints:	
		<xsl:value-of
		select="concat('https://server.company.com/documents/invoice_print.show?c_rptno=',./INVOICE_NO)"/>
11	XDO_LINK_?INVOICE_NO?	
12		

The report output is displayed as shown in [Figure 6–24](#). The logic that is defined in the XDO_METADATA sheet is applied to create a hyperlink for each INVOICE_NO entry.

Figure 6–24 Example of the Report Output

Customer	Priority	Invoice Number
Aaron	3	5952174
Abner	3	8280605
Acheson	2	7604618
Addison	3	9645172
Aeschelle	3	9616238
Agers	2	9685385
Ahab	3	9644915
Aiden	3	9663648
Ajackal	3	9685016
Akkers	3	9706403
DZIMA	3	9645249
CGUO	3	9706193

6.5.1.4 Importing and Calling a Subtemplate

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

Note: The Template Builder for Excel does not support preview for templates that import subtemplates.

To import the subtemplate, enter the command shown in [Table 6–8](#).

Table 6–8 Importing a Subtemplate

Column A Entry	Column B Entry
XDO_SUBTEMPLATE_?n? where <i>n</i> is a unique identifier. For example: XDO_SUBTEMPLATE_?1? .	<xsl:import href="xdoxsl:///path to subtemplate.xsb?"/> For example: <xsl:import href="xdoxsl:///Shared Folders/Financial Reports/SubTemplates/MySubTemplate.xsb?"/>

To call the subtemplate, declare the cell name for which the results should be returned in Column A, then enter the call-template syntax with any other XSL processing to be performed. The commands are shown in [Table 6-9](#).

Table 6-9 Calling a Subtemplate

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

For more information on XSL subtemplates and creating the subtemplate object in the catalog, see [Chapter 14, "Designing XSL Subtemplates."](#)

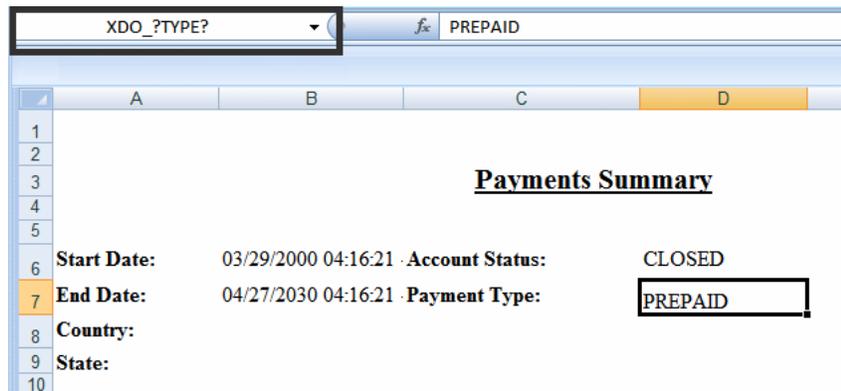
Example: Importing and Calling a Subtemplate

Assume you have the following subtemplate uploaded to the BI Publisher catalog as PaymentsSummary-SubTemplate.xsb. This subtemplate evaluates 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 the Excel template, you have defined a field with the XDO Defined Name XDO_?TYPE?, which is populated based on the string returned from code performed in the subtemplate, as shown in [Figure 6-25](#).

Figure 6-25 Populated XDO_?TYPE? Field



Enter the commands shown in [Table 6–10](#) in the Data Constraints region.

Table 6–10 Commands for Data Constraints Region

Column A Entry	Column B Entry
XDO_SUBTEMPLATE_?1?	<xsl:import href="xdoxsl://Shared Folders/Financial Reports/SubTemplates/PaymentsSummary-SubTemplate.xsb?"/>
XDO_?TYPE?	<xsl:call-template name="BRM_PAY_TYPES"> <xsl:with-param name="pPayType" select="string('10000')"/> </xsl:call-template>

The XDO_SUBTEMPLATE_?1? function imports the subtemplate from the BI Publisher catalog.

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

6.5.1.5 Referencing Java Extension Libraries

You can include the reference to a Java extension library in the template and then call methods from this library to perform processing in the template. Use the command shown in [Table 6–11](#) to reference the Java extension libraries.

Table 6–11 Referencing 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.example.com/XSL/Transform/java/example.com.xmlpublisher.reports.BIPEExtension"?>

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

Example: Calling a Java Extension Library

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

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

After you have declared the library as shown above, specify the cell to which you want to apply the method by entering the XDO defined name in Column A and calling the function in Column B. [Table 6–12](#) shows example commands.

Table 6–12 Example: Calling a Java Extension Library

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

The entries in the XDO_METADATA sheet to declare and call the Java extension libraries are shown in [Figure 6–26](#).

Figure 6–26 Entries in the XDO_METADATA Sheet to Declare and Call the Java Extension Libraries

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

6.5.2 Formatting Functions That Rely on Specific Data Attribute Values

The following commands require that specific formatting attributes be present in the XML data file. A summary list of the commands is shown in [Table 6–13](#). See the corresponding section for details on usage.

Table 6–13 Commands for Specific Formatting Attributes

Function	Command
Section 6.5.2.1, "Defining Border and Underline Styles"	XDO_STYLE_n_?cell object name?
Section 6.5.2.2, "Skipping a Row"	XDO_SKIPROW_?cell object name?

6.5.2.1 Defining Border and Underline Styles

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

With the XDO_STYLE command you specify the cell to which to apply the style, the logic to determine when to apply the style, and the style type to apply. The style value must be present in the XML data. [Table 6–14](#) provides examples.

Table 6–14 Defining Border and Underline Styles

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

BI Publisher supports the normal Excel style types and values as shown in [Table 6–15](#).

Table 6–15 Excel Style Types and Values

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

You can also set a color using one of the types shown in [Table 6–16](#).

Table 6–16 Color Types

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

BI Publisher also supports the underline type with the values shown in [Table 6–17](#).

Table 6–17 Underline Types

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

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

Example: Defining Styles

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

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

- borderStyle
- underLineStyle
- borderColor

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

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

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

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

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

</G_DEPT>

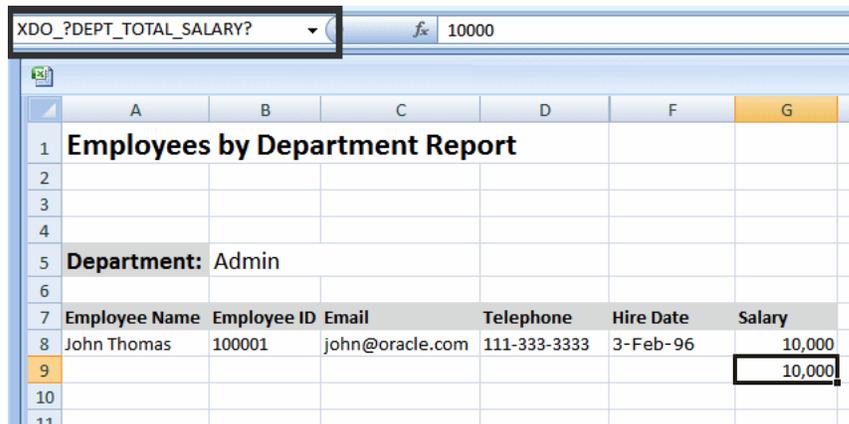
...

</EMPLOYEES>

To define a style:

1. In the Excel template, assign the defined name XDO_?DEPT_TOTAL_SALARY? to the field that is to display the DEPT_TOTAL_SALARY from the data, as shown in Figure 6-27.

Figure 6-27 Assigning the Defined Name XDO_?DEPT_TOTAL_SALARY?



2. In the XDO_METADATA sheet, enter the following:
 - To define the top border style, use the entries shown in Table 6-18.

Table 6-18 Top Border Style

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

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

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

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

- To define the top border color, use the entries shown in Table 6-19.

Table 6-19 Top Border Color

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

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

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

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

- To define the underline style, use the entries shown in [Table 6–20](#).

Table 6–20 Underline Style

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

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

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

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

[Figure 6–28](#) shows the three entries in the Data Constraints region.

Figure 6–28 Entries for Data Constraints

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

When you run the report, the style commands are applied to the XDO_?DEPT_TOTAL_SALARY? cell, as shown in [Figure 6–29](#).

Figure 6–29 A Generated Report Showing Style Commands Applied to the XDO_?DEPT_TOTAL_SALARY? Cell

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

6.5.2.2 Skipping a Row

Use the XDO_SKIPROW command to suppress the display of a row of data in a table when the results of an evaluation defined in Column B return the case insensitive string "True". Example entries are shown in Table 6–21.

Table 6–21 Skipping a Row

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

Example: Skipping a Row Based on Data Element Attribute

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

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

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

```

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

...

</EMPLOYEES>

```

To suppress the display of the row of the employee data when the MANAGER attribute is set to "True", enter the entries shown in [Table 6–22](#) in the Data Constraints section.

Table 6–22 *Skipping a Row Based on Data Element Attribute*

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

The output from this template is shown in [Figure 6–30](#). Note that the employee Michael Hartstein is not included in the report.

Figure 6–30 Output for a Skipped Row

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

6.5.3 Grouping Functions

Use the functions shown in [Table 6–23](#) to create groupings of data in the template.

Table 6–23 Creating Groupings of Data

Function	Command
Section 6.5.3.1, "Grouping the data"	XDO_GROUP_?group element?
Section 6.5.3.3, "Regrouping the Data"	XDO_REGROUP_?

6.5.3.1 Grouping the data

Use the XDO_GROUP command to group flat data when the layout requires a specific data grouping, for example, to split the data across multiple sheets. Example entries are shown in [Table 6–24](#).

Table 6–24 Grouping the Data

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

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

6.5.3.2 Handling the Generated XDO Define Names in Nested Groups

When XDO_GROUP_? is used in a nested group, the ranges of XDO define names in the final report become meaningless. In this case, do not refer to the define names in formulas in the final report. You can disable the XDO markup activity in the final report using the command "XDO_MARKUP_?". [Table 6–25](#) shows the usage of XDO_MARKUP_? in the XDO_METADATA sheet.

Table 6–25 Using XDO_MARKUP_?

Column A Entry	Column B Entry
XDO_MARKUP_?	"false" (The cell must be formatted as Text in the Excel Format Cells dialog.)

In addition, if your template includes a large number of defined names and these are used in multiple levels of nested groups, Excel may not be able to handle the number of generated defined names. In this case, use the XDO_MARKUP_? command to disable markup for the generated report.

When set to "false", BI Publisher does not produce any defined names for any result produced by XDO_GROUP_?

6.5.3.3 Regrouping the Data

The XDO_REGROUP regroups the data by declaring the structure using the defined names. The XDO_REGROUP logic is a shortened form of the XDO_GROUP logic and does not require the XSLTcoding requirements in the XDO_METADATA sheet. The definition must therefore be directly on XDO_REGROUP_? define names, or on any other definition on the XDO_METADATA sheet. Entries are shown in [Table 6–26](#).

Table 6–26 Regrouping the Data

Column A Entry	Column B Entry
XDO_REGROUP_?	XDO_REGROUP_? ?UniqueGroupID?levelName?groupByName?sortByName?sortByName? where <ul style="list-style-type: none"> ■ UniqueGroupID is the ID of the group. It can be the same as the levelName or you can assign it unique name. ■ levelName is the XML level tag name in the XML data file or XDO_CURRGRP_ that represents the current-group() in the context of nested grouping. XDO_CURRGRP_ should be used for all inner groups when more than one nesting group exists in your template. ■ groupByName is the field name that you want to use for the GroupBy operation for the current group. This name can be empty if the XDO_REGROUP_? command is used for the most inner group. ■ sortByName is the field name that you want to sort the group by. You can have multiple sortBy fields. If no sortByName is declared, then the data from the XML file is not sorted.

[Table 6–27](#), [Table 6–28](#), and [Table 6–29](#) show an example of how to create three nested groupings.

Table 6–27 Creating Nested Groupings, Example 1

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

In the definition shown in [Table 6–27](#), the most outer group is defined as PAYMENTSUMMARY_Q1, and it is grouped by PAY_TYPE_NAME

Table 6–28 Creating Nested Groupings, Example 2

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

The definition shown in [Table 6–28](#) creates a second outer group. The group is assigned the name COUNTRY_GRP and it is grouped by the element COUNTRY.

Table 6–29 Creating Nested Groupings, Example 3

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

The definition shown in [Table 6–29](#) creates the inner group STATEGRP and it includes a sortByName parameter: STATE.

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

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

- 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:
 - a. Navigate to the report in the catalog.
 - b. Click **Edit**.
 - c. Click **Add New Layout**.
 - d. Click **Upload**.
 - e. Complete the fields in the Upload dialog and select "XSL Stylesheet (HTML/XML/Text)" as the template **Type**.
 - f. After upload, click **View a List**. Deselect **Active**, so that users do not see this template as an option when they view the report.

Note: For testing purposes, you might 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.

- g. Save the report definition.
3. In the Excel template, on the XDO_METADATA sheet, in the Header section, enter the file name for the **Preprocess XSLT File** parameter. For example:
splitByBrand.xsl

6.6.1 XSLT Preprocessing Examples: Splitting Flat Data into Multiple Sheets

This section includes two examples of using an XSLT preprocess file to group flat data so that it can be split into multiple sheets in Excel. The examples are:

- [Splitting the Data by a Specific Field](#)
- [Splitting the Data by Count of Rows](#)

Splitting the data by row count is an option when your report data exceeds the sheet row size of Excel 2003 (65,536 rows per sheet).

Both examples use the following XML data:

```
<ROWSET>
  <ROW>
    <Products.Type>COATINGS</Products.Type>
    <Products.Brand>Enterprise</Products.Brand>
    <Markets.Region>CENTRAL REGION</Markets.Region>
    <Markets.District>CHICAGO DISTRICT</Markets.District>
    <Periods.Year>2000</Periods.Year>
    <Measures.Dollars>1555548.0</Measures.Dollars>
  </ROW>
  <ROW>
    <Products.Type>COATINGS</Products.Type>
    <Products.Brand>Enterprise</Products.Brand>
    <Markets.Region>EASTERN REGION</Markets.Region>
    <Markets.District>NEW YORK DISTRICT</Markets.District>
    <Periods.Year>2000</Periods.Year>
    <Measures.Dollars>1409228.0</Measures.Dollars>
  </ROW>
  ...
</ROWSET>
```

6.6.1.1 Splitting the Data by a Specific Field

This example demonstrates how to use an XSLT preprocess file to create a grouping in the data that will enable the splitting of the data across multiple Excel sheets based on the grouping. This example groups the sample data by the `Products.Brand` field.

1. Create an XSLT file to group the data.

The following sample XSLT file groups the data according to `<Products.Brand>` and creates a high level element `<BrandGroup>` for each of those groups.

```
<?xml version="1.0" encoding="utf-8" ?>
  <xsl:stylesheet version="2.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
    <xsl:template match="/">
```

```

<ROWSET>
  <xsl:for-each-group select="/ROWSET/ROW" group-by="./Products.Brand">
    <xsl:variable name="var_brand" select="current-grouping-key()" />
    <BrandGroup>
      <xsl:attribute name="name">
        <xsl:value-of select="$var_brand" />
      </xsl:attribute>
      <xsl:copy-of select="current-group()" />
    </BrandGroup>
  </xsl:for-each-group>
</ROWSET>
</xsl:template>
</xsl:stylesheet>

```

When applied to the data sample, this XSLT file generates intermediate data as follows:

```

<ROWSET>
  <BrandGroup name="Enterprise">
    <ROW>
      <Products.Type>COATINGS</Products.Type>
      <Products.Brand>Enterprise</Products.Brand>
      <Markets.Region>CENTRAL REGION</Markets.Region>
      <Markets.District>CHICAGO DISTRICT</Markets.District>
      <Periods.Year>2000</Periods.Year>
      <Measures.Dollars>1555548.0</Measures.Dollars>
    </ROW>
    ...
  </BrandGroup>
  ...
</ROWSET>

```

2. Save the XSLT file as `splitByBrand.xsl` and upload the file to the report definition in the BI Publisher catalog. Select "XSL Stylesheet (HTML/XML/Text)" as the template **Type**.
3. In the Excel template file, in the XDO_METADATA sheet, enter the following:
 - For the **Preprocess XSLT File** parameter, enter "splitByBrand.xsl"
 - In the **Data Constraints** region, make the entries shown in [Table 6–30](#) to split the data into multiple sheets based on the `<BrandGroup>` element created by the results of the XSLT preprocessing.

Table 6–30 Entries in Data Constraints Region to Split by Brand

Column A Entry	Column B Entry
XDO_SHEET_?	<?//BrandGroup?>
XDO_SHEET_NAME_?	<?./@name?>

The sample entries in the XDO_METADATA sheet are shown in [Figure 6–31](#).

Figure 6–31 XDO_METADATA Sheet for the Split by Brand Example

	A	B
1	Version	
2	ARU-dbdrv	
3	Extractor Version	
4	Template Code	
5	Template Type	TYPE_EXCEL_TEMPLATE
6	Preprocess XSLT File	splitByBrand.xsl
7	Last Modified Date	
8	Last Modified By	
9		
10	Data Constraints:	
11	XDO_SHEET_?	<?//BrandGroup?>
12	XDO_SHEET_NAME_?	<?./@name?>
13		

4. Upload the Excel template file to the report definition in the BI Publisher catalog.

6.6.1.2 Splitting the Data by Count of Rows

This example demonstrates how to use an XSLT preprocess file to group the sample XML data by the count of occurrences of /ROWSET/ROW and then configure the Excel template to create a new sheet for each occurrence of the newly created group.

1. Create an XSLT file to create groups in the data according to a size specified in a variable.

The following sample XSLT file groups the occurrences of /ROWSET/ROW according to the value of \$var_size and creates a high level element <CountGroup> for each of those groups.

```
<?xml version="1.0" encoding="utf-8" ?>
<xsl:stylesheet version="2.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="/">
    <ROWSET>
      <xsl:variable name="var_size" select="3" />
      <xsl:for-each select="/ROWSET/ROW">
        <xsl:variable name="var_pos" select="position()" />
        <xsl:variable name="var_mod" select="$var_pos mod($var_size)" />

        <xsl:if test="$var_mod = 1">
          <xsl:variable name="var_groupNum" select="($var_pos - $var_mod) div
number($var_size) + 1" />
          <xsl:element name="CountGroup">
            <xsl:attribute name="name">
              <xsl:value-of select="concat('Group', $var_groupNum)" />
            </xsl:attribute>
            <xsl:for-each select="/ROWSET/ROW[position() > ($var_pos - 1) and
position() < ($var_pos + $var_size)]">
              <xsl:copy-of select="." />
            </xsl:for-each>
          </xsl:element>
        </xsl:if>
      </xsl:for-each>
    </ROWSET>
  </xsl:template>
</xsl:stylesheet>
```

When applied to the data sample, this XSLT file generates intermediate data as follows:

```
<ROWSET>
  <CountGroup name="Group1">
    <ROW>
      <Products.Type>COATINGS</Products.Type>

      <Products.Brand>Enterprise</Products.Brand>

      <Markets.Region>CENTRAL REGION</Markets.Region>

      <Markets.District>CHICAGO DISTRICT</Markets.District>

      <Periods.Year>2000</Periods.Year>

      <Measures.Dollars>1555548.0</Measures.Dollars>

    </ROW>
    ...
  </CountGroup>
  ...
</ROWSET>
```

2. Save the XSLT file as `splitByCount.xsl` and upload the file to the report definition in the BI Publisher catalog. Select "XSL Stylesheet (HTML/XML/Text)" as the template **Type**.
3. In the Excel template file, in the XDO_METADATA sheet, enter the following:
 - For the **Preprocess XSLT File** parameter, enter "splitByCount.xsl"
 - In the **Data Constraints** region, make the entries shown in [Table 6–31](#).

Table 6–31 Entries in Data Constraints Region to Split by Count

Column A Entry	Column B Entry
XDO_SHEET_?	<?//CountGroup?>
XDO_SHEET_NAME_?	<?./@name?>

4. Upload the Excel template file to the report definition in the BI Publisher catalog.

6.7 Using the Template Viewer to Debug a Template

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

The Template Viewer is installed when you install the Template Builder for Word; see [Section 6.1.5, "Desktop Tools for Excel Templates"](#) for more information.

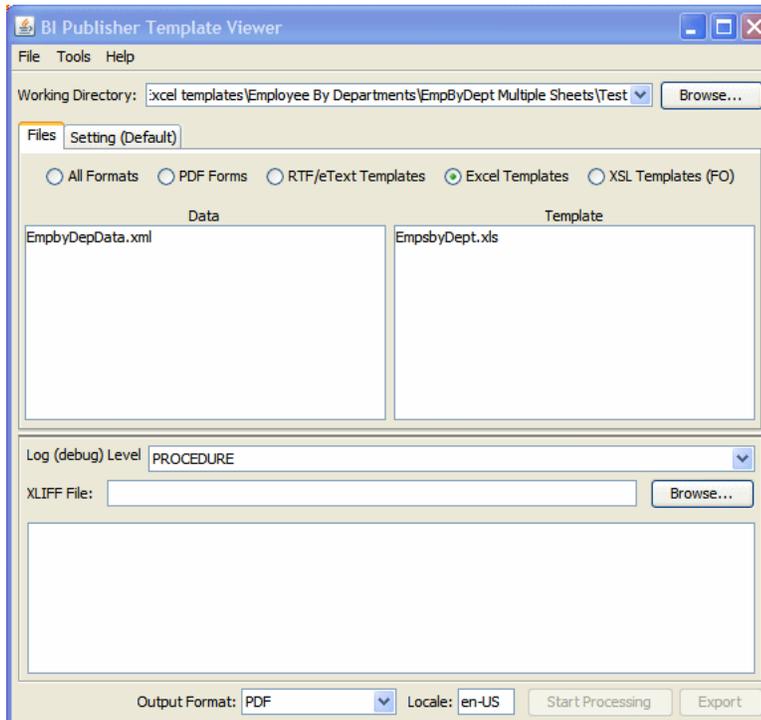
To preview with the Template Viewer and view log messages:

1. Open the Template Viewer:

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

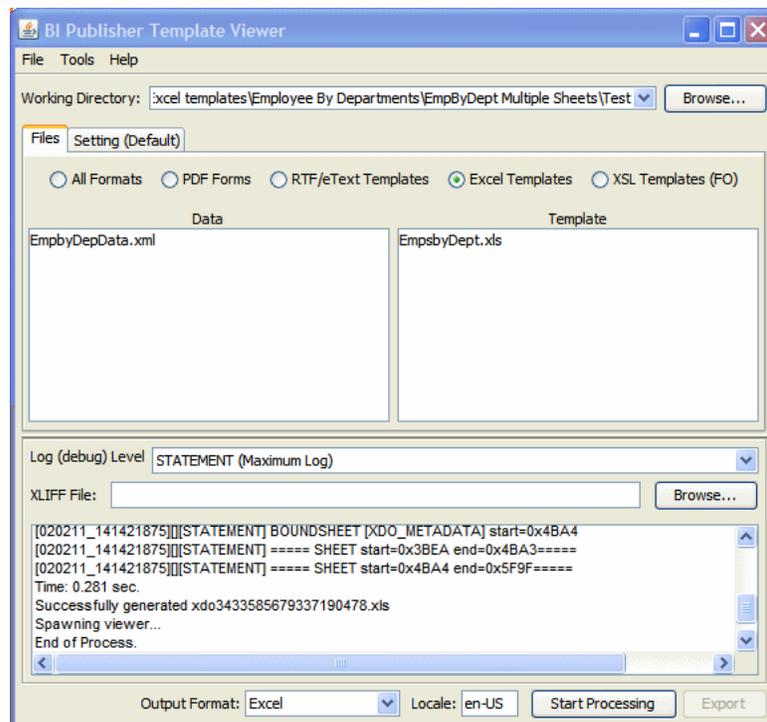
2. Click **Browse** to locate the folder that contains the sample data file and template file. The data file and template file must reside in the same folder.
3. Select **Excel Templates**. The **Data** and **Template** regions display all .xml files and all .xls files present in the directory, as shown in [Figure 6–32](#).

Figure 6–32 The Data and Template Regions Showing All .xml and .xls Files



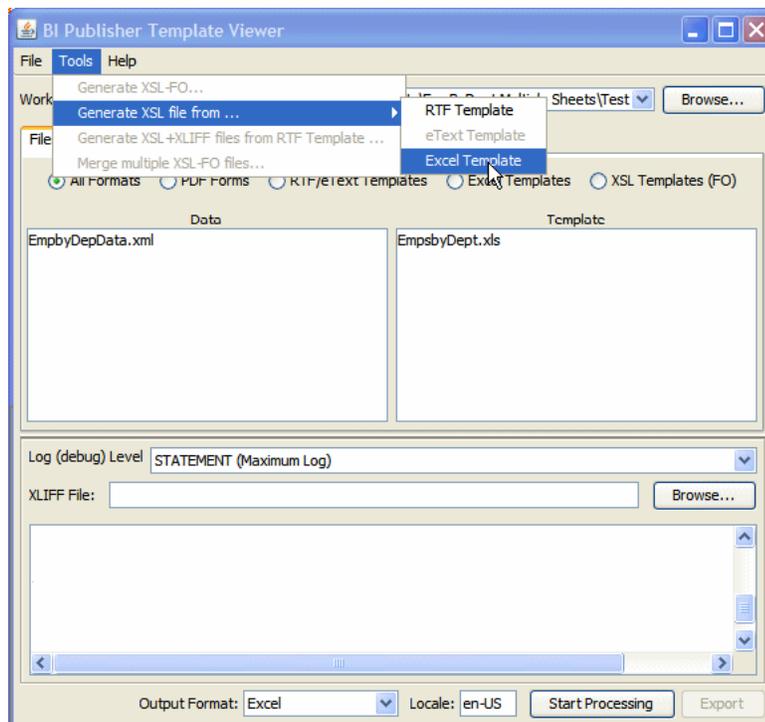
4. Click the appropriate data and template files to select them.
5. Select the log level.
6. From the **Output Format** list, select Excel.
7. Click **Start Processing**.

The Template Viewer merges the selected data with the selected template and spawns the appropriate viewer. View any log messages in the message box, as shown in [Figure 6–33](#).

Figure 6–33 Log Messages**To view the generated XSL:**

1. In the Template Viewer, select the data and template files and choose Excel output.
2. On the **Tools** menu, select **Generate XSL file from** and then choose **Excel Template**, as shown in [Figure 6–34](#).

Figure 6–34 The Excel Template Option



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

Creating PDF Templates

This chapter describes creating PDF templates for BI Publisher.

This chapter includes the following sections:

- [Section 7.1, "Overview"](#)
- [Section 7.2, "Requirements"](#)
- [Section 7.3, "Designing the Template"](#)
- [Section 7.4, "Adding Markup to the Template"](#)
- [Section 7.5, "Creating a Placeholder"](#)
- [Section 7.6, "Defining Groups of Repeating Fields"](#)
- [Section 7.7, "Adding Page Numbers and Breaks"](#)
- [Section 7.8, "Performing Calculations"](#)
- [Section 7.9, "Completed PDF Layout Example"](#)
- [Section 7.10, "Runtime Behavior"](#)
- [Section 7.11, "Creating a Layout from a Predefined PDF Form"](#)
- [Section 7.12, "Adding or Designating a Field for a Digital Signature"](#)

7.1 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 [Section 7.11, "Creating a Layout from a Predefined PDF Form."](#)

If you are designing the template, then when you have converted to PDF, the 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 the original document and then convert back to PDF.

For this reason, the PDF template is not recommended for documents that require frequent updates. However, it is appropriate for forms that have a fixed template, such as invoices or purchase orders.

7.2 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 the template. Regardless of version that you are using to design the template, you must save the PDF file as Adobe Acrobat 5.0 (PDF specification version 1.4).

Note: If you are using a later version of Adobe Acrobat Professional, then you must use the **Reduce File Size Option** (available from the **Document** menu or from the **File** menu depending on the version) to save the file as **Adobe Acrobat 5.0 compatible**.

For PDF conversion, BI Publisher supports any PDF conversion utility, such as Adobe Acrobat Distiller.

7.3 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.

[Figure 7-1](#) shows a template for a sample purchase order. It was designed using Microsoft Word and converted to PDF using Adobe Acrobat Distiller.

Figure 7-1 Template for a Sample Purchase Order

The following is the XML data that is 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>
```

7.4 Adding Markup to the Template

After you have converted a document to PDF, you define form fields that 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 [Section 4.2, "Associating the XML Data to the Template Layout."](#)

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 the template and the form fields. You therefore must place the fields exactly where you want the data to display on the template.

7.5 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 the version. If you are using Adobe Acrobat 9 Pro, then from the **Forms** menu, select **Add or Edit Fields**.

7.5.1 Naming the Placeholder

The name of the placeholder must match the XML source field name.

7.5.2 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, then refer to the documentation for details.

To create a text placeholder:

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 the 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 the fields, you must make the template compatible with Adobe Acrobat 5.0. From the **Document** menu, select **Reduce File Size**. From the **Make Compatible with** list, select **Adobe Acrobat 5.0 and later**.

7.5.3 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/updatable state of the field in the output PDF. See [Section 7.10.2, "Setting Fields as Updatable or Read Only."](#)
 - 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

7.5.4 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 causes 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.

To create a check box:

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 the 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.

7.5.5 Creating a Radio Button Group

A radio button group is used to display options from which only one can be selected. For example, the XML data file contains a field called <SHIPMENT_METHOD>. The possible values for this field are "Standard" or "Overnight". You represent this field in the 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, then refer to the documentation for details.

To create a radio button group:

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 the 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 defaults 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 want to change any of the properties, then 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.

7.6 Defining Groups of Repeating Fields

In the PDF layout, you explicitly define the area on the page that contains 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 [Section 7.10.1, "Placement of Repeating Fields."](#)

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.

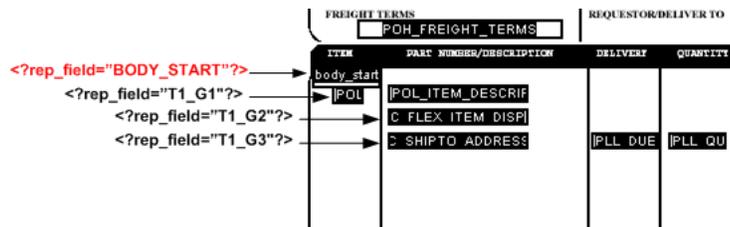
For example, the group in the sample report is laid out in three rows.

- 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"?>`

Figure 7–2 shows the entries for the **Short Description/Tooltip** field.

Figure 7–2 Entries for the Short Description/Tooltip Field



- (Optional) Align the fields. To ensure proper alignment of a row of fields, it is recommended that you use Adobe Acrobat's alignment feature.

7.7 Adding Page Numbers and Breaks

This section describes how to add the following page features to the PDF layout:

- Section 7.7.1, "Adding Page Numbers"
- Section 7.7.2, "Adding Page Breaks"

7.7.1 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.

To add page numbers:

- Decide the position on the layout where you want the page number to be displayed.
- Create a placeholder field called @pagenum@ (See Section 7.5.2, "Creating a Text Placeholder.")
- 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, then the start value that is assigned in the layout is overridden. If no start value is assigned, then it defaults to 1.

7.7.2 Adding Page Breaks

You can define a page break in the 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>

```

Suppose the report requirement is to display the salary information for each employee by department as shown in [Figure 7-3](#).

Figure 7-3 Department Salary Summary Example

Department Salary Summary

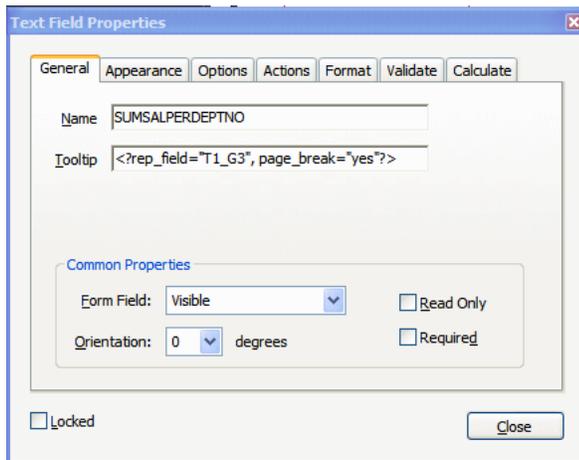
<code>body_start</code>	Dept No.	Emp No	Emp Name	Job	Salary
	<code>DEPTNO</code>	<code>EMPNO</code>	<code>ENAME</code>	<code>JOB</code>	<code>SAL</code>
					<code>SUMSALPERDEP</code>

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 Text Field Properties dialog for the field is shown in [Figure 7-4](#).

Figure 7-4 The Text Field Properties Dialog



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 [Figure 7-5](#).

Figure 7-5 A Sample Report with Data

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

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

The page breaks after each department.

7.8 Performing Calculations

Adobe Acrobat provides a calculation function in the **Field Properties** dialog box.

To create a field to display a calculated total on a 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.

7.9 Completed PDF Layout Example

Figure 7-6 shows the completed PDF layout.

Figure 7-6 A Completed PDF Layout

ORACLE
E-Business Suite

Purchase Order

PURCHASE ORDER NO. | REVISION | PAGE
IPOH PO N | 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: IPOH VENDOR NAME
IPOH VENDOR ADDRESS III
IPOH VENDOR ADR INFO
IPOH VENDOR COUNTRY

CUSTOMER ACCOUNT NO. | VENDOR NO. | DATE OF ORDER/YEAR | REVISED ENTRY/YEAR
IPOH CUST | IPOH VE | IPOH CR | IPOH AR |

PAYMENT TERMS | SHIP VIA | FOR
IPOH PAYMENT TERMS | IPOH SHIP V | IPOH FOB

FREIGHT TERMS | REQUESTOR/DELIVER TO | CONTACT TELEPHONE
IPOH FREIGHT TERMS

ITEM	ITEM NUMBER/DESCRIPTION	QUANTITY	QUANTITY	UNIT	UNIT PRICE	NETAMOUNT	TAX
body_start	IPOH IPOH ITEM DESC IPOH ITEM IPOH SHIPTO ADD		IPOH QTY		IPOH PRICE	IPOH AMT	IPOH TAX
body_end							

Total | IPOH AMOUNT P

Oracle E-Business Suite

7.10 Runtime Behavior

The following sections describe runtime behavior of PDF templates:

- Section 7.10.1, "Placement of Repeating Fields"
- Section 7.10.2, "Setting Fields as Updatable or Read Only"
- Section 7.10.3, "Overflow Data"

7.10.1 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 displays 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 is placed this calculated distance below the first occurrence.

Second occurrence, multiple rows:

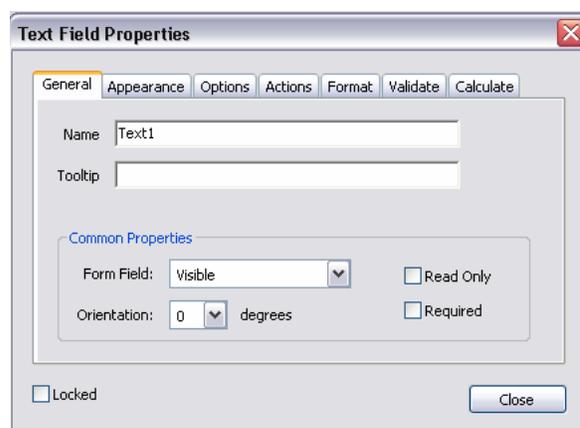
If the first group contains multiple rows, then the second occurrence of the group is placed the calculated distance below the last row of the first occurrence.

The distance between the rows within the group is maintained as defined in the first occurrence.

7.10.2 Setting Fields as Updatable 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 [Figure 7-7](#):

Figure 7-7 Read Only Option in The 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, as described in [Chapter 10, "Setting Report Processing and Output Document Properties"](#):

- 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 updatable:

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 overrides the settings of "all-field-readonly".

To remove all fields from the output PDF:

Set the property "remove-pdf-fields" to "true".

7.10.3 Overflow Data

When multiple pages are required to accommodate the occurrences of repeating rows of data, each page displays identically except for the defined repeating area, which displays 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 displays all data from the purchase order form with the continuation of the item rows.

7.11 Creating a Layout from a Predefined PDF Form

There are many PDF forms available online that you may want to use as layouts for the report data. For example, government forms that your company is required to submit. You can use these downloaded PDF files as the 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 [Section 7.11.1, "Determining If a PDF Has Form Fields Defined"](#) if you are unsure. If the PDF form already has fields defined, then 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 the XML data file. See [Section 7.11.2, "Using a Predefined PDF Form as a Layout by Renaming the Form Fields."](#)
- Use BI Publisher's Data Model Editor to rename the XML element names in the data file to match the field names in the PDF form. For information, see the "Structuring Data" chapter in *Oracle Fusion Middleware Data Modeling Guide for Oracle Business Intelligence Publisher*.

If the form fields are not already defined in the downloaded PDF, then you must create them. See [Section 7.4, "Adding Markup to the Template"](#) for instructions on inserting the form field placeholders.

7.11.1 Determining If a PDF Has Form Fields Defined

To determine if the PDF form has form fields defined:

1. Open the document in Adobe Acrobat Reader or Adobe Acrobat Professional.
2. Click **Highlight Fields**. Form fields that exist in the document are highlighted.

To get a list of the field names:

1. Open the document in Adobe Acrobat Professional.
2. From the **Form** menu, select **Add or Edit Fields**. The field names display in the document as well as in the **Fields** pane.

7.11.2 Using a Predefined PDF Form as a Layout by Renaming the Form Fields

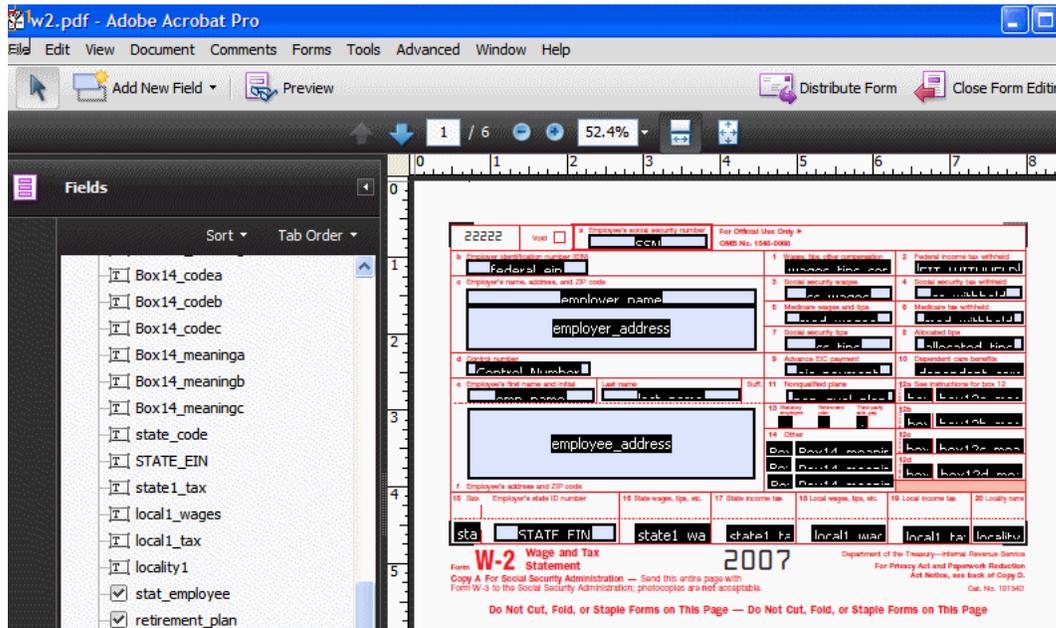
To use a predefined PDF form as a layout:

1. Download or import the PDF file to the local system.

2. Open the file in Adobe Acrobat Professional.
3. From the **Form** menu, select **Add or Edit Fields**. This highlights text fields that have already been defined.

Figure 7–8 shows a sample W-2 PDF form after selecting **Add or Edit Fields** to highlight the text fields.

Figure 7–8 A Sample W-2 PDF Form After Selecting the Add or Edit Fields Option



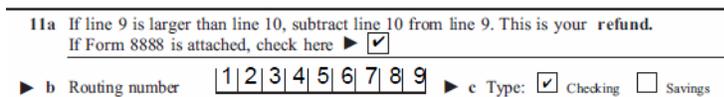
To map the existing form fields to the data from the incoming XML file, rename the fields to match the element names in the XML file.

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-click menu.
5. In the **Name** field, enter the element name from the input XML file.
6. Repeat for all fields that you want populated by the data file.
7. When all fields have been updated, click **Close Form Editing**.
8. **IMPORTANT:** Make the 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**.
9. Save the layout.

7.11.3 Using the Comb of Characters Option

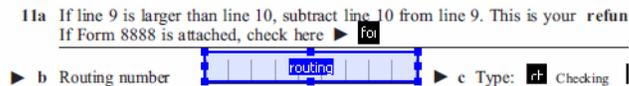
The comb of characters option for a PDF form field in Adobe Acrobat spreads the text evenly across the width of the text field. Use this option when the form field requires the characters to be entered in specific positions, as the Routing number field shown in Figure 7–9:

Figure 7–9 Example of PDF Form Field Requiring Characters in Specific Positions



To use this feature, perform the following:

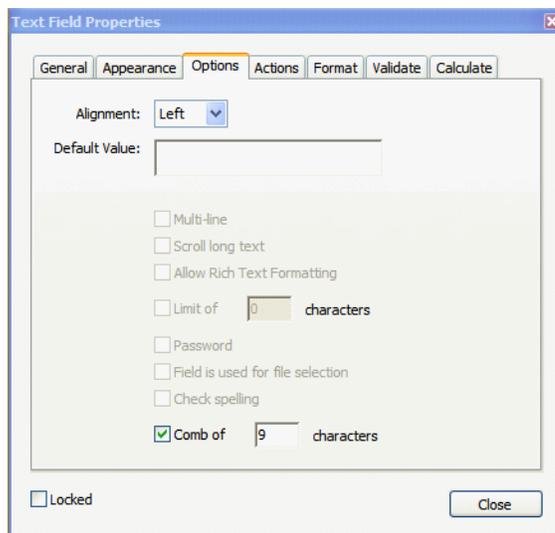
1. In Adobe Acrobat Professional, add the form field as a text field. An example is shown in the following figure:



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

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

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



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

When you run the report, the characters comprising the value for the routing field will be spread across the text field as shown in [Figure 7–10](#):

Figure 7–10 Example with Data

11a If line 9 is larger than line 10, subtract line 10 from line 9. This is your refund .	
If Form 8888 is attached, check here <input checked="" type="checkbox"/>	
▶ b Routing number	5 4 2 3 9 7 3 2 5 ▶ c Type: <input checked="" type="checkbox"/> Checking <input checked="" type="checkbox"/> Savings

Figure 7–10 shows how the data will display in the field when the data for the routing field does not contain the full nine characters and the **Alignment** option is set to left:

Figure 7–11 Example of Left-Aligned Data

11a If line 9 is larger than line 10, subtract line 10 from line 9. This is your refund .	
If Form 8888 is attached, check here <input checked="" type="checkbox"/>	
▶ b Routing number	5 4 2 3 9 7 3 ▶ c Type: <input checked="" type="checkbox"/> Checking <input checked="" type="checkbox"/> Savings

7.12 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 the 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 the PDF template for the digital signature. For more information and a description of the other required tasks and options, see the "Implementing a Digital Signature" section in *Oracle Fusion Middleware Administrator's Guide for Oracle Business Intelligence Publisher*.

7.12.1 About Signature Field Options

For PDF templates you have the following options for designating a digital signature field for the 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 the PDF template does not already include a signature field. See [Section 7.12.2, "Adding a Signature Field."](#)
- Use an existing signature field in the PDF template.

Use this option if the 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 [Section 7.12.3, "Configuring the Report to Insert the Digital Signature at Runtime."](#)
- 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 [Section 10.4, "PDF Digital Signature Properties."](#)

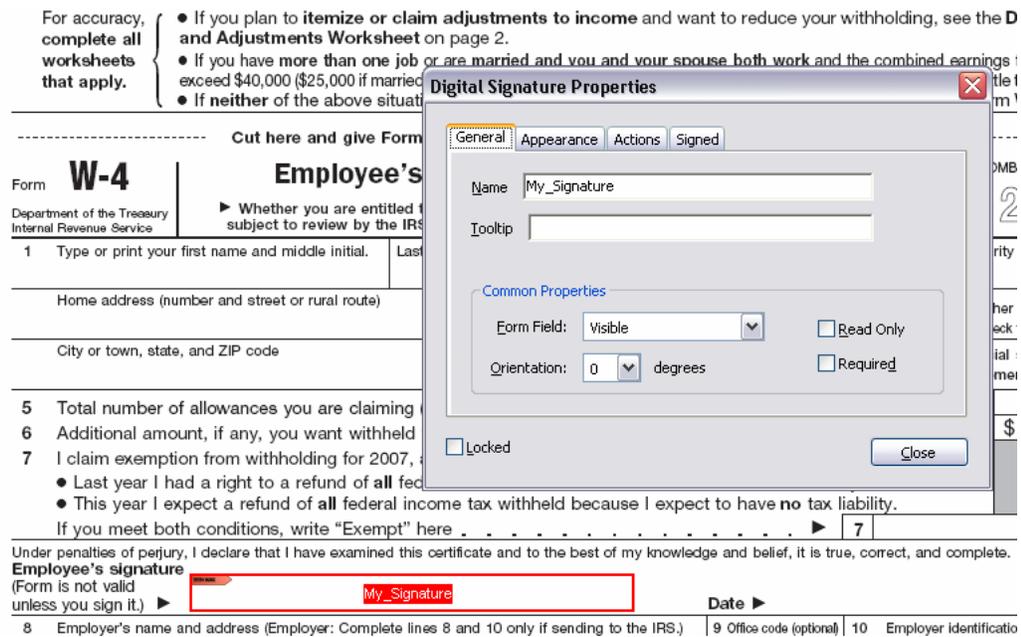
All three options require setting configuration properties for the report in the Report Properties page after you have uploaded the template.

7.12.2 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 prompts you to enter a name for the field.
4. Enter a name for the signature field. [Figure 7-12](#) shows an inserted digital signature field called "My_Signature."

Figure 7-12 An Inserted Digital Signature Field Called My_Signature



5. Save the template.
6. Proceed to [Section 7.12.3, "Configuring the Report to Insert the Digital Signature at Runtime."](#)

7.12.3 Configuring the Report to Insert the Digital Signature at Runtime

After you have uploaded the PDF template to the report definition (see [Section 2.5.2, "Adding a Layout by Uploading a Template File"](#)), you must enable digital signature and specify the signature field in the Report Properties.

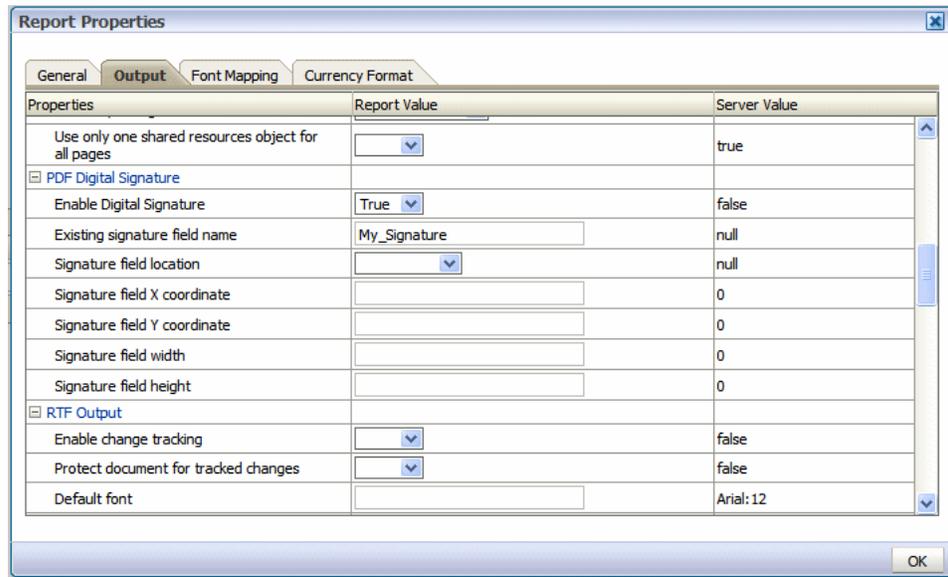
To configure the report to insert the digital signature:

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.

Figure 7–13 The My_Signature Field 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 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 eText Templates

BI Publisher uses eText templates to generate the flat files for EDI and EFT transactions. This chapter describes creating the eText templates in Microsoft Word

This chapter includes the following sections:

- [Section 8.1, "Overview"](#)
- [Section 8.2, "Structure of eText Templates"](#)
- [Section 8.3, "Constructing the Data Tables"](#)
- [Section 8.4, "Command Rows"](#)
- [Section 8.5, "Structure of the Data Rows"](#)
- [Section 8.6, "Setting Up Command Tables"](#)
- [Section 8.7, "Creating a Filler Block"](#)
- [Section 8.8, "Expressions, Control Structures, and Functions"](#)
- [Section 8.9, "Identifiers, Operators, and Literals"](#)

8.1 Overview

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 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.

Figure 8–2 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.

Figure 8–2 The Relationship Between the XML Data Hierarchy and the Template Level

```

?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>Citi Bank</PaymentsCommon:FinancialInstitution>
    </PaymentsCommon:TrxnParties>
    <PaymentsCommon:Encryption>H</PaymentsCommon:Encryption>
    <PaymentsCommon:FileID>10180300001</PaymentsCommon:FileID>
    <PaymentsCommon:BaseRecordCount>94457</PaymentsCommon:BaseRecordCount>
  </PaymentsCommon:RequestHeader>
  <OutBoundPayments:BatchRequest>
    <OutBoundPayment>
    <OutBoundPayment>
    <OutBoundPayment>
    <OutBoundPayment>
    <OutBoundPayment>
    ...
  </OutBoundPayments:BatchRequest>

```

<LEVEL>	RequestHeader			
<POSITION>	<LENGTH>	<FORMAT>	<PAD>	<DATA>
<NEW RECORD>		FileHeaderRec		
1	3	Alpha		'FID'
4	4	Number	L, '0'	FileID
8	1	Alpha	R, ''	Encryption
9	6	Date, YYYYMMDD		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 generates the nested records of the child tables before moving on to the next parent instance.

8.2.1 Command Rows, Data Rows, and Data Column Header Rows

Figure 8–3 shows the placement of Command Rows, Data Rows, and Data Column Header Rows.

8.4 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.

8.4.1 Level Command

The level command associates a table with an XML element. The parameter for the level command is an XML element. The level is 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 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 is 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>	RequestHeader
<NEW RECORD>	FileHeaderRecordA
Data rows for the FileHeaderRecordA	

<LEVEL>	Batch
<NEW RECORD>	BatchHeaderRecordA
Data rows for the BatchHeaderRecordA	
<NEW RECORD>	BatchHeaderRecordB
Data rows for the BatchHeaderRecordB	

<LEVEL>	Payment
<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 the 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 generates the printed records that are shown in [Table 8-1](#).

Table 8-1 Printed Records

Record Order	Record Type	Description
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

Table 8–1 (Cont.) Printed Records

Record Order	Record Type	Description
17	InvoiceRecordA	For Batch2, Payment1, Invoice1
18	InvoiceRecordA	For Batch2, Payment1, Invoice2
19	InvoiceRecordA	For Batch2, Payment1, Invoice3
20	BatchFooterRecordC	For Batch2
21	BatchFooterRecordD	For Batch2
22	FileFooterRecordB	One footer record for the EFT file

8.4.2 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 are 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 the COUNT function [Section 8.8.3, "Functions"](#) for more information.

Consecutive new record commands (or empty records) are not allowed.

8.4.3 Sort Ascending and Sort Descending Commands

Use the sort ascending and sort descending commands to sort the instances of a level. Enter the elements that you want 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.

8.4.4 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 applies 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.

8.5 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, because all columns after the required ones are ignored.

The usage rules for these columns are as follows:

8.5.1 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.

8.5.2 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 is truncated. Data is not padded if it is less than the maximum length.

8.5.3 Format Column

Specifies the data type and format setting. There are three accepted data types:

- Alpha
- Number
- Date

Refer to [Section 8.9.4, "Field-Level Key Words"](#) for their usage.

8.5.4 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 [Section 8.6, "Setting Up Command Tables"](#) for details on these commands.

[Table 8–2](#) shows sample Data, Format Specifier, and Output. The Output assumes the default group and decimal separators.

Table 8–2 Sample Data, Format Specifier, and Output

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

8.5.5 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.

8.5.6 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 as shown in [Table 8–3](#).

Table 8–3 Mapping of EDI Delimiter-Based Data Types to eText Data Types

ASC X12 Data Type	Format Template Data Type
A - Alphabetic	Alpha

Table 8–3 (Cont.) Mapping of EDI Delimiter-Based Data Types to eText Data Types

ASC X12 Data Type	Format Template Data Type
AN -Alphanumeric	Alpha
B - Binary	Number
CD - Composite data element	N/A
CH - Character	Alpha
DT - Date	Date
FS - Fixed-length string	Alpha
ID - Identifier	Alpha
IV - Incrementing Value	Number
Nn - Numeric	Number
PW - Password	Alpha
R - Decimal number	Number
TM - Time	Date

Assume the setup commands shown in [Table 8–4](#).

Table 8–4 Setup Commands

Name	Command
NUMBER THOUSANDS SEPARATOR	.
NUMBER DECIMAL SEPARATOR	,

[Table 8–5](#) 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.

Table 8–5 Data, Format Specifier, and Output for the Specified Setup Commands

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

8.5.7 Pad

Pad 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, ''

8.5.8 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 [Section 8.8, "Expressions, Control Structures, and Functions."](#)

8.5.9 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.

8.5.10 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.

8.6 Setting Up Command Tables

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 [Figure 8-4](#).

Figure 8–4 Example Setup Tables

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>	`InvoiceTrxnReportingCat', (IF InvoiceTrxnReportingCat = 'V' THEN `InvoiceDEVTransitGoods' END IF), (IF InvoiceTrxnReportingCat = 'V' THEN `InvoiceDEVGoodsIndexNum' END IF), (IF InvoiceTrxnReportingCat = 'V' THEN `InvoiceDEVPassingTrade' END IF)
<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

8.6.1 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, then it is padded. If longer, it is 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 are displayed 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.

8.6.2 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 is 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, [Table 8–6](#) shows five payments under a batch.

Table 8–6 Example of 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 generate the four groups (instances) shown in [Table 8–7](#) for the new level.

Table 8–7 The Five Payments Generated

Payment Group Instance	Group Criteria	Payments in Group
Group1	PaymentDate1, PayeeName1	Payment1, Payment4
Group2	PaymentDate1, PayeeName2	Payment3
Group3	PaymentDate1, PayeeName3	Payment5
Group4	PaymentDate2, PayeeName1	Payment2

The order of the new instances is the order in which the records 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. [Table 8-8](#) and [Table 8-9](#) show the generated output.

Scenario 1: Group Records Only

EFT File Structure:

- BatchRec
 - PaymentGroupHeaderRec
 - PaymentGroupFooterRec

Table 8-8 Example of Group Records Only

Record Sequence	Record Type	Description
1	BatchRec	NA
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)
8	PaymentGroupHeaderRec	For group 4 (PaymentDate2, PayeeName1)
9	PaymentGroupFooterRec	For group 4 (PaymentDate2, PayeeName1)

Scenario 2: Group Records and Individual Records

EFT File Structure:

BatchRec

- PaymentGroupHeaderRec
 - PaymentRec
- PaymentGroupFooterRec

Table 8-9 Example of Group Records and Individual Records

Record Sequence	Record Type	Description
1	BatchRec	NA
2	PaymentGroupHeaderRec	For group 1 (PaymentDate1, PayeeName1)
3	PaymentRec	For Payment1
4	PaymentRec	For Payment4

Table 8–9 (Cont.) Example of Group Records and Individual Records

Record Sequence	Record Type	Description
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
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.

8.6.3 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" always refers to the EFT record. The database record is 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:

8.6.4 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).

8.6.5 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, then all the records in the level have the same sequence number. The record-based increment assigns 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.

8.6.6 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.

8.6.7 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.

8.6.8 Element Subcommand

The element subcommand specifies the operation used to generate each element. An element is a child-level expression that is concatenated to generate the concatenation string.

8.6.9 Delimiter Subcommand

The delimiter subcommand specifies the delimiter to separate the concatenated items in the string.

8.6.10 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, [Table 8–10](#) shows five invoices in a payment.

Table 8–10 Example of 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 concatenation definition shown in [Table 8–11](#):

Table 8–11 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 is as follows:

```
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, then use either TRUNCATE function or the SUBSTR function as follows:

```
TRUNCATE(ConcatenatedInvoiceInfo, 40)
```

```
SUBSTR(ConcatenatedInvoiceInfo, 1, 40)
```

Either of these statements result in:

```
car_parts_inv0001,car_parts_inv0002,car_
```

To isolate the next forty characters, use the SUBSTR function:

```
SUBSTR(ConcatenatedInvoiceInfo, 41, 40)
```

to get the following string:

```
parts_inv0003,car_parts_inv0004,car_par
```

8.6.11 Invalid Characters and Replacement Characters Commands

Some formats require a different character set than the one that was used to enter the data in Oracle Applications. For example, some German formats require the output file in ASCII, but the data was entered in German. If there is a mismatch between the original and target character sets you can define an ASCII equivalent to replace the original. For example, you would replace the German unlauded "a" with "ao".

Some formats do not allow certain characters. To ensure that known invalid characters are not transmitted in the output file, use the invalid characters command to flag occurrences of specific characters.

To use the replacement characters command, specify the source characters in the left column and the replacement characters in the right column. You must enter the source characters in the original character set. This is the only case in a format template in which you use a character set not intended for output. Enter the replacement characters in the required output character set.

For DELIMITER_BASED formats, if there are delimiters in the data, you can use the escape character "?" to retain their meaning. For example,

First name?+Last name equates to First name+Last name

Which source?? equates to Which source?

Note that the escape character itself must be escaped if it is used in data.

The replacement characters command can be used to support the escape character requirement. Specify the delimiter as the source and the escape character plus the delimiter as the target. For example, the command entry for the preceding examples is as follows:

<REPLACEMENT CHARACTERS>	
+	?+
?	??
<END REPLACEMENT CHARACTERS>	

The invalid character command has a single parameter that is a string of invalid characters that causes the system to error out.

The replacement character process is performed before or during the character set conversion. The character set conversion is performed on the XML extract directly, before the formatting. After the character set conversion, the invalid characters are checked in terms of the output character set. If no invalid characters are found, then the system proceeds to formatting.

8.6.12 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, then the system sets "\n" as default new record character.

8.6.13 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" is used.

8.6.14 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 commands as follows:

<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 [Section 8.5.3, "Format Column."](#)

8.6.15 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. Valid values are "UPPER" and "LOWER". Enter the command as follows:

<CASE CONVERSION>	UPPER
-------------------	-------

8.7 Creating a Filler Block

For FIXED_POSITION_BASED templates, you can use a filler block to define a specific block size for the eText output. When the actual data does not fill the specified block size, the remainder of the block is filled with a specified filler character.

For example, if you define a BLOCK SIZE of 9, and the eText output generated is only three lines of text, then the remaining six lines are filled with the specified FILLER CHARACTER.

The commands used are:

- <BEGIN FILLER BLOCK> - this signifies the beginning of the block. Enter a name for this block.
- <FILLER CHARACTER> - enter a character or string to use to "fill" the remainder of the block when the data does not fill it.

Example entries for <FILLER CHARACTER> are:

To fill the block with the ? character, enter the FILLER CHARACTER command as shown:

<FILLER CHARACTER>	?
--------------------	---

To fill the block with the string "abc", enter the FILLER CHARACTER command as shown:

<FILLER CHARACTER>	abc'
--------------------	------

To fill the block with empty spaces, enter the FILLER CHARACTER command as shown:

<FILLER CHARACTER>	
--------------------	--

- <BLOCK SIZE> - enter an integer to specify the size of the block in lines of text.

These commands must be used before the template definition starts.

Enter the following command at the end of the block:

- <END FILLER BLOCK> - signifies the end of the block. Enter the name already specified for this block in the <BEGIN FILLER BLOCK> command.

Figure 8–5 shows an example of filler block usage.

Figure 8–5 Example of Filler Block Usage

<BEGIN FILLER BLOCK>	PaymentsBlock
<FILLER CHARACTER>	?
<BLOCK SIZE>	5

<LEVEL>		OutboundPayment				
<SORT ASCENDING>		Payee/Name				
<POSITION>	<LENGTH>	<FORMAT>	<PAD>	<DATA>	<COMMENTS>	
<NEW RECORD>		EntryDetailRecCR				
1	1	Number		6	Record type	
2	2	Alpha	L, ' '	IF PaymentAmount/Value = 0 THEN '23' ELSE '22' END IF	Transaction Code	
4	8	Alpha	L, '0'	PayeeBankAccount/Bank Number	Receiving DFI Identification. Let BI Publisher do the truncation	
12	1	Alpha	L, '0'	SUBSTR (PayeeBankAccou nt/BankNumber, 9, 1)	Check digit. The validation code will make sure that the BankNumber is 9 digits. You don't need to use the LENGTH function here.	
13	17	Alpha	L, '0'	PayeeBankAccount/Bank AccountNumber	DFI Account Number	
30	10	Number, Integer	L, '0'	PaymentAmount/Value*1 00	Amount	
40	15	Alpha	L, '0'	PaymentNumber/CheckNu mber	Identification Number	
55	22	Alpha	R, ' '	Payee/Name	Receiving Company Name/ID Number	
77	2	Alpha	R, ' '		Discretionary Data	
79	1	Number		0	Addenda record Indicator	
80	8	Alpha	L, '0'	/OutboundPaymentInstr uction/PayerInstrumen t/BankAccount/BankNum ber	ODFI Routing Number	

8.8.2 Control Structures

The only supported control structure is "IF-THEN-ELSE". It can be used in an expression. The syntax is:

```
IF <boolean_expressionA> THEN
    <numeric or character expression1>
[ELSIF <boolean_expressionB THEN
    <numeric or character expression2>]
...
[ELSE
    <numeric or character expression3]
END IF
```

Generally the control structure must evaluate to a number or an alphanumeric string. The control structure is considered to a numeric or character expression. The ELSIF and ELSE clauses are optional, and there can be as many ELSIF clauses as necessary. The 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;
```

8.8.3 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 increases 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, then the function counts 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 returns the total number of payments in the batch. However, if the current level is RequestHeader, the COUNT returns the total number of payments in the file across all batches. If the argument is a record type, the count function counts 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")
```

 produces 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

8.9 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.

8.9.1 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

8.9.2 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.

8.9.3 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.

8.9.4 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).

8.9.5 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.

8.9.6 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.

[Table 8–12](#) lists the operators and describes their usage.

Table 8–12 Operators

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 "()" is 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.
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.

8.9.7 Reference to XML Extract Fields and XPATH Syntax

XML elements can be used in any expression. At runtime they are 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 to refer to the PayeeName element in a Payment data table, 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 must reference the BatchName element, you can specify ../BatchName. The "../" provides 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 must 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 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 can find the correct context for them.

If you use relative path syntax, then 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, then 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, then you must 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.

8.9.8 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, then the Chinese characters do 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.

Creating Flash Templates

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. This chapter describes how to create Flash templates.

This chapter includes the following sections:

- [Section 9.1, "Overview"](#)
- [Section 9.2, "Building a Flash Template"](#)
- [Section 9.3, "Uploading the Flash Template to the Report Definition"](#)
- [Section 9.4, "Setting Properties for PDF Output"](#)

9.1 Overview

This chapter demonstrates 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. Reference the Flex documentation for these more advanced features.

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 describes 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.

9.1.1 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, the 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 the data model, see the *Oracle Fusion Middleware Data Modeling Guide for Oracle Business Intelligence Publisher*.

9.1.2 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 by modifying the Flash configuration settings:

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.

9.2 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

9.2.1 Adding the Data Source

To add the data source:

1. Generate a sample data file from the 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:

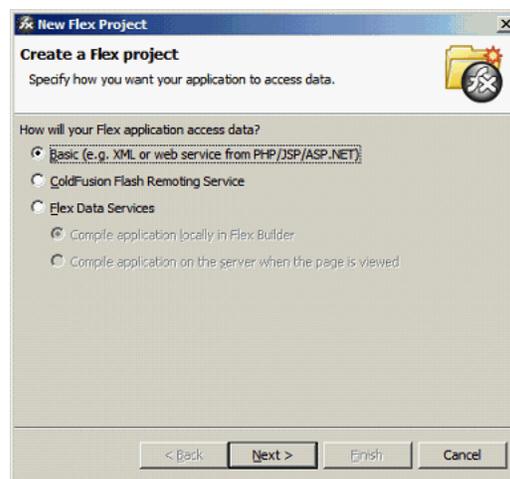
```

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

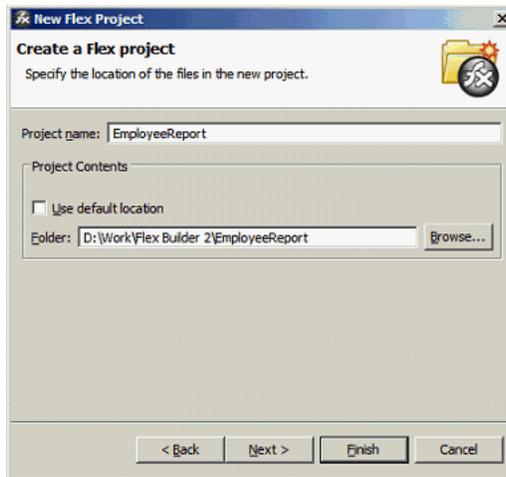
```

2. Open the Flex IDE and create a new Flex Project; select the "Basic" data access method, as shown in [Figure 9-1](#).

Figure 9-1 Example of Selecting the Basic Data Access Method



In the next dialog, give the project a name as shown in [Figure 9-2](#). The name that you use here is assigned to the template file name that you create.

Figure 9–2 Example of Specifying a Project Name

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 the 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 that you provided.

At runtime the BI Publisher server generates the runtime data from the report and injects 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.

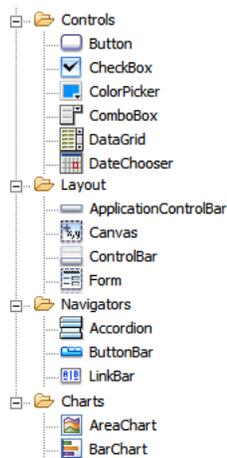
9.2.2 Creating the Layout

The Flex IDE creates a default canvas for you to drop objects onto. You can modify the canvas as required to suit the report.

Important: If you intend to embed the Flash output in a PDF document, then you must set the Width and Height of the template in the **Size** region of the **Layout** properties. Even if you want 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.

[Figure 9–3](#) shows an example of the available objects in the Component Navigator pane.

Figure 9–3 Example of the Available Objects in the Component Navigator Pane

These objects can be dragged and dropped to the design palette.

To create the layout:

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 [Figure 9–4](#).

Figure 9–4 Example of a Layout

Column 1	Column 2	Column 3

By default three columns are generated. In the next section, [Binding the Layout Objects to the Data Source](#), you override the default in the MXML code.

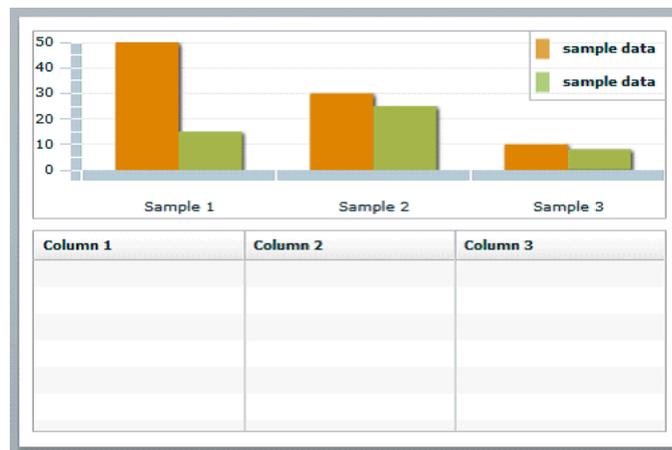
9.2.2.1 Adding a Chart

If you have purchased the charting option, then you can add charts to the layout.

To add a chart to the layout:

1. Make some room for the chart in the 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. When 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 [Figure 9–5](#).

Figure 9–5 A Sample Chart



This is a sample chart. You bind it to the data in the next section.

9.2.3 Binding the Layout Objects to the Data Source

When 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.

9.2.3.1 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 see that the first line of the DataGrid code has been highlighted for you. This feature is useful if you have built complex Flex templates and must 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:columns>
</mx:DataGrid>
```

```

        <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 table in [Figure 9–6](#).

Figure 9–6 Example of a Table

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 the 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 sample data. Select Run, then Run EmployeeReport. This opens a new browser window and renders the table with the sample data.

9.2.3.2 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" />
```

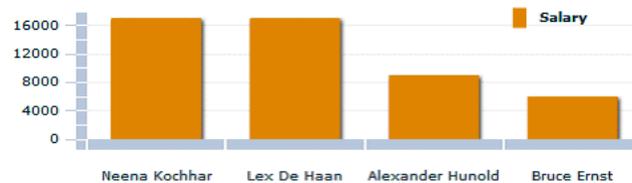
2. To bind the data source to the chart object, add the `dataProvider` attribute to the `<mx:ColumnChart>` element as follows:

```
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 chart in [Figure 9–7](#).

Figure 9–7 A Chart Showing Salary by Employee



Make the following updates to the code:

- Add a `<horizontalAxis>` element to define the element from the data source that is 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" />
```

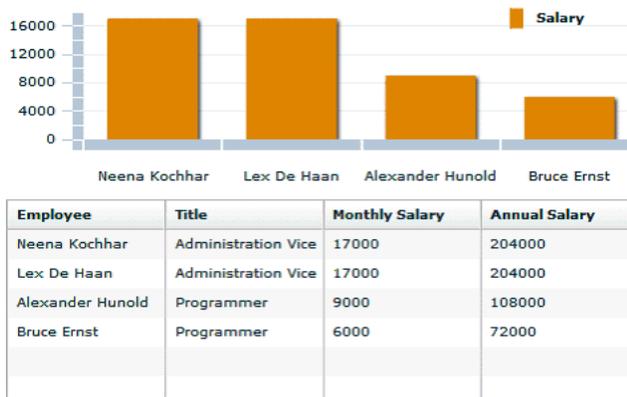
Note in the preceding sample, the `<mx:horizontalAxis>` element has been added and the `categoryField` attribute has the `NAME` data element assigned. This element is required to render the chart.

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 sample data. You should get an output showing the chart above the tabulated data as shown in [Figure 9–8](#).

Figure 9–8 Output Showing a Chart Above the Tabulated Data



9.3 Uploading the Flash Template to the Report Definition

To upload the template to the report definition:

1. Navigate to the 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.

9.4 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.

Note: To produce PDF output, you must specify the height and width of the template in the Flex Builder. See [Section 9.2.2, "Creating the Layout."](#)

To set properties for the PDF output:

1. Navigate to the 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.

Figure 9–9 shows the **Flash** heading in the Formatting tab.

Figure 9–9 The Flash Heading in the Formatting Tab

Properties	Report Value	Server Value
Set all fields as read only in output	<input type="checkbox"/>	true
Maintain each field's read only setting	<input type="checkbox"/>	false
Flash		
Page width of wrapper document	<input type="text"/>	792 (in points, 11 inch)
Page height of wrapper document	<input type="text"/>	612 (in points, 8.5 inch)
Start x position of Flash area in PDF	<input type="text"/>	18 (in points, 0.25 inch)
Start y position of Flash area in PDF	<input type="text"/>	18 (in points, 0.25 inch)
Width of Flash area	<input type="text"/>	Same as flash width in points in swf
Height of Flash area	<input type="text"/>	Same as flash height in points in swf
CSV output		
CSV delimiter	<input type="text"/>	,
Remove leading and trailing white space	<input type="checkbox"/>	false

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 is 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.

Setting Report Processing and Output Document Properties

This chapter describes how to configure report processing and output document properties in BI Publisher.

This chapter includes the following sections:

- Section 10.1, "Overview"
- Section 10.2, "PDF Output Properties"
- Section 10.3, "PDF Security Properties"
- Section 10.4, "PDF Digital Signature Properties"
- Section 10.5, "PDF/A Output Properties"
- Section 10.6, "PDF/X Output Properties"
- Section 10.7, "RTF Output Properties"
- Section 10.8, "HTML Output Properties"
- Section 10.9, "FO Processing Properties"
- Section 10.10, "RTF Template Properties"
- Section 10.11, "PDF Template Properties"
- Section 10.12, "Flash Template Properties"
- Section 10.13, "CSV Output Properties"
- Section 10.14, "Excel 2007 Output Properties"
- Section 10.15, "All Outputs"
- Section 10.16, "Defining Font Mappings"

10.1 Overview

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 takes precedence.

To set a property at the report level:

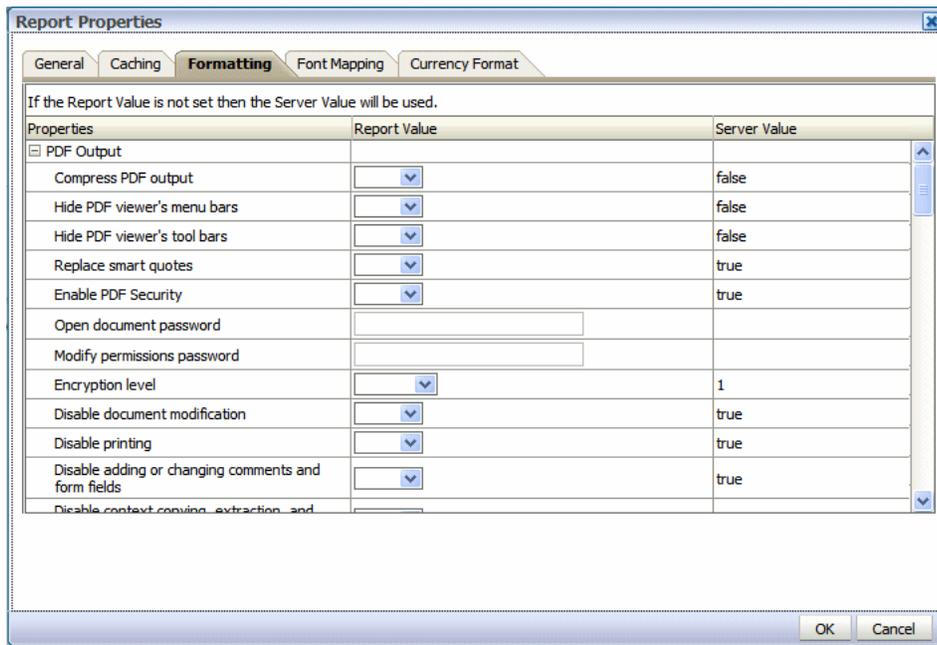
1. Open the report in the Report Editor, and then
2. Click **Properties** to display the **Report Properties** dialog.

3. Click the **Formatting** tab to display the formatting properties.

For each property, **Report Value** is updatable and the **Server Value** is shown for reference.

The Formatting tab of the Report Properties dialog is shown in [Figure 10-1](#).

Figure 10-1 Report Properties, Formatting Tab



For information on setting the properties at the server level, see "Defining Run-Time Configurations" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Business Intelligence Publisher*.

10.2 PDF Output Properties

[Table 10-1](#) describes the properties that are available for PDF output.

Table 10-1 PDF Output Properties

Property Name	Description
Compress PDF output	<p>Default: true</p> <p>Description: Specify "true" or "false" to control compression of the output PDF file.</p> <p>Internal Name: pdf-compression</p>
Hide PDF viewer's menu bars	<p>Default: false</p> <p>Description: 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>Internal Name: pdf-hide-menubar</p>

Table 10–1 (Cont.) PDF Output Properties

Property Name	Description
Hide PDF viewer's tool bars	<p>Default: false</p> <p>Description: Specify "true" to hide the viewer application's toolbar when the document is active.</p> <p>Internal Name: pdf-hide-toolbar</p>
Replace smart quotes	<p>Default: true</p> <p>Description: Set to "false" if you do not want curly quotes replaced with straight quotes in the PDF output.</p> <p>Internal Name: pdf-replace-smartquotes</p>
Use only one shared resources object for all pages	<p>Default: true</p> <p>Description: 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"> ■ Viewing may take longer for a large file with many SVG objects ■ If you choose to break up the file by using Adobe Acrobat to extract or delete portions, then the edited PDF files are larger because the single shared resource object (that contains all of the SVG objects for the entire file) is included with each extracted portion. <p>Setting this property to "false" creates a resource object for each page. The file size is larger, but the PDF viewing is faster and the PDF can be broken up into smaller files more easily.</p> <p>Internal Name: pdf-use-one-resources</p>
PDF Navigation Panel Initial View	<p>Default: Bookmarks Open</p> <p>Description: Controls the navigation panel view that is presented when a user first opens a PDF report. The following options are supported:</p> <ul style="list-style-type: none"> ■ Panels Collapsed - displays the PDF document with the navigation panel collapsed. ■ Bookmarks Open (default) - displays the bookmark links for easy navigation. ■ Pages Open - displays a clickable thumbnail view of each page of the PDF. <p>Internal Name: pdf-pagemode</p>

10.3 PDF Security Properties

Use the properties described in [Table 10–2](#) to control the security settings for the output PDF documents.

Table 10–2 PDF Security Properties

Property Name	Description
Enable PDF Security	<p>Default: false</p> <p>Description: If you specify "true," then the output PDF file is encrypted. You can then also specify the following properties:</p> <ul style="list-style-type: none"> ▪ Open document password ▪ Modify permissions password ▪ Encryption Level <p>Internal Name: pdf-security</p>
Open document password	<p>Default: N/A</p> <p>Description: This password is required for opening the document. It enables users to open the document only. This property is enabled only when "Enable PDF Security" is set to "true". 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>Internal Name: pdf-open-password</p>
Modify permissions password	<p>Default: N/A</p> <p>Description: This password enables users to override the security setting. This property is effective only when "Enable PDF Security" is set to "true". 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>Internal Name: pdf-permissions-password</p>
Encryption level	<p>Default: 2 - high</p> <p>Description: Specify the encryption level for the output PDF file. The possible values are:</p> <ul style="list-style-type: none"> ▪ 0: Low (40-bit RC4, Acrobat 3.0 or later) ▪ 1: Medium (128-bit RC4, Acrobat 5.0 or later) ▪ 2: High (128-bit AES, Acrobat 7.0 or later) <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"> ▪ Disable printing ▪ Disable document modification ▪ Disable context copying, extraction, and accessibility ▪ Disable adding or changing comments and form fields <p>When Encryption level is set to 1 or higher, the following properties are available:</p> <ul style="list-style-type: none"> ▪ Enable text access for screen readers ▪ Enable copying of text, images, and other content ▪ Allowed change level ▪ Allowed printing level <p>Internal Name: pdf-encryption-level</p>

Table 10-2 (Cont.) PDF Security Properties

Property Name	Description
Disable document modification	<p>Default: false</p> <p>Description: Permission available when "Encryption level" is set to 0. When set to "true", the PDF file cannot be edited.</p> <p>Internal Name: pdf-no-changing-the-document</p>
Disable printing	<p>Default: false</p> <p>Description: Permission available when "Encryption level" is set to 0. When set to "true", printing is disabled for the PDF file.</p> <p>Internal Name: pdf-no-printing</p>
Disable adding or changing comments and form fields	<p>Default: false</p> <p>Description: 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>Internal Name: pdf-no-accff</p>
Disable context copying, extraction, and accessibility	<p>Default: false</p> <p>Description: Permission available when "Encryption level" is set to 0. When set to "true", the context copying, extraction, and accessibility features are disabled.</p> <p>Internal Name: pdf-no-cceda</p>
Enable text access for screen readers	<p>Default: true</p> <p>Description: 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>Internal Name: pdf-enable-accessibility</p>
Enable copying of text, images, and other content	<p>Default: false</p> <p>Description: 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>Internal Name: pdf-enable-copying</p>
Allowed change level	<p>Default: 0</p> <p>Description: Permission available when "Encryption level" is set to 1 or higher. Valid Values are:</p> <ul style="list-style-type: none"> ■ 0: none ■ 1: Allows inserting, deleting, and rotating pages ■ 2: Allows filling in form fields and signing ■ 3: Allows commenting, filling in form fields, and signing ■ 4: Allows all changes except extracting pages <p>Internal Name: pdf-changes-allowed</p>

Table 10–2 (Cont.) PDF Security Properties

Property Name	Description
Allowed printing level	<p>Default: 0</p> <p>Description: Permission available when "Encryption level" is set to 1 or higher. Valid values are:</p> <ul style="list-style-type: none"> ▪ 0: None ▪ 1: Low resolution (150 dpi) ▪ 2: High resolution <p>Internal Name: pdf-printing-allowed</p>

10.4 PDF Digital Signature Properties

The properties described in [Table 10–3](#) 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 the output PDF documents, see "Implementing a Digital Signature" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Business Intelligence Publisher*.

Note: 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 the output report. The 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 the document.

Note: If you enable digital signature, but do not set any location properties, the digital signature placement defaults to the top left of the document.

Table 10–3 PDF Digital Signature Properties

Property Name	Description
Enable Digital Signature	<p>Default: false</p> <p>Description: Set this to "true" to enable digital signature for the report.</p> <p>Internal Name: signature-enable</p>

Table 10–3 (Cont.) PDF Digital Signature Properties

Property Name	Description
Existing signature field name	<p>Default: N/A</p> <p>Description: This property applies to PDF layout templates only. If the report is based on a PDF template, then you can enter a field from the PDF template in which to place the digital signature.</p> <p>Internal Name: signature-field-name</p>
Signature field location	<p>Default: top-left</p> <p>Description: This property can apply to RTF or PDF layout templates. This property provides a list that contains the following values: Top Left, Top Center, Top Right. Choose one of these general locations and BI Publisher inserts 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>Internal Name: signature-field-location</p>
Signature field X coordinate	<p>Default: 0</p> <p>Description: 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>Internal Name: signature-field-pos-x</p>
Signature field Y coordinate	<p>Default: 0</p> <p>Description: 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>Internal Name: signature-field-pos-y</p>
Signature field width	<p>Default: 0</p> <p>Description: Enter in points (72 points equal one inch) the desired width of the inserted digital signature field. This applies only if you are also setting the properties Signature field x coordinate and Signature field Y coordinate.</p> <p>Internal Name: signature-field-width</p>
Signature field height	<p>Default: 0</p> <p>Description: Enter in points (72 points equal one inch) the desired height of the inserted digital signature field. This applies only if you are also setting the properties Signature field x coordinate and Signature field Y coordinate.</p> <p>Internal Name: signature-field-height</p>

10.5 PDF/A Output Properties

Set the properties described in [Table 10–4](#) to configure PDF/A output. For more information on PDF/A output see [Section 2.9, "Generating PDF/A and PDF/X Output."](#)

Table 10–4 PDF/A Output Properties

Property Name	Description
PDF/A ICC Profile Data	<p>Default: Default profile data provided by JVM</p> <p>Description: The name of the ICC profile data file, for example: CoatedFOGRA27.icc</p> <p>The ICC (International Color Consortium) profile is a binary file describing the color characteristics of the environment where this PDF/A file is intended to be displayed. (For more information, see the article at http://en.wikipedia.org/wiki/ICC_profile).</p> <p>The ICC profile that you select must have a major version below 4.</p> <p>To use a specific profile data file other than the default settings in the JVM, obtain the file and place it under <code><publisher repository>/Admin/Configuration</code>. When you set this property, you must also set a value for PDF/A ICC Profile Info (pdfa-icc-profile-info).</p> <p>Internal Name: pdfa-icc-profile-data</p>
PDF/A ICC Profile Info	<p>Default: sRGB IEC61966-2.1</p> <p>Description: ICC profile information (required when pdfa-icc-profile-data is specified)</p> <p>Internal Name: pdfa-icc-profile-info</p>
PDF/A file identifier	<p>Default: Automatically generated file identifier</p> <p>Description: One or more valid file identifiers set in the xmpMM:Identifier field of the metadata dictionary. To specify more than one identifier, separate values with a comma (,).</p> <p>Internal Name: pdfa-file-identifier</p>
PDF/A document ID	<p>Default: None</p> <p>Description: Valid document ID. The value is set in the xmpMM:DocumentID field of the metadata dictionary.</p> <p>Internal Name: pdfa-document-id</p>
PDF/A version ID	<p>Default: None</p> <p>Description: Valid version ID. The value is set in the xmpMM:VersionID field of the metadata dictionary.</p> <p>Internal Name: pdfa-version-id</p>
PDF/A rendition class	<p>Default: None</p> <p>Description: Valid rendition class. The value is set in the xmpMM:RenditionClass field of the metadata dictionary.</p> <p>Internal Name: pdfa-rendition-class</p>

10.6 PDF/X Output Properties

Set the properties described in [Table 10–5](#) to configure PDF/X output. The values that you set for these properties will depend on the printing device. Note the following restrictions on other PDF properties:

- pdf-version - value above 1.4 is not allowed for PDF/X-1a output
- pdf-security - must be set to False
- pdf-encryption-level - must be set to 0
- pdf-font-embedding - must be set to true

For more information on PDF/X output see [Section 2.9, "Generating PDF/A and PDF/X Output."](#)

Table 10-5 PDF/X Output Properties

Property Name	Description
PDF/X ICC Profile Data	<p>Default: None</p> <p>Description: (Required) The name of the ICC profile data file, for example: CoatedFOGRA27.icc.</p> <p>The ICC (International Color Consortium) profile is a binary file describing the color characteristics of the intended output device. (For more information, see the article at http://en.wikipedia.org/wiki/ICC_profile). For production environments, the color profile may be provided by your print vendor or by the printing company that prints the generated PDF/X file. The file must be placed under <bi publisher repository>/Admin/Configuration.</p> <p>Profile data is also available from Adobe (http://www.adobe.com/support) or colormangement.org (http://www.colormangement.org/).</p> <p>Internal Name: pdfx-dest-output-profile-data</p>
PDF/X output condition identifier	<p>Default: None</p> <p>Description: (Required) The name of one of the standard printing conditions registered with ICC (International Color Consortium). The list of standard CMYK printing conditions to use with PDF/X-1a is provided on the following ICC website: http://www.color.org/chardata/drsection1.xalter. The value that you enter for this property is a valid "Reference name," for example: FOGRA43.</p> <p>Choose the appropriate value for the intended printing environment. This name is often used to guide automatic processing of the file by the consumer of the PDF/X document, or to inform the default settings in interactive applications.</p> <p>Internal Name: pdfx-output-condition-identifier</p>
PDF/X output condition	<p>Default: None</p> <p>Description: A string describing the intended printing condition in a form that will be meaningful to a human operator at the site receiving the exchanged file. The value is set in OutputCondition field of OutputIntents dictionary.</p> <p>Internal Name: pdfx-output-condition</p>
PDF/X registry name	<p>Default: http://www.color.org</p> <p>Description: A registry name. Set this property when the pdfx-output-condition-identifier is set to a characterization name that is registered in a registry other than the ICC registry.</p> <p>Internal Name: pdfx-registry-name</p>

Table 10–5 (Cont.) PDF/X Output Properties

Property Name	Description
PDF/X version	<p>Default: PDF/X-1a:2003</p> <p>Description: The PDF/X version set in GTS_PDFXVersion and GTS_PDFXConformance fields of Info dictionary. PDF/X-1a:2003 is the only value currently supported.</p> <p>Internal Name: pdfx-version</p>

10.7 RTF Output Properties

Set the properties described in [Table 10–6](#) to configure RTF output.

Table 10–6 RTF Output Properties

Property Name	Description
Enable change tracking	<p>Default: false</p> <p>Description: Set to "true" to enable change tracking in the output RTF document.</p> <p>Internal Name: rtf-track-changes</p>
Protect document for tracked changes	<p>Default: false</p> <p>Description: Set to "true" to protect the document for tracked changes.</p> <p>Internal Name: rtf-protect-document-for-tracked-changes</p>
Default font	<p>Default: Arial:12</p> <p>Description: Use this property to define the font style and size in RTF output when no other font has been defined. This is particularly useful to control the sizing of empty table cells in generated reports. Enter the font name and size in the following format <FontName>:<size> for example: Arial:12. Note that the font you choose must be available to the BI Publisher processing engine at runtime. See Section 10.16, "Defining Font Mappings" for information on installing fonts for the BI Publisher server and also for the list of fonts predefined for BI Publisher.</p> <p>Internal Name: rtf-output-default-font</p>

10.8 HTML Output Properties

The properties described in [Table 10–7](#) can be set to govern HTML output files.

Table 10–7 HTML Output Properties

Property Name	Description
Show header	<p>Default: true</p> <p>Description: Set to "false" to suppress the template header in HTML output.</p> <p>Internal Name: html-show-header</p>
Show footer	<p>Default: true</p> <p>Description: Set to "false" to suppress the template footer in HTML output.</p> <p>Internal Name: html-show-footer</p>

Table 10–7 (Cont.) HTML Output Properties

Property Name	Description
Replace smart quotes	<p>Default: true</p> <p>Description: Set to "false" if you do not want curly quotes replaced with straight quotes in the HTML output.</p> <p>Internal Name: html-replace-smartquotes</p>
Character set	<p>Default: UTF-8</p> <p>Description: Specifies the output HTML character set.</p> <p>Internal Name: html-output-charset</p>
Make HTML output accessible	<p>Default: false</p> <p>Description: Specify true if you want to make the HTML output accessible.</p> <p>Internal Name: make-accessible</p>
Use percentage width for table columns	<p>Default: true</p> <p>Description: Set this property to true to render table columns according to a percentage value of the total width of the table rather than as a value in points. This property is especially useful if the browser renders tables with extremely wide columns. Setting this property to true improves the readability of the tables.</p> <p>Internal Name: html-output-width-in-percentage</p>
View Paginated	<p>Default: false</p> <p>Description: When set to true, HTML output will render in the report viewer with pagination features. These features include:</p> <ul style="list-style-type: none"> ■ Generated table of contents ■ Navigation links at the top and bottom of the page ■ Ability to skip to a specific page within the HTML document ■ Search for strings within the HTML document using the browser's search capability ■ Zoom in and out on the HTML document using the browser's zoom capability <p>Note that these features are supported for online viewing through the report viewer only.</p>

10.9 FO Processing Properties

The properties described in [Table 10–8](#) can be set to govern FO processing.

Table 10–8 FO Processing Properties

Property Name	Description
Use BI Publisher's XSLT processor	<p>Default: true</p> <p>Description: Controls BI Publisher's parser usage. If set to false, then XSLT is not parsed.</p> <p>Internal Name: xslt-xdoparser</p>

Table 10–8 (Cont.) FO Processing Properties

Property Name	Description
Enable scalable feature of XSLT processor	<p>Default: false</p> <p>Description: 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>Internal Name: xslt-scalable</p>
Enable XSLT runtime optimization	<p>Default: true</p> <p>Description: When set to "true", the overall performance of the FO processor is increased and the size of the temporary FO files generated in the temp directory is significantly decreased. Note that for small reports (for example 1-2 pages) the increase in performance is not as marked. To further enhance performance when you set this property to true, it is recommended that you set the property Extract attribute sets to "false". See Section 10.10, "RTF Template Properties."</p> <p>Internal Name: xslt-runtime-optimization</p>
Enable XPath Optimization	<p>Default: false</p> <p>Description: When set to "true", the XML data file is analyzed for element frequency. The information is then used to optimize XPath in XSL.</p> <p>Internal Name: xslt-xpath-optimization</p>
Pages cached during processing	<p>Default: 50</p> <p>Description: 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>Internal Name: system-cache-page-size</p>
Bidi language digit substitution type	<p>Default: None</p> <p>Description: Valid values are "None" and "National". When set to "None", Eastern European numbers are used. When set to "National", Hindi format (Arabic-Indic digits) is used. This setting is effective only when the locale is Arabic, otherwise it is ignored.</p> <p>Internal Name: digit-substitution</p>
Disable variable header support	<p>Default: false</p> <p>Description: If "true", prevents variable header support. Variable header support automatically extends the size of the header to accommodate the contents.</p> <p>Internal Name: fo-prevent-variable-header</p>
Add prefix to IDs when merging FO	<p>Default: false</p> <p>Description: 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>Internal Name: fo-merge-conflict-resolution</p>

Table 10–8 (Cont.) FO Processing Properties

Property Name	Description
Enable multithreading	<p>Default: false</p> <p>Description: 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>Internal Name: fo-multi-threads</p>
Disable external references	<p>Default: true</p> <p>Description: 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 the system. Set this to "false" if the report or template calls external files.</p> <p>Internal Name: xdk-secure-io-mode</p>
FO Parsing Buffer Size	<p>Default: 1000000</p> <p>Description: Sets the size of the buffer for the FO Processor. When the buffer is full, the elements from the buffer are 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 affects the memory consumption of the system.</p> <p>Internal Name: fo-chunk-size</p>
Enable XSLT runtime optimization for sub-template	<p>Default: true</p> <p>Note: The default is true on the BI Publisher server. If you call the FOProcessor directly, the default is false.</p> <p>Description: 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>Internal Name: xslt-do-import</p>
Enable PPTX native chart support	<p>Default: true</p> <p>Description: This property applies to PowerPoint 2007 output. When set to true, charts in PowerPoint 2007 output are rendered as native PowerPoint (PPTX) charts. If this property is set to false, then the chart is rendered as an embedded PNG image.</p> <p>Internal Name: pptx-native-chart</p>

10.10 RTF Template Properties

The properties described in [Table 10–9](#) can be set to govern RTF templates.

Table 10–9 RTF Template Properties

Property Name	Description
Extract attribute sets	<p>Default: Auto</p> <p>Description: The RTF processor automatically extracts attribute sets within the generated XSL-FO. The extracted sets are placed in an extra FO block, which can be referenced. This improves processing performance and reduces file size. Valid values are:</p> <ul style="list-style-type: none"> ■ Enable - extract attribute sets for all templates and subtemplates ■ Auto - extract attribute sets for templates, but not subtemplates ■ Disable - do not extract attribute sets <p>Internal Name: rtf-extract-attribute-sets</p>
Enable XPath rewriting	<p>Default: true</p> <p>Description: When converting an RTF template to XSL-FO, the RTF processor automatically rewrites the XML tag names to represent the full XPath notations. Set this property to "false" to disable this feature.</p> <p>Internal Name: rtf-rewrite-path</p>
Characters used for checkbox	<p>Default: Albany WT J;9746;9747/A</p> <p>Description: The BI Publisher default PDF output font does not include a glyph to represent a checkbox. If the template contains a checkbox, use this property to define a Unicode font for the representation of checkboxes in the PDF output. You must define the Unicode font number for the "checked" state and the Unicode font number for the "unchecked" state using the following syntax: fontname;<unicode font number for true value's glyph >;<unicode font number for false value's glyph></p> <p>Example: Albany WT J;9746;9747/A Note that the font that you specify must be made available to BI Publisher at runtime.</p> <p>Internal Name: rtf-checkbox-glyph</p>

10.11 PDF Template Properties

The properties described in [Table 10–10](#) can be set to govern PDF templates.

Table 10–10 PDF Template Properties

Property Name	Description
Remove PDF fields from output	<p>Default: false</p> <p>Description: Specify "true" to remove PDF fields from the output. When PDF fields are removed, data entered in the fields cannot be extracted.</p> <p>Internal Name: remove-pdf-fields</p>
Set all fields as read only in output	<p>Default: true</p> <p>Description: 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 updatable, set this property to "false".</p> <p>Internal Name: all-field-readonly</p>

Table 10–10 (Cont.) PDF Template Properties

Property Name	Description
Maintain each field's read only setting	<p>Default: false</p> <p>Description: 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."</p> <p>Internal Name: all-fields-readonly-asis</p>

10.12 Flash Template Properties

The properties described in [Table 10–11](#) can be set to govern Flash templates.

Table 10–11 Flash Template Properties

Property Name	Description
Page width of wrapper document	<p>Default: 792</p> <p>Description: Specify in points the width of the output PDF document. The default is 792, or 11 inches.</p> <p>Internal Name: flash-page-width</p>
Page height of wrapper document	<p>Default: 612</p> <p>Description: Specify in points the height of the output PDF document. The default is 612, or 8.5 inches.</p> <p>Internal Name: flash-page-height</p>
Start x position of Flash area in PDF	<p>Default: 18</p> <p>Description: 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>Internal Name: flash-startx</p>
Start y position of Flash area in PDF	<p>Default: 18</p> <p>Description: 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>Internal Name: flash-starty</p>
Width of Flash area	<p>Default: Same as flash width in points in swf</p> <p>Description: 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>Internal Name: flash-width</p>
Height of Flash area	<p>Default: Same as flash height in points in swf</p> <p>Description: 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>Internal Name: flash-height</p>

10.13 CSV Output Properties

Use the properties described in [Table 10–12](#) to control comma-separated value output.

Table 10–12 CSV Output Properties

Property Name	Description
CSV delimiter	Default: , Description: Specifies the character used to delimit the data in comma-separated value output. Other options are: Semicolon (;), Tab (\t) and Pipe ().
Remove leading and trailing white space	Default: false Description: Specify "True" to remove leading and trailing white space between data elements and the delimiter.

10.14 Excel 2007 Output Properties

Use the properties described in [Table 10–13](#) to control Excel 2007 output.

Table 10–13 Excel 2007 Output Properties

Property Name	Description
Show grid lines	Default: false Description: Set to true to show the Excel table grid lines in the report output.
Page break as a new sheet	Default: true Description: When set to "True" a page break that is specified in the report template generates a new sheet in the Excel workbook.
Minimum column width	Default: 3 (in points, 0.04 inch) Description: When the column width is less than the specified minimum and it contains no data, the column is merged with the preceding column.
Minimum row height	Default: 1 (in points, 0.01 inch) Description: When the row height is less than the specified minimum and it contains no data, the row is removed.

10.15 All Outputs

The property in [Table 10–14](#) applies to all outputs.

Table 10–14 All Outputs

Property Name	Description
Hide version number in output	Default: true Description: 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 want to include the version of BI Publisher (for example, Oracle BI Publisher 11.1.1.4.0) that generated the document, then set this property to false.
Use 11.1.1.5 compatibility mode	Reserved. Do not update unless instructed by Oracle.

10.16 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.

10.16.1 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 [Section 10.16.4, "BI Publisher's Predefined Fonts."](#)

The predefined fonts are located in the Oracle Business Intelligence Oracle home, in: `ORACLE_HOME/common/fonts`. If you want to map to another font, then you must place the font in this directory to make it available to BI Publisher at runtime. If the environment is clustered, then 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 the environment, then fonts are located in `$JAVA_HOME/jre/lib/fonts`.

10.16.2 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 apply to the selected report only.

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

10.16.3 Creating a Font Mapping

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

To create a Font Mapping:

1. Under RTF Templates or PDF Templates, select **Add Font Mapping**.
2. Enter the following on the **Add Font Mapping** page:
 - Base Font - enter the font family that is 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, then 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 [Section 10.16.4, "BI Publisher's Predefined Fonts."](#)

10.16.4 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 [Table 10–15](#).

Table 10–15 *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
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 [Table 10–16](#). All TrueType fonts are subsetted and embedded into PDF.

Table 10–16 TrueType Fonts

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)
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)
10	Andale Duospace WT TC	bold	bold	ADUOTCB.ttf	TrueType (Traditional Chinese flavor, Fixed width)

10.16.4.1 Included Barcode Fonts

BI Publisher also includes the barcode fonts listed in [Table 10–17](#).

Table 10–17 Included Barcode Fonts

Font File	Supported Algorithm
128R00.TTF	code128a, code128b, and code128c
B39R00.TTF	code39, code39mod43
UPCR00.TTF	upca, upce

Part II

Creating Style Templates and Sub Templates

This part describes how to create and implement style templates and sub templates. It includes the following chapters:

- [Chapter 11, "Creating and Implementing Style Templates"](#)
- [Chapter 12, "Understanding Subtemplates"](#)
- [Chapter 13, "Designing RTF Subtemplates"](#)
- [Chapter 14, "Designing XSL Subtemplates"](#)

Creating and Implementing Style Templates

A style template is an RTF template that contains style information that can be applied to other RTF layouts to achieve a consistent look and feel across your enterprise reports.

This chapter describes how to create and implement style templates. It includes the following sections:

- [Section 11.1, "Understanding Style Templates"](#)
- [Section 11.2, "Creating a Style Template RTF File"](#)
- [Section 11.3, "Uploading a Style Template File to the Catalog"](#)
- [Section 11.4, "Assigning a Style Template to a Report Layout"](#)
- [Section 11.5, "Updating a Style Template"](#)
- [Section 11.6, "Adding Translations to a Style Template Definition"](#)

11.1 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

11.1.1 About Styles Defined in the Style Template

The styles of the following elements can be defined in the style template:

11.1.1.1 Paragraph and Heading Styles

You can create a paragraph style in a style template. When this same named style is used in a report layout, the report layout inherits 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 strike through).

11.1.1.2 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.

11.1.1.3 Header and Footer Content

The header and footer regions of the style template are 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, then it is overwritten.

11.1.2 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 the style template.
3. Save this document as a .RTF file.
4. To ensure that you do not lose custom styles in Microsoft Word, also save the document as a Word Template file (.dot) or save the 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 the 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 applies the styles, header, and footer from the style template.

11.2 Creating a Style Template RTF File

The following sections describe how to define the style types in the Microsoft Word document. For more complete information see the Microsoft Word documentation.

11.2.1 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 the report as well.

To define a paragraph style type:

1. In the 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 the style. Choose style type: Paragraph. Format the 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 the new style is displayed in the list of available formats in the Styles and Formatting task pane.
5. Choose the new style and make an entry in the style template to display the style.

To apply the paragraph style type in the document:

1. Position the cursor within the paragraph (or text) to which you want to apply the style.
2. Select the style from list of available formats in the Styles and Formatting task pane. The style is applied to the paragraph.

To modify an existing style type:

1. In the 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.

11.2.2 Defining Styles for Tables

To define a table style type:

1. In the 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 the style. Choose style type: Table. Format the 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 the new style is displayed in the list of available formats in the Styles and Formatting task pane.
5. Choose the new style and make an entry in the style template to display the style.

To apply the table style type in the document:

1. Position the cursor within the table to which you want to apply the style.
2. Select the table style from list of available formats in the Styles and Formatting task pane. The style is applied to the table.

11.2.3 Defining a Header and Footer

You can define a header and footer in the style template. The contents and sizing of the header and footer in the style template are applied to the report layouts.

Important: If a header and footer have been defined in the report layout, then they are overwritten. The header and footer from the style template are applied.

To define a header and footer:

1. In the 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.

11.3 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.

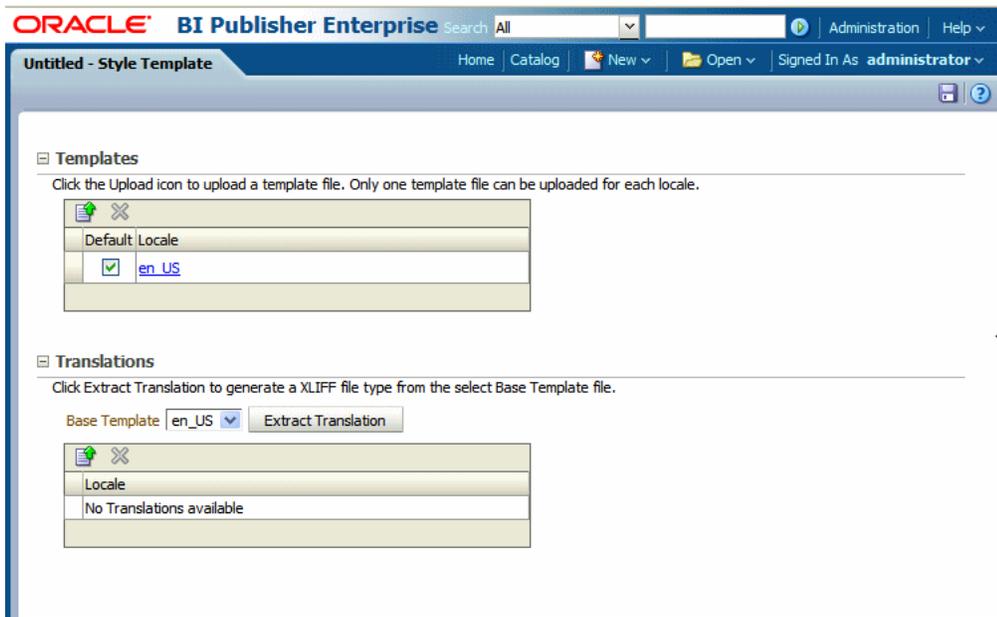
To upload a style template file:

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 (shown in Figure 11-1), use the **Browse** button to select the Template File. Select rtf as the **Type**, and select the appropriate **Locale**, as shown in Figure 11-2. Then click **OK**.

Figure 11-1 Upload Template File Dialog



Figure 11-2 Select the Appropriate Locale



The style template file is displayed in the Templates region as the locale name that 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 is automatically named as the locale regardless of the name that you give the file before upload.

6. If you are uploading multiple localized files, then select the file that is to be used as the default. For more information on localization of template files see [Section 11.6, "Adding Translations to a Style Template Definition."](#)

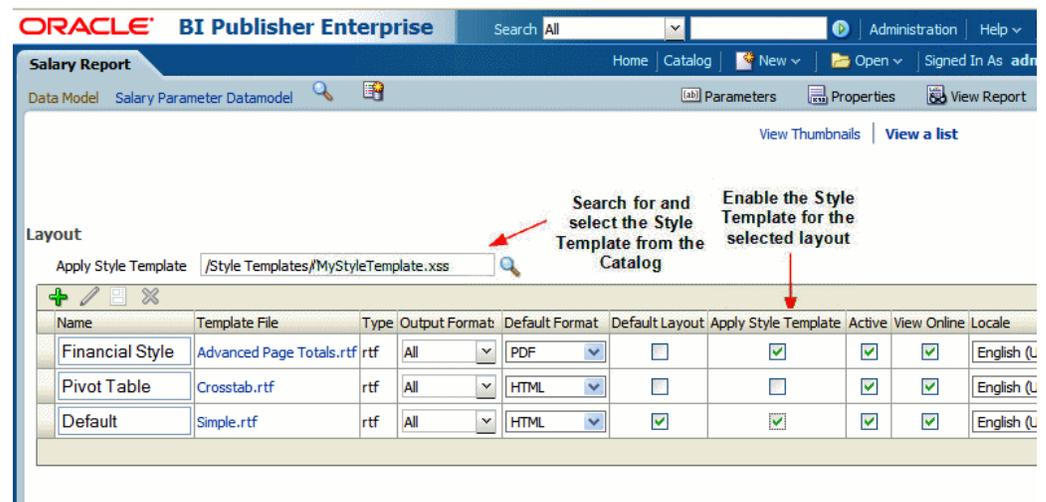
11.4 Assigning a Style Template to a Report Layout

To assign 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 the style template from the BI Publisher catalog.
3. For the layout templates that you want to use the style template, select the **Apply Style Template** box for the template. Note that the box is only enabled for RTF templates.

Figure 11–3 shows the actions required to enable a style template in the Report Editor.

Figure 11–3 Enabling a Style Template in the Report Editor



11.5 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.

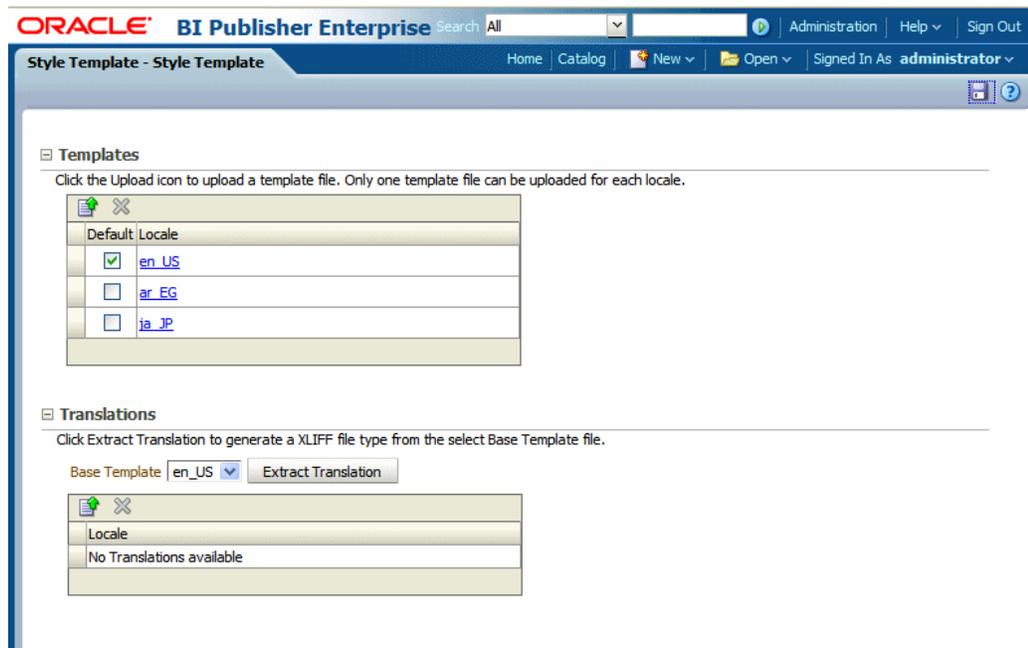
3. Delete the existing file.
4. Upload the edited file, choosing the same locale.

11.6 Adding Translations to a Style Template Definition

Style templates offer the same support for translations as RTF template files.

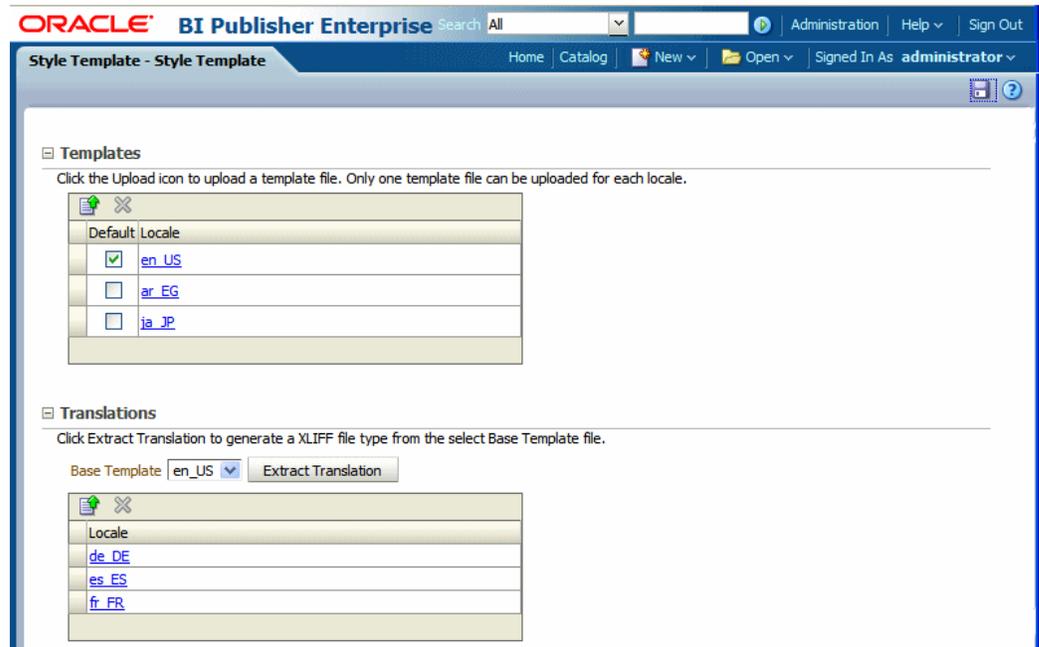
You can upload multiple translated RTF files under a single Style Template definition and assign the appropriate locale. These are displayed in the Templates region, as shown in [Figure 11-4](#).

Figure 11-4 *Templates Region*



Or you can generate an XLIFF (.xlf) file of the translatable strings, translate the strings, and upload the translated file. These are displayed in the Translations region, as shown in [Figure 11-5](#).

Figure 11–5 Translations Region



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 are 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 are displayed in the Translations region of the Style Template definition.

Understanding Subtemplates

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 chapter describes concepts for using subtemplates in BI Publisher.

This chapter includes the following sections:

- [Section 12.1, "What is a Subtemplate?"](#)
- [Section 12.2, "Supported Locations for Subtemplates"](#)
- [Section 12.3, "Testing Subtemplates from the Desktop"](#)
- [Section 12.4, "Creating the Sub Template Object in the Catalog"](#)
- [Section 12.5, "Calling a Subtemplate from an External Source"](#)

For information on designing an RTF subtemplate, see [Chapter 13, "Designing RTF Subtemplates."](#) For information on designing an XSL subtemplate, see [Chapter 14, "Designing XSL Subtemplates."](#)

12.1 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

12.1.1 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.

12.1.2 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.

12.2 Supported Locations for Subtemplates

It is recommended that you upload 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 [Section 12.5, "Calling a Subtemplate from an External Source"](#) for more information.

12.3 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 a 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 the 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.

12.4 Creating the Sub Template Object in the Catalog

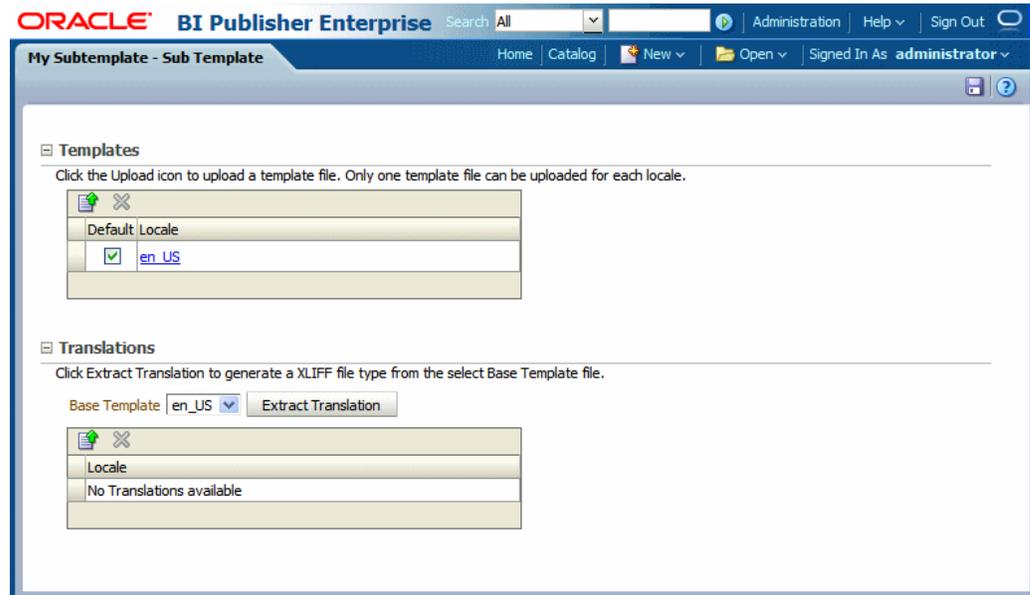
To upload a subtemplate file:

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 the 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**.

The subtemplate file is displayed in the **Templates** region as the locale name that 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**. Figure 12–1 shows a Sub Template named "My Subtemplate".

Figure 12–1 Sub Template



6. (RTF Sub Templates only) If you are uploading multiple localized files, then select the file that is to be used as the default. For more information on localization of template files, see [Section 13.6, "Adding Translations to an RTF Subtemplate."](#)

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 is automatically named as the locale regardless of the name that 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 use the Name that you choose here with the .xsb extension when you import the Sub Template object (for example: MySubtemplate.xsb).

12.5 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 subtemplates in the catalog.

Note that localization is not supported for subtemplates that are maintained outside the catalog.

12.5.1 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?>
```

12.5.2 Importing Subtemplates Outside the Catalog on the Same Server

If the subtemplate is located on the server, but not in the BI Publisher catalog, then enter the following:

```
<?import:file://{template_path}?>
```

where

template_path is the path to the subtemplate file on the server

For example:

```
<?import:file://c:/Folder/mySubtemplate.rtf?>
```

12.5.3 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 describes how to use RTF subtemplates to create and reuse functionality across multiple BI Publisher reports.

This chapter includes the following sections:

- [Section 13.1, "Understanding RTF Subtemplates"](#)
- [Section 13.2, "Process Overview for Creating and Implementing RTF Subtemplates"](#)
- [Section 13.3, "Creating an RTF Subtemplate File"](#)
- [Section 13.4, "Calling a Subtemplate from a Main Template"](#)
- [Section 13.5, "When to Use RTF Subtemplates"](#)
- [Section 13.6, "Adding Translations to an RTF Subtemplate"](#)

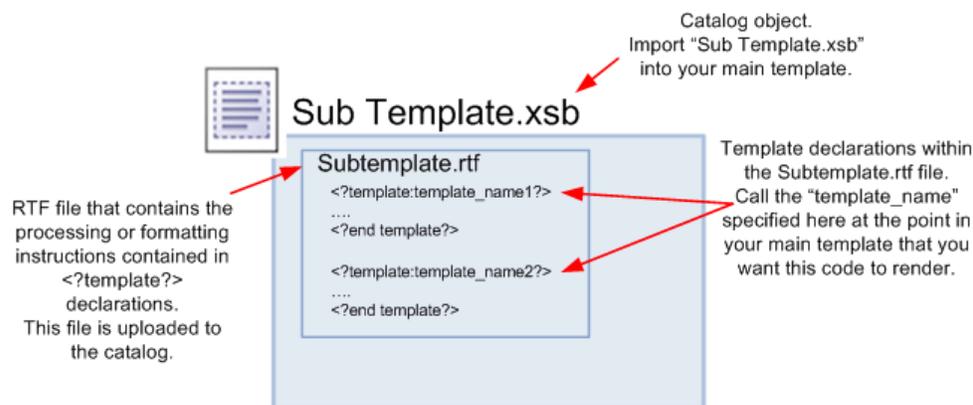
13.1 Understanding RTF Subtemplates

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.

Figure 13-1 illustrates the composition of an RTF Sub Template.

Figure 13-1 Composition of an RTF Sub Template



13.2 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 that contains the common components or processing instructions that you want 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 the template and subtemplate.

Tip: You can use the BI Publisher Desktop Template Viewer to test the 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 the local environment. See [Section 12.3, "Testing Subtemplates from the Desktop."](#)

4. Upload the main template to the report definition and create the Sub Template object in the catalog. See [Section 12.4, "Creating the Sub Template Object in the Catalog."](#)

13.3 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, [Figure 13–2](#) shows a sample RTF file that contains two subtemplates, one named commonHeader and one named commonFooter.

Figure 13–2 Sample RTF File

<?template:commonHeader?>	
ORACLE	Report Date: 17 August 2009
<?end template?>	
<?template:commonFooter?>	
Oracle Corporation 600 Oracle Parkway Redwood Shores CA 94065	Page 1 of 1
<?end template?>	

13.4 Calling a Subtemplate from a Main Template

To implement the subtemplate in a main template, you must make two entries in the 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.

13.4.1 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?>
```

Note: If the subtemplate resides in a personal folder under My Folders, the command to import the subtemplate is:

```
<?import:xdoxsl:///~username/path to subtemplate.xsb?>
```

where *username* is your user name.

For example, if user "myuser" uploads a subtemplate called "Template1" to a folder called "Subtemplates" under "My Folders", the correct import statement is:

```
<?import:xdoxsl:///~myuser/Subtemplates/Template1.xsb?>
```

13.4.2 Calling the Subtemplate to Render Its Contents

To call 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:

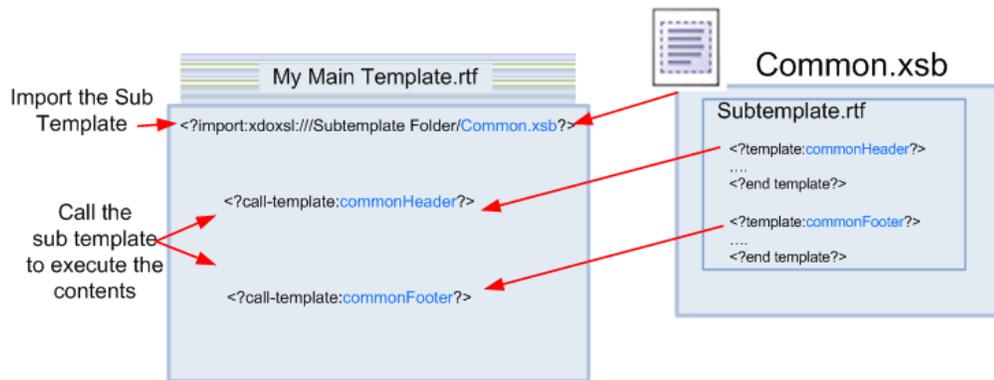
```
<?call-template:template_name?>
```

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).

Figure 13–3 illustrates the entries required in a main template.

Figure 13–3 Required Entries in Main Template



13.4.3 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}?>
```

13.4.4 Example

In this example, your company address is a fixed string that is displayed in all your templates. Rather than reproduce the string in all the templates, you can place it in one subtemplate and reference it from all the others.

To place the string in a subtemplate and reference it:

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 assigns the object the .xsb extension).
4. In the 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 the main template, in the location you want the address to appear, enter

```
<?call-template:MyAddress?>
```

At runtime the contents of the MyAddress subtemplate are 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: [Section 13.5, "When to Use RTF Subtemplates."](#)

13.5 When to Use RTF Subtemplates

Following are several common use-cases for RTF subtemplates.

13.5.1 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 are simplified and require updating only the subtemplate instead of each individual layout.

13.5.2 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 the 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.

13.5.2.1 Example

Assume you have a report that is sent to customers in India and the United States. You must 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.

The subtemplate file might 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?>
```

To call the subtemplate with the appropriate address format:

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 the 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 is properly applied, depending on the value of COUNTRY_CODE in the data.

13.5.3 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.

13.5.3.1 Example

Assume in the report data model that 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 the 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.

To display the 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 the 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 is determined based on the value of DeptName. For more information on CHOOSE statements in RTF templates, see [Section 4.10.4, "Inserting Choose Statements."](#)

13.5.3.2 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.

13.5.3.3 Example

This example illustrates setting up a subtemplate to contain a formula to calculate interest.

The subtemplate performs the interest calculation on the data in this report and passes the result back to the main template. The subtemplate accommodates the possibility that multiple reports that call this functionality might have different tag names for the components of the formula.

Assume that 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>
```

To set up a subtemplate to contain a formula for calculating interest:

1. In an RTF file, create a template declaration called "calcInterest". In this subtemplate 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 the 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

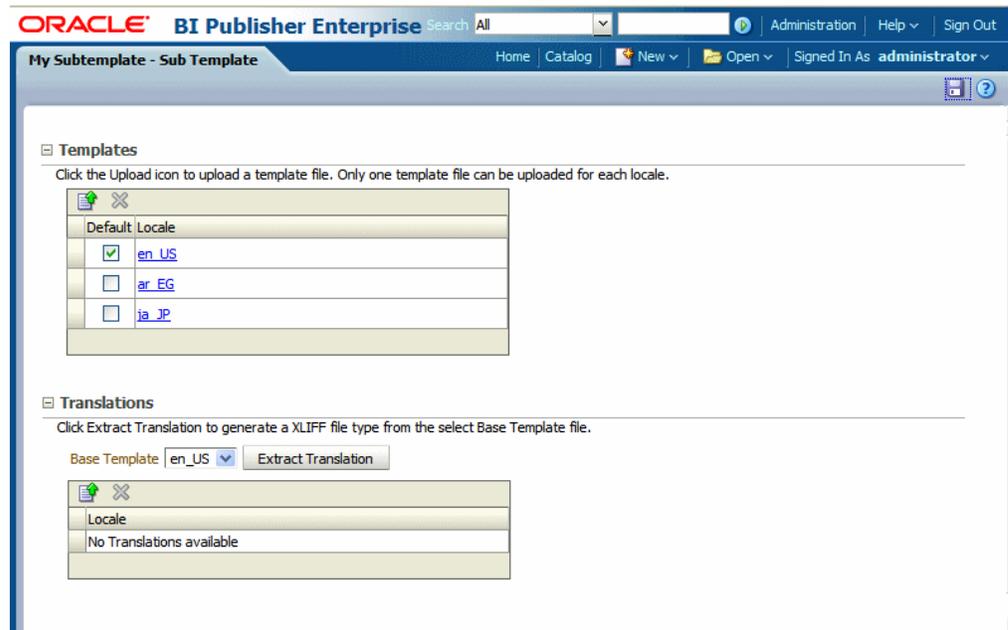
where the call to the subtemplate is made. Use this feature, for example, if you want to keep the results on the same line as a string of text that precedes the call.

13.6 Adding Translations to an RTF Subtemplate

RTF subtemplates offer the same support for translations as RTF template files.

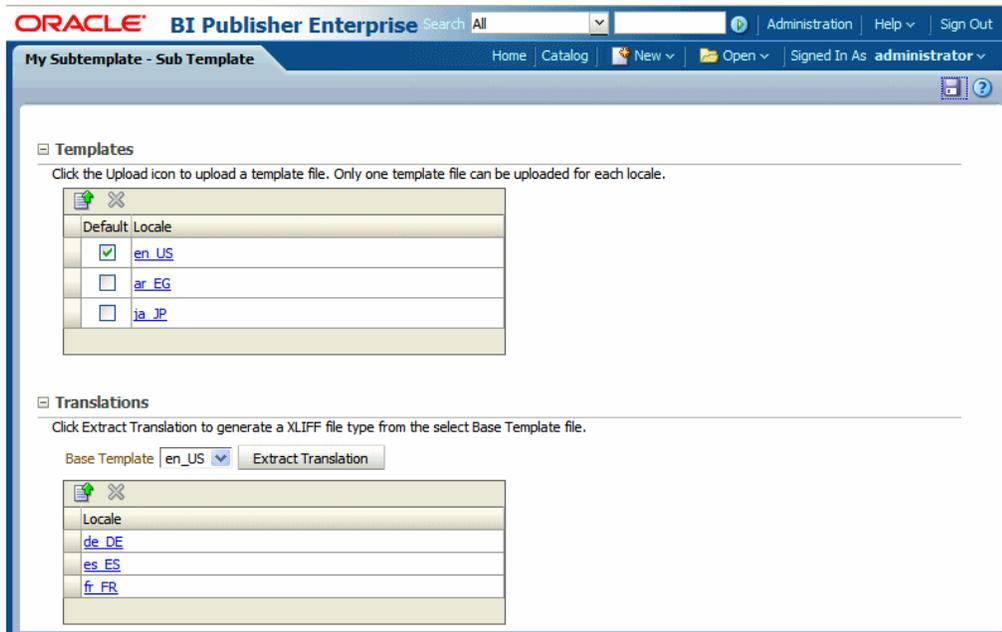
You can upload multiple translated RTF files under a single Sub Template definition and assign the appropriate locale. These are displayed in the Templates region, as shown in [Figure 13-4](#).

Figure 13-4 *Templates Region*



Or you can generate an XLIFF (.xlf) file of the translatable strings, translate the strings, and upload the translated file. These are displayed in the Translations region, as shown in [Figure 13-5](#).

Figure 13–5 Translations Region



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 are 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 are displayed in the Translations region of the Sub Template definition.

For more information on translations, see [Chapter 15, "Translation Support Overview and Concepts."](#)

Designing XSL Subtemplates

This chapter describes how to create XSL subtemplates to create reusable advanced functionality for your RTF templates.

This chapter includes the following sections:

- [Section 14.1, "Understanding XSL Subtemplates"](#)
- [Section 14.2, "Process Overview for Creating and Implementing XSL Subtemplates"](#)
- [Section 14.3, "Creating an XSL Subtemplate File"](#)
- [Section 14.4, "Calling an XSL Subtemplate from the Main Template"](#)
- [Section 14.5, "Creating the Sub Template Object in the Catalog"](#)
- [Section 14.6, "Example Uses of XSL Subtemplates"](#)

14.1 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. When uploaded to BI Publisher as a Sub Template object in the Catalog, this XSL file can be called from other RTF templates to execute the formatting or processing commands.

XSL subtemplates can handle complex data and layout requirements. Use XSL subtemplates to transform the data structure for a section of a report (for example, for a chart) or to create a style sheet to manage a complex layout.

14.1.1 Where to Put XSL Code in the RTF Main Template

When you call the XSL subtemplate within a 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 the RTF template.

For more information on inserting form fields in an RTF template see [Section 5.4.1, "Inserting a Field."](#)

14.2 Process Overview for Creating and Implementing XSL Subtemplates

Creating and implementing an XSL subtemplate consists of the following steps:

1. Create the XSL file that contains the common components or processing instructions to include in other templates.

An 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 that includes 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 the report definition and create the Sub Template object in the catalog.

14.3 Creating an XSL Subtemplate File

Enter the instructions in an editor that enables you to save the file as type ".xsl". 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>
```

Table 14–1 describes the components of the template declaration.

Table 14–1 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>
```

14.4 Calling an XSL Subtemplate from the Main Template

To implement the subtemplate in the main template, make two entries in the 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.

14.4.1 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?>
```

14.4.2 Calling the Subtemplate

The template statements that you defined within the XSL subtemplate file are applied to 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 method applies all the templates that are defined in the XSL subtemplate to the `data_element` specified. Based on the data content of `data_element`, appropriate functions in those templates are applied. See the following use case for a detailed example: [Section 14.6.1, "Handling XML Data with HTML Formatting."](#)

- By calling a template by name:

```
<xsl:call-template name="templateName"/>
```

This method calls the template by name and the template executes, similar to a function call. Here also parameters can be passed to the template call, similarly to an RTF subtemplate. See the next section: [Section 14.4.3, "Passing Parameters to an XSL Subtemplate."](#)

See the following use case for a detailed example: [Section 14.6.2, "Dynamically Applying Formatting to a Portion of Data."](#)

14.4.3 Passing Parameters to an XSL Subtemplate

To pass parameters to the XSL subtemplate:

1. 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>
```

14.5 Creating the Sub Template Object in the Catalog

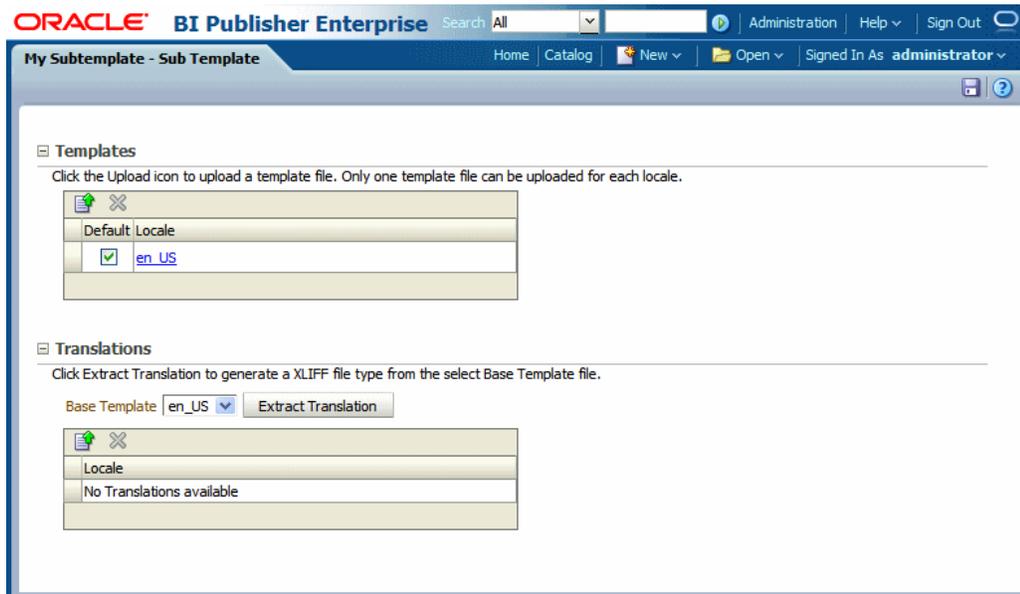
To upload the subtemplate file:

- On the global header click **New** and then click **Sub Template**. This launches an untitled Sub Template page.
- In the **Templates** region, click **Upload** to launch the **Upload Template File** dialog.
- Browse for and select the subtemplate file.
 - Type:** Select xsl for an XSL subtemplate file.
 - Locale:** Select the appropriate locale for the subtemplate file.
- Click **Upload**.

The subtemplate file is displayed in the **Templates** region as the locale name that you selected (for example: en_US).

- Click **Save**. In the **Save As** dialog choose the catalog folder in which to save the Sub Template. Enter the **Name** and click **Save**. Figure 14–1 shows a Sub Template named "My Subtemplate".

Figure 14–1 Sub Template



The Sub Template object is saved with the extension ".xsb". Use the Name that you choose here with the .xsb extension when you import the Sub Template to the report (for example: MySubtemplate.xsb).

Translations are not supported for XSL Sub Templates.

14.6 Example Uses of XSL Subtemplates

The following are examples of formatting that can be achieved in a report by using XSL subtemplates:

- [Section 14.6.1, "Handling XML Data with HTML Formatting"](#)
- [Section 14.6.2, "Dynamically Applying Formatting to a Portion of Data"](#)

14.6.1 Handling XML Data with HTML Formatting

If you have XML data that already contains HTML formatting and you want to preserve that formatting in the report, then 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 the data uses a simple
 for a break, then you must 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
- - bold tag

Assume a report requirement to display this to retain the formatting supplied by these tags as shown in [Figure 14-2](#).

Figure 14-2 Sample HTML Tags

Project Name	Project Management
Project Scope	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.
Project Definition	Information about current projects is not readily available to executives. 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>
```

To use this XSL syntax:

1. Upload this XSL subtemplate file to the BI Publisher catalog location: Shared Folders/Projects. Save this subtemplate file as "htmlmarkup.xsb".
2. In the main template enter the following to import the subtemplate file:

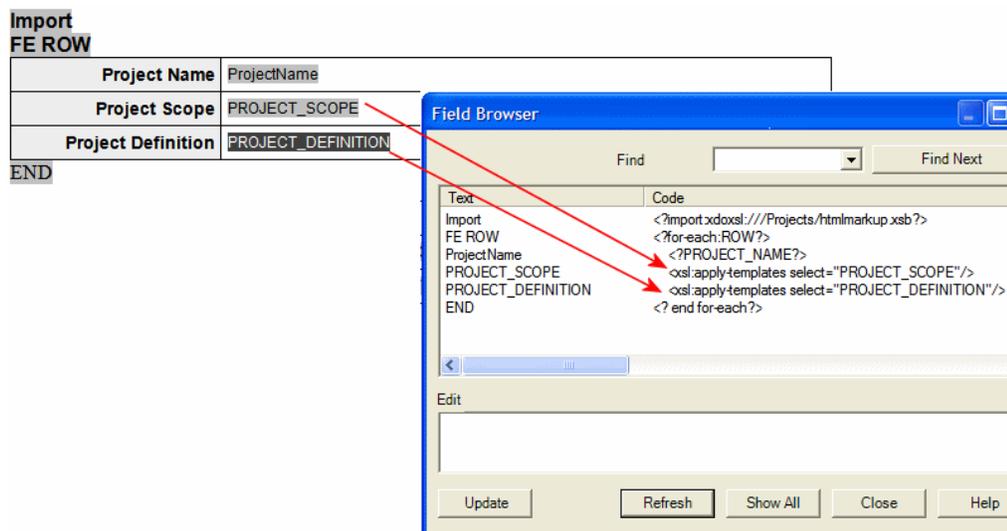
```
<?import:xdoxsl:///Projects/htmlmarkup.xsb?>
```

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" />
```

Figure 14-3 shows the field definitions in the template.

Figure 14-3 Field Definitions



The command tells the processor to apply all templates to the value of the element PROJECT_SCOPE and PROJECT_DEFINITION. It then cycles through the subtemplate functions looking for a match.

14.6.2 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 CO₂ is expected to display as CO₂ and H₂O is expected to display as H₂O.

```
<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 accepts the FORMULA field as a parameter, and then read one character at a time. It compares the character with 0 - 9 digits, and if there is a match, then that character is 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 character 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>
```

```
</xsl:call-template>
</xsl:if>
</xsl:template>
```

To use this XSL template statement:

1. Save this file as chemical.xsl.
2. Follow the instructions in [Section 12.4, "Creating the Sub Template Object in the Catalog."](#) Assume that you name the Sub Template "Chemical" (it is saved as Chemical.xsb) and place it in the following location: Shared Folders/Subtemplates.

3. In the main RTF template enter the import syntax:

```
<?import:xdoxsl:///Subtemplates/Chemical.xsb?>
```

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 are shown as expected: H₂O.

Part III

Translating Reports and Catalog Objects

This part provides information about translating reports and catalog objects. It includes the following chapters:

- [Chapter 15, "Translation Support Overview and Concepts"](#)
- [Chapter 16, "Translating Individual Templates"](#)
- [Chapter 17, "Translating Catalog Objects, Data Models, and Templates"](#)

Translation Support Overview and Concepts

This chapter provides an overview of concepts related to report and catalog translation in BI Publisher.

It includes the following sections:

- [Section 15.1, "Translation Support Overview"](#)
- [Section 15.2, "Working with Translation Files"](#)
- [Section 15.3, "Locale Selection Logic"](#)

15.1 Translation Support Overview

BI Publisher supports translation in two ways, as described in the following sections:

- [Section 15.1.1, "What Is Catalog Translation?"](#)
- [Section 15.1.2, "What Is Template Translation?"](#)

15.1.1 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 *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 see the item translations that are appropriate for the user interface language that they selected in their My Account preferences. Users see report

translations that are appropriate for the Report Locale that they selected in their My Account preferences.

15.1.2 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 need only the final report documents translated. For example, you must generate translated invoices to send to German and Japanese customers.

15.2 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:

- [Section 15.2.1, "What Is an XLIFF?"](#)
- [Section 15.2.2, "Structure of the XLIFF File"](#)

15.2.1 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>

15.2.2 Structure of the XLIFF File

The generated XLIFF file has the following structure:

```
<xliff>
  <file>
    <header>
      <body>
        <trans-unit>
          <source>
          <target>
          <note>
```

[Figure 15–1](#) shows an excerpt from an untranslated XLIFF file.

Figure 15–1 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="orphan"
    <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="aeccc17e" 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>

```

15.2.2.1 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 the International Organization for Standardization Web site at <http://www.iso.org/iso/home.html>

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 the target language. Table 15–1 shows examples of source-language and target-language attribute values appropriate for the given translations.

Table 15–1 Source-language and Target-language Attribute Values

Translation (Language/Territory)	source-language value	target-language value
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

15.2.2.2 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 is 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>
```

Note: Do not edit or delete the embedded data field tokens or you affect the merging of the XML data with the template.

15.2.2.3 <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>
```

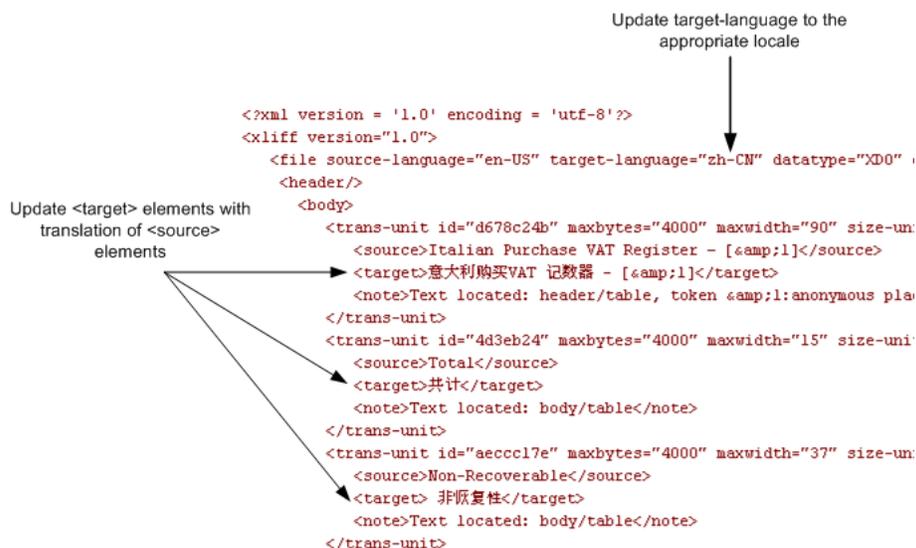
When you initially export the XLIFF file for translation, the source and target elements are all identical. To create the translation for this template, enter the appropriate translation for each source element string in its corresponding <target> element.

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>
```

[Figure 15–2](#) shows the sample XLIFF file from the previous figure updated with the Chinese translation.

Figure 15–2 Sample XLIFF File Updated with Chinese Translation



15.3 Locale Selection Logic

BI Publisher applies a translation based on the user's selected Report Locale. BI Publisher first tries 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, then BI Publisher tries 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), then BI Publisher selects 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)

With the same set of translations, if the locale selected is French (Switzerland), then the EmployeeTemplate.rtf (fr) is applied. Now if the available translations are 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) is applied. Even though there is a language match, BI Publisher does not match the different locales.

Therefore, to ensure that a French language translation is used when French is the selected language, regardless of the selected locale, 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 describes how to create and upload translated template files to provide translations for specific templates.

This chapter includes the following sections:

- [Section 16.1, "Overview"](#)
- [Section 16.2, "Types of Translations"](#)
- [Section 16.3, "Using the XLIFF Option"](#)
- [Section 16.4, "Using the Localized Template Option"](#)

16.1 Overview

The following template types can be translated individually:

- RTF layout files
- style templates
- subtemplates
- BI Publisher layouts (.xpt)

This chapter describes how to create and upload translated template files when you want to provide translations only for specific templates.

To translate the layouts within the broader scope of the catalog, see [Chapter 17, "Translating Catalog Objects, Data Models, and Templates."](#)

16.2 Types of Translations

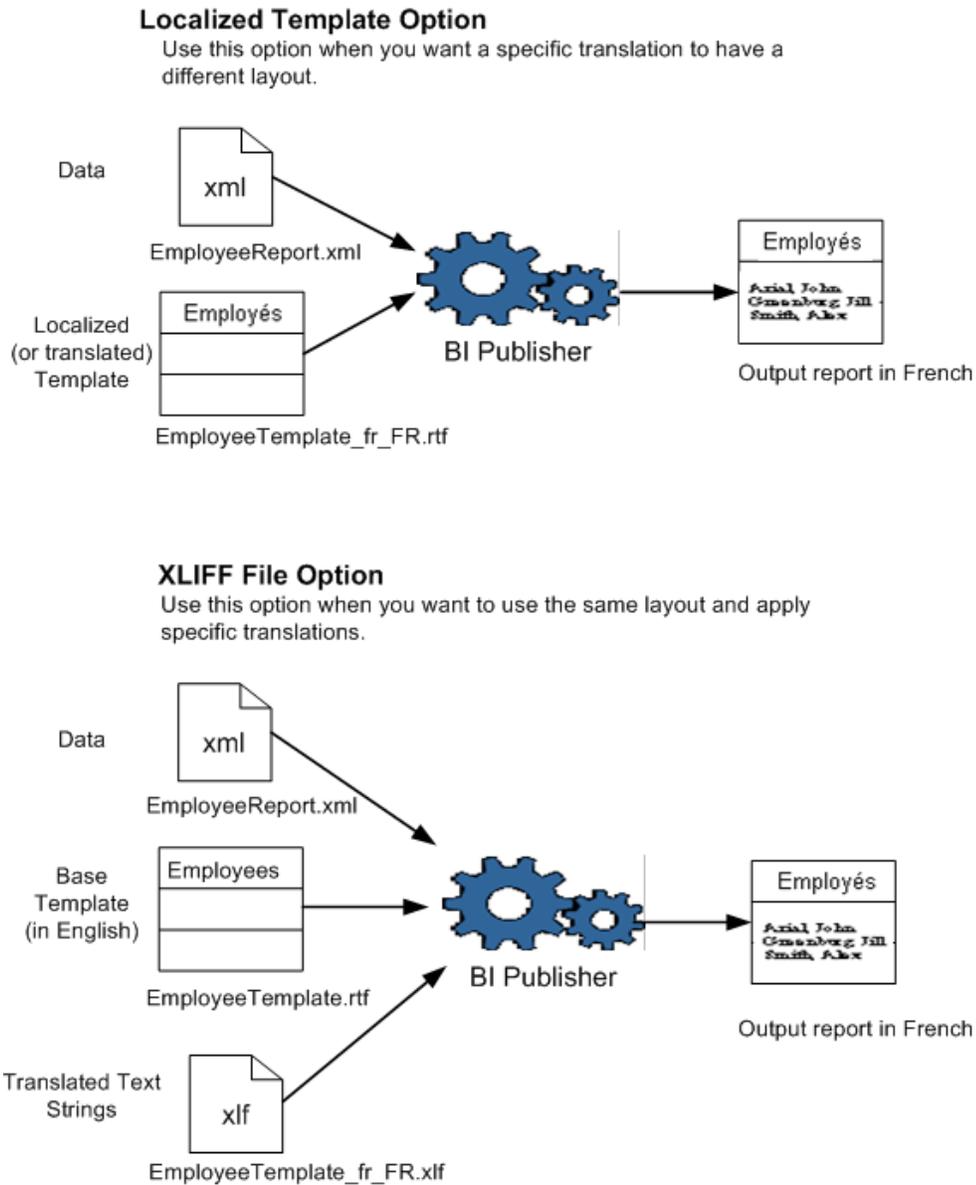
There are two options for adding translations for templates:

- Create a separate RTF template that is translated (a localized template). This option is available 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 require only translation of the text strings of the template layout, then use the XLIFF option.

The diagrams in [Figure 16–1](#) illustrate the translation concepts.

Figure 16–1 Translation Concepts

16.3 Using the XLIFF Option

The following sections describe using the XLIFF option to translate files:

- [Section 16.3.1, "Generating the XLIFF from a Template"](#)
- [Section 16.3.2, "Translating the XLIFF"](#)
- [Section 16.3.3, "Uploading the Translated XLIFF to BI Publisher"](#)

16.3.1 Generating the XLIFF from a Template

There are two methods for generating an XLIFF for a single template file, as described in the following sections:

- [Section 16.3.1.1, "Generating the XLIFF from the Template Builder"](#) (not supported for XPT templates)

- [Section 16.3.1.2, "Generating the XLIFF from the Layout Properties Page"](#)

16.3.1.1 Generating the XLIFF from the Template Builder

Note: This procedure assumes that you have installed the BI Publisher Template Builder for Microsoft Word. See "Get Started," in the *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 the 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.

16.3.1.2 Generating the XLIFF from the Layout Properties Page

You can generate the XLIFF file for report layout templates or for style templates and subtemplates.

To generate the XLIFF file 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 file to a local directory.

To generate the XLIFF file for style templates and subtemplates:

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 file to a local directory.

16.3.2 Translating the XLIFF

When you have downloaded the XLIFF file, it can be sent to a translation provider, or using a text editor, you can enter the translation for each string. See [Section 15.2.2, "Structure of the XLIFF File"](#) 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 associate these translations to the original template. See [Section 16.3.3, "Uploading the Translated XLIFF to BI Publisher."](#)

16.3.3 Uploading the Translated XLIFF to BI Publisher

To upload the translated XLIFF:

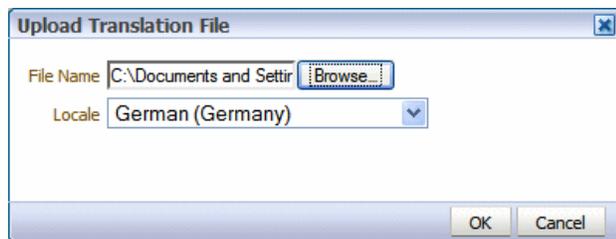
1. Navigate to the report, subtemplate, or style template in the catalog and click **Edit** to open it for editing.

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 **Translations** region, click the **Upload** toolbar button.
3. In the **Upload Translation File** dialog (shown in [Figure 16-2](#)), locate the file in a local directory and select the **Locale** for this translation.

Figure 16-2 Upload Translation File Dialog



4. Click OK to upload the file and view it in the **Translations** table.

16.4 Using the Localized Template Option

If you must design a different layout for the reports that you present for different localizations, then you can create a 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 following sections describe using the localized template option:

- [Section 16.4.1, "Designing the Localized Template File"](#)
- [Section 16.4.2, "Uploading the Localized Template to BI Publisher"](#)

16.4.1 Designing the Localized Template File

Use the same tools that you used to create the base template file, translating the strings and customizing the layout as desired for the locale.

16.4.2 Uploading the Localized Template to BI Publisher

To upload the localized template:

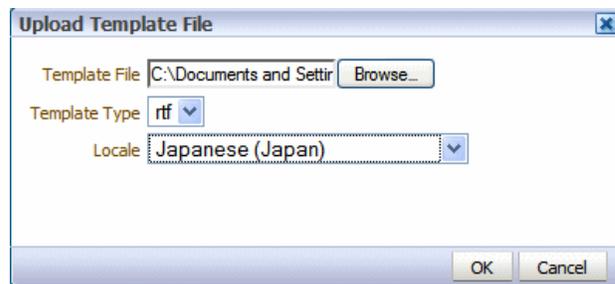
1. Navigate to the report, subtemplate, or style template in the catalog and click **Edit** to open it for editing.

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 (shown in [Figure 16-3](#)), locate the file in a local directory, select rtf as the **Template Type** and select the **Locale** for this template file.

Figure 16-3 Upload Template File Dialog



4. Click **OK** to upload the file and view it in the **Templates** table.

Translating Catalog Objects, Data Models, and Templates

This chapter describes translating BI Publisher catalog objects, data models, and templates using the Export XLIFF function that is available at the catalog level.

This chapter includes the following sections:

- [Section 17.1, "Overview"](#)
- [Section 17.2, "What Can Be Translated?"](#)
- [Section 17.3, "Exporting the XLIFF File"](#)
- [Section 17.4, "Identifying and Updating the Object Tags"](#)
- [Section 17.5, "Importing the XLIFF File"](#)

17.1 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 are 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).

If BI Publisher is integrated with Oracle Business Intelligence Enterprise Edition, then 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 *Oracle Fusion Middleware System Administrator's Guide for Oracle Business Intelligence Enterprise Edition* for information about components that can be translated in an integrated catalog.

17.2 What Can Be Translated?

[Table 17-1](#) shows what strings can be translated.

Table 17–1 *Strings That Can Be Translated*

Object	What Can Be Translated	Preference That Determines Translation Displayed
Folder	Name Description	UI Language (applies to all)
Data Model	Name Description Data Display Name	UI Language (applies to all)
Report	Name Description Layout Names Data Model Reference Parameter Name	UI Language (applies to all)
Style Template	Name Static text in the template	UI Language Report Locale
Sub Template	Name Static text in the template	UI Language Report Locale
BI Publisher Layouts (.xpt)	Static text in the layout	Report Locale
RTF Layouts	Static text in the layout	Report Locale

17.2.1 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.

17.3 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**, as shown in [Figure 17–1](#).

Figure 17–1 *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.

17.4 Identifying and Updating the Object Tags

For information on how to manually update the XLIFF files with translation strings, see [Section 15.2.1, "What Is an XLIFF?"](#)

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 [Figure 17–2](#).

Figure 17–2 Values for Two-Letter Language Code

```

<?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>
  </body>
</xliff>

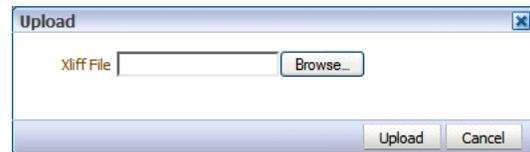
```

17.5 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 [Figure 17–3](#).

Figure 17–3 Upload Dialog

3. Click **Browse** to locate the translated file and then select the appropriate locale from the list.
4. Click **Upload**.

Part IV

Reference Information

This part provides reference information for Oracle BI Publisher. It includes the following appendices:

- [Appendix A, "Techniques for Handling Large Output Files"](#)
- [Appendix B, "Extended Function Support in RTF Templates"](#)
- [Appendix C, "Designing Accessible Reports"](#)
- [Appendix D, "Supported XSL-FO Elements"](#)

Techniques for Handling Large Output Files

This appendix describes techniques that are available to improve performance when the report generates very large PDF output files.

It includes the following sections:

- [Section A.1, "Reusing Static Content"](#)
- [Section A.2, "Generating Zipped PDF Output"](#)
- [Section A.3, "Implementing PDF Splitting for an RTF Template"](#)
- [Section A.4, "Implementing PDF Splitting for a PDF Template"](#)

A.1 Reusing Static Content

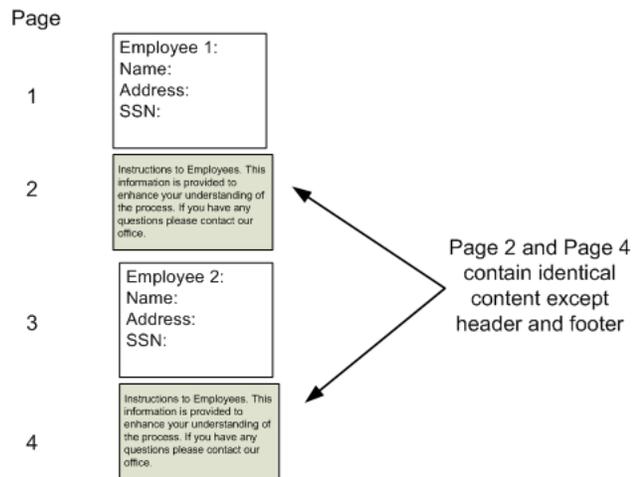
This section describes how to reuse static, repeating content in a PDF report output to reduce the overall PDF file size. This section contains the following topics:

- [Section A.1.1, "What Is Static Content Reuse?"](#)
- [Section A.1.2, "Limitations of this Feature"](#)
- [Section A.1.3, "Defining Reusable Content in an RTF Template"](#)
- [Section A.1.4, "Example"](#)

A.1.1 What Is Static Content Reuse?

If the 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), then 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 expected output shown in [Figure A-1](#).

Figure A-1 W-2 Form Report 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 includes the static content in the generated PDF document only once and references it in other places when needed, thereby reducing the overall output file size.

A.1.2 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.

A.1.3 Defining Reusable Content in an RTF Template

To define the static content to be reused, use the following tags around the content in the 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.

A.1.4 Example

The following example illustrates an implementation of this feature. The sample report generates one occurrence per employee. The generated report has employee-specific

information on the front page of each occurrence, and static instructions that print on the back of each occurrence. A section break occurs after each employee to reset page numbering.

Figure A-2 illustrates this template structure.

Figure A-2 RTF Template Structure

<?for-each@section:employee?>

Lastname	Firstname	MI
Address	City	State
Year	Employer	Earnings

Page Break.....

<?reusable-static-content:?>

<p>Notice to Employee</p> <p>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</p>	<p>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</p>
--	---

<?end reusable-static-content?>

<?end for-each?>

A.2 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.

A.2.1 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).

A.2.2 Design Time Considerations

To enable report splitting, the report designer must determine the following:

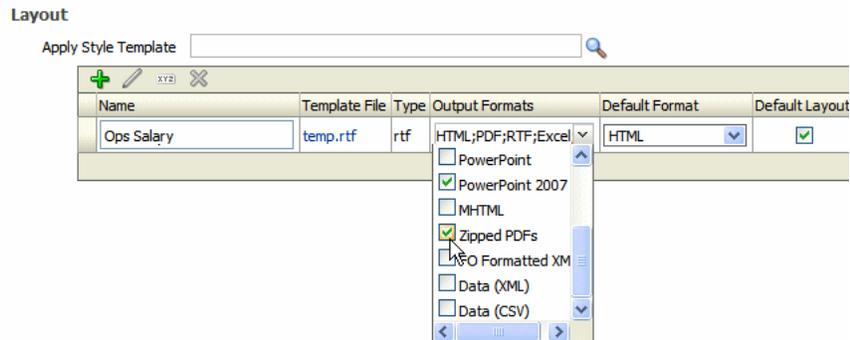
- Select a "repeat" element to serve as the counter.

- Determine how many instances of the repeat element occur per PDF file.
- Select which data elements to include in the generated index file.

A.2.3 Selecting the Output Type

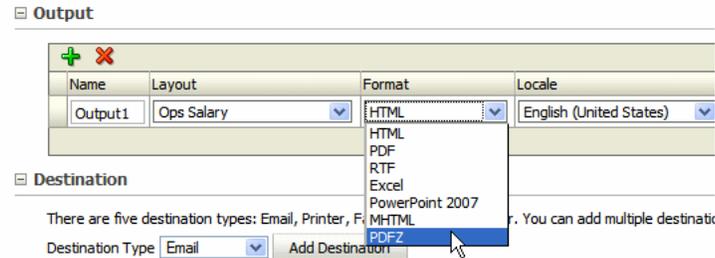
After uploading the template to the report definition, enable Zipped PDFs as an output type, as shown in [Figure A-3](#).

Figure A-3 Enabling Zipped PDFs as an Output Type



When scheduling the report, select PDFZ as the output type, as shown in [Figure A-4](#).

Figure A-4 Selecting PDFZ as the Output Type



A.3 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:

- [Section A.3.1, "Entering the Commands in an RTF Template"](#)
- [Section A.3.2, "Example - split by each department"](#)

A.3.1 Entering the Commands in an RTF Template

When you design a template to use this feature, you must add commands to specify the following:

- What element in the data is repeated (using the simple for-each command)
- How many occurrences of the element are 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 the 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 that you choose that is 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 provides 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 are written to the index files.

A.3.2 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 the 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?>
```

A.4 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.

A.4.1 Entering the Commands in the PDF Template

To enable this feature for a PDF template, enter the three form fields listed in [Table A-1](#) in the template with the specified commands in the "Tooltip" field.

Table A-1

Form Field Name	Tooltip Command
REPEAT-ELEMENT	<?repeat-element: <i>element name</i> ?> where <i>element_name</i> is the XML tag name of the repeating element that is counted. Example: <?repeat-element:emp_id?>
CATALOG-INDEX-INFO	<?catalog-index-info:'Name'; <i>element_name</i> ?> where 'Name' is the label that appears in the index file for the <i>element_name</i> that you specify. The index generates a "From" and "To" listing for each file in the zipped set. Example: <?catalog-index-info:'Last Name';LAST_NAME?> Note that you can include multiple occurrences of the catalog-index-info command to include multiple data elements in the index file.
SPLIT-COUNT	<?split-count: <i>n</i> ?> where <i>n</i> is the number of occurrences of the repeat-element that triggers the creation of a new file. Example: <?split-count:10000?>

Extended Function Support in RTF Templates

This appendix describes SQL and XSL functions extended by BI Publisher for use in RTF templates.

It includes the following sections:

- [Section B.1, "Extended SQL and XSL Functions"](#)
- [Section B.2, "XSL Equivalents"](#)
- [Section B.3, "Using FO Elements"](#)

B.1 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.

You cannot mix xdox statements with XSL expressions in the same context. For example, assume that you had two elements, `FIRST_NAME` and `LAST_NAME` to concatenate into a 30-character field and right pad the field with the character "x". You could not use the following:

Incorrect:

```
<?xdofx:rpadd(concat(FIRST_NAME, LAST_NAME), 30, 'x')?>
```

because `concat` is an XSL expression. Instead, you could use the following:

Correct:

```
<?xdofx:rpadd(FIRST_NAME || LAST_NAME), 30, 'x')?>
```

The supported functions are shown in [Table B-1](#).

Table B-1 Supported Functions

SQL Statement or XSL Expression	Usage	Description
2+3	<?xdoxf:2+3?>	Addition
2-3	<?xdoxf:2-3?>	Subtraction
2*3	<?xdoxf:2*3?>	Multiplication
2 div 3	<?xdoxf:2 div 3?>	Division
2**3	<?xdoxf:2**3?>	Exponential
3 2	<?xdoxf:3 2?>	Concatenation
sdiv()	<?xdoxslt:sdiv(num1,num2, string)?>	Safe divide function returns a specified value if the result of the function is not a number (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'.
lpad('aaa',10, '.')	<?xdoxf:lpad('aaa',10, '.')?>	The lpad function pads the left side of a string with a specific set of characters. The syntax for the lpad function is: lpad(string1,padded_length,[pad_string])string1 is the string to pad characters to (the left-hand side).padded_length is the number of characters to return.pad_string is the string that is padded to the left-hand side of string1 .
rpad('aaa',10, '.')	<?xdoxf:rpad('aaa',10, '.')?>	The rpad function pads the right side of a string with a specific set of characters. The syntax for the rpad function is: rpad(string1,padded_length,[pad_string]).string1 is the string to pad characters to (the right-hand side).padded_length is the number of characters to return.pad_string is the string that is padded to the right-hand side of string1
trim()	<?xdoxslt:trim(' a ')?>	Removes spaces in a string. Enter the text to be trimmed, the function returns the trimmed text.
ltrim()	<?xdoxslt:ltrim(' a ')?>	Removes the leading white spaces in a string.
rtrim()	<?xdoxslt:rtrim(' a ')?>	Removes the trailing white spaces in a string.

Table B-1 (Cont.) Supported Functions

SQL Statement or XSL Expression	Usage	Description
truncate	<?xdoxslt:truncate (<i>number</i> [, <i>integer</i>])?>	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. Example: <?xdoxslt:truncate(-2.3333)?> returns -2 Example: <?xdoxslt:truncate(2.7777, 2)?> returns 2.77 Example: <?xdoxslt:truncate(27.7777, -1)?> returns 20
replicate	<?xdoxslt:replicate('string', <i>integer</i>)?>	The replicate function replicates the specified string the specified number of times. Example: <?xdoxslt:replicate('oracle', 3)?> returns oracleoracleoracle
decode('xxx','bbb','ccc','xxx','ddd')	<?xdofx:decode('xxx','bbb','ccc','xxx','ddd')?>	The decode function has the functionality of an IF-THEN-ELSE statement. The syntax for the decode function is: decode(expression, search, result [,search, result]...[, default]) <i>expression</i> is the value to compare. <i>search</i> is the value that is compared against <i>expression</i> . <i>result</i> is the value returned, if <i>expression</i> is equal to <i>search</i> . <i>default</i> is returned if no matches are found.
Instr('abcabcabc','a',2)	<?xdofx:Instr('abcabcabc','a',2)?>	The <i>instr</i> function returns the location of a substring in a string. The syntax for the <i>instr</i> function is: instr(string1,string2,[start_position],[nth_appearance]) <i>string1</i> is the string to search. <i>string2</i> is the substring to search for in <i>string1</i> . <i>start_position</i> is the position in <i>string1</i> where the search starts. 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> . <i>nth_appearance</i> is the <i>nth</i> appearance of <i>string2</i> .
substr('abcdefg',2,3)	<?xdofx:substr('abcdefg',2,3)?>	The <i>substr</i> function allows you to extract a substring from a string. The syntax for the <i>substr</i> function is: substr(string, start_position, length) <i>string</i> is the source string. <i>start_position</i> is the position for extraction. The first position in the string is always 1. <i>length</i> is the number of characters to extract.

Table B-1 (Cont.) Supported Functions

SQL Statement or XSL Expression	Usage	Description
left	<?xdoxslt:left('abcdefg', 3)?>	Enables you to extract the specified number of characters from a string, starting from the left. The syntax is left(string, Numchars) For example, <?xdoxslt:left('abcdefg', 3)?> returns abc
right	<?xdoxslt:right('abcdefg', 3)?>	Enables you to extract the specified number of characters from a string, starting from the right. The syntax is right(string, Numchars) For example, <?xdoxslt:right('abcdefg', 3)?> returns efg
replace(name,'John','Jon')	<?xdox:replace(name,'John','Jon')?>	The replace function replaces a sequence of characters in a string with another set of characters. The syntax for the replace function is: replace(string1,string_to_replace,[replacement_string])string1 is the string to replace a sequence of characters with another set of characters.string_to_replace is the string that is searched for in string1.replacement_string is optional. All occurrences of string_to_replace are replaced with replacement_string in string1.
to_number('12345')	<?xdox:to_number('12345')?>	Function to_number converts char, a value of CHAR, VARCHAR2, NCHAR, or NVARCHAR2 datatype that contains a number in the format that is specified by the optional format model fmt, to a value of NUMBER datatype.
format_number	<?xdoxslt:format_number(12345, n, \$_XDOLOCALE)?>	Converts a number to a string and formats the number according to the locale specified in \$_XDOLOCALE and to the number of decimal positions specified in n using Java's default symbols. For example: <?xdoxslt:format_number(-12345, 2, 'fr-FR')?> returns -12 345,00
format_number	<?xdoxslt:format_number(12345, n, s1, s2,\$_XDOLOCALE)?>	Converts a number to a string and uses the specified separators: s1 for the thousand separator and s2 for the decimal separator. For example: <?xdoxslt:format_number(12345, 2, 'g', 'd', \$_XDOLOCALE)?> returns 12g345d00
pat_format_number	<?xdoxslt:pat_format_number(12345, '##,##0.00', \$_XDOLOCALE)?>	Returns a number formatted with the specified pattern. For example: <?xdoxslt:pat_format_number(12345, '##,##0.00', \$_XDOLOCALE)?> returns 12,345.00

Table B-1 (Cont.) Supported Functions

SQL Statement or XSL Expression	Usage	Description
to_char(12345)	<?xdoxf:to_char('12345')?>	Use the TO_CHAR function to translate a value of NUMBER datatype to VARCHAR2 datatype.
to_date	<?xdoxf:to_date (char [,fmt [, 'nlsparam']])	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.
format_date()	<?xdoxslt:format_date(./AnyDate,'yyyy-MM-dd','M M/dd/yyyy',\$_XDOLOCALE,\$_XDOTIMEZONE)?>	Reads date in one format and creates in another format.
sysdate()	<?xdoxf:sysdate()?>	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. See Section 4.15.13, "Displaying the System Date (sysdate) in Reports" for information on properly formatting the sysdate in report output.
current_date()	<?xdoxslt:current_date(\$_XDOLOCALE,\$_XDOTIMEZONE)?> Example: <?xdoxslt:current_date('ja-JP','Asia/Tokyo')?>	Returns the current date in "yyyy-MM-dd" format in the given locale and timezone. This function supports only the Gregorian calendar.
current_time()	<?xdoxslt:current_time(\$_XDOLOCALE,\$_XDOTIMEZONE)?> Example: <?xdoxslt:current_time('ja-JP','Asia/Tokyo')?>	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.

Table B-1 (Cont.) Supported Functions

SQL Statement or XSL Expression	Usage	Description
date_diff	<pre><?xdoxslt:date_diff('y', 'YYYY-MM-DD', 'YYYY-MM-DD', \$_XDOLOCALE, \$_XDOTIMEZONE)?></pre>	<p>This function provides a method to get the difference between two dates in the given locale. The dates must be in "yyyy-MM-dd" format. This function supports only the Gregorian calendar. The syntax is as follows:</p> <pre><?xdoxslt:date_diff('format', 'YYYY-MM-DD', 'YYYY-MM-DD', \$_XDOLOCALE, \$_XDOTIMEZONE)?></pre> <p>where format is the time value for which the difference is to be calculated. Valid values are:</p> <ul style="list-style-type: none"> ▪ y - for year ▪ m - for month ▪ w - for week ▪ d - for day ▪ h - for hour ▪ mi - for minute ▪ s - for seconds ▪ ms - for milliseconds <p>Example: <pre><?xdoxslt:date_diff('y', '2000-04-08', '2001-05-01', \$_XDOLOCALE, \$_XDOTIMEZONE)?></pre> returns 1</p> <p>Example: <pre><?xdoxslt:date_diff('m', '2001-04-08', '2000-02-01', \$_XDOLOCALE, \$_XDOTIMEZONE)?></pre> returns -14</p> <p>Example: <pre><?xdoxslt:date_diff('d', '2006-04-08', '2006-04-01', \$_XDOLOCALE, 'America/Los_Angeles')?></pre> returns -7</p>
sec_diff	<pre><?xdoxslt:sec_diff('2000-04-08T20:00:00', '2000-04-08T21:00:00', \$_XDOLOCALE, \$_XDOTIMEZONE)?></pre>	<p>This function provides a method to get the difference between two dates in seconds in the given locale. The dates must be in "yyyy-MM-dd'T'HH:mm:ss". This function supports only Gregorian calendar. Example: <pre><?xdoxslt:sec_diff('2000-04-08T20:00:00', '2000-04-08T21:00:00', \$_XDOLOCALE, \$_XDOTIMEZONE)?></pre> returns 3600</p>
get_day	<pre><?xdoxslt:get_day('2000-04-08', \$_XDOLOCALE)?></pre>	<p>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. Example: <pre><?xdoxslt:get_day('2000-04-08', \$_XDOLOCALE)?></pre> returns 8</p>

Table B-1 (Cont.) Supported Functions

SQL Statement or XSL Expression	Usage	Description
get_month	<?xdoxslt:get_month('2000-04-08', \$_XDOLOCALE)?>	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. Example: <?xdoxslt:get_month('2000-04-08', \$_XDOLOCALE)?> returns 4
get_year	<?xdoxslt:get_year('2000-04-08', \$_XDOLOCALE)?>	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. Example: <?xdoxslt:get_year('2000-04-08', \$_XDOLOCALE)?> returns 2000
month_name	<?xdoxslt:month_name(1, 0, \$_XDOLOCALE)?>	This function provides a method to get the name of the month in the given locale. This function supports only the Gregorian calendar. The syntax for this function is: <?xdoxslt:month_name(<i>month</i> , [<i>abbreviate?</i>], \$_XDOLOCALE)?> where <i>month</i> is the numeric value of the month (January = 1) and [<i>abbreviate?</i>] is the value 0 for do not abbreviate or 1 for abbreviate. Example: <?xdoxslt:month_name(12, 1, 'fr-FR')?> returns dec. Example: <?xdoxslt:month_name(1, 0, \$_XDOLOCALE)?> returns January
maximum	<?xdoxslt:maximum(ELEMENT_NAME)?>	Returns the maximum value of the element in the set.
abs	<?xdoxslt:abs(-123.45)?>	Returns the absolute value of the number entered. Example: <?xdoxslt:abs(-123.45)?> Returns: 123.45
chr	<?xdoxf: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	<?xdoxf:ceil(<i>n</i>)?>	CEIL returns smallest integer greater than or equal to <i>n</i> .
floor	<?xdoxf:floor(<i>n</i>)?>	FLOOR returns largest integer equal to or less than <i>n</i> .
round (SQL function)	<?xdoxf:round (<i>number</i> [, <i>integer</i>])?>	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. Example: <?xdoxf:round (2.777)?> returns 3 Example: <?xdoxf:round (2.777, 2)?> returns 2.78

Table B-1 (Cont.) Supported Functions

SQL Statement or XSL Expression	Usage	Description
round (XSLT function)	<?xdoxslt:round (<i>number</i> [, <i>integer</i>])?>	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. Example: <?xdoxslt:round (2.777)?> returns 3 Example: <?xdoxslt:round (2.777, 2)?> returns 2.78
lower	<?xdofx:lower (char)?>	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> .
upper	<?xdofx:upper(char)?>	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> .
length	<?xdofx:length(char)?>	The "length" function returns the length of <i>char</i> . LENGTH calculates length using characters as defined by the input character set.
greatest	<?xdofx:greatest (<i>expr</i> [, <i>expr</i>]...)?>	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.
least	<?xdofx:least (<i>expr</i> [, <i>expr</i>]...)?>	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.
next_element	<?xdoxslt:next_element(current-group(),',','element-name')?>	Method to get the next element in the current group. Returns the element that occurs after the element named. For example: <?xdoxslt:next_element(current-group(),',','employee')? > returns the element that occurs in the current group after "employee".
prev_element	<?xdoxslt:prev_element(current-group(),',','element-name')?>	Method to get the previous element in the current group. Returns the element that occurs before the element named. For example: <?xdoxslt:prev_element(current-group(),',','employee')? > returns the element that occurs in the current group before "employee".

Table B-1 (Cont.) Supported Functions

SQL Statement or XSL Expression	Usage	Description
set_array	<code><?xdoxslt:set_array(\$_XDOCTX, '<name of hash table>', n, '<value>')?></code>	Sets a value in a hash table. Syntax is <code><?xdoxslt:set_array(\$_XDOCTX, '<name of hash table>', n, '<value>')?></code> where <code>\$_XDOCTX</code> is required to set the context, <code><name of hash table></code> is the name that you supply for your table <code>n</code> is the index of the hash table <code><value></code> is the value to set in the hash table. For example: <code><?xdoxslt:set_array(\$_XDOCTX, 'Employee', 2, 'Jones')?></code> See <code>get_array</code> below.
get_array	<code><?xdoxslt:get_array(\$_XDOCTX, '<name of hash table>', n)?></code>	Returns the value at the specified index of the hash table. Syntax is <code><?xdoxslt:get_array(\$_XDOCTX, '<name of hash table>', n)?></code> where <code>\$_XDOCTX</code> is required to set the context, <code><name of hash table></code> is the name you supplied for your table in <code>set_array</code> <code>n</code> is the index value of the element you want returned. For example, used in conjunction with the <code>set_array</code> example above, <code><?xdoxslt:get_array(\$_XDOCTX, 'Employee', 2)?></code> returns Jones

Table B-2 shows supported combination functions.

Table B-2 Supported Combination Functions

SQL Statement	Usage
<code>(2+3/4-6*7)/8</code>	<code><?xdoxf:(2+3/4-6*7)/8?></code>
<code>lpad(substr('1234567890',5,3),10,'^')</code>	<code><?xdoxf:lpad(substr('1234567890',5,3),10,'^')?></code>
<code>decode('a','b','c','d','e','1') instr('321',1,1)</code>	<code><?xdoxf:decode('a','b','c','d','e','1') instr('321',1,1)?></code>

B.1.1 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

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

Table B-3 describes the function attributes.

Table B-3 Function Attributes

Attribute	Description	Valid Value
amount	The number to be transformed.	Any number

Table B-3 (Cont.) Function Attributes

Attribute	Description	Valid Value
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 governs the number of digits after the decimal point. The currency code does not generate a currency symbol in the output.	An integer, such as 2; or a currency code, such as 'USD'.
caseType	The case type of the output.	Valid values are: 'CASE_UPPER', 'CASE_LOWER', 'CASE_INIT_CAP'
decimalStyle	Output type of the decimal fraction area.	Valid values are: 'DECIMAL_STYLE_FRACTION1', 'DECIMAL_STYLE_FRACTION2', 'DECIMAL_STYLE_WORD'

Table B-4 displays the example function as entered in an RTF template and the returned output.

Table B-4 Function as Entered in an RTF Template

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

B.2 XSL Equivalents

Table B-5 lists the BI Publisher simplified syntax with the XSL equivalents.

Table B-5 BI Publisher Syntax with XSL Equivalents

Supported XSL Elements	Description	BI Publisher Syntax
<xsl:value-of select="name">	Placeholder syntax	<?name?>
<xsl:apply-templates select="name">	Applies a template rule to the current element's child nodes.	<?apply:name?>
<xsl:copy-of select="name">	Creates a copy of the current node.	<?copy-of:name?>
<xsl:call-template name="name">	Calls a named template to be inserted into/applied to the current template.	<?call:name?>
<xsl:sort select="name">	Sorts a group of data based on an element in the dataset.	<?sort:name?>

Table B-5 (Cont.) BI Publisher Syntax with XSL Equivalents

Supported XSL Elements	Description	BI Publisher Syntax
<xsl:for-each select="name">	Loops through the rows of data of a group, used to generate tabular output.	<?for-each:name?>
<xsl:choose>	Used in conjunction with when and otherwise to express multiple conditional tests.	<?choose?>
<xsl:when test="exp">	Used in conjunction with choose and otherwise to express multiple conditional tests	<?when:expression?>
<xsl:otherwise>	Used in conjunction with choose and when to express multiple conditional tests	<?otherwise?>
<xsl:if test="exp">	Used for conditional formatting.	<?if:expression?>
<xsl:template name="name">	Template declaration	<?template:name?>
<xsl:variable name="name">	Local or global variable declaration	<?variable:name?>
<xsl:import href="url">	Import the contents of one stylesheet into another	<?import:url?>
<xsl:include href="url">	Include one stylesheet in another	<?include:url?>
<xsl:stylesheet xmlns:x="url">	Define the root element of a stylesheet	<?namespace:x=url?>

B.3 Using FO Elements

You can use most FO elements in an RTF template inside the Microsoft Word form fields. The FO elements listed in [Table B-6](#) 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 D, "Supported XSL-FO Elements."](#)

Table B-6 FO Elements

FO Element	BI Publisher Syntax
<fo:page-number-citation ref-id="id">	<?fo:page-number-citation:id?>
<fo:page-number>	<?fo:page-number?>
<fo:ANY NAME WITHOUT ATTRIBUTE>	<?fo:ANY NAME WITHOUT ATTRIBUTE?>

Designing Accessible Reports

This appendix describes techniques for designing reports to increase accessibility of report output to users with disabilities. Accessibility support is for HTML output only.

This appendix includes the following sections:

- [Section C.1, "Avoiding Nested Tables or Separated Tables"](#)
- [Section C.2, "Defining a Document Title"](#)
- [Section C.3, "Defining Alternative Text for an Image"](#)
- [Section C.4, "Defining a Table Summary"](#)
- [Section C.5, "Defining a Table Column Header"](#)
- [Section C.6, "Defining a Table Row Header"](#)
- [Section C.7, "Sample Supported Tables"](#)

C.1 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.

[Figure C-1](#) shows a simple table.

Figure C-1 Simple Table

Header1	Header2	Header3

[Figure C-2](#) shows an example of a nested table: A table is inserted inside a table-cell.

Figure C-2 Nested Table

Column Header1	Column Header2	Column Header3				
	Cell with inserted table					
	<table border="1"> <tr> <td>Nested table</td> <td></td> </tr> <tr> <td></td> <td></td> </tr> </table>	Nested table				
Nested table						

C.1.1 Examples

The following are examples of table structures that BI Publisher does and does not support for accessibility.

C.1.1.1 Nested Tables

BI Publisher does not support accessibility when nested tables are used in a report, as shown in [Figure C-3](#). In this example, BI Publisher cannot tell to which column data "C1R1data" belongs.

Figure C-3 Unsupported Table Layout: Nested Tables in a Report

	Column header 1	Column Header2
Row Header1	C1R1data	C2R1data
	C1R2data	C2R2data

Remove the nested table as shown in [Figure C-4](#).

Figure C-4 Supported Table Layout: Nested Tables Removed

	Column header 1	Column Header2
Row Header1	C1R1data	C2R1data
	C1R2data	C2R2data

C.1.2 Table Headers Must Not Be Separated from the Table Body

The example shown in [Figure C-5](#) is not supported because the header, table body and accessibility fields exist in three different tables.

Figure C-5 Unsupported Table Layout: Header, Body, and Accessibility Fields in Separate Tables

	Column Header 1	Column Header 2	Column Header 3
UC1	Row Header 1	C1R1Data	C2R1Data
	Row Header 2	C1R2Data	C2R2Data
UC2	Subtotal	Total 1	Total 2

These three tables should be joined into one to support accessibility, as shown in [Figure C-6](#).

Figure C-6 Supported Table Layout: Three Tables Joined into One

ACC	Column Header 1	Column Header 2	Column Header 3
UC1	Row Header 1	C1R1Data	C2R1Data
	Row Header 2	C1R2Data	C2R2Data
UC2	Subtotal	Total 1	Total 2

C.2 Defining a Document Title

You can define a document title. The procedure differs slightly depending on the version of Microsoft Word.

To define or change a document title in Microsoft Word 2007:

1. Click the **Office** button, click **Prepare**, and then click **Properties**.

To define or change a document title in previous versions of Word:

1. On the **File** menu, click **Properties**, then click the **Summary** tab.

C.3 Defining Alternative Text for an Image

To define alternative text for an image in the 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.

C.4 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.

C.5 Defining a Table Column Header

You can define a table column header. The procedure differs slightly depending on the version of Microsoft Word.

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.

To define a table column header in previous 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**.

C.6 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 [Figure C-7](#), 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.

Figure C-7 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

C.7 Sample Supported Tables

[Figure C-8](#), [Figure C-9](#), [Figure C-10](#), and [Figure C-11](#) display sample tables for which accessibility is supported.

Figure C-8 Supported Tables: 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

Figure C–9 Supported Tables: 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

Figure C–10 Supported Tables: 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

Figure C–11 Supported Tables: 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

This appendix lists the XSL-FO elements that are supported by BI Publisher in RTF templates.

- [Section D.1, "Supported XSL-FO Elements"](#)

D.1 Supported XSL-FO Elements

[Table D-1](#) 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.

Table D-1 Supported XSL-FO Elements

Element	Supported Content Elements	Supported Attributes
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

Table D-1 (Cont.) Supported XSL-FO Elements

Element	Supported Content Elements	Supported Attributes
bidirectional-override	bidirectional-override external-graphic instream-foreign-object inline leader page-number page-number-citation basic-link	inline-properties
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

Table D-1 (Cont.) Supported XSL-FO Elements

Element	Supported Content Elements	Supported Attributes
conditional-page-master-reference	N/A	master-reference page-position <ul style="list-style-type: none"> ▪ first ▪ last ▪ rest ▪ any ▪ inherit odd-or-even <ul style="list-style-type: none"> ▪ odd ▪ even ▪ any ▪ inherit blank-or-not-blank <ul style="list-style-type: none"> ▪ blank ▪ not-blank ▪ any ▪ inherit
external-graphic	N/A	graphic-properties src
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

Table D-1 (Cont.) Supported XSL-FO Elements

Element	Supported Content Elements	Supported Attributes
list-item	list-item-label list-item-body	block-properties
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
page-sequence	static-content flow	inheritable-properties id master-reference initial-page-number <ul style="list-style-type: none"> ■ auto ■ <page-number> force-page-count <ul style="list-style-type: none"> ■ auto ■ end-on-even ■ end-on-odd ■ end-on-even-layout ■ end-on-odd-layout ■ no-force ■ inherit 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
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

Table D-1 (Cont.) Supported XSL-FO Elements

Element	Supported Content Elements	Supported Attributes
root	bookmark-tree layout-master-set page-sequence	inheritable-properties
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"> ■ 0 ■ 90 ■ 180 ■ 270 ■ -90 ■ -180 ■ -270 ■ 0deg ■ 90deg ■ 180deg ■ 270deg ■ -90deg ■ -180deg ■ -270deg ■ inherit writing-mode <ul style="list-style-type: none"> ■ lr-tb
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 D-1 (Cont.) Supported XSL-FO Elements

Element	Supported Content Elements	Supported Attributes
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
table-row	table-cell	inheritable-properties id
wrapper	inline page-number page-number-citation basic-link block block-container table wrapper	inheritable-properties id

D.1.1 Property Groups Table

[Table D-2](#) lists the supported properties belonging to the attribute groups defined in [Table D-1](#).

Table D-2 Supported Properties of the Attribute Groups

Property Group	Properties
area-properties	clip overflow (visible, hidden) reference-orientation <ul style="list-style-type: none"> ▪ 0 ▪ 90 ▪ 180 ▪ 270 ▪ -90 ▪ -180 ▪ -270 ▪ 0deg ▪ 90deg ▪ 180deg ▪ 270deg ▪ -90deg ▪ -180deg ▪ -270deg ▪ inherit writing-mode (lr-tb, rl-tb, lr, rl) baseline-shift (baseline, sub, super) vertical-align
block-properties	inheritable-properties id

Table D–2 (Cont.) Supported Properties of the Attribute Groups

Property Group	Properties
border-padding-back ground-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 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

Table D–2 (Cont.) Supported Properties of the Attribute Groups

Property Group	Properties
box-size-properties	height width
character-properties	font-properties text-decoration
empty-inline-properties	character-properties border-padding-background-properties id color
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
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"> ■ auto ■ absolute ■ fixed ■ inherit
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

Table D-2 (Cont.) Supported Properties of the Attribute Groups

Property Group	Properties
keeps-and-breaks-properties-block	keeps-and-breaks-properties-inline
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
margin-properties-CSS	margin margin-bottom margin-left margin-right margin-top

Table D–2 (Cont.) Supported Properties of the Attribute Groups

Property Group	Properties
margin-properties-in-line	margin-properties-block space-start space-start.optimum space-end space-end.optimum position <ul style="list-style-type: none"> ▪ static ▪ relative ▪ absolute ▪ fixed ▪ inherit top left
region-properties	border-padding-background-properties area-properties region-name
side-region-properties	region-properties extent

A

- accessibility
 - checking for using Template Builder, 5-30
- Adobe Flash
 - designing templates, 9-1
 - required settings for PDF output, 9-2
- alignment
 - RTF template, 4-36

B

- background support
 - RTF templates, 4-40
- barcode formatting
 - custom, 4-106
- barcodes
 - included in BI Publisher, 4-104
- bidirectional language alignment
 - RTF template, 4-36
- body tags
 - RTF template, 4-13
- bookmarks
 - generating PDF bookmarks from an RTF template, 4-50
 - inserting in RTF templates, 4-47
- brought forward/carried forward page totals, 4-68

C

- calculations in PDF layout, 7-11
- calendar profile option, 4-102
- calendar specification, 4-102
- cell highlighting
 - conditional in RTF templates, 4-63
- charts
 - building in RTF templates, 4-16
- check box placeholder
 - creating in PDF layout, 7-5
- check box support
 - RTF templates, 4-52
- choose statements, 4-59
- clip art support, 4-25
- columns
 - fixed width in tables, 4-38
- conditional columns

- rtf template, 4-60
- conditional formatting, 4-56
 - table rows, 4-62
- conditional formatting features, 4-56
- configuration properties
 - precedence of levels, 10-1
- connections
 - setting maximum for an interactive report, 3-17
- context command, 4-107
- cross-tab reports, 4-85

D

- date fields in RTF templates, 4-38
- dates
 - formatting in Excel templates, 6-12
- digital signature
 - adding signature field to a PDF layout, 7-17
 - setting properties, 10-6
- drawing support, 4-25
- drop-down form field support
 - RTF templates, 4-53
- dynamic data columns, 4-88
 - example, 4-89
- dynamic table of contents in RTF template, 4-50

E

- end on even page, 4-46
- etext data tables, 8-4
- etext template command rows, 8-5
- even page
 - force report to end on, 4-46
- Excel templates
 - formatting dates, 6-12

F

- file size
 - techniques for reducing, A-1
- filler block
 - etext templates, 8-21
- fixed-width columns
 - RTF templates, 4-38
- Flash templates
 - configuration properties, 9-10

- designing, 9-1
- uploading to the BI Publisher server, 9-10

FO

- supported elements, D-1

FO elements

- using in RTF templates, 4-112, B-11

fonts

- external, 4-103
- mapping, 10-16
- setting up, 4-103

footers

- RTF template, 4-13

for-each-group XSL 2.0 standard, 4-73

form field method

- inserting placeholders, 4-7

form field properties options in PDF template, 7-5

form fields in the PDF template, 7-4

formatting options in PDF templates, 7-5

G

groups

- basic RTF method, 4-11
- defining in PDF layout, 7-7
- defining in RTF template, 4-10
 - syntax, 4-10
- form field method, 4-11
- grouping scenarios in RTF templates, 4-10
- in RTF templates, 4-5

H

headers and footers

- different first page, 4-14
- different odd and even pages, 4-14
- inserting placeholders, 4-13
- multiple, 4-13
- resetting within one output file, 4-83
- RTF template, 4-13

hidden text

- support in RTF templates, 4-36

horizontal table break, 4-89

HTML output

- controlling table widths, 10-11

hyperlinks

- bookmarks, 4-47
- dynamic, 4-47
- inserting in RTF template, 4-47
- internal, 4-47
- static, 4-47

I

IF statements, 4-57

- in free-form text, 4-57

if-then-else statements, 4-58

images

- including in RTF template, 4-15

IN predicate

- If-Then-Else control structure
- etext templates, 8-24

L

last page

- support for special content, 4-44

layout editor

- features, 3-1

list component

- layout editor, 3-60

M

markup

- adding to the PDF template, 7-4
- adding to the RTF template, 4-6

max connections

- setting for an interactive report, 3-17

multicolumn page support, 4-39

multiple headers and footers

- RTF template, 4-13

N

Namespace support in RTF template, 4-112

native page breaks and page numbering

- Microsoft Word, 4-36

nulls

- how to test for in XML data, 4-73

number-to-word conversion, B-9

O

output formats

- limiting by report, 5-5, 5-6

overflow data in PDF layouts, 7-14

P

page breaks

- PDF layouts, 7-8
- RTF template, 4-36, 4-41

page breaks and page numbering

- native support, 4-36

page number

- setting initial
- RTF templates, 4-43

page numbers

- PDF layouts, 7-8
- restarting within one output file, 4-83
- RTF template, 4-36

page totals

- brought forward/carried forward, 4-68
- inserting in RTF template, 4-65

PDF layouts

- completed example, 7-12
- creating from downloaded file, 7-14
- defining groups, 7-7
- definition of, 7-1
- overflow data, 7-14
- page breaks, 7-8
- page numbering, 7-8
- placeholders

- check box, 7-5
 - radio button group, 7-6
- placement of repeating fields at runtime, 7-12
- runtime behaviors, 7-12
- PDF output
 - handling large files, A-1
 - invalid, 4-85
- PDF output properties, 10-2
- PDF security properties, 10-3
- PDF template
 - adding markup, 7-4
 - placeholders
 - text, 7-4
 - types of, 7-4
 - saving as Adobe Acrobat 5.0 compatible, 7-2
 - sources for document templates, 7-2
 - supported modes, 7-2
 - when to use, 7-1
- pivot table
 - designing in RTF templates, 4-85
- placeholders
 - basic RTF method, 4-7
 - form field RTF method, 4-7
 - in PDF templates, 7-4
 - in RTF templates, 4-5
 - defining, 4-5, 4-7
 - inserting in the header and footer of RTF templates, 4-13
 - PDF layouts
 - check box, 7-5
 - radio button group, 7-6
 - PDF templates
 - text, 7-4
 - types of, 7-4
- PowerPoint output
 - design considerations, 4-114
- predefined fonts, 10-18
- properties
 - setting at template level, 4-82

R

- radio button group
 - creating in PDF layouts, 7-6
- regrouping, 4-73
- Rich Text Format (RTF)
 - definition, 4-2
- row breaking
 - preventing in RTF templates, 4-37
- row formatting
 - conditional, 4-62
- RTF placeholders
 - syntax, 4-7
- RTF template
 - adding markup, 4-6
 - applying design elements, 4-3
 - definition, 4-2
 - designing, 4-3
 - groups, 4-5
 - including images, 4-15

- native formatting features, 4-36
- placeholders, 4-5
- prerequisites, 4-2
- sample template design, 4-4
- supported modes, 4-3
 - basic method, 4-3
 - form field method, 4-3
 - using XSL or XSL
 - FO, 4-4

- RTF template design
 - headers and footers, 4-13
- RTF template placeholders, 4-7
- running totals
 - RTF templates, 4-71

S

- sample RTF template
 - completed markup, 4-9
- section context command, 4-83
- security
 - PDF properties, 10-3
- setting the initial page number
 - RTF templates, 4-43
- shape support, 4-25
- sorting
 - RTF template, 4-72
- SQL functions
 - BI Publisher syntax for, B-1
 - using in RTF templates, 4-109
- SQL functions extended for BI Publisher, B-1
- style templates, 11-1
- syntax
 - RTF template placeholder, 4-7
- sysdate
 - displaying and formatting in RTF templates, 4-99

T

- table features
 - fixed-width columns, 4-38
 - preventing rows breaking across pages
 - RTF template, 4-37
 - repeating table headers
 - RTF template, 4-37
 - RTF template, 4-37
 - text truncation, 4-38
- table of contents support
 - RTF template, 4-50
 - dynamic TOC, 4-50
- tables
 - controlling table widths in HTML output, 10-11
 - horizontal table break, 4-89
- Template Builder, 5-3
 - editing field properties, 5-27
 - prerequisites, 5-2
 - preview options, 5-27
- text placeholder
 - creating in PDF template, 7-4
- text truncation in tables, 4-38

totals

- brought forward/carried forward, 4-68
- inserting page totals in RTF template, 4-65
- running
 - RTF templates, 4-71

U

updateable variables

- RTF templates, 4-79

V

variables

- RTF templates, 4-79

W

watermarks

- RTF templates, 4-40

X

XML data file

- example, 4-4

XML file

- how to read, 4-5

XSL

FO elements

- using in RTF templates, 4-109

XSL elements

- apply a template rule, 4-112
- BI Publisher syntax for, B-10
- call template, 4-113
- copy the current node, 4-113
- define the root element of the style sheet, 4-113
- import a style sheet, 4-113
- template declaration, 4-113
- using in RTF templates, 4-112
- variable declaration, 4-113