

# ActuateOne™

One Design  
One Server  
One User Experience

**Designing Spreadsheets**  
using BIRT Spreadsheet Designer

Information in this document is subject to change without notice. Examples provided are fictitious. No part of this document may be reproduced or transmitted in any form, or by any means, electronic or mechanical, for any purpose, in whole or in part, without the express written permission of Actuate Corporation.

© 1995 - 2012 by Actuate Corporation. All rights reserved. Printed in the United States of America.

Contains information proprietary to:

Actuate Corporation, 951 Mariners Island Boulevard, San Mateo, CA 94404

[www.actuate.com](http://www.actuate.com)

[www.birt-exchange.com](http://www.birt-exchange.com)

The software described in this manual is provided by Actuate Corporation under an Actuate License agreement. The software may be used only in accordance with the terms of the agreement. Actuate software products are protected by U.S. and International patents and patents pending. For a current list of patents, please see <http://www.actuate.com/patents>.

Actuate Corporation trademarks and registered trademarks include:

Actuate, ActuateOne, the Actuate logo, Archived Data Analytics, BIRT, BIRT 360, BIRT Data Analyzer, BIRT Performance Analytics, Collaborative Reporting Architecture, e.Analysis, e.Report, e.Reporting, e.Spreadsheet, Encyclopedia, Interactive Viewing, OnPerformance, Performancesoft, Performancesoft Track, Performancesoft Views, Report Encyclopedia, Reportlet, The people behind BIRT, X2BIRT, and XML reports.

Actuate products may contain third-party products or technologies. Third-party trademarks or registered trademarks of their respective owners, companies, or organizations include:

Mark Adler and Jean-loup Gailly ([www.zlib.net](http://www.zlib.net)): zlib. Adobe Systems Incorporated: Flash Player. Apache Software Foundation ([www.apache.org](http://www.apache.org)): Axis, Axis2, Batik, Batik SVG library, Commons Command Line Interface (CLI), Commons Codec, Derby, Hive driver for Hadoop, Shindig, Struts, Tomcat, Xalan, Xerces, Xerces2 Java Parser, and Xerces-C++ XML Parser. Castor ([www.castor.org](http://www.castor.org)), ExoLab Project ([www.exolab.org](http://www.exolab.org)), and Intalio, Inc. ([www.intalio.org](http://www.intalio.org)): Castor. Codejock Software: Xtreme Toolkit Pro. Eclipse Foundation, Inc. ([www.eclipse.org](http://www.eclipse.org)): Babel, Data Tools Platform (DTP) ODA, Eclipse SDK, Graphics Editor Framework (GEF), Eclipse Modeling Framework (EMF), and Eclipse Web Tools Platform (WTP), licensed under the Eclipse Public License (EPL). Bits Per Second, Ltd. and Graphics Server Technologies, L.P.: Graphics Server. Gargoyl Software Inc.: HtmlUnit, licensed under Apache License Version 2.0. GNU Project: GNU Regular Expression, licensed under the GNU Lesser General Public License (LGPLv3). HighSlide: HighCharts. IDAutomation.com, Inc.: IDAutomation. Jason Hsueh and Kenton Varda ([code.google.com](http://code.google.com)): Protocole Buffer. IDR solutions Ltd.: JBIG2, licensed under the BSD license. ImageMagick Studio LLC.: ImageMagick. InfoSoft Global (P) Ltd.: FusionCharts, FusionMaps, FusionWidgets, PowerCharts. Matt Inger ([sourceforge.net](http://sourceforge.net)): Ant-Contrib, licensed under Apache License Version 2.0. Matt Ingenthron, Eric D. Lambert, and Dustin Sallings ([code.google.com](http://code.google.com)): Spymemcached, licensed under the MIT OSI License. International Components for Unicode (ICU): ICU library. jQuery: jQuery, licensed under the MIT License. Yuri Kanivets ([code.google.com](http://code.google.com)): Android Wheel gadget, licensed under the Apache Public License (APL). KL Group, Inc.: XRT Graph, licensed under XRT for Motif Binary License Agreement. LEAD Technologies, Inc.: LEADTOOLS. Bruno Lowagie and Paulo Soares: iText, licensed under the Mozilla Public License (MPL). Microsoft Corporation (Microsoft Developer Network): CompoundDocument Library. Mozilla: Mozilla XML Parser, licensed under the Mozilla Public License (MPL). MySQL Americas, Inc.: MySQL Connector. Netscape Communications Corporation, Inc.: Rhino, licensed under the Netscape Public License (NPL). OOPS Consultancy: XMLTask, licensed under the Apache License, Version 2.0. Oracle Corporation: Berkeley DB, Java Advanced Imaging, JAXB, JDK, Jstl. PostgreSQL Global Development Group: pgAdmin, PostgreSQL, PostgreSQL JDBC driver. Progress Software Corporation: DataDirect Connect XE for JDBC Salesforce, DataDirect JDBC, DataDirect ODBC. Rogue Wave Software, Inc.: Rogue Wave Library SourcePro Core, tools.h++. Sam Stephenson ([prototype.conio.net](http://prototype.conio.net)): prototype.js, licensed under the MIT license. Sencha Inc.: Ext JS. ThimbleWare, Inc.: JMemcached, licensed under the Apache Public License (APL). World Wide Web Consortium (W3C)(MIT, ERCIM, Keio): Flute, JTIty, Simple API for CSS. XFree86 Project, Inc.: ([www.xfree86.org](http://www.xfree86.org)): xvfb. ZXing authors ([code.google.com](http://code.google.com)): ZXing, licensed under the Apache Public License (APL).

All other brand or product names are trademarks or registered trademarks of their respective owners, companies, or organizations.

Document No. 120201-2-961002 October 5, 2012

# Contents

<b>About <i>Designing Spreadsheet Reports</i> .....</b>	<b><i>xix</i></b>
---	-------------------

## Part 1

## **Getting Started**

### Chapter 1

<b>Building your first spreadsheet reports .....</b>	<b>3</b>
--	----------

About Actuate BIRT Spreadsheet Designer .....	4
About the report design process .....	4
About the BIRT Spreadsheet Designer user interface .....	5
Tutorial 1: Creating a report using the listing wizard .....	6
Task 1: Connect to a data source .....	6
Task 2: Create a data set and a query .....	8
Task 3: Use the listing wizard to create a data range .....	12
Task 4: Preview your report .....	15
Tutorial 2: Creating a report from scratch .....	16
Task 1: Connect to a data source .....	16
Task 2: Construct a data set and a query .....	17
Task 3: Construct a data range .....	19
Task 4: Add data to the report .....	25
Task 5: Format the report .....	30
Task 6: Preview the report .....	34
Next steps .....	35

### Chapter 2

<b>Understanding report design .....</b>	<b>37</b>
--	-----------

Planning a report .....	38
Content considerations .....	38
Layout strategies .....	39
Security considerations .....	40
Distribution strategies .....	41
Designing a report .....	41
Connecting to the data source .....	41
Specifying report data .....	41
Structuring report layout .....	42
Viewing a report .....	44
Using BIRT Spreadsheet Designer Run options .....	44
Using View with Excel options .....	45
Using the View as PDF option .....	45

Distributing the report .....	45
Understanding BIRT Spreadsheet Designer files .....	46

## Part 2

## Accessing data

### Chapter 3

<b>Accessing a data source .....</b>	<b>49</b>
About BIRT Spreadsheet Designer data sources .....	50
Accessing multiple data sources .....	51
About the BIRT Spreadsheet Designer example databases .....	51
Connecting to a data source .....	51
Creating a data source connection .....	52
Modifying a data source connection .....	53
About accessing additional types of data sources .....	53
Using a custom data source type in a report design .....	54
Developing a custom data driver .....	54
About data sets .....	56
Creating a data set .....	56
Previewing data set results .....	57
Adding a computed field to a data set .....	58
Combining multiple data sets .....	61
About data set objects .....	61
Creating a joined data set .....	61
Modifying a data set .....	64
Defining a default data set .....	64
Running data set queries in priority order .....	65
Forcing a data set to refresh at view time .....	66

### Chapter 4

<b>Connecting to a database or JDBC data source .....</b>	<b>69</b>
Accessing a database .....	70
Connecting to a database or other JDBC data source .....	70
Creating a JDBC data source connection .....	71
Modifying a JDBC data source connection .....	73
Adding a custom JDBC driver .....	74
Accessing data using a SQL query .....	76
Creating a SQL query using BIRT Spreadsheet Designer .....	77
About table joins .....	78
Selecting data elements using the query editor .....	80
Creating a query using Design View .....	81
Creating a query using SQL View .....	83
Suppressing the catalog, schema, or table name in a SQL query .....	84

Creating a computed field using the query editor .....	84
Summarizing row data in the query editor .....	85
Filtering query results .....	87
Using a condition to filter a query .....	87
Using a parameter to filter a query .....	89
About creating a parameter in a query .....	89
Using a static parameter .....	89
Using an ad hoc parameter .....	90
Modifying a query .....	91
Adding and removing joins .....	91
Modifying join properties .....	92
Modifying columns using the query editor .....	93
Adding a table or column .....	93
Moving a column .....	93
Sorting data rows .....	94
Hiding a column .....	94
Deleting a field from the query .....	94
About refreshing metadata .....	95
Refreshing source metadata .....	95
Refreshing result set metadata .....	95
Running multiple data set queries simultaneously .....	96
Accessing a stored procedure .....	96
Creating a data set that uses a stored procedure call .....	96

## Chapter 5

<b>Accessing an Actuate information object .....</b>	<b>99</b>
About information objects .....	100
Working with an information object .....	100
Preparing to access information object data .....	100
Connecting to an Actuate information object .....	101
Examining Information Object Query Builder .....	103
Opening Information Object Query Builder .....	104
Choosing an information object query editor .....	105
Using the expression builder .....	106
Creating an information object query in the Basic Design interface .....	107
Creating a customized graphical information object query .....	111
Selecting one or more information objects .....	113
Hiding column categories .....	114
Defining output columns .....	115
Setting column properties .....	116
Specifying a join .....	117
About joins .....	117
Optimizing joins .....	119

Filtering data .....	121
Creating a filter condition .....	122
Creating multiple filter conditions .....	130
Prompting for filter values .....	133
Setting dynamic filter prompt properties .....	135
Grouping data .....	138
Creating a GROUP BY clause .....	140
Removing a column from the GROUP BY clause .....	142
Filtering on an aggregate column .....	144
Defining parameters .....	145
Specifying a parameter's prompt properties .....	147
Setting parameter properties .....	149
Setting source parameters .....	151
Synchronizing source parameters .....	152
Creating a textual information object query .....	153
Displaying output columns .....	155
Displaying parameters .....	155
Displaying information object query output .....	156

## Chapter 6

### **Accessing data in a text file ..... 157**

Accessing data from a text file .....	158
Connecting to a text file .....	159
Creating a data set for a delimited text file .....	160
Creating a data set for a fixed-width text file .....	162

## Part 3

### **Developing a spreadsheet report**

## Chapter 7

### **Laying out a report ..... 167**

About layout components .....	168
Comparing layout options .....	169
About the listing wizard .....	169
Creating a data range manually .....	170
About a pivot range .....	171
Constructing a data range .....	173
Understanding row and column sections .....	173
About data range groups .....	174
Creating a sheet-level group .....	174
About sorting and grouping .....	176
Creating a grouped section from a data field .....	177

Modifying data range components .....	178
Editing a section .....	179
Deleting a section .....	179
Renaming a section .....	179
Changing a group key .....	179
Inserting a row or column .....	180
Deleting a cell, row, or column .....	180
Hiding a row or column .....	181
Hiding empty sections in a report .....	181
Setting row height and column width .....	182
Using the size report script function .....	182
Using multiple data ranges in a report design .....	184
Using multiple data ranges on a single worksheet .....	184
Using multiple data ranges to display data on different worksheets .....	186

## Chapter 8

### **Formatting a report ..... 189**

About BIRT Spreadsheet Designer formatting options .....	190
Formatting numbers .....	190
Accessing number formatting options .....	191
Creating a custom currency format .....	191
Merging cells .....	192
About the merge report script function .....	192
Merging a range of adjacent cells .....	192
Merging cells with repeating values .....	193
Merging cells in a heading .....	194
Formatting styles .....	195
Applying a theme .....	196
Using conditional formatting .....	197
Creating a conditional format .....	198
Removing a conditional format .....	200
Creating a conditional format with an expression .....	201
Creating a report outline .....	201
About outline components .....	202
Creating a report outline .....	203
Removing an outline or an outline level .....	206
Setting outline display options .....	207
Working with workbooks and worksheets .....	208

## Chapter 9

### **Adding content to a data range ..... 211**

About adding content .....	212
Adding data fields to a data range .....	212

About dragging and dropping data fields .....	212
Working with report script functions .....	213
Using the report script function builder .....	214
Adding detail to a data range .....	215
Creating a detail report script function .....	216
Setting up data filters .....	219
Defining a data subset .....	219
Changing condition order .....	224
Using include and exclude to filter data .....	224
Selecting data from multiple data sets .....	225
Creating totals and subtotals .....	226
Using Excel SUM and SUBTOTAL functions .....	226
Using the Excel SUM function .....	227
Using the Excel SUBTOTAL function .....	229
Using Excel functions in different locales .....	230
Summarizing data in hierarchies .....	231
About an even data hierarchy .....	231
About an uneven data hierarchy .....	232
Retrieving data from an uneven hierarchy .....	232
Using defined names and other reference tools .....	235
About defined names .....	236
Usage guidelines for defined names .....	236
Creating a defined name .....	236
Using a conditionally defined name .....	238
About cell references .....	238
Using the cells report script function .....	239
About section references .....	239
About worksheet and workbook references .....	239
Working with data that contains null values .....	240
Handling null values with function expressions .....	241
Handling null values in Excel formulas .....	243
Changing the appearance of null data values .....	243
Working with static content .....	244
Using the setEntry function to locate static content .....	244
Displaying static and dynamic text in the same cell .....	245
Using a report macro to simplify a custom report script function .....	245
Validating data entry .....	247
 Chapter 10	
<b>Using a chart .....</b>	<b>249</b>
About charts .....	250
Understanding chart elements .....	250
Understanding chart types .....	251



About area charts .....	252
About bar and column charts .....	252
About bubble charts .....	253
About combination charts .....	254
About difference charts .....	254
About doughnut charts .....	255
About line charts .....	256
About pie charts .....	257
About scatter charts .....	257
About step charts .....	258
About stock charts .....	259
About study charts .....	259
Tutorial 3: Creating a chart in a spreadsheet report .....	260
Task 1: Create a report design .....	260
Task 2: Create a data range in the report design .....	262
Task 3: Create a chart object using chart wizard .....	262
Task 4: Refine the chart appearance .....	265
Task 5: Add descriptive information to the chart .....	265
Task 6: Relocate and resize the chart .....	266
Placing a chart in a report design .....	266
Comparing and contrasting data using chart elements .....	266
Displaying one chart type many times .....	267
Showing many data series in one chart .....	268
Separating a chart from a data range .....	270
Using more than one worksheet .....	270
Using more than one chart type .....	271
 Chapter 11	
<b>Charting a data range .....</b>	<b>273</b>
Designing a chart .....	274
Understanding the axes of a chart .....	274
About the axes .....	274
Defining the axes .....	275
Plotting different chart types .....	275
Creating a chart object .....	277
About Chart Wizard .....	277
Selecting options for a new chart .....	277
Linking a chart to a data range .....	278
Working with Data Mapping .....	278
Mapping a chart element to a data range .....	278
Creating a new section for data mapping .....	280
Working with Source Data .....	280
Updating the data range for a chart .....	281

Selecting a different data range .....	281
Selecting series layout .....	282
Refreshing a chart .....	282
Working with Series .....	282
Working with Titles .....	284
Refining a chart .....	284
Working with an element in a chart area .....	284
Selecting a chart area .....	284
Selecting a chart element .....	285
Showing tips .....	285
Formatting the chart area .....	285
Changing a chart type .....	285
Changing a color or pattern in the chart area .....	286
Changing the line that borders a chart area .....	286
Changing a font characteristic in the chart area .....	287
Working with hidden data .....	287
Displaying a non-numeric data value in a chart .....	288
Resetting a chart layout .....	288
Formatting an element .....	288
Showing and hiding an element .....	288
Changing the color or pattern in an element .....	289
Changing the line that borders an element .....	289
Changing a font characteristic for an element .....	290
Aligning text in an element .....	290
Changing the number format in an element .....	291
Working with a data series .....	291
Formatting a data series .....	291
Changing shape, color and size of points in a data series .....	291
Assigning different colors to each data point in a single data series .....	292
Setting different colors for positive and negative values in a data series .....	292
Plotting a data series on a second y-axis .....	292
Displaying y-error bars in a data series .....	293
Hiding data labels for a data series .....	293
Changing display order of data series .....	294
Stacking series .....	294
Plotting data points as percentages .....	295
Formatting data series in specific chart types .....	295
Formatting a bar or column chart .....	295
Setting space between groups and shapes .....	296
Changing the shape of a bar or column .....	297
Showing series lines .....	297
Formatting a bubble chart .....	298
Setting bubble size .....	298

Rearranging bubbles that overlap	299
Adjusting axis scale on a bubble chart	299
Formatting a combination chart	299
Formatting a doughnut chart	299
Separating sectors of a doughnut chart	299
Sizing the center of a doughnut chart	300
Changing the start angle of doughnut sectors	301
Formatting a pie chart	301
Exploding sectors of a pie chart	301
Changing the start angle of pie sectors	302
Displaying data label lines	302
Formatting a data series in a pie of pie chart	302
Adding reference lines to a chart	304
Adding drop lines	305
Adding high-low lines	305
Adding up or down bars	305
Changing the width of up or down bars	305
Working with axes	306
Showing and hiding lines and labels	306
Showing or hiding lines on an axis	306
Showing and hiding descriptions on an axis	307
Placing and scaling an axis	307
Describing elements in a data series	311
Displaying data labels	311
Working with a trendline	312
About linear trendlines	312
About logarithmic trendlines	312
About polynomial trendlines	312
About power trendlines	313
About exponential trendlines	313
About moving average trendlines	313
Adding and modifying a trendline	314
<b>Chapter 12</b>	
<b>Designing a customizable report</b>	<b>315</b>
About customizable reports	316
Understanding parameter types	317
Using a run-time parameter	317
Using a view-time parameter	317
Using run-time and view-time parameters in a report	318
Using a system parameter	318
Creating a parameter	319
Defining required attributes for a parameter	320

Defining a parameter name .....	320
Defining a parameter data type .....	320
Listing parameters on Requester Page .....	321
Using a text box .....	321
Using a list .....	322
Limiting how many parameter values Requester Page lists .....	325
Setting options for Requester Page users .....	325
Allowing users to provide null or empty values .....	326
Hiding characters that a user types .....	326
Creating a tooltip .....	326
Creating a defined name for a parameter .....	326
Hiding a parameter on Requester Page .....	326
Grouping parameters .....	327
Creating a parameter group .....	328
Creating a cascading parameter group .....	329
Providing a value for a parameter .....	330
Working with system parameters .....	331
Providing a report parameter value that uses locale-specific formatting .....	331

## Chapter 13

### **Working with a pivot range ..... 333**

About pivot ranges .....	334
Examining a sample pivot range .....	334
Recognizing the parts of a pivot range .....	335
Creating a pivot range .....	336
About sources of pivot range data .....	336
Working with Pivot Field List .....	339
Adding data fields to a pivot range .....	340
Improving performance when adding fields to a pivot range .....	342
Moving a field in a pivot range .....	342
Changing the location of a pivot range .....	342
Working with pivot range appearance .....	343
Renaming a pivot range item .....	343
Formatting a pivot range field .....	343
Using tabular or outline layout .....	344
Formatting a pivot range .....	345
Preserving pivot range formatting .....	346
Working with pivot range layout .....	346
Arranging page field layout .....	346
Adding and deleting the page area in a pivot range .....	347
Arranging data fields .....	348
Arranging items in a pivot range field .....	349
Centering labels for row and column fields .....	349

Hiding a pivot range field .....	350
Hiding a pivot range item .....	350
Hiding an item in the pivot range page area .....	350
Hiding inner field details .....	351
Consolidating a pivot range view .....	352
Expanding a pivot range view .....	352
Showing detail for a data field .....	354
Working with pivot range data .....	354
Updating pivot range data .....	354
Changing the source for pivot range data .....	354
Working with empty fields and error values .....	355
Working with pivot range calculations .....	355
About creating a pivot range calculation .....	355
Working with a calculated field or a calculated item .....	356
Changing the expression a calculation uses .....	358
Using variance data .....	359
Working with totals .....	360
Working with a subtotal .....	360
Working with a grand total .....	361
Working with a block total .....	361
Working with pivot range formulas .....	362
Grouping pivot range data .....	362
Grouping date, time, and number fields .....	362
Grouping selected pivot range items .....	364
Sorting pivot range data .....	365
Filtering pivot range data .....	366
Referring to pivot range data .....	367
Referring to pivot range data from inside the pivot range .....	367
Referring to pivot range data from outside the pivot range .....	368
Printing a pivot range .....	370
 Chapter 14	
<b>Designing a secure report .....</b>	<b>371</b>
About report security .....	372
Restricting user access .....	372
Allowing selected user actions .....	373
About password requirements .....	374
Setting a password for a workbook .....	374
Setting a password for workbook structure .....	375
Setting a password for a worksheet .....	375
Unlocking selected cells in a worksheet .....	376
Locking a graphical object .....	376
Hiding a worksheet .....	377

Hiding formula results .....	377
Hiding formula text .....	378
Using file encryption .....	378
About SmartSheet security .....	379
About security IDs .....	380
Using security IDs in a data range .....	380
About creating grant expressions .....	381
Adding a grant expression to a report element .....	383
Testing a grant expression .....	383
Using SmartSheet security to deliver secure reports .....	384
Using SmartSheet security on a data set .....	384
Using SmartSheet security on a worksheet .....	385

## Chapter 15

### **Using a hyperlink ..... 389**

About hyperlinks .....	390
Understanding hyperlink targets and target types .....	390
Working with hyperlinks .....	391
Working with a hyperlinked cell .....	392
Working with hyperlink text .....	392
Working with a fixed hyperlink .....	392
Working with a variable hyperlink .....	394
Using hyperlinks in a BIRT iServer environment .....	395
Linking to a report in an Encyclopedia volume .....	395
Creating a hyperlink that varies at run time .....	396

## Chapter 16

### **Using a spreadsheet report on a production system ..... 397**

Publishing a spreadsheet report to a BIRT iServer System .....	398
Preparing to publish a report .....	398
Creating a BIRT iServer profile .....	399
Publishing a report .....	400
Managing file versions .....	401
Copying file properties .....	401
Naming a report .....	401
Configuring a data source for a production environment .....	403
Using a connection configuration file .....	404
Setting up a connection configuration file .....	404
Examining a sample connection configuration file .....	409

## Chapter 17

### **Personalizing BIRT Spreadsheet Designer ..... 411**

Setting BIRT Spreadsheet Designer preferences .....	412
---	-----

Changing general designer preferences .....	412
Viewing a report in BIRT Spreadsheet Designer .....	413
Moving a toolbar .....	414
Changing language preferences .....	415
Changing compatibility preferences .....	415
Selecting a target file format .....	415
Selecting the default palette .....	416
Selecting encoding options .....	416
Changing data preferences .....	417
Changing error logging preferences .....	418
Setting the location for online help files .....	419
Showing tips on charts .....	419
Setting workbook properties .....	420
Controlling workbook element display .....	420
Using type markers .....	421
Modifying the color palette .....	421
About calculation preferences .....	424
Setting calculation preferences .....	424
Recalculating worksheet formulas .....	425
Setting sheet properties .....	425
Selecting evaluation rules .....	425
Controlling worksheet appearance .....	426
Improving spreadsheet report performance .....	428
Modifying the BIRT Spreadsheet Designer configuration file .....	428
Improving BIRT Spreadsheet Designer performance .....	429
Allocating memory .....	429
Optimizing a query .....	429
Using report script functions .....	429

## Part 4

# Extending BIRT Spreadsheet Designer

## Chapter 18

### **Report script reference guide ..... 433**

About BIRT Spreadsheet Designer report script .....	434
Placing report script functions in a data range .....	435
Using mathematical operators with report script functions .....	436
Using a list .....	437
Using operators with lists .....	437
Sorting a list .....	437
Using wildcards with report script functions .....	438
Using multiple report script functions in a cell or section .....	438

Accessing a non-default data set .....	438
Adding a comment .....	439
BIRT Spreadsheet Designer report script function overview .....	439
BIRT Spreadsheet Designer report script function reference .....	442
average .....	442
cells .....	443
chart .....	443
col .....	444
color .....	444
count .....	444
date .....	445
day .....	446
delete .....	446
detail .....	446
distinct .....	447
error .....	447
eval .....	448
exclude .....	448
extract .....	448
first .....	449
fontcolor .....	450
format .....	450
formula .....	451
getEntry .....	451
group .....	452
groupNum .....	453
groupVal .....	453
hour .....	454
hyperlink .....	454
if .....	455
include .....	455
isnull .....	455
last .....	456
len .....	456
lock .....	457
lower .....	457
max .....	457
merge .....	458
min .....	458
minute .....	458
mod .....	459
month .....	459
name .....	459



now	460
nth	460
pageBreak	461
product	461
quarter	462
rollup	462
round	463
row	463
rownum	463
second	464
security	464
sectioncount	464
select	465
setchart	465
setentry	466
sheet	467
size	467
stdev	468
stdevp	469
substitute	469
sum	470
time	470
todate	470
tonumber	471
toText	471
trim	472
trunc	472
upper	473
values	473
var	474
varp	474
write	474
year	474

Chapter 19	
<b>Working with callback classes</b>	<b>477</b>
About callback classes	478
Writing a callback class	479
Using the BookModel object	479
Understanding a simple example	480
Using import statements	480
Using try-catch blocks	481
Using getLock() and releaseLock()	481

Compiling the Java file .....	481
Deploying a callback class .....	482
Deploying a callback class to BIRT Spreadsheet Designer .....	482
Deploying a callback class to Actuate BIRT iServer System .....	483
Testing and debugging a callback class .....	483
Logging messages from a callback class .....	483
Using a Logger object in BIRT Spreadsheet Designer .....	484
Using a Logger object on BIRT iServer System .....	484
Modifying and retesting a callback class .....	485
Testing in a Java development environment .....	485
Testing a callback on Actuate BIRT iServer System .....	485
Configuring BIRT iServer System for remote Java debugging .....	486
Testing the BIRT iServer System Java processes .....	487
Writing a multiple-class callback .....	489
Examining a multiple-class callback example .....	489
Specifying a package for additional classes .....	490
Deploying additional classes .....	491
Locating the example source files .....	492

## Chapter 20

### **Examining samples of callback classes ..... 495**

About the callback examples .....	496
Using the BIRT Spreadsheet API .....	496
Using the BIRT Spreadsheet API Javadoc .....	497
Understanding the examples .....	498
Using and testing the examples .....	498
Understanding the SetPrintAreas example .....	498
Understanding the ModifyQuery example .....	502
Understanding the DynamicImage example .....	505
Understanding the CellLocking example .....	507
Understanding the CellFormatting example .....	509
Understanding the WordFormatting example .....	513
Understanding the CustomNumberFormatting example .....	514
Understanding the FileWriting example .....	518
Understanding the DrawingObjects example .....	520
Understanding the DrawingCharts example .....	522
Understanding the CreatePivotRange example .....	524
Understanding the CreatingADataRange example .....	530

## Chapter 21

### **Working with VBA ..... 535**

Including VBA functionality in a spreadsheet report .....	536
Creating and using a template file .....	536

Creating and naming a template file .....	536
Testing VBA code with a spreadsheet report .....	537
Publishing a report with a VBA template .....	537
Testing VBA code with Excel 2007 .....	537
Using a form control object .....	538
Reviewing a VBA template example .....	538
Using workbook and worksheet code names .....	541

## Part 5

# Working in multiple locales using BIRT Spreadsheet technology

## Chapter 22

### **Using BIRT Spreadsheet Designer with multiple locales ..... 545**

Deploying BIRT Spreadsheet Designer in a translated language .....	546
About BIRT Spreadsheet Designer translated languages .....	546
Using code to override locale settings .....	547
Deploying BIRT Spreadsheet Designer in a supported language .....	548
Using regional settings with BIRT Spreadsheet Designer .....	549
Using international symbols in XML .....	549
Viewing localized currency .....	550
Viewing localized dates .....	550
Using non-English versions of BIRT Spreadsheet Designer .....	550
Understanding BIRT Spreadsheet Designer limitations .....	550

## Appendix A

### **BIRT Spreadsheet Designer internationalization information ..... 553**

About internationalization information .....	554
Format symbols .....	554
Values and errors .....	555
Function names .....	555
Using the Japanese translation .....	570

## Appendix B

### **Comparing Excel and BIRT Spreadsheet Designer ..... 573**

About Microsoft Excel compatibility .....	574
Working with graphical object differences .....	574
Working with worksheet differences .....	575
Working with formula differences .....	575
Understanding formula character limits .....	575
Working with differences in arithmetic functions .....	576
Working with array formulas .....	578
Allowing an incorrect number of arguments .....	578
Working with pivot range differences .....	578

Working with VBA differences .....579

**Glossary ..... 581**

**Index ..... 641**

# About Designing Spreadsheet Reports

---

*Designing Spreadsheet Reports* describes how to use the graphical user interface of BIRT Spreadsheet Designer to create spreadsheet reports.

*Designing Spreadsheet Reports* includes the following chapters:

- *About Designing Spreadsheet Reports.* This chapter provides an overview of this guide.
- *Part 1. Getting Started.* This part provides an overview of the report design process.
- *Chapter 1. Building your first spreadsheet reports.* This chapter introduces the report design process and provides two tutorials that show two different techniques for designing spreadsheet reports.
- *Chapter 2. Understanding report design.* This chapter presents planning questions to consider before beginning a new report design, and provides more information about the major tasks in the report design process.
- *Part 2. Accessing data.* This part provides information and procedures for accessing data from a database or other data source.
- *Chapter 3. Accessing a data source.* This chapter describes the overall process of accessing data, including accessing data from external data sources or from a data set object created by a previous query.
- *Chapter 4. Connecting to a database or JDBC data source.* This chapter provides information about accessing and retrieving data from a database or JDBC data source. This chapter describes how to use the graphical and textual query editors to access tables in the data source and how to access data from a stored procedure.
- *Chapter 5. Accessing an Actuate information object.* This chapter provides information about accessing and retrieving data from information objects created in Actuate Information Object Designer.

- *Chapter 6. Accessing data in a text file.* This chapter provides information about accessing and retrieving data from a text file.
- *Part 3. Developing a spreadsheet report.* This part describes how to manipulate and format report data, how to add features such as conditional formatting, charts and hyperlinks, and how to deploy a completed report.
- *Chapter 7. Laying out a report.* This chapter discusses basic concepts for arranging the information in a report and then describes how to set up report sections, including group sections.
- *Chapter 8. Formatting a report.* This chapter provides information about formatting spreadsheet reports.
- *Chapter 9. Adding content to a data range.* This chapter describes different types of data, explains how to supply the data for your report, and covers special techniques for working with data.
- *Chapter 10. Using a chart.* This chapter discusses charts and supported chart types, presents a tutorial about creating a chart object, and provides examples of using a chart in a spreadsheet report.
- *Chapter 11. Charting a data range.* This chapter discusses designing a chart, describes how to link a chart to a data range and explains how to refine the appearance of a chart by formatting chart elements.
- *Chapter 12. Designing a customizable report.* This chapter describes how to use parameters to create a report design that users can customize for their needs.
- *Chapter 13. Working with a pivot range.* This chapter describes how to create and manipulate a pivot range.
- *Chapter 14. Designing a secure report.* This chapter describes how to use BIRT Spreadsheet Designer's security features, such as cell locking and SmartSheet security, to maintain information integrity in your spreadsheet reports.
- *Chapter 15. Using a hyperlink.* This chapter describes how to set up and use a hyperlink in a spreadsheet report.
- *Chapter 16. Using a spreadsheet report on a production system.* This chapter provides information about publishing a spreadsheet report to a server. This chapter describes how to change a data source connection when running a report, enabling migration of reports to production data sources without modifying the report design. This chapter also describes how to add VBA functionality to a published report.
- *Chapter 17. Personalizing BIRT Spreadsheet Designer.* This chapter explains how to customize the BIRT Spreadsheet Designer environment.
- *Part 4. Extending BIRT Spreadsheet Designer.* This part describes how to use scripting and commands to enhance a report.

- *Chapter 18. Report script reference guide.* This chapter is a guide to the report script functions you use to build a report.
- *Chapter 19. Working with callback classes.* This chapter explains how to use the BIRT Spreadsheet API to develop report callbacks for use with BIRT Spreadsheet Designer.
- *Chapter 20. Examining samples of callback classes.* This chapter offers several example callback classes.
- *Chapter 21. Working with VBA.* This chapter explains how to use VBA to add features such as macros, auto shapes, or toolbar items to a spreadsheet report.
- *Part 5. Working in multiple locales using BIRT Spreadsheet technology.* This part describes how to create reports that work in multiple locales.
- *Chapter 22. Using BIRT Spreadsheet Designer with multiple locales.* This chapter describes how to create and deploy a report in a different locale.
- *Appendix A. BIRT Spreadsheet Designer internationalization information.* This appendix lists format symbols, values and errors, and function names that you use to create a report for use in multiple locales.
- *Appendix B. Comparing Excel and BIRT Spreadsheet Designer.* This appendix describes differences between Microsoft Excel files and the Excel files that BIRT Spreadsheet Designer produces.
- *Glossary.* This chapter provides definitions of terms used in Actuate BIRT Spreadsheet Designer product and documentation.

The examples and screenshots in this book use data from two sample relational databases, Classic Models and Securities, that Actuate ships with the BIRT Spreadsheet Designer product.





# Part One



**Getting Started**



# 1

## **Building your first spreadsheet reports**

This chapter contains the following topics:

- About Actuate BIRT Spreadsheet Designer
- Creating a report using the listing wizard
- Creating a report from scratch

---

## About Actuate BIRT Spreadsheet Designer

An Actuate spreadsheet report differs from a Microsoft Excel spreadsheet in two significant ways:

- BIRT Spreadsheet Designer retrieves data directly from a data source. Each time the report is run, BIRT Spreadsheet Designer retrieves the data from the source, thereby keeping report output up-to-date.
- BIRT Spreadsheet Designer's Design Editor provides a powerful, structured environment in which to develop flexible, textured reports. Using this environment, you can create simple spreadsheet reports or design robust reports that contain a combination of parameters, computed fields, charts, pivot ranges, and security features.

The tutorials that follow show you how to design two different reports. In the first, you use the listing wizard to quickly design a simple report. In the second, you use the design editor's rich work area to create a more complex report.

### About the report design process

The BIRT Spreadsheet Designer report design process consists of the following tasks, whether you use a wizard to help you or design the report entirely from scratch:

- Connect to a data source.  
A data source is the location from which BIRT Spreadsheet Designer retrieves the data for your report. You specify the data source to use for each report. A data source can be a relational database, a flat file, an SAP database, an Actuate information object, or a custom source that you define.
- Create a data set and a query.  
You use a data set to specify the tables to use from the data source. You use a query to specify the data columns to use from each table.
- Construct a data range.  
You create a data range in a report design to structure the report layout. A data range consists of row and column sections and groups. You can use the listing wizard to construct a simple data range, or design one of your own, from scratch.
- Add data to the data range.  
You can drag a data element from a data set and drop it into the data range. Alternatively, you can use the listing wizard to perform this step. Either way, you use the BIRT Spreadsheet Designer report script function builder to help you specify how data displays on the report. For example, the write function specifies that customer name data displays as text, while the count function

stipulates that quantity data displays as a number, and the sum function specifies that currency amounts display as totals or subtotals. A report script function is a combination of simple formulas, data fields, and filter statements.

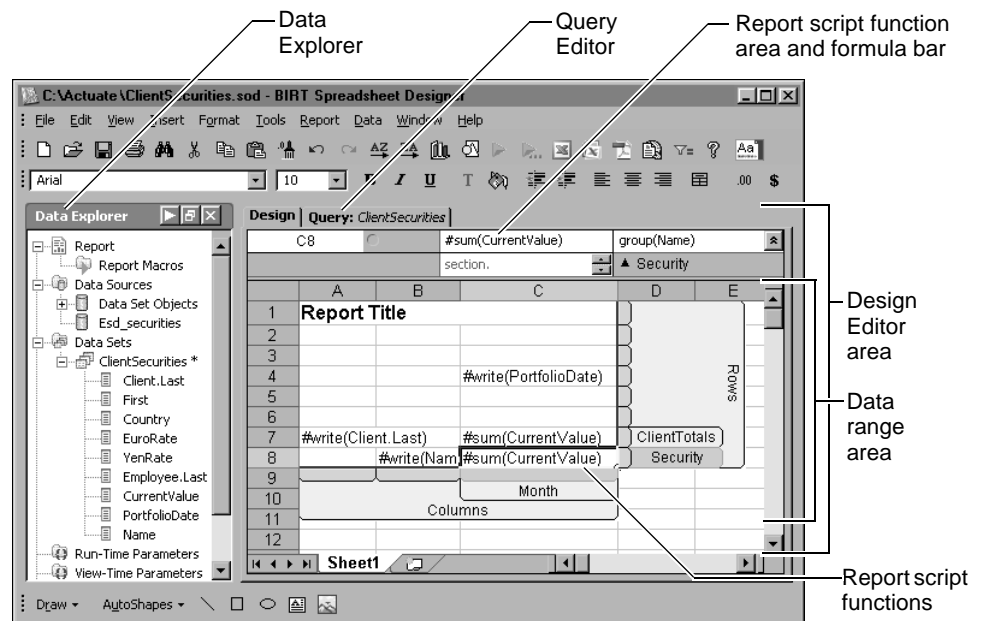
- **Format the report design.**  
You can add several types of formatting to a spreadsheet report: bold, italic, different fonts and font sizes, borders, and conditional format and formulas.
- **Preview report results.**  
You can preview report output to determine which aspects of the design to modify before releasing the report to end users.

The tutorials that follow show you how to perform each task.

## About the BIRT Spreadsheet Designer user interface

Use the query editor to select data elements and create data sets. BIRT Spreadsheet Designer generates query syntax as you select columns from each table. In the query editor toolbar, select Tools to view a query using Design or SQL view.

Use the design editor to perform most report design tasks, such as placing, formatting, and manipulating data fields in a data range. Figure 1-1 highlights primary elements of the design editor user interface. Use both editors to complete the following tutorials.



**Figure 1-1** BIRT Spreadsheet Designer design editor

For more information about the BIRT Spreadsheet Designer user interface, see Chapter 2, “Understanding report design.”

## Tutorial 1: Creating a report using the listing wizard

In this tutorial, you create a simple report using the BIRT Spreadsheet Designer listing wizard. The listing wizard performs two significant steps for you:

- Constructs a data range based on the data set you design
- Helps you specify the location of the data on the report

### Task 1: Connect to a data source

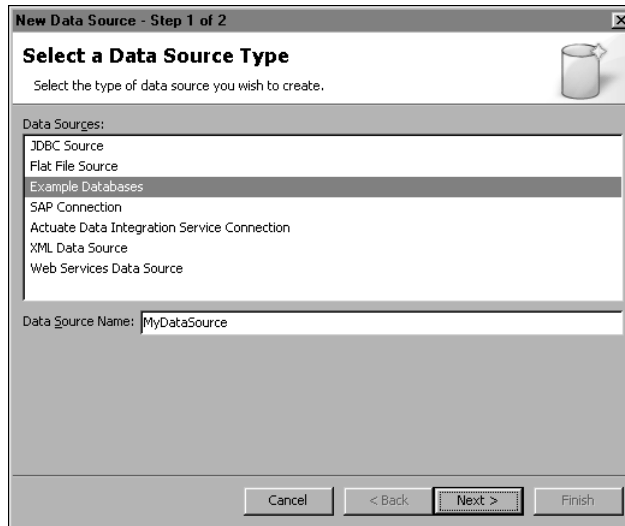
In this part of Tutorial 1, you create a connection to the Classic Models database.

- 1 Open BIRT Spreadsheet Designer.
- 2 In Data Explorer, right-click Data Sources, then choose Create Data Source, as shown in Figure 1-2.

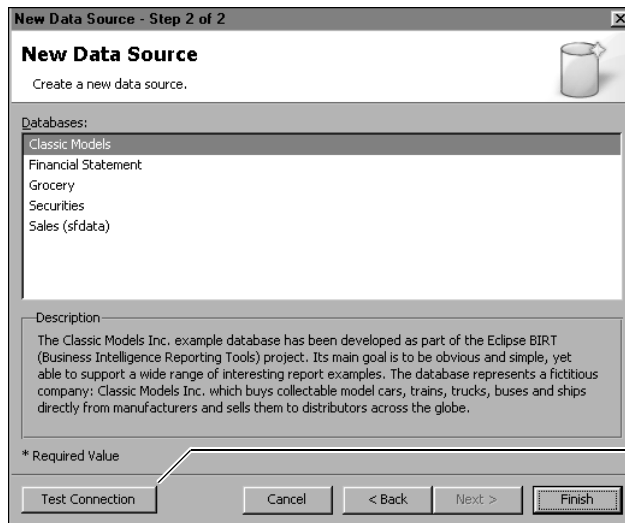


**Figure 1-2** Choosing Create Data Source

- 3 On New Data Source—Step 1 of 2, take the following actions, as shown in Figure 1-3, then choose Next:
  - In Data Sources, select Example Databases.
  - In Data Source Name, type  
MyDataSource
- 4 On New Data Source—Step 2 of 2, take the following actions, as shown in Figure 1-4:
  - In Databases, select Classic Models.
  - Choose Test Connection.



**Figure 1-3** Creating a new data source, step 1



**Figure 1-4** Creating a new data source, step 2

- 5 When the Connection test succeeded message appears, as shown in Figure 1-5, choose OK.



**Figure 1-5** Connection test succeeded message

- 6 On New Data Source—Step 2 of 2, choose Finish.

The data source that you created now appears in Data Explorer, under Data Sources, as shown in Figure 1-6.

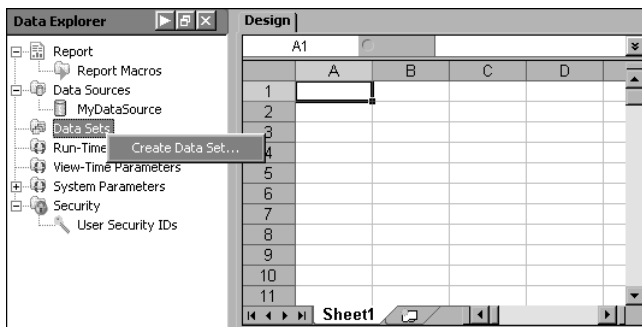


**Figure 1-6** New data source in Data Explorer

## Task 2: Create a data set and a query

In this part of Tutorial 1, you create a data set using a pre-defined query table called a view. A pre-defined view contains columns from a number of different tables, which simplifies the process of creating a data set. To construct a data set from scratch, you typically use multiple tables, create joins between them, then select columns from each table. You learn to construct a data range from scratch in Tutorial 2.

- 1 In Data Explorer, right-click Data Sets, then choose Create Data Set, as shown in Figure 1-7.

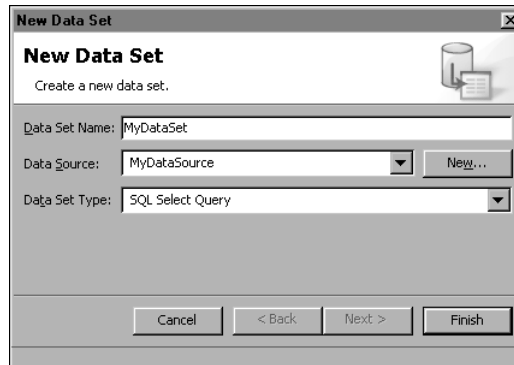


**Figure 1-7** Creating a new data set

- 2 On New Data Set, do the following steps, as shown in Figure 1-8.

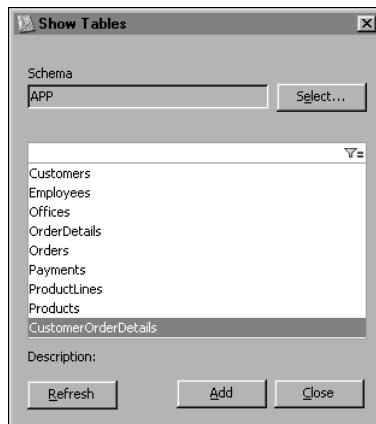


- 1 In Data Set Name, type:  
MyDataSet
- 2 In Data Source, ensure that MyDataSource is selected, then choose Finish.



**Figure 1-8** Naming the data set

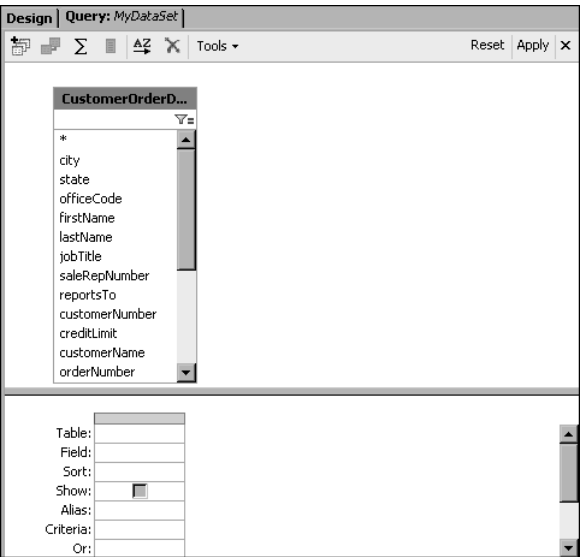
- 3 On Show Tables, select CustomerOrderDetails, as shown in Figure 1-9. Leave the Schema field unchanged.



**Figure 1-9** Selecting a data element using Show Tables

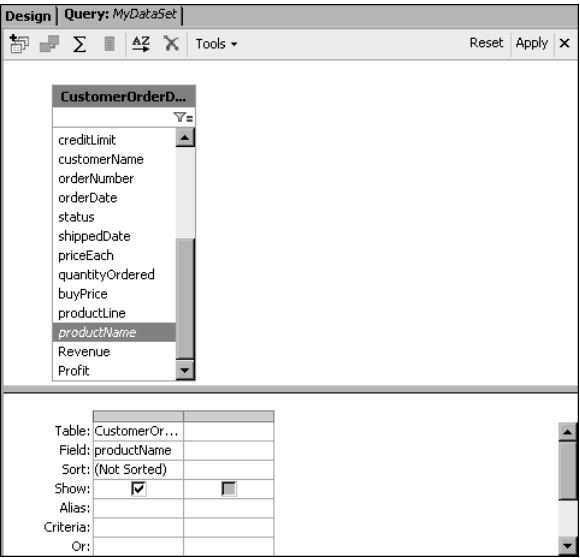
- 4 Choose Add, then choose Close.

The data element that you chose appears in Query Editor, as shown in Figure 1-10.



**Figure 1-10** Query Editor design view after adding a data element

- 5 In CustomerOrderDetails, double-click ProductName. The table and field that you chose then appears as a column in the lower pane of Query Editor, as shown in Figure 1-11.

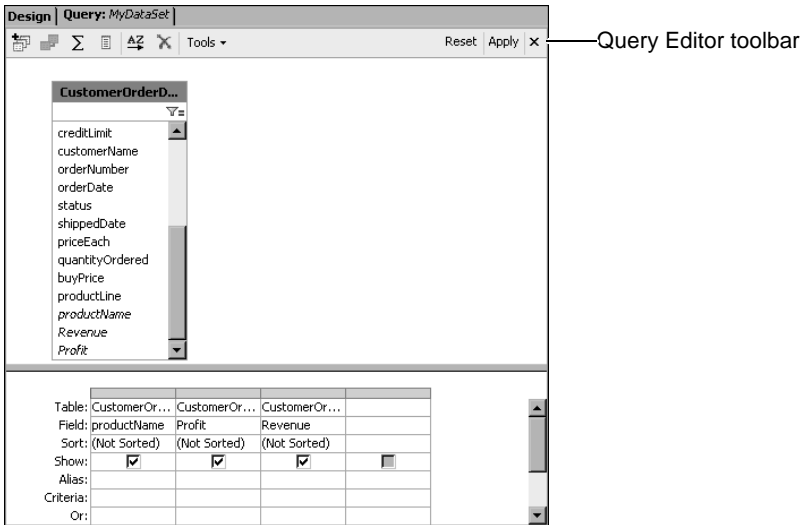


**Figure 1-11** A column added in Query Editor

- 6 Double-click the following fields in CustomerOrderDetails to add them to the query:

- Profit
- Revenue

The lower pane of Query Editor now appears as shown in Figure 1-12. Data will appear on the generated report in the same order as the columns appear in the lower pane, unless you change the order elsewhere in the report design.

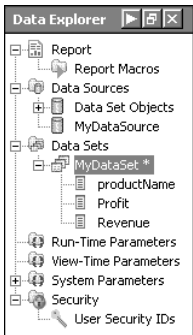


**Figure 1-12** Column in Query Editor



- 7 Choose Apply on the query editor toolbar to save the query.

MyDataSet appears in Data Explorer. Data fields that you added to MyDataSet appear under the data set name in the expanded tree view, as shown in Figure 1-13. To contract the view, select a minus sign located to the left of a listed element.

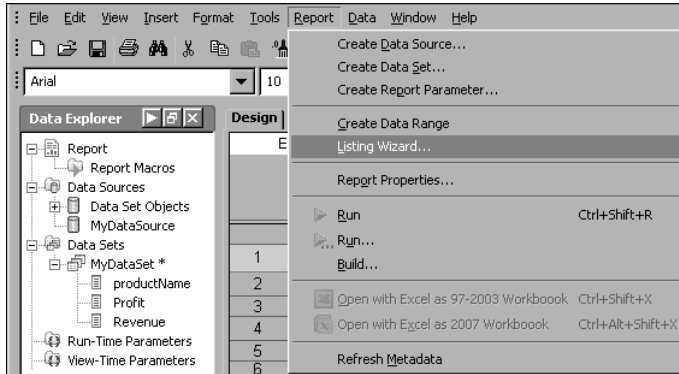


**Figure 1-13** Expanded view of data set

### Task 3: Use the listing wizard to create a data range

In this part of the tutorial, you use the listing wizard to create a data range and to specify the location of data on the range.

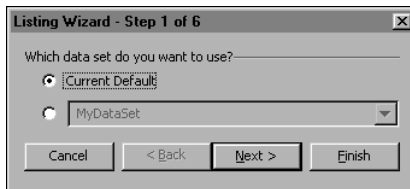
- 1 In Design Editor, choose Report➤Listing Wizard from the main menu, as shown in Figure 1-14.



**Figure 1-14** Select Listing wizard

- 2 On Listing Wizard—Step 1 of 6, accept the default data set, Current Default, as shown in Figure 1-15, then choose Next.

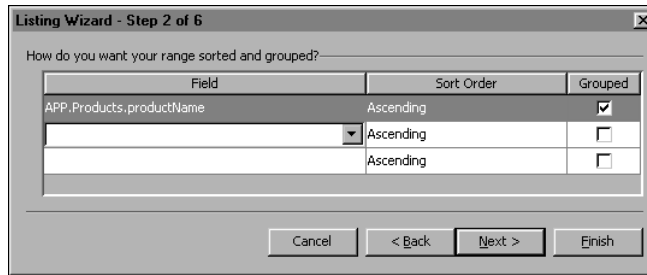
The default data set is the one you created in Task 2. When your report contains multiple data sets, you can choose any one of them as a data source.



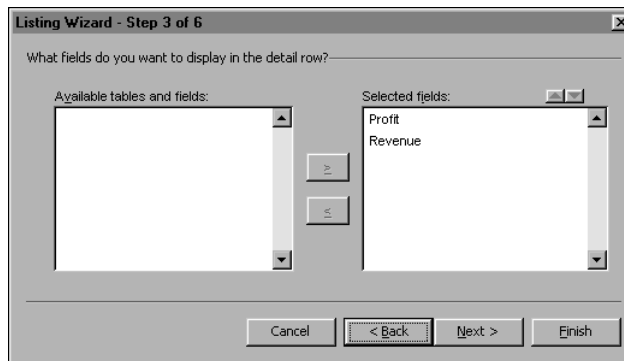
**Figure 1-15** Listing Wizard, step 1

- 3 On Listing Wizard—Step 2 of 6, do the following steps, as shown in Figure 1-16, then choose Next:
  - 1 Select the blank cell or the down arrow under Field, then select APP.Products.productName.
  - 2 In the row that contains APP.Products.productName, select Grouped.
- 4 On Listing Wizard—Step 3 of 6, verify that Profit and Revenue appear under Selected Fields, as shown in Figure 1-17, then choose Next.

This action causes both selected fields, Profit and Revenue, appear in the report detail row.

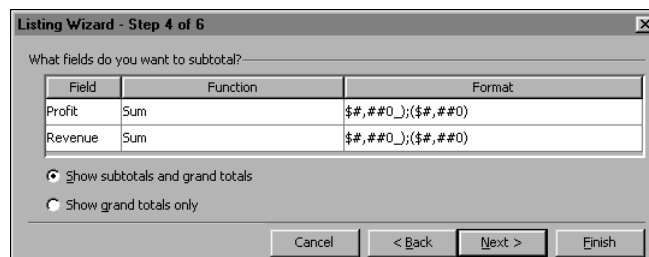


**Figure 1-16** Listing Wizard, step 2



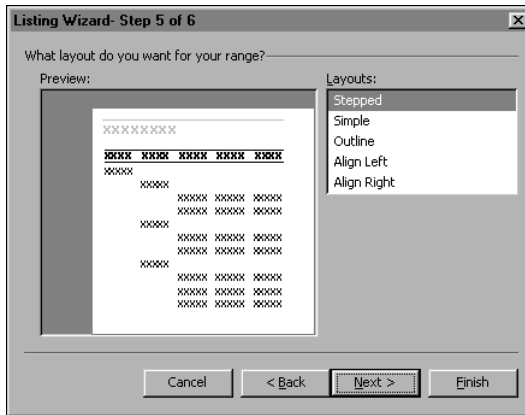
**Figure 1-17** Listing Wizard, step 3

- 5 On Listing Wizard—Step 4 of 6, do the following steps, as shown in Figure 1-18, then choose Next:
  - 1 In the Function field for Profit, select sum.
  - 2 In the Format field for Profit, select:  
\$#,##0\_) ; (\$#,##0)
  - 3 Repeat steps 5-1 and 5-2 for Revenue.
  - 4 Ensure that the default radio button, Show subtotals and grand totals, is selected.



**Figure 1-18** Listing Wizard, step 4

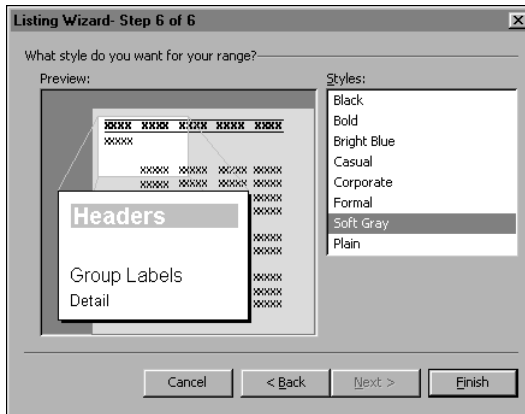
- 6 On Listing Wizard—Step 5 of 6, leave the Stepped layout selected, as shown in Figure 1-19, then choose Next.



**Figure 1-19** Listing Wizard, step 5

- 7 On Listing Wizard—Step 6 of 6, select Soft Gray, as shown in Figure 1-20, then choose Finish.

This step specifies the fonts for the report and the background color of the column headings.



**Figure 1-20** Listing Wizard, step 6

The data range that you created using the listing wizard now appears in Design Editor, as shown in Figure 1-21.

Notice the following:

- Data is grouped by product name. The product name group spans rows 2-5, and columns A-C.

- The detail row, row 3, contains both profit and revenue. You specified detail row contents in step 3 of the listing wizard.
- Subtotal formulas for profit and revenue appear in row 4. You specified to include subtotals in step 4 of the listing wizard.
- Grand total formulas appear in row 6. You specified to include grand totals in step 4 of the listing wizard.
- Row one contains column headings. You indicated the background color of the column headings in step 6 of the listing wizard.

	A	B	C	D	E	F	G	H	I	J
1	product	Profit	Reven							
2	#productName									
3		#Profit	#Revenue	Detail	productName	Range	Rows			
4		#formula	#formula							
5										
6		#formu	#formu							
7										
8	Columns									

**Figure 1-21** The data range after using the listing wizard

- 8 Choose File→Save Design As to save the report design as a .sod file.

## Task 4: Preview your report

In this part of the tutorial, you run the report and view the results.

- 1 In Design Editor, double the width of column A so that you can see the full name.
- 2 In cell A1, improve the appearance of the title by changing the text to:  
Product Name
- 3 Choose Run. The report appears on View, as shown in Figure 1-22. Profit and Revenue subtotals appear at the end of each product group. To see profit and revenue grand totals, scroll down to the bottom of the report, to row 3325.
- 4 Choose File→Save Design to save the changes you made to the report design.
- 5 Choose File→Save View As to name and save the generated view. The view is not automatically saved when you save the design.
- 6 Choose File→Close to close the report design.

Design   View   Query: MyDataSet			
B42		1407.36	
	A	B	C
1	Product Name	Profit	Revenue
2	18th century schooner		
3		1054.35	3771.57
4		903.07	4937.73
5		894.6	3200.12
6		700.5	3170.7
7		1109.4	4650.02
8		840.6	3804.84
9		1068.94	3456.8
10		1946.4	5898.72
11		722.4	4180.68
12		1419.25	4301.15
13		1032.15	3914.05
14		1179.6	4473.2
15		542.98	3342.54
16		1532.09	5072.71
17		619.29	2348.43
18		1505.28	5539.94
19		459.58	2271.06
20		1108.23	4484.17
21		1659	5776
22		1511.26	4887.2
23		727.42	3774
24		571.33	3123.87
25		783.87	3171.73
26		1413.12	5200.76
27		1382.4	5087.7
28		1561.69	4937.63
29		932.58	3649.8
30		\$29,181	\$112,427
31			
32	18th Century Vintage Horse Carriage		
33		1233.48	3541.6
34		1935.12	4607.68
35		1440.93	4052.75

Profit and revenue data grouped by product name

Profit and revenue subtotals by product name

**Figure 1-22** Viewing report results

## Tutorial 2: Creating a report from scratch

In this tutorial, you construct a report design from scratch. After you create a data set and query from multiple tables and columns, you construct a data range using a combination of row and column sections and groups. After constructing the data range, you add data to the report by dragging and dropping data fields from the Data Explorer into the range. Next, you add formatting to the report before running it.

### Task 1: Connect to a data source

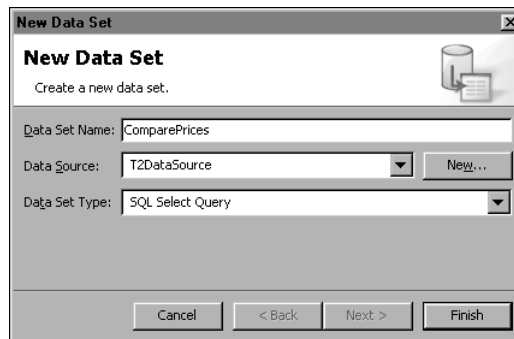
Open a new design, and follow the instructions in Tutorial 1, Task 1, to create a connection to the Classic Models database. The screen images for Tutorial 2 use T2DataSource as the name of the data source.



## Task 2: Construct a data set and a query

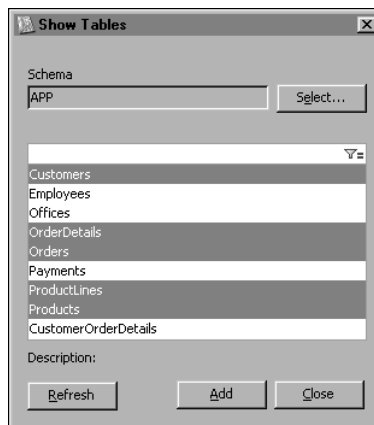
In this part of Tutorial 2, you construct a data set using multiple columns from multiple tables. You then create a database query based on the contents of the data set.

- 1 In Data Explorer, right-click Data Sets and choose Create Data Set.
- 2 On New Data Set, do the following steps, as shown in Figure 1-23, then choose Finish:
  - 1 In Data Set Name, type:  
`ComparePrices`
  - 2 In Data Source, select the data source you created in Tutorial 2, Task 1.



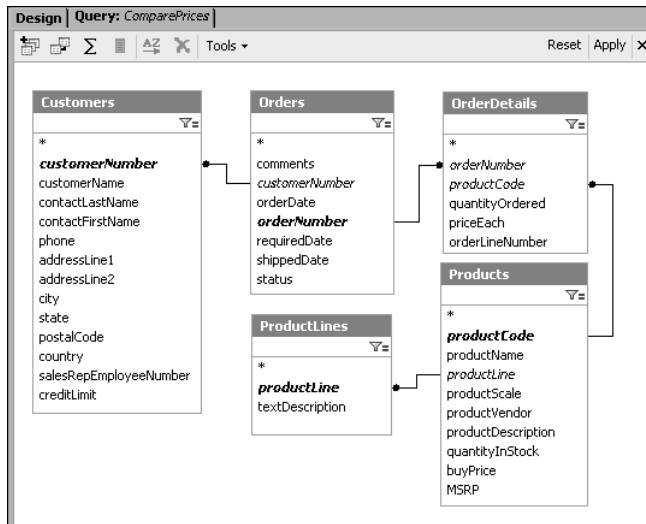
**Figure 1-23** Creating a new data set

- 3 On Show Tables, hold down the Ctrl key and select the following tables, as shown in Figure 1-24. Leave the Select Schema field unchanged.



**Figure 1-24** Selecting data tables

- Customers
  - OrderDetails
  - Orders
  - ProductLines
  - Products
- 4 Choose Add, then Close.
  - 5 In Query Editor, drag each table and drop it in a different location so that you can see all the joins, as the example in Figure 1-25 shows.



**Figure 1-25** Viewing tables and joins in Query Editor

- 6 Double-click the following columns to add them to the query:
  - ProductLines.productLine
  - Products.productName
  - Customers.customerName
  - OrderDetails.quantityOrdered
  - Products.MSRP
  - Products.buyPrice
  - Orders.orderDate

The lower pane of Query Editor appears as shown in Figure 1-26.

Table:	ProductLines	Products	Customers	OrderDetails	Products	Products	Orders	
Field:	productLine	productName	customerName	quantityOrd...	MSRP	buyPrice	orderDate	
Sort:	(Not Sorted)	(Not Sorted)	(Not Sorted)	(Not Sorted)	(Not Sorted)	(Not Sorted)	(Not Sorted)	
Show:	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Alias:								
Criteria:								
Or:								

**Figure 1-26** After adding columns to a query

Data appears on the generated report in the same order, from left to right, as the columns appear in Query Editor unless you change the order elsewhere in the report design.

- 7 In the column that shows orderDate, in Criteria, type the following filter expression, as shown in Figure 1-27:

Between 01/01/2003 and 12/31/2004

Table:	ProductLines	Products	Customers	OrderDetails	Products	Products	Orders	
Field:	productLine	productName	customerName	quantityOrd...	MSRP	buyPrice	orderDate	
Sort:	(Not Sorted)	(Not Sorted)	(Not Sorted)	(Not Sorted)	(Not Sorted)	(Not Sorted)	(Not Sorted)	
Show:	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Alias:								
Criteria:							Between 01/01/2003 and 12/31/2004	
Or:								

**Figure 1-27** Adding a filter criteria to orderDate

Apply

- 8 On the query editor toolbar, choose Apply to save the query.

The name of the data set you created now appears in Data Explorer under Data Sets.

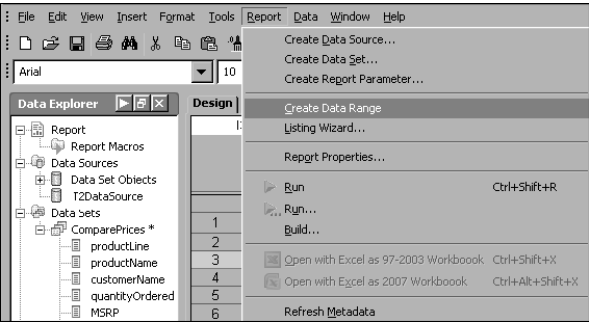
### Task 3: Construct a data range

In this part of Tutorial 2, you add row and column sections and groups to construct a data range.

#### How to create row sections

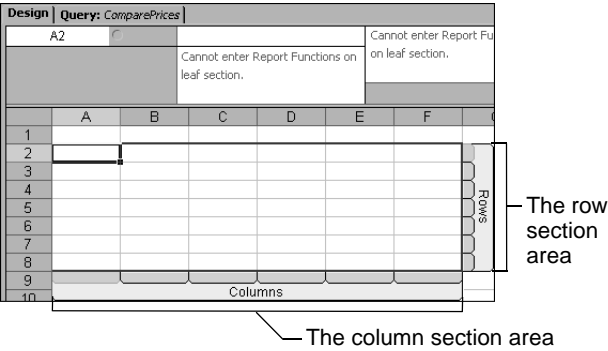
Use row and column sections to indicate the rows and columns that contain the data that displays in the report.

- 1 On Design Editor, select cells A2 through F8.
- 2 Then choose Report>Create Data Range from the main menu, as shown in Figure 1-28.



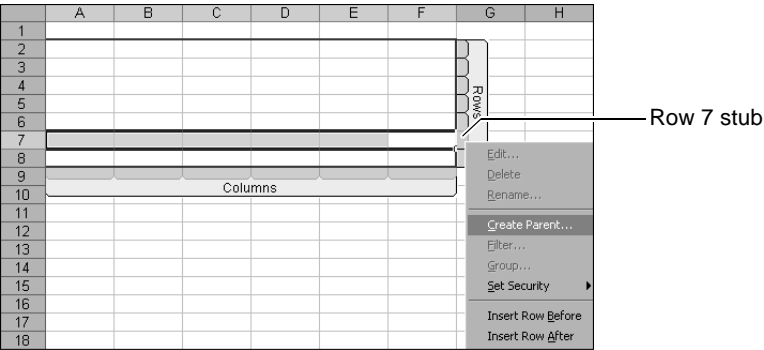
**Figure 1-28** Choosing Create Data Range

The data range appears on Design, as shown in Figure 1-29.



**Figure 1-29** A blank data range

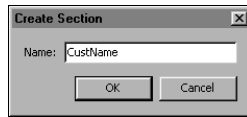
- 3 Right-click the row 7 stub, between columns F and G, then choose Create Parent from the context menu, as shown in Figure 1-30.



**Figure 1-30** Adding a parent section

Create Section appears.

- 4 On Create Section, in Name, replace Section 1 with Custname, as shown in Figure 1-31. Then, choose OK.



**Figure 1-31** Naming a section

The data range appears as shown in Figure 1-32.

A screenshot of a spreadsheet grid. The columns are labeled A through H, and the rows are labeled 1 through 10. A shaded area covers rows 6, 7, and 8, spanning columns A through F. This area is labeled 'CustName' on the right side. The label 'Columns' is at the bottom, and 'Rows' is on the right side of the grid.

**Figure 1-32** After adding the first section

- 5 Select the row 5 stub and drag your cursor over the stubs for rows 6, 7 and 8 so that you select all four stubs at once. Then, right-click and choose Create Parent from the context menu.
- 6 On Create Section, in Name replace Section1 with ProdName, as shown in Figure 1-33. Then, choose OK.



**Figure 1-33** Naming the second section

The data range now appears as shown in Figure 1-34. The ProdName section is the parent to the CustName section.

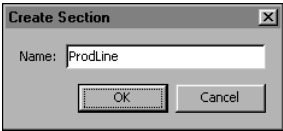
A screenshot of a spreadsheet grid. The columns are labeled A through I, and the rows are labeled 1 through 10. A shaded area covers rows 6, 7, and 8, spanning columns A through F, labeled 'CustName'. A larger shaded area covers rows 5, 6, 7, and 8, spanning columns A through G, labeled 'ProdName'. The label 'Columns' is at the bottom, and 'Rows' is on the right side of the grid.

**Figure 1-34** After adding the second section

- 7 Select the row 3 stub and drag your cursor over the stubs for rows 4-8 so that you select all six stubs at once. Then, right-click and choose Create Parent from the context menu.

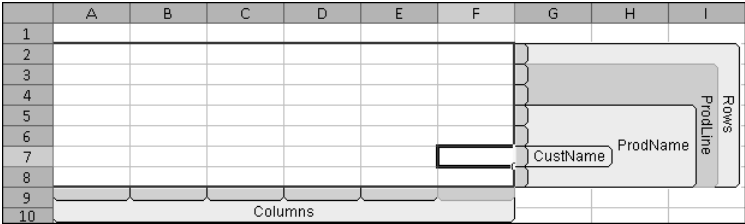
- 8 On Create Section, replace the existing text with the following name, as shown in Figure 1-35. Then choose OK.

ProdLine



**Figure 1-35** Naming the third section

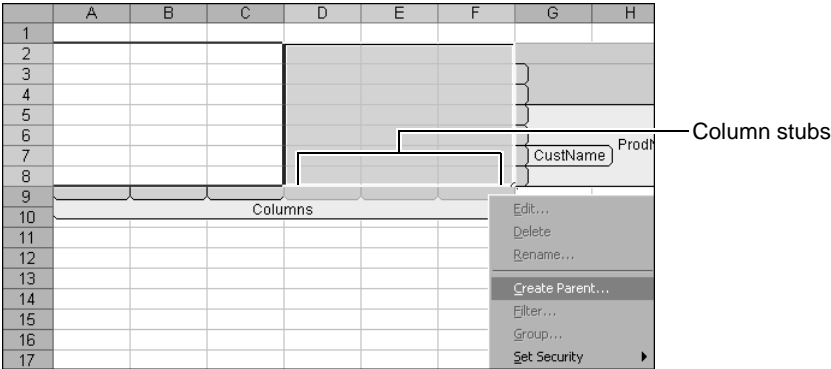
The data range appears as shown in Figure 1-36. The ProdLine section is the parent to the ProdName and CustName sections.



**Figure 1-36** After adding the third section

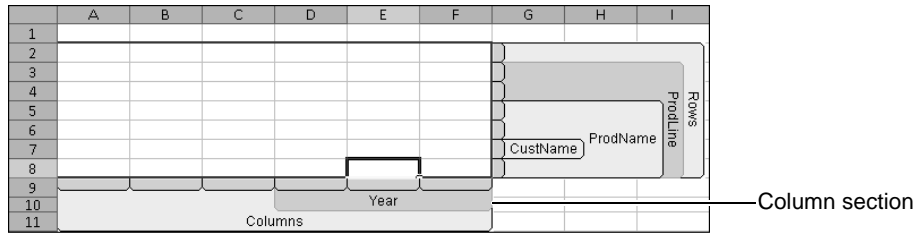
**How to create column sections**

- 1 Select the column stubs for columns D-F and choose Create Parent, as shown in Figure 1-37.



**Figure 1-37** Creating a parent column section

- 2 On Create Section, in Name, replace Section1 with Year. Then, choose OK.  
The data range now appears as shown in Figure 1-38.

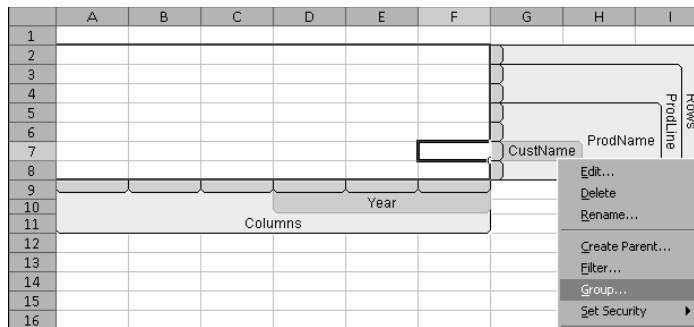


**Figure 1-38** After adding a column section

### How to create section groups

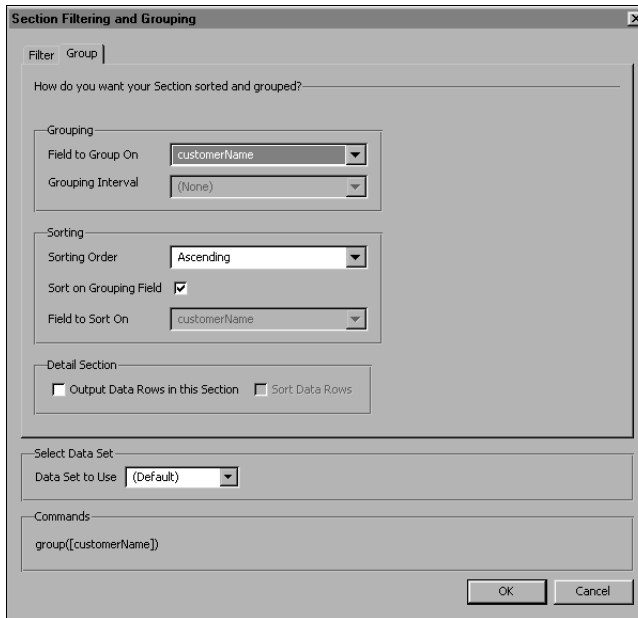
You create section groups when you want data to display in groups. For example, in this tutorial, you group customer data first by product line and then by product name. You then group all of that information by year. When you run the report later in the tutorial, you see how the groups display data in an organized, easy-to-read manner.

- 1 In Design Editor, right-click the CustName section tab, then choose Group from the context menu, as shown in Figure 1-39.



**Figure 1-39** Creating a group

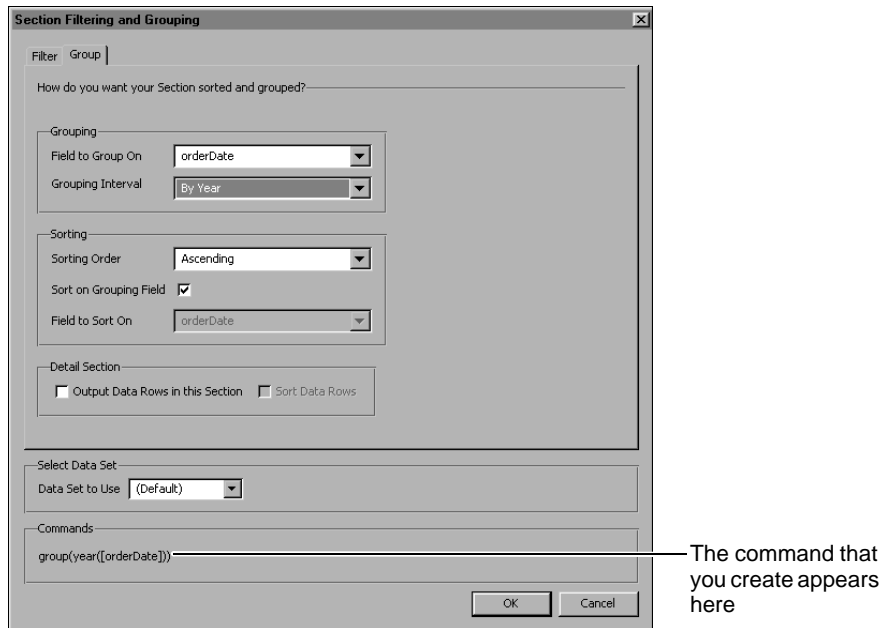
- 2 On Section Filtering and Grouping, do the following steps, as shown in Figure 1-40. Then, choose OK.
  - 1 In Field to Group On, select customerName.
  - 2 In Sorting Order, confirm that Ascending is selected.
  - 3 Ensure that Sort on Grouping Field is selected.
- 3 Right-click the ProdName section tab, then choose Group.
- 4 On Section Filtering and Grouping, do the following steps. Then, choose OK.
  - 1 In Field to Group On, select productName.
  - 2 In Sorting Order, confirm that Ascending is selected.
  - 3 Ensure that Sort on Grouping Field is selected.



**Figure 1-40** Creating the CustName group

- 5 Right-click the ProdLine section tab, then choose Group.
- 6 On Section Filtering and Grouping, do the following steps. Then, choose OK.
  - 1 In Field to Group On, select productLine.
  - 2 In Sorting Order, confirm that Ascending is selected.
  - 3 Ensure that Sort on Grouping Field is selected.
- 7 Right-click the Year section tab, then choose Group.
- 8 On Section Filtering and Grouping, do the following steps, as shown in Figure 1-41. Then, choose OK.
  - 1 In Field to Group On, select orderDate.
  - 2 In Grouping Interval, select By Year.
  - 3 In Sorting Order, confirm that Ascending is selected.
  - 4 Ensure that Sort on Grouping Field is selected.





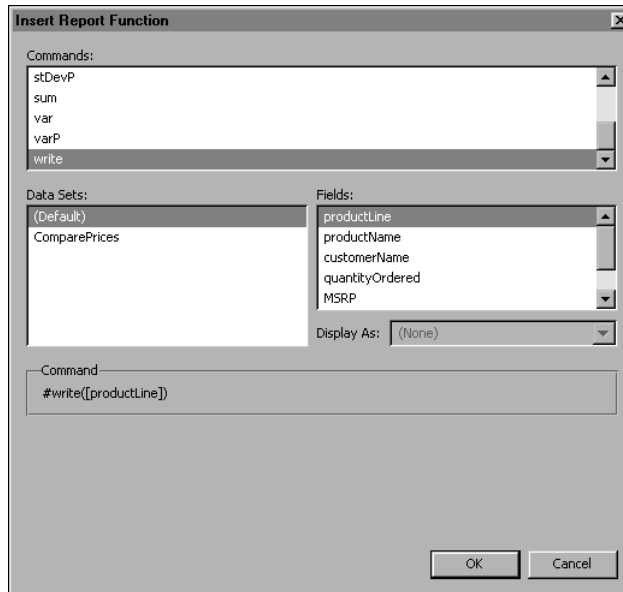
**Figure 1-41** Grouping the Year section

## Task 4: Add data to the report

In this part of Tutorial 2, you add data to the cells in each section. To do this, drag columns from the data set and drop them into the appropriate cell in the data range. During the drag-and-drop process, you use the report script function builder to create a command that tells BIRT Spreadsheet Designer how to calculate and display the data.

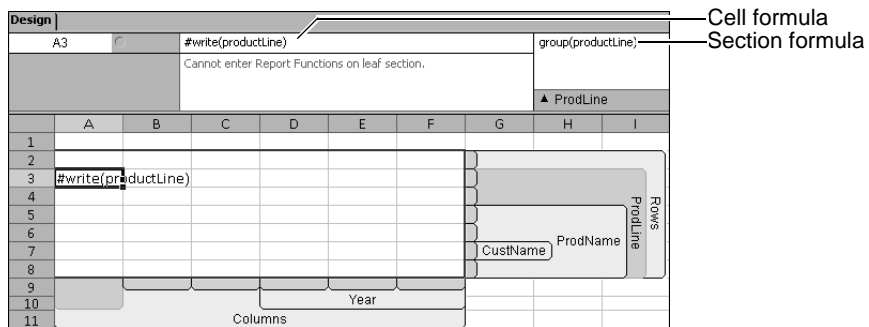
### How to add data fields to the report

- 1 To add the productLine data field to the data range:
  - 1 Drag productLine from the ComparePrices data set in Data Explorer and drop it into cell A3.
  - 2 On Insert Report Script Function, do the following steps, as shown in Figure 1-42. Then, choose OK.
    - 1 In Commands, confirm that write is selected.
    - 2 In Fields, select productLine.
    - 3 In Data Sets, confirm that (Default) is selected.



**Figure 1-42** Inserting the write function

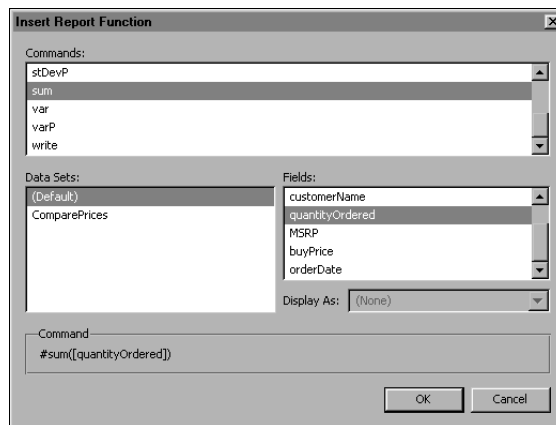
The data range now appears as shown in Figure 1-43. When you select a section tab, the formula for the section appears at the right side of the design editor. When you select a cell, the formula for the cell appears at the top center of the design editor.



**Figure 1-43** Data range after adding a report script function to cell A3

- 2 To add the productName data field to the data range:
  - 1 Drag productName from Data Explorer and drop it into cell B5.
  - 2 On Insert Report Script Function, do the following steps. Then, choose OK.
    - 1 In Commands, confirm that write is selected.
    - 2 In Fields, select productName.

- 3 In Data Sets, confirm that (Default) is selected.
- 3 To add the customerName data field to the data range:
  - 1 Drag customerName from Data Explorer and drop it into cell C7.
  - 2 On Insert Report Script Function, do the following steps. Then, choose OK.
    - 1 In Commands, confirm that write is selected.
    - 2 In Fields, select customerName.
    - 3 In Data Sets, confirm that (Default) is selected.
- 4 To add the quantityOrdered data field to the data range:
  - 1 Drag quantityOrdered from Data Explorer and drop it into cell D7.
  - 2 On Insert Report Script Function, do the following steps, as shown in Figure 1-44. Then, choose OK.
    - 1 In Commands, select sum.
    - 2 In Fields, select quantityOrdered.
    - 3 In Data Sets, confirm that (Default) is selected.



**Figure 1-44** Inserting the sum function

The data range now appears as shown in Figure 1-45.

	A	B	C	D	E	F
1						
2						
3	#write(productLine)					
4						
5		#write(productName)				
6						
7			#write(customerName)	#sum(quantityOrdered)		
8						
9						
10					Year	

**Figure 1-45** Viewing the data range after adding quantityOrdered

- 5 To add the MSRP data field to the data range:
  - 1 Drag MSRP from Data Explorer and drop it into cell E7.
  - 2 On Insert Report Script Function, do the following steps. Then, choose OK.
    - 1 In Commands, select write.
    - 2 In Fields, select MSRP.
    - 3 In Data Sets, confirm that (Default) is selected.
- 6 To add the buyPrice data field to the data range:
  - 1 Drag buyPrice from Data Explorer and drop it into cell F7.
  - 2 On Insert Report Script Function, do the following steps. Then, choose OK.
    - 1 In Commands, confirm that sum is selected.
    - 2 In Fields, select buyPrice.
    - 3 In Data Sets, confirm that (Default) is selected.

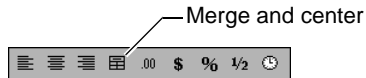
The data range now appears as shown in Figure 1-46.

	A	B	C	D	E	F
1						
2						
3	#write(productLine)					
4						
5		#write(productName)				
6						
7			#write(customerName)	#sum(quantityOrdered)	#write(MSRP)	#sum(buyPrice)
8						
9						
10					Year	

**Figure 1-46** Viewing the data range after adding buyPrice

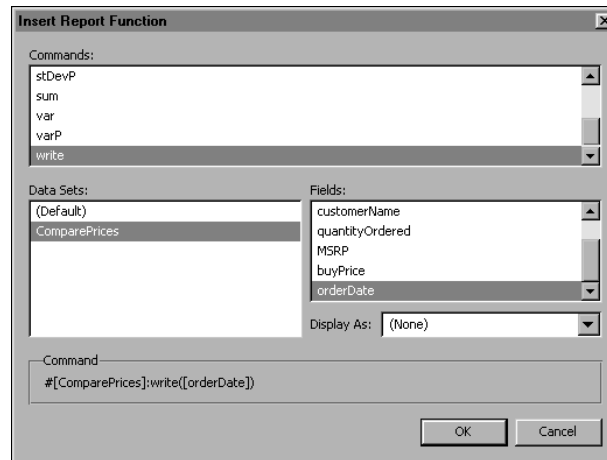
#### How to create a merged data cell

- 1 Select cells D2 through F2, then choose the Merge and Center icon on the formatting toolbar, as shown in Figure 1-47. This action merges the three cells into one large cell, D2, and centers the data it displays.



**Figure 1-47** Choosing merge and center

- 2 To add the orderDate data field to the data range:
  - 1 Drag orderDate from Data Explorer and drop it into the merged cell, D2.
  - 2 On Insert Report Script Function, do the following steps, as shown in Figure 1-48. Then, choose OK.
  - 1 In Commands, confirm that write is selected.



**Figure 1-48** Inserting a report script function in a merged cell

- 2 In Data Sets, select ComparePrices. When merging cells, you must indicate the specific data set, even if the data set is the default data set.
- 3 In Fields, select orderDate.

The data range now appears as shown in Figure 1-49.

	A	B	C	D	E	F	G	H	I
1									
2				#ComparePrices:write(orderDate)					
3	#write(productLine)						<div> <div>ProdLine</div> <div>ProdName</div> </div>		
4		#write(productName)							
5									
6									
7			#write(cust	#sum(quant	#write(MSR	#sum(buyP	CustName	ProdName	
8									
9									
10					Year				
11									

**Figure 1-49** Viewing the data range after adding orderDate

- 3 Choose File→Save Design As to name and save the report.



- 4 Choose Run to view the report before adding formatting. The report appears as shown in Figure 1-50.

Design	View									
	A36									
	A	B	C	D	E	F	G	H	I	
1										
2					37627			37988		
3	Classic Cars									
4										
5		1948 Porsche 356-A Roadster								
6										
7			Australian C	0		0	55	77	53.9	
8			Blauer See	41	77	53.9	0		0	
9			Collectable	26	77	53.9	0		0	
10			Cruz & Sons	38	77	53.9	0		0	
11			Down Unde	0		0	21	77	53.9	
12			Euro+ Shop	45	77	53.9	20	77	53.9	
13			FunGiftIde	0		0	37	77	53.9	
14			Herkku Gift	22	77	53.9	0		0	
15			L'ordine Sou	0		0	33	77	53.9	
16			Land of Toy	0		0	29	77	53.9	
17			Marseille M	43	77	53.9	0		0	
18			Men 'R' US F	38	77	53.9	0		0	
19			Mini Auto V	20	77	53.9	0		0	
20			Mini Gifts D	48	77	53.9	0		0	
21			Online Diec	45	77	53.9	0		0	
22			Oulu Toy Su	0		0	46	77	53.9	
23			Signal Gift S	0		0	29	77	53.9	
24			Toms Spezi	0		0	20	77	53.9	
25			UK Collecta	0		0	23	77	53.9	
26			Vida Sport,	0		0	91	77	107.8	

**Figure 1-50** Viewing report results before formatting

Notice that the report has no column headings, the dates are not formatted, and the price columns do not display numbers in a currency format.

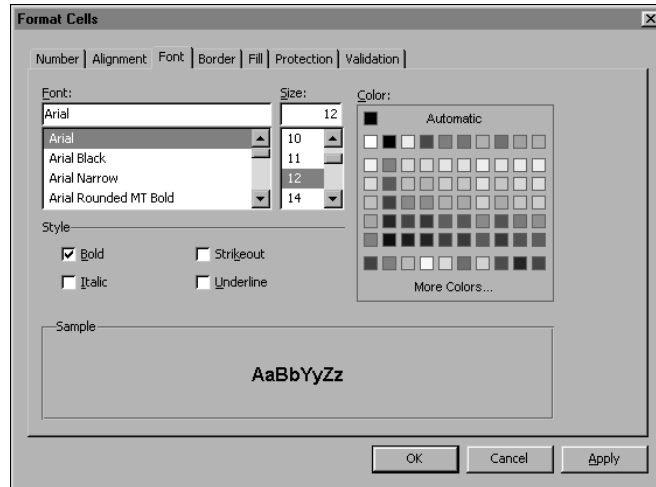
## Task 5: Format the report

In this part of Tutorial 2, you add and format a report title and column headings. You also format the data cells to show, as appropriate, dollar signs and actual dates.

### How to format a report title

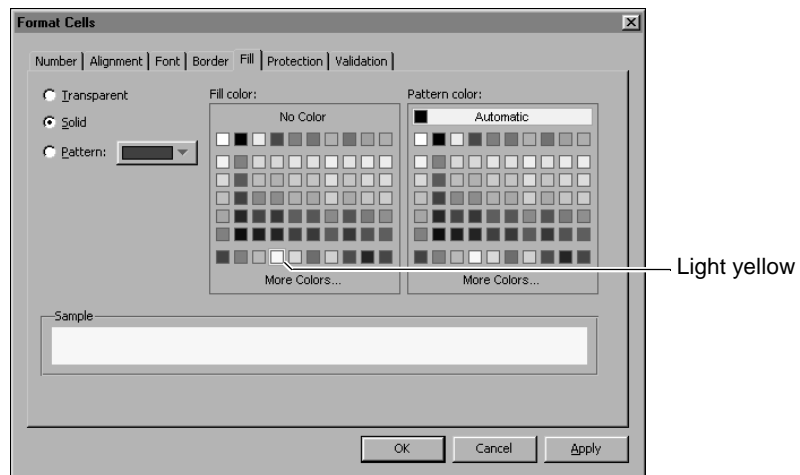
- 1 On Design, in cell A1, type:  
MSRP and Purchase Price
- 2 Press Enter.
- 3 Right-click row 1. From the context menu, choose Format Cells.
- 4 On Format Cells, select Font.
- 5 On Font, do the following steps, as shown in Figure 1-51. Then, choose Apply.
  - 1 In Font, select Arial.
  - 2 In Size, select 12.

- 3 In Style, select Bold.
- 6 On Format Cells, select Fill.



**Figure 1-51** Format font attributes

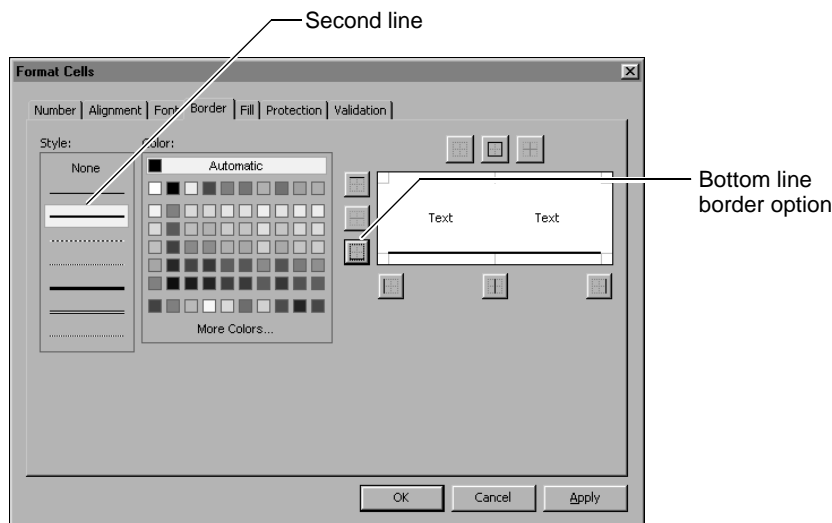
- 7 On Fill, do the following steps, as shown in Figure 1-52. Then, choose Apply.
  - 1 Select Solid.
  - 2 In Fill Color, select light yellow.



**Figure 1-52** Format font attributes

- 8 On Format Cells, select Border.
- 9 On Border, do the following steps, as shown in Figure 1-53. Then, choose OK.

- 1 In Style, select the second line.
- 2 Select the bottom line border option on the right side of the dialog.



**Figure 1-53** Format border attributes

The title row now looks similar to the one shown in Figure 1-54.

	A	B	C	D	E	F	G	H	I
1	<b>MSRP and Purchase Price</b>								
2				#ComparePrices:write(orderDate)					
3	#write(productLine)								
4									
5		#write(productName)							
6									
7			#write(cust	#sum(quant	#write(MSR	#sum(buyP	CustName	ProdName	
8									
9									
10					Year				
11									

**Figure 1-54** The data range after formatting the title

### How to format column headings

- 1 Type the listed heading in each of the following cells:
  - In A2, type Product Line, then press Enter.
  - In B4, type Product Name, then press Enter.
  - In C6, type Customer Name, then press Enter.
  - In D6, type Quantity, then press Enter.
  - In E6, type MSRP, then press Enter.
  - In F6, type Purchase Price, then press Enter.



- 2 Use the formatting toolbar to bold and center each heading. Also add bold to the merged date cell.

The data range now appears as shown in Figure 1-55.

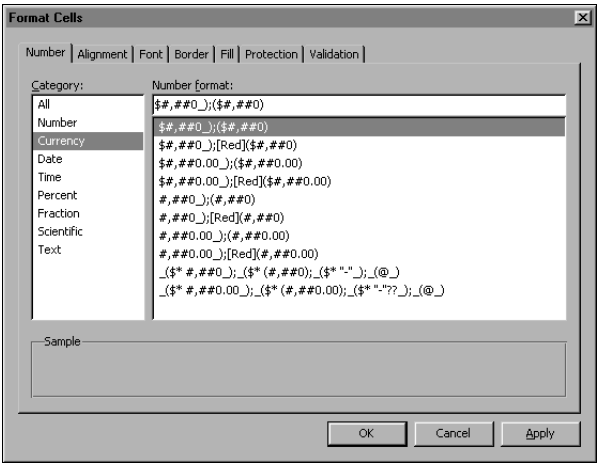
	A	B	C	D	E	F	G	H	I
1	<b>MSRP and Purchase Price</b>								
2	Product Line			#ComparePrices:write(orderDate)					
3	#write(productLine)								
4	Product Name								
5	#write(productName)								
6	Customer Name	Quantity	MSRP	Purchase Price					
7	#write(cust	#sum(quant	#write(MSR	#sum(buyP					
8									
9									
10									
11									

**Figure 1-55** The data range after formatting column headings

### How to format data cells

- 1 Select cells E7 and F7. Right-click. From the context menu, choose Format Cells.
- 2 On Format Cells, select Number.
- 3 On Number, do the following steps, as shown in Figure 1-56. Then, choose OK.
  - 1 In Category, select Currency.
  - 2 In Number Format, select:

\$#,##0\_);( \$#,##0)

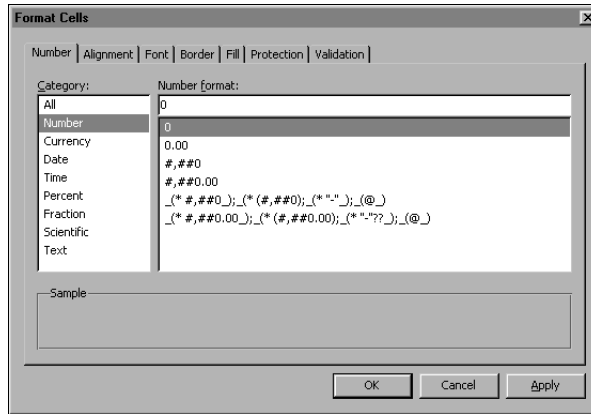


**Figure 1-56** Formatting currency

- 4 Right-click cell D7. From the context menu, choose Format Cells.

- 5 On Format Cells, select Number.
- 6 On Number, do the following steps, as shown in Figure 1-57. Then, choose OK.
  - 1 In Category, select Number.
  - 2 In Number Format, select:
 

0



**Figure 1-57** Formatting a simple number cell

- 7 Right-click the merged date cell, D2, then choose Format Cells.
- 8 On Format Cells—Number, do the following steps, as shown in Figure 1-58. Then, choose OK.
  - 1 In Category, select Date.
  - 2 In Number Format, type:
 

YYYY

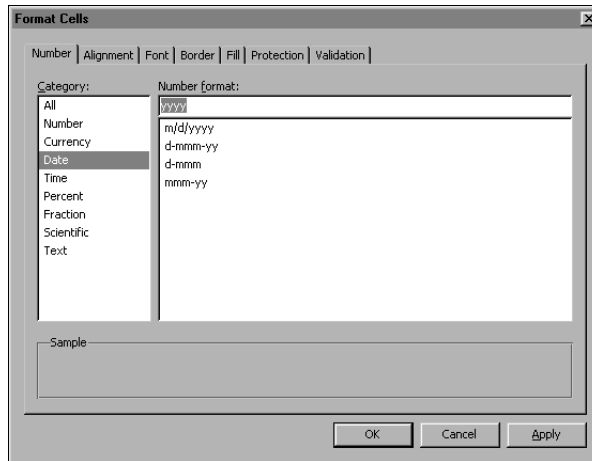


- 9 Save the report design.

## Task 6: Preview the report

In this task, you run the report and review its results.

- 1 Double the size of column C.



**Figure 1-58** Formatting a date cell



- 2 Choose Run. The report appears in View, as shown in Figure 1-59. Use the scroll bar to look through the report. As you scroll, notice the following:
  - The date cells are formatted to display dates by year, and product data is displayed for each year.
  - The MSRP and Purchase Price cells are formatted to display prices in the U.S. currency format.
  - Data is grouped by product line, such as Classic Cars, then product name group, such as 194 8 Porsche 356-A Roadster.
  - The names of the customers who purchased each product are displayed within the product name and line groups, along with the quantity they purchased, the Manufacturer's Suggested Retail Price and price they actually paid.

## Next steps

The next chapter, "Understanding report design," provides more detail about the report design process used in BIRT Spreadsheet Designer. It also provides more information about how to use the BIRT Spreadsheet Designer user interface.

Design		View									
E8		77									
	A	B	C	D	E	F	G	H	I	J	
1	MSRP and Purchase Price										
2	Product Line		2003				2004				
3	Classic Cars										
4	Product Name										
5	1948 Porsche 356-A Roadster										
6		Customer Name	Quantity	MSRP	Purchase	Quantity	MSRP	Purchase	Price		
7		Australian Collectors,	0	\$0	\$0	1	\$77	\$54			
8		Blauer See Auto, Co.	1	\$77	\$54	0	\$0	\$0			
9		Collectables For Less	1	\$77	\$54	0	\$0	\$0			
10		Cruz & Sons Co.	1	\$77	\$54	0	\$0	\$0			
11		Down Under Souvenir	0	\$0	\$0	1	\$77	\$54			
12		Euro+ Shopping Chat	1	\$77	\$54	1	\$77	\$54			
13		FunGiftIdeas.com	0	\$0	\$0	1	\$77	\$54			
14		Herkku Gifts	1	\$77	\$54	0	\$0	\$0			
15		L'ordine Souveniers	0	\$0	\$0	1	\$77	\$54			
16		Land of Toys Inc.	0	\$0	\$0	1	\$77	\$54			
17		Marseille Mini Autos	1	\$77	\$54	0	\$0	\$0			
18		Men 'R' US Retailers,	1	\$77	\$54	0	\$0	\$0			
19		Mini Auto Werke	1	\$77	\$54	0	\$0	\$0			
20		Mini Gifts Distributors	1	\$77	\$54	0	\$0	\$0			
21		Online Diecast Creati	1	\$77	\$54	0	\$0	\$0			
22		Oulu Toy Supplies, In	0	\$0	\$0	1	\$77	\$54			
23		Signal Gift Stores	0	\$0	\$0	1	\$77	\$54			
24		Toms Spezialitäten, L	0	\$0	\$0	1	\$77	\$54			
25		UK Collectables, Ltd.	0	\$0	\$0	1	\$77	\$54			
26		Vida Sport, Ltd	0	\$0	\$0	2	\$154	\$108			
		Sheet1									

**Figure 1-59** Formatted report results

# 2

## Understanding report design

This chapter contains the following topics:

- Planning a report
- Designing a report
- Distributing the report
- Understanding BIRT Spreadsheet Designer files

---

## Planning a report

An effective report design takes planning. Before beginning the design, it's useful to consider report specifics such as purpose, audience, data display, and data availability. For example, do you want managers and sales representatives to view the same data, or provide different views according to their roles and information needs? Do you want a sales report to display data for all regions combined or by individual regions or both? Do you want totals and subtotals to appear at the top of a data group or at the bottom? In addition, you must be sure that the data source you want to use can provide the necessary information. If one data source cannot adequately provide the data, how can you design the report to use multiple sources?

Before you design the report in BIRT Spreadsheet Designer, create a prototype or mock-up of your report design to help define how the report should look. Ask actual users to verify that the mock-up effectively presents the information they need. If you are replacing an older report with a BIRT Spreadsheet Designer report, designing a similar layout can help viewers use the new report with more confidence.

### Content considerations

In the context of spreadsheet report design, content means data. By the time you begin using BIRT Spreadsheet Designer, you should be able to answer the following questions about the content of your report:

- What is the purpose of the report and who will use it?  
Purpose is tied to audience, which, along with data availability, drives content selection and presentation. The more specific you are in determining purpose and audience, the more effective your design will be. For example, a report for high-level executives looking for data summaries uses and displays data differently than a report used by financial planners to perform a detailed analysis. In this situation, you could design a pivot range report for the financial analysts and a simple data range report for the executives.

There are other situations in which one report using view-time or run-time parameters can be designed to serve multiple audiences. For example, you can enable managers to view overall totals by region and representative, while allowing sales reps to view only their own data or data for their own region or office.

- What kinds of data should the report display?  
Typically, a report for financial analysts displays more varied data in more detail than a summary report. For example, in a global financial planning report, you could display monetary fluctuations in multiple currencies and with multiple values. You can display currency data by sum and percent, or by

a more complicated formula. You can display date information by month, quarter, and year.

- What is your data source and does it contain the data you need?  
A data source is the location from which BIRT Spreadsheet Designer retrieves the data that appears in the report. Supported data sources include relational databases, Actuate information objects, and other applications or spreadsheets.

After you identify the data source for a particular report, you should confirm that it contains the data you need. For example, does the database contain financial data for multiple currencies or just one? Does it contain data for the period you want to cover in the report? Can you get the information you need from one data source or will you need to use multiple ones? If you must use multiple sources, which ones and are they available for your use?

- How do you want or need to modify report data?  
BIRT Spreadsheet Designer provides several options for modifying report data, including the use of parameters, computed fields, and filters. A view-time parameter enables end users to type or select a value when opening a report. A run-time parameter requires users to type or select a value when they run a report. For example, you can require sales reps to select a sales region at view-time, or require that managers type in an ID number at run time.

If your data source does not contain exactly the data field you need, you can create a computed field that does. For example, if your data source does not contain a profit field, but does contain fields that, combined, can calculate profit, you can use those fields to create a computed field.

You can also use data filters that narrow or tailor report output to better suit audience needs. For example, you can filter date fields to return data from a specific day or period, or filter a list of products to return only results for particular products.

## Layout strategies

Layout impacts a report viewer's use and understanding of report data. In BIRT Spreadsheet Designer, you use row and column sections and groups to structure the report layout.

- Which data should you display in rows and which in columns?  
A spreadsheet typically displays name information, such as client and product names, by row, and calculated data such as currency amounts and quantities, in columns. You can use a mock-up to help determine which data to place in rows and columns.

It is also helpful to consider the use and placement of subtotal and grand total rows. Should the report display only subtotals or grand totals or both? And

should you place the subtotals before or after a data group? For example, should you display order totals for each product at the top row or bottom row for each product group?

- How should you group the data?

Data groups provide enormous flexibility in terms of data presentation and use. A hierarchical arrangement can be particularly useful. For example, a top-level group named All Regions could contain subgroups for the Northeast, Southwest, Upper Midwest, and South regions. You could use parameters or security features to allow access by group, so that sales managers could see data for all groups, while sales reps could see data only for their own group.

In a different scenario, you could create a top-level group called Portfolio Managers. Nested beneath this group, in hierarchical order, you could create a subgroup for each manager's clients, and beneath the client subgroup, another subgroup for the stock owned by each client. You could further group this data by country or office, and instruct BIRT Spreadsheet Designer to display data for different countries on different spreadsheets in the same workbook.

- How do you want to format the report?

Formatting can help report viewers identify data and data trends. For example, you can use a color format condition to highlight values over or under a certain amount, and use a conditional formula to call attention to sales reps whose monthly sales figures fall below a certain average.

In addition, you can add titles and column headings to the report and format them with different font styles, colors, borders, and other options to help viewers identify data groups. You can also reorganize a data range by adding or removing rows, columns, sections, and groups.

## Security considerations

BIRT Spreadsheet Designer provides several security options to restrict user access at different levels. For example, you can restrict a user's view by worksheet or by column, row, or cell. In addition, you can require a password to view a worksheet or a particular column or row in a worksheet.

You employ security options when designing the report. Before you begin, consider the following security questions:

- How do you want to restrict user access?

Do you want to restrict access simply by hiding a row or column or a group of rows or columns? Or do you want to require the entry of a password? Do you want to display calculated results, but hide the formula that created the result? Do you want to restrict access to certain operations, such as saving or deleting a workbook or a worksheet?



- Do you want to tie a user's role or login ID to a customized view of the report? The BIRT Spreadsheet Designer SmartSheet security option works by tying a user's role or login ID to a customized view of a report, which enables you to use the same report to provide different views to different users. You can restrict a user's view by placing conditions on a data set, a data range section, or a worksheet. These conditions, called grant expressions, are used to evaluate the user's security identifiers.

## Distribution strategies

You should also consider in advance how to distribute the finished report. For example, will you be e-mailing reports to a list, or publishing reports to an Actuate BIRT iServer System and using its built-in notification feature? Will users be viewing them from the web, or from an Excel file on a shared network drive?

Do you plan to use security features in your report design that require users to login to view the report, or to view part of a report? If so, how do you ensure that users get the login information they need?

---

## Designing a report

After you complete the planning phase, you can begin using BIRT Spreadsheet Designer. The tutorials in Chapter 1, "Building your first spreadsheet reports," showed you how to perform the primary tasks necessary to design a spreadsheet report. This section provides additional introductory information about each task. The remaining chapters in this book provide detailed instruction about how to implement each task using the BIRT Spreadsheet Designer interface and options.

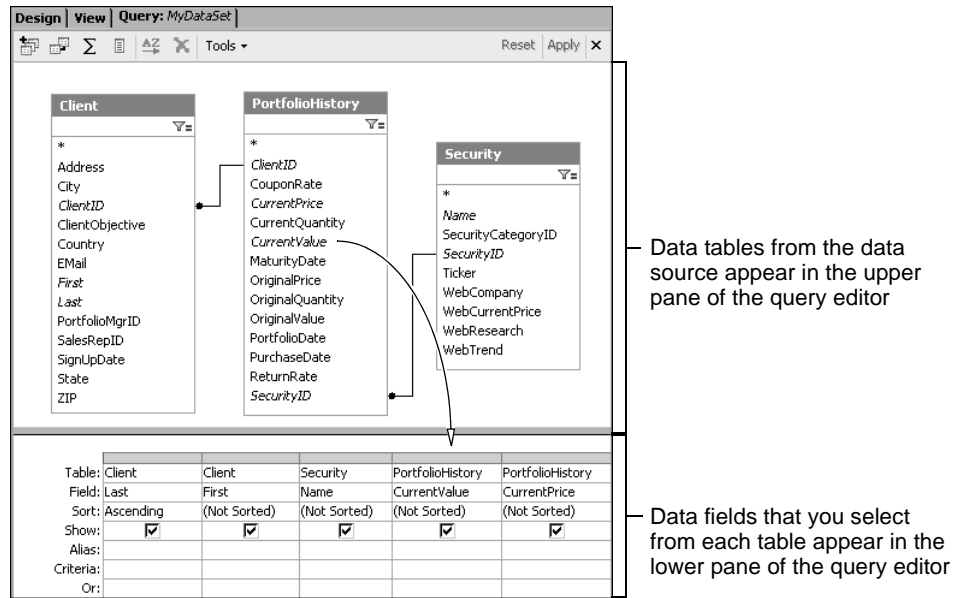
## Connecting to the data source

After you identify the appropriate data source during the planning process, you use the BIRT Spreadsheet Designer interface to create a data source connection. For more information about configuring data source connections, refer to the chapters in Part Two, *Accessing data*.

## Specifying report data

In BIRT Spreadsheet Designer, you use a data set and a query to specify the data to retrieve from the data source. A data set consists of tables, and a query, of columns from those tables. Data columns are also known as data fields. You can apply a filter condition and a sort order to query columns. You create the data set first, and from that, the query. The query refines your data set selections.

You use the BIRT Spreadsheet Designer query editor to select the data tables and design the query. Figure 2-1 shows how tables and columns appear in the query editor.



**Figure 2-1** Viewing an example query in the query editor, design view

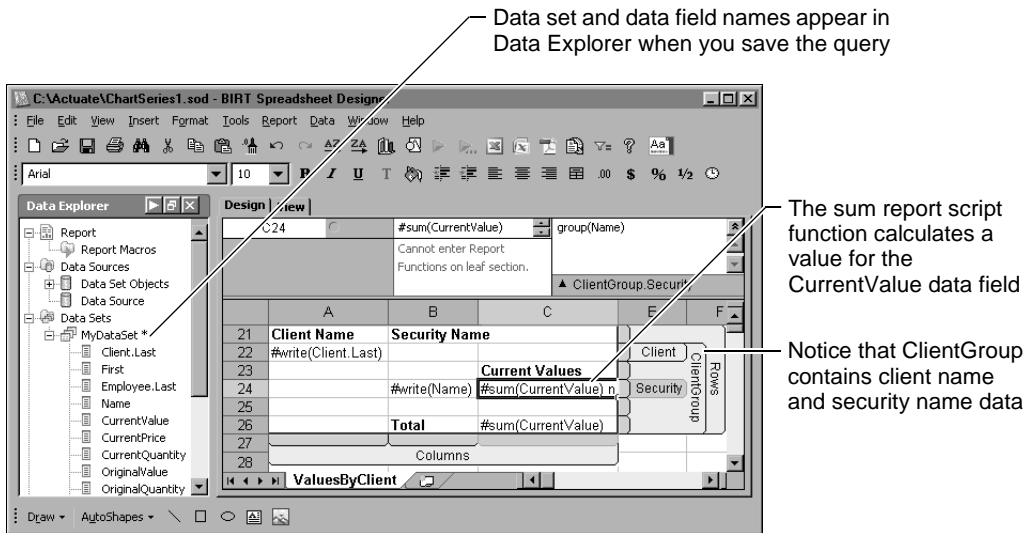
## Structuring report layout

In BIRT Spreadsheet Designer, you design the report layout in a data range, which provides the report's structure. A data range consists of row and column sections, which you organize into groups. After creating the data range, you add data set fields from the data set.

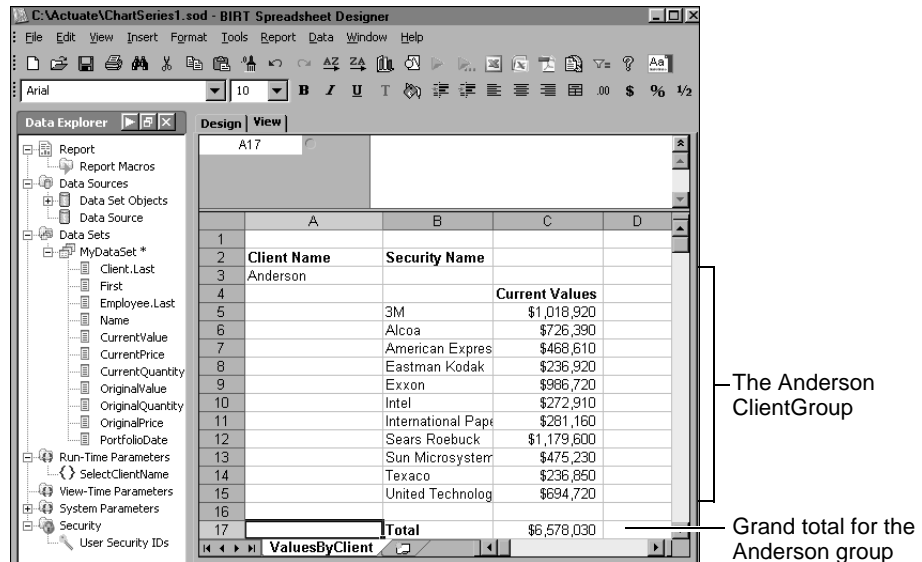
When you add data set fields, you specify a report script function for that data. For example, a typical report script function for a currency value is Sum, which instructs BIRT Spreadsheet Designer to display those values as sums. The BIRT Spreadsheet Designer report script function builder assists you in assigning report script functions to data set fields.

Figure 2-2 shows how a sample data range looks in the design editor after data set fields and functions have been added. The purpose of this sample report is to display the value of each security in a client's stock portfolio. Data is grouped by client name, and also by security name.

Figure 2-3 shows how the data range that appears in Figure 2-2 displays results on a generated report. Notice that security names are listed within the Anderson ClientGroup. Current values are summed for each security belonging to Anderson. A grand total for the Anderson group appears in cell C17.



**Figure 2-2** A sample data range as it appears in Design Editor



**Figure 2-3** A sample generated report

You can lay out the report design on a single worksheet or using multiple sheets. Like Microsoft Excel, a BIRT Spreadsheet Designer spreadsheet is contained in a workbook. A workbook initially consists of one worksheet, to which you can add more. BIRT Spreadsheet Designer treats each worksheet as an individual

spreadsheet. A workbook with multiple sheets can contain sheets that share data and report script functions or that are completely independent of one another.

In Chapter 8, “Formatting a report,” you learn a number of techniques for adding and manipulating sheets in a workbook.

## Viewing a report

You can check your progress as you design a report by viewing a spreadsheet executable file in BIRT Spreadsheet Designer. You can also view report output in Excel, or in Acrobat Reader. Viewing a report executable file shows you how the report output appears with live data and formatting.

## Using BIRT Spreadsheet Designer Run options

Use either of the following options to view a spreadsheet executable file in BIRT Spreadsheet Designer:



- Choose Report→Run to generate a spreadsheet report executable file. The report output appears on View, as shown in Figure 2-4.

Design | **View** Select View

L27

	A	B	C	D	E	F	G
1	<b>MSRP vs. Purchase Price</b>						
2	<b>Product Line</b>		2003				
3	Classic Cars						
4	<b>Product Name</b>						
5	1948 Porsche 356-A Roadster						
6		<b>Customer Name</b>	<b>Quantity</b>	<b>MSRP</b>	<b>Purchase</b>	<b>Quantity</b>	
7		Australian Collector	0	\$0	\$0	1	
8		Blauer See Auto, Co	1	\$77	\$54	0	
9		Collectables For Le	1	\$77	\$54	0	
10		Cruz & Sons Co.	1	\$77	\$54	0	
11		Down Under Souver	0	\$0	\$0	1	
12		Euro+ Shopping Ch	1	\$77	\$54	1	
13		FunGiftIdeas.com	0	\$0	\$0	1	
14		Herkku Gifts	1	\$77	\$54	0	
15		L'ordine Souvenirs	0	\$0	\$0	1	
16		Land of Toys Inc.	0	\$0	\$0	1	
17		Marseille Mini Autos	1	\$77	\$54	0	

**Figure 2-4** Examining report output on View



- Choose Report→Run with Parameters to generate a spreadsheet report that uses run- or view-time parameters. Requester Page appears for entry of parameter values. After you enter values on Requester Page, select OK to generate the spreadsheet report executable file.

If a report takes longer than 1.5 seconds to run, Report Progress appears. To stop report execution, select Cancel.

## Using View with Excel options



Select Report→View with Excel as 97-2003 workbook to create an XLS file and open the spreadsheet report in Excel 2003.



Select Report→View with Excel as 2007 workbook to create an XLSX file and open the spreadsheet report in Excel 2007.

When you generate a BIRT Spreadsheet Designer report in Excel, keep in mind that BIRT Spreadsheet Designer supports a larger number of rows and columns than Excel. Excel truncates the length and width of extremely large reports. Also, when you generate an Excel 2007 file (an XLSX or XLSM file), the exported output does not support the inclusion of drawing objects, such as auto shapes, forms, and pictures. For more information about how to use BIRT Spreadsheet Designer with Excel, see Appendix B, “Comparing Excel and BIRT Spreadsheet Designer.”

If you use Visual Basic for Applications (VBA) to add a report feature such as an action button that updates a database, you must preview the report in Excel. Because VBA only works in Excel files, the Excel preview option enables you to see how the VBA code works with your report design.

## Using the View as PDF option



Select View as PDF to create a portable document format (PDF) file and open the spreadsheet report in Acrobat Reader.

---

## Distributing the report

After you complete the report design, you generate a spreadsheet executable (.sox) file, which you publish then distribute as an instance file for end users to view.

The following list summarizes the steps necessary to distribute a report to end users:

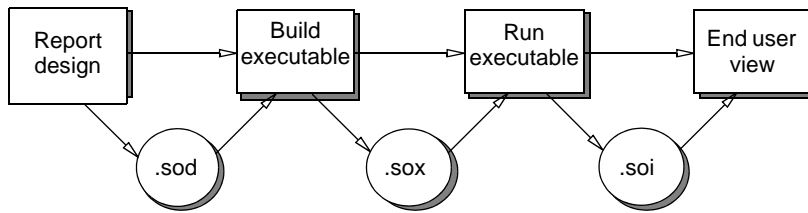
- Generate the spreadsheet executable (.sox) file from your design file.
- Publish the executable to a server, such as BIRT iServer or an application server.
- Run a job on the server to create a spreadsheet object instance (.soi) file.
- The end user uses Actuate Information Console or another application to view the instance file.

---

## Understanding BIRT Spreadsheet Designer files

BIRT Spreadsheet Designer creates and uses a number of files during the report development process. The first time you preview your work, BIRT Spreadsheet Designer creates a workspace folder to contain the files that it uses during the generation process. Typically, the folder contains cached data files, a copy of the generated report in Excel, and a run.soi file. Report design and executable files do not appear in the workspace folder. You can delete the workspace folder at any time.

Figure 2-5 illustrates the stages at which BIRT Spreadsheet Designer generates the primary spreadsheet report files you use.



**Figure 2-5** BIRT Spreadsheet Designer file generation process

The following list describes the main BIRT Spreadsheet Designer files you use:

- Spreadsheet object design (.sod) file  
The SOD file contains your report design.
- Spreadsheet object executable (.sox) file  
The preview process generates the SOX file. If you use callback classes or VBA code, the BIRT Spreadsheet Designer compiler searches for errors before creating the SOX.
- Spreadsheet object instance (.soi) file  
If you have the Actuate BIRT Spreadsheet Option licensed on an Actuate BIRT iServer System, BIRT iServer runs the SOX file and creates the SOI file. End users use the SOI to view the generated report.

# Part Two

---

**Accessing data**





## Accessing a data source

This chapter contains the following topics:

- About BIRT Spreadsheet Designer data sources
- Connecting to a data source
- About data sets
- Adding a computed field to a data set
- Combining multiple data sets
- Modifying a data set

---

## About BIRT Spreadsheet Designer data sources

To generate spreadsheets using BIRT Spreadsheet Designer, you connect to a data source to retrieve information instead of typing a formula into a cell. Connecting to a data source, such as a JDBC database or an Actuate Information Object, keeps the data accurate and up-to-date. After connecting to a data source, you create a data set and a query to specify the information the data source retrieves.

BIRT Spreadsheet Designer supports the following data sources:

- **Relational databases**  
To access data from a relational database such as Oracle, Microsoft Access, or Informix, you use an ODBC or JDBC driver.
- **Text files**  
You can use information stored in a delimited or fixed-width flat file to retrieve information.
- **Actuate information objects**  
An Actuate information object uses the Actuate Data Integration Service Connect to provide access to diverse data sources, including data streams. Information objects can access and combine information from databases, and XML files.
- **Custom data drivers**  
If you use a custom data driver, BIRT Spreadsheet Designer includes the driver in the list of available data sources when you set up a data source connection.

This chapter provides general information about connecting to and querying an external data source. Subsequent chapters provide specific information about connecting to and querying each type of data source. Table 3-1 lists the chapter associated with each type.

**Table 3-1** Data source types and their corresponding chapters

Data source type	Chapter describing the fields
Actuate Information Object and Data Integration Service connection	Chapter 5, "Accessing an Actuate information object"
Custom data driver	Chapter 3, "Accessing a data source"
Flat file source	Chapter 6, "Accessing data in a text file"
JDBC source	Chapter 4, "Connecting to a database or JDBC data source"

## Accessing multiple data sources

BIRT Spreadsheet Designer enables you to access data from different sources using the same report. For example, you can use a different data source when you publish a report than you do when you design it. You can also combine data from different sources or queries.

Typically, a report developer uses test data, rather than production data, when developing a report. On the server, the report must use the production data source. To use a different data source than the one specified in a report design, use a configuration file that migrates a report from one data source to the next. For more information about using a data source configuration file, see Chapter 2, “Migrating reports without changing the report design.”

## About the BIRT Spreadsheet Designer example databases

The screen images and instructional procedures in this manual, use the BIRT Spreadsheet Designer example databases to demonstrate product use. The following list describes the sample databases:

- Classic Models contains customer and sales information about the purchase and sale of collectible model cars, trucks, motorcycles, and boats.
- Financial Statement contains financial data divided by operational codes. Use this database to practise using multiple rows and column groups and outlining.
- Grocery contains sales and product information for a medium-size grocery chain. Use this database to learn how to develop data ranges that use data from data sources that have hierarchical data structures.
- Securities contains client, employee, price and quantity information about the growth of client stock portfolios over time.
- Sales (sfdata) contains sales forecast data for a fictitious manufacturing company.

The example databases install with the BIRT Spreadsheet Designer product. All of the data is fictional and designed to support report designers in learning how to use BIRT Spreadsheet Designer.

---

## Connecting to a data source

To create a connection to an external data source, you first specify the source and provide values for the properties BIRT Spreadsheet Designer uses to create the connection. For example, you provide a file name for a file data source or login information for a database. You then create a data set and specify the data fields

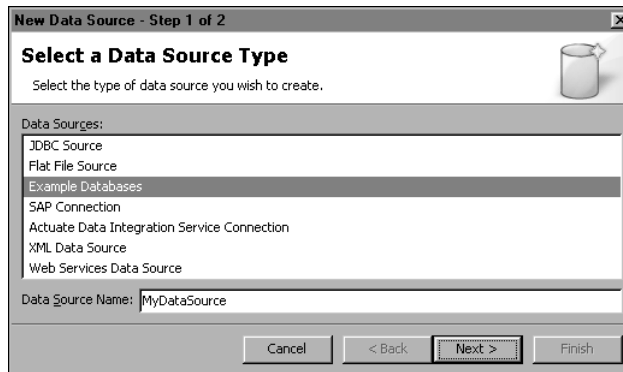
to extract from the source. For more information about creating data sets, see “Creating a data set,” later in this chapter.

When additional properties are available for a data source, you can view and edit them after setting up the connection. The number and kind of additional properties vary according to the source type.

## Creating a data source connection

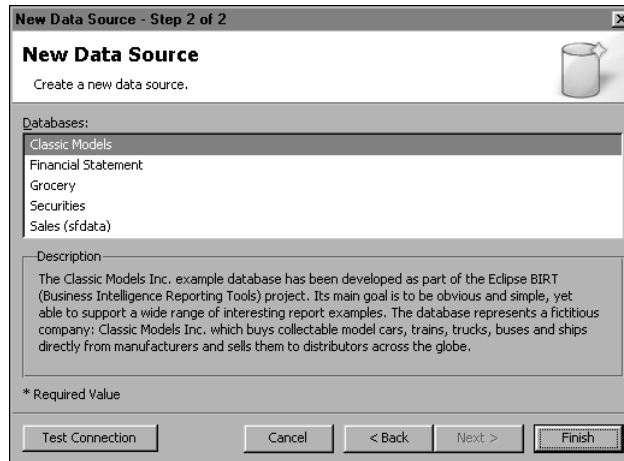
The following procedure uses a sample database to illustrate the steps you take to connect to a data source. The steps for accessing other types of data sources are similar.

- 1 In Actuate BIRT Spreadsheet Designer, choose Report→Create Data Source.
- 2 On New Data Source—Step 1 of 2, take the following steps, then choose Next:
  - 1 In Data Sources, select the type of data source to use. Figure 3-1 shows Example Databases selected.
  - 2 In Data Source Name, type a name for the data source.



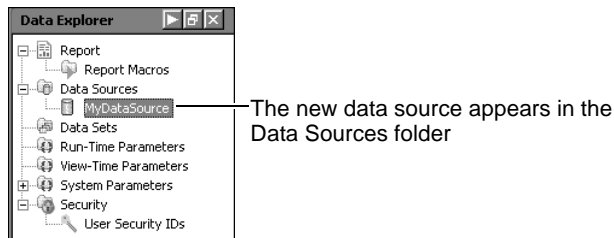
**Figure 3-1** Step 1 in connecting to a data source

- 3 On New Data Source—Step 2 of 2, perform the following steps:
  - 1 Provide the necessary values to establish a connection using the source type you selected. The fields available on New Data Source—Step 2 of 2 vary by source type. If you choose Examples Databases as a source, you select a sample database on New Data Source—Step 2 of 2, as shown in Figure 3-2.
  - 2 Use the Test Connection button to confirm the connection you are creating.



**Figure 3-2** Step 2 in connecting to a data source

- 4 Choose Finish. The data source that you configured appears in Data Explorer, as shown in Figure 3-3.



**Figure 3-3** Data source appears in Data Explorer

## Modifying a data source connection

To edit data source driver information or properties, right-click a data source name in Data Explorer and choose Edit from the context menu.

To delete a data source, right-click a data source name in Data Explorer and choose Delete from the context menu.

To rename a data source, right-click a data source name in Data Explorer, choose Rename from the context menu, type the new name, and then press Enter.

## About accessing additional types of data sources

A report developer can access a variety of a data sources using predefined data drivers. To provide access to other types of data, a Java programmer can create a custom data driver. Actuate supports drivers defined by the open data access (ODA) extension points in the Eclipse Data Tools Platform (DTP). A report

developer chooses the data source from a list of possible types of data sources in the Actuate design tool. The list does not distinguish between custom and native data sources. When a report developer chooses a data source type, the ODA driver provides a design-time user interface for querying the data source.

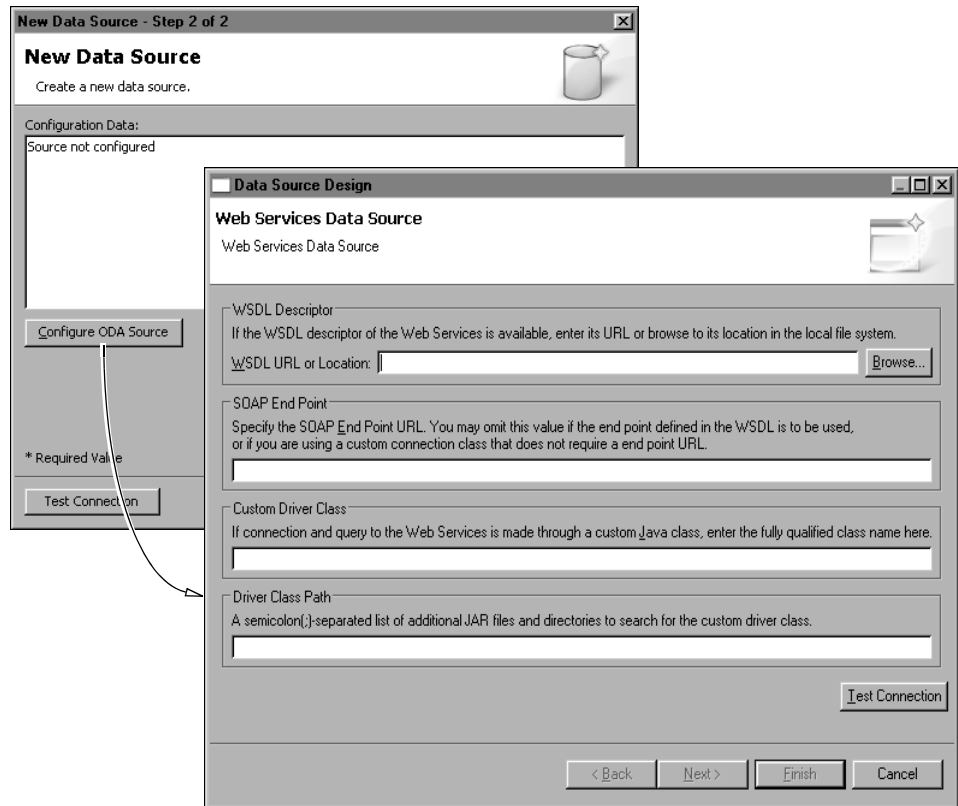
An ODA driver supports both design-time and run-time functionality. The Actuate design tool uses the driver to build a connection to the data source, retrieve parameter and data row definitions, and compile these definitions for use at run time. At run time, Actuate BIRT iServer loads the driver. Then, the driver creates the connection and data source instance and fetches the requested data. Each ODA driver supports one type of connection and can support multiple instances of that connection type. Each driver can support multiple data sources. The driver must be installed in the client environment where you design the report and on Actuate BIRT iServer where you run the report.

## **Using a custom data source type in a report design**

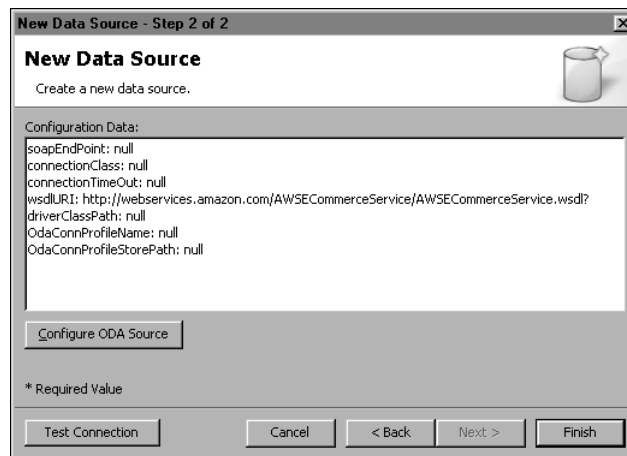
A custom data source type appears in the list of available data source types when you choose to create a new data source. The second step in creating a data source of a custom type is the same for all custom data source types. The data source wizard provides the ability to configure the data source by choosing **Configure ODA Source**. This choice launches the user interface for the particular data source. Figure 3-4 shows the interface for the WSDL data source that BIRT Spreadsheet Designer provides. The XML data source provides a user interface for configuration in the same way. After you provide the configuration values for the data source, you choose **Finish** on the custom interface. The configuration values appear in **New Data Source**, as shown in Figure 3-5. You can then use this data source to create a data set, as described later in this chapter.

## **Developing a custom data driver**

To develop a custom data driver, you must create Eclipse plug-ins that implement the DTP ODA extension points. For information on these extension points, read the documentation provided for the Data Tools Platform on the Eclipse web site, <http://www.eclipse.org>. The book, *Integrating and Extending BIRT*, published by Addison-Wesley, provides fully worked examples of developing an ODA driver. You can download the source code for the ODA driver extension examples from <http://www.actuate.com/birt/contributions>. Both BIRT Spreadsheet Designer and an Actuate BIRT iServer System access custom data drivers located in the `\Program Files\Actuate11\oda\eclipse\plugins` folder. To deploy an ODA driver to the BIRT Spreadsheet Deployment Kit, place the driver plug-ins in the application's `WEB-INF\platforms\dropins\eclipse\plugins` folder.



**Figure 3-4** Launching the user interface to configure an ODA WSDL data source



**Figure 3-5** Configuration values for an ODA WSDL data source

---

## About data sets

After setting up a data source connection, you must create a BIRT Spreadsheet Designer data set. A data set specifies the data to retrieve from a data source. For example, when you create a data set from a JDBC data source, you specify the database tables. When you create a data set from a delimited text file data source, you specify the delimiting character that separates the fields.

After you create the data set, you select fields from the data source to define data set contents. For example, when connecting to a relational database, you use a SQL query to select the data fields to use from the tables in the data set.

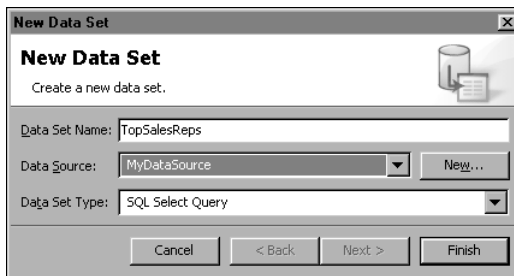
A BIRT Spreadsheet Designer workbook can contain a single data set, or multiple ones. You can use multiple data sets to create a joined data set, or you can use fields from each set as you would from a single set. For more information about using a joined data set in a report design, see “Combining multiple data sets,” later in this chapter.

## Creating a data set

The following procedure uses a relational database to illustrate the process of creating a data set. Though the process of creating a data set is similar for each type of data source, the procedure for selecting fields varies according to data source. BIRT Spreadsheet Designer provides a user interface for each of the built-in data sources. The interface for the data source type appears when you finish creating the data set. Custom data sources, described earlier in this chapter, provide their own interfaces.

### How to create a data set

- 1 In BIRT Spreadsheet Designer, choose Report→Create Data Set. Alternatively, you can right-click Data Sets in Data Explorer, and choose Create Data Set from the context menu.
- 2 On New Data Set, in Data Set Name, type a name for your data set, as shown in Figure 3-6.

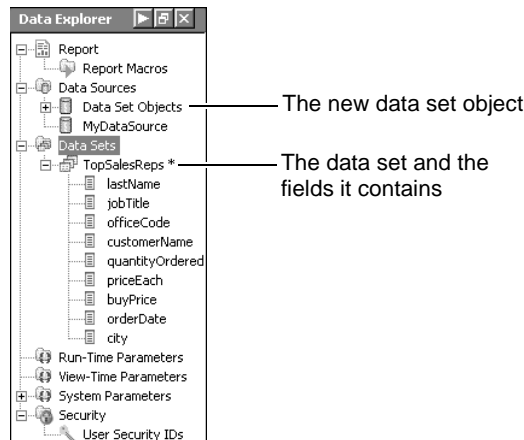


**Figure 3-6** Creating a new data set



- 3 In Data Source, select the data source.
- 4 In Data Set Type, select the type of data set to create.
- 5 Choose Finish. The name of the new data set appears in the Data Explorer, under Data Sets. BIRT Spreadsheet Designer displays a dialog or pane in which to select data set fields, depending on the type of data source you chose in step 3. Because this sample procedure uses a JDBC connection as its data source, the Show Tables dialog appears next. For instructions on creating a query for a JDBC connection, see Chapter 6, “Creating a database or JDBC query.”

After you have selected the data set fields, a plus sign appears next to the new data set in Data Explorer. Choose the plus sign to expand the list of fields contained in the new data set, as shown in Figure 3-7. The data set object for the data set appears in the Data Set Objects folder of Data Explorer. You can use the data set object when creating a joined data set of two or more data sets.



**Figure 3-7** Data Explorer displaying the new data set

## Previewing data set results

After you create the data set and query, you can preview query results before publishing the report. The preview option enables you to verify that the query returns the data you need.

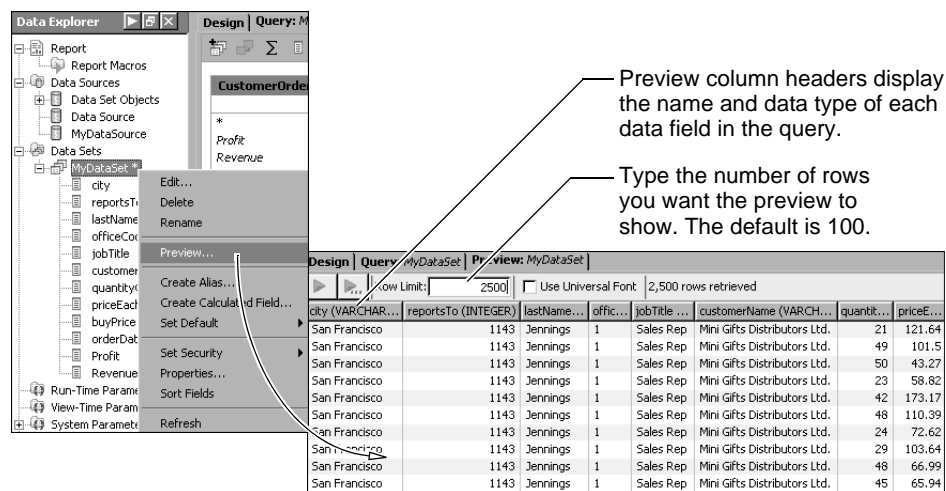
To preview query results, right-click a data set in Data Explorer, and select Preview, as shown in Figure 3-8. Query results appear on the Preview tab, also shown in Figure 3-8. After previewing, you can use the query editor to change the data set or query, or choose one of the Run report options to view the generated report.

You can perform the following actions in Preview mode:

- To modify the number of rows returned, type a new value in Row Limit.

- To display rows that contain Unicode values, select Use Universal Font.
- To refresh the current preview, choose Refresh the preview using the current parameter values.
- To preview the data set using parameter values, choose Refresh the preview using new parameter values.
- Close Preview.

After you preview a query, you can return to the query editor to refine the query, or you can run the report.



**Figure 3-8** Previewing data from a data set

## Adding a computed field to a data set

A computed field is a data field that you create to perform a calculation or action that cannot be performed by a field already in the data source. A computed field is also known as a calculated field.

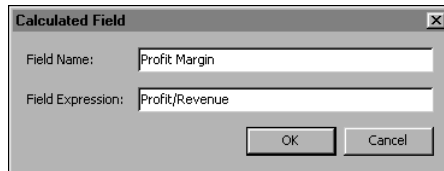
You can use the following elements to create a computed field:

- Any field in the data set
- Any mathematical or logical expression and operator that the spreadsheet engine can interpret

You can create a computed field using Data Explorer. For a JDBC data source, you can also use the query editor. This section provides instruction about using Data Explorer to create a computed field. For instruction about using the query editor to do so, see Chapter 6, “Creating a database or JDBC query.” When you create a

computed field in Data Explorer, you can use it in a data range but not in a pivot range.

To create a computed field in Data Explorer, right-click the data set, and choose Create Calculated Field from the context menu. You type a name and an expression for the calculated field. Figure 3-9 shows the use of two fields from a data set, Revenue and Profit, to create a computed field called Profit Margin.



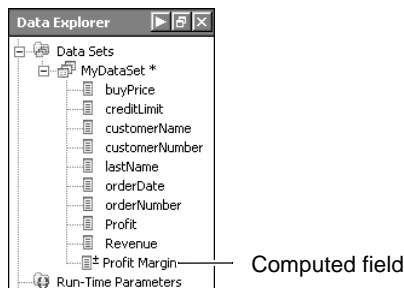
**Figure 3-9** Creating a computed field

In this example, the expression for Profit Margin divides Profit by Revenue. You can also use more complex and conditional expressions. For example, to display A if CustomerCredit is greater than 600 and B if not, use the following expression:

```
if (CustomerCredit>600, "A", "B")
```

Though you cannot use aggregate expressions, such as sum, in a computed field, you can use most other standard spreadsheet operators.

After you create a computed field, it appears in Data Explorer with the other fields in the data set. A plus and minus sign differentiate a computed field from the other data fields in the set, as shown in Figure 3-10. You add a computed field to a data range and assign a report function to it just as for any other data field.



**Figure 3-10** Computed field in data set

To edit a computed field, right-click its name and choose Edit from the context menu.

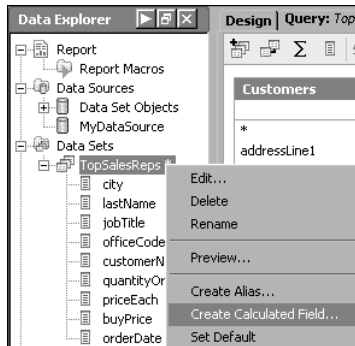
To delete a computed field, right-click its name and choose Delete from the context menu.

To rename a computed field, right-click the computed field's name and choose Rename from the context menu, type the new name, then press Enter. Any

expressions that use the field now contain a question mark. To correct this problem, retype the field name or drag and drop the renamed field as necessary.

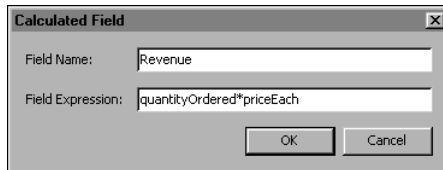
### How to create a computed field

- 1 In Data Explorer, right-click the name of the data set and choose Create Calculated Field from the context menu, as shown in Figure 3-11.



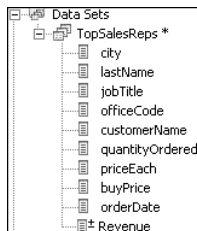
**Figure 3-11** Selecting Create Calculated Field

- 2 On Calculated field, type a name and a field expression, as shown in Figure 3-12. Then choose OK.



**Figure 3-12** Creating the revenue field

The computed field now appears in the data set, as shown in Figure 3-13.



**Figure 3-13** The computed field, Revenue

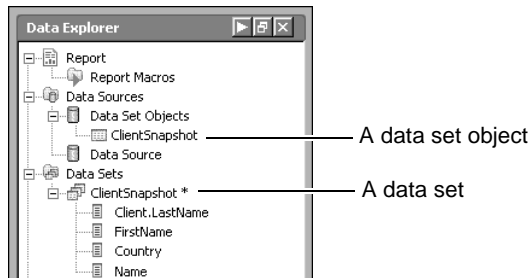
---

## Combining multiple data sets

You can use multiple data sets in a report design to retrieve data from multiple sources. For example, you could create three data sets that retrieve data from three different sources: an information object, an Oracle database, and a flat file. After creating the individual data sets, you use data set objects to form a joined data set. The joined data set acts as a query to retrieve data from all three data sources. A BIRT Spreadsheet Designer joined data set resembles a Microsoft Access make-table query, which makes a new, independent table by retrieving a subset of fields from an existing table. The make-table query improves efficiency because you can reuse rules applied on an existing table to create a new table.

### About data set objects

Every time you create a data set, BIRT Spreadsheet Designer creates a corresponding data set object that appears in Data Explorer. The data set object appears in the list of data sources and contains cached data from the data set. You use the cached data to create a joined data set combining data from multiple sources. Figure 3-14 shows a data set and a corresponding data set object, both called ClientSnapshot.



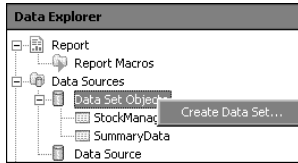
**Figure 3-14** Data Explorer showing a data set and data set object

### Creating a joined data set

When you create a joined data set, you choose a data set object as your data source. The contents of the combined data sets then appear in the query editor. You create the query by selecting items from the data set contents. A joined data set uses the same query editor as a data set created with a JDBC data source.

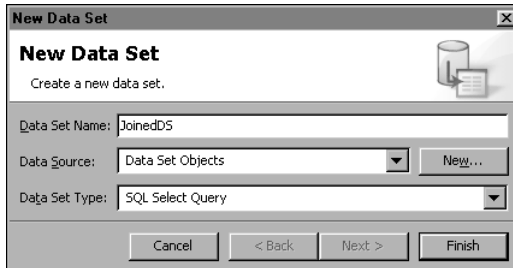
#### How to create a joined data set

- 1 In Data Explorer, under Data Sources, right-click Data Set Objects and choose Create Data Set, as shown in Figure 3-15.



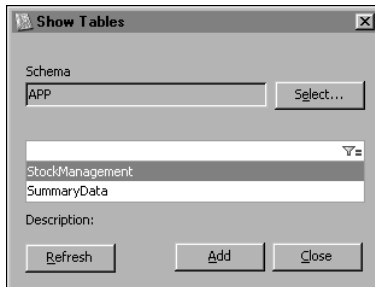
**Figure 3-15** Creating a joined data set from data set objects

- 2 On New Data Set, name the joined data set, and ensure that Data Set Objects is selected as the data source, shown in Figure 3-16. Then choose Finish.



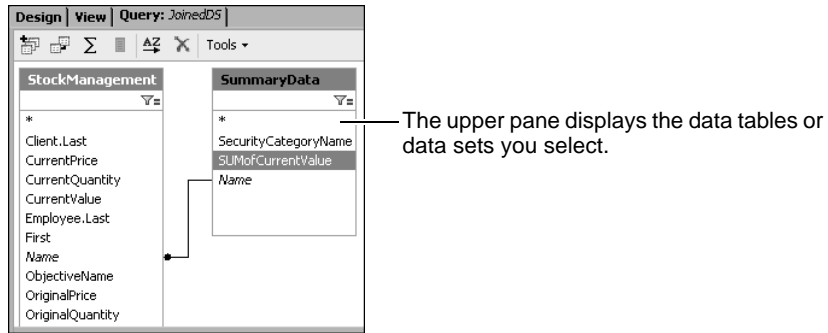
**Figure 3-16** Using a data set object as a data source

Show Tables appears, as shown in Figure 3-17. Notice that the Show Tables dialog displays the names of both data set objects.



**Figure 3-17** Specifying data set objects as tables

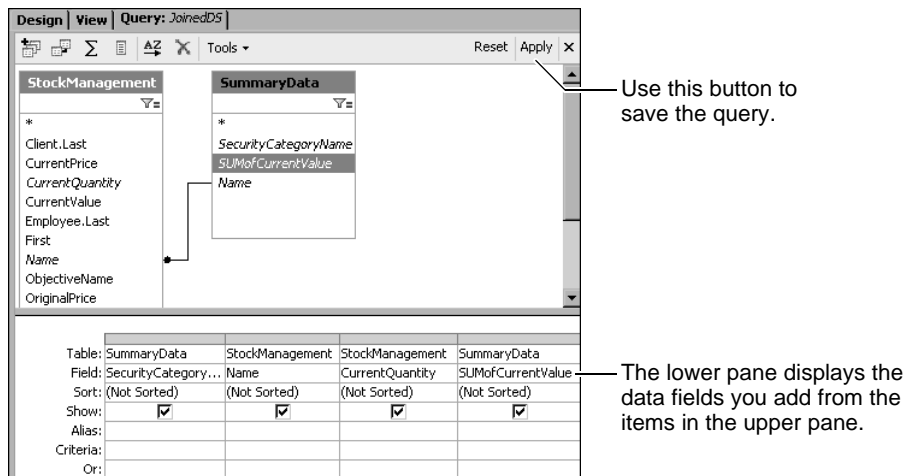
- 3 On Show Tables, select the data set or data sets from which to select data fields. Choose Add, then choose Close. The data fields contained in each data set appear in the upper pane of the query editor. The data sets are joined on a common field, as shown in Figure 3-18.



**Figure 3-18** The query editor after adding data sets

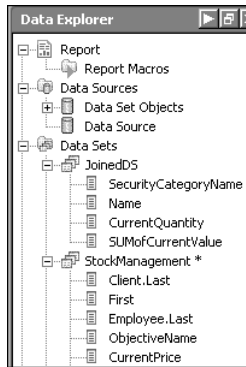
At least one join must exist between the data sets. For more information about using and creating joins, see Chapter 6, “Creating a database or JDBC query.”

- 4 To create the query, double-click the data fields to use in the report design. The data fields then appear in the lower pane, as shown in Figure 3-19.



**Figure 3-19** Selecting data sets

- 5 To save the query, choose Apply. The joined data set now appears in Data Explorer, as shown in Figure 3-20.



**Figure 3-20** Viewing the joined data set

You can further refine this type of query by adding filter and sort conditions. For more information about filtering and sorting a query on a joined data set, see Chapter 6, “Creating a database or JDBC query.”

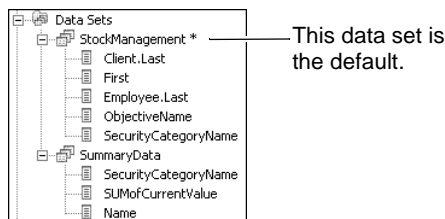
## Modifying a data set

To rename a data set, right-click the data set name in Data Explorer, then choose Rename. The name becomes an editable field. Type the new name, then press Enter.

To edit an existing query, in Data Explorer, right-click the data set and choose Edit. The relevant dialog box or pane appears in which you can edit the query for that data source type.

## Defining a default data set

If your report design contains multiple data sets, you can identify one as the default data set for a data range, worksheet, or workbook. If you do not specify a specific data set to use as the default when multiple data sets exist, BIRT Spreadsheet Designer applies the default status to the first data set you created. A lone data set in a report design is always the default data set. An asterisk denotes the default data set for the current worksheet, as shown in Figure 3-21.

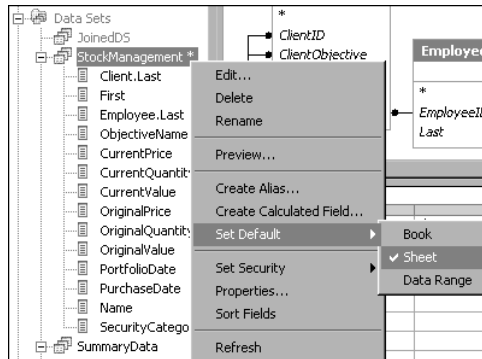


**Figure 3-21** The default data set



## How to change the default data set

- 1 In Data Explorer, right-click on the data set, and select Set Default, as shown in Figure 3-22. Three setting options appear.



**Figure 3-22** Setting the default position

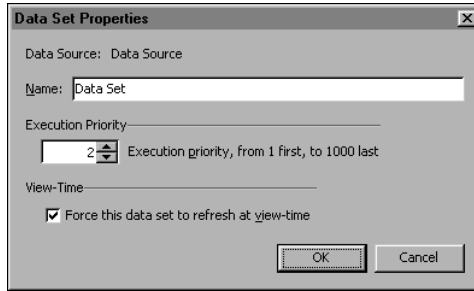
- 2 Choose one of the three available settings, shown in the following list. BIRT Spreadsheet Designer applies the setting you select.
  - **Book**  
Choose this setting to make the data set the default data set for the entire design file.
  - **Sheet**  
Choose this setting to make the data set the default data set for the current worksheet. In the example shown in Figure 3-22, Sheet is selected.
  - **Data Range**  
Choose this setting to make the data set the default data set for the currently selected data range.

## Running data set queries in priority order

In some report designs having multiple data sets, the rows that one data set returns depend on values provided by another. In this case the first data set query must run before the dependent query.

To specify the order in which data set queries run, set an execution priority value for each data set. A data set query executes in order from highest, 1 to lowest, 1000, priority value. The default execution priority value is 500. Multiple data sets with priority value 500 execute concurrently or, in unspecified order. Execution priority values affect only the order of run-time queries.

Figure 3-23 shows DataSet with execution priority value 2, and set to refresh at view time.



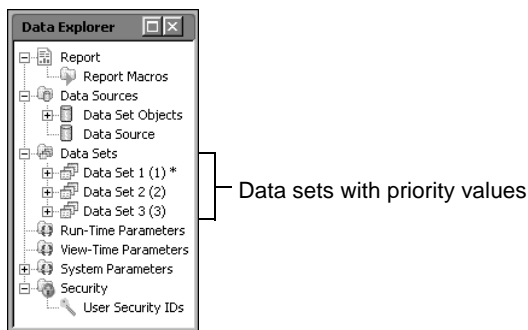
**Figure 3-23** Setting data set properties

### How to set the execution priority property for a data set

- 1 In Data Explorer, under Data Sets, right-click a data set name, then choose Properties.
- 2 On Data Set Properties, under Execution Priority, type or use the up or down arrow to select a priority value between 1 and 1000.

The execution priority value that you define for a data set query appears in Data Explorer, enclosed in parentheses at the right side of the data set name.

The example report design shown in Figure 3-24 contains 3 data sets. A priority order value appears in Data Explorer next to each data set name. Data Set 1, the default data set in this example, has been assigned the highest priority value, 1. Queries for data set 2 and data set 3 run second and third, respectively.



**Figure 3-24** Multiple data sets with priority order set

## Forcing a data set to refresh at view time

BIRT Spreadsheet Designer typically activates a query at run time, when you run an executable file on a report server. Users who open an instance of a published report see the cached data from when the server ran the report. Alternatively, you can specify that a query run at view time, when a user opens a view of a report.

Use the view-time option if you anticipate changes to report data between run time and view time. For example, you can design a report with two data sets, one that contains historical data, and another that contains year-to-date data. Refreshing the year-to-date data at view time ensures that the most recent changes to report data are available to end users. The execution priority of a data set does not affect the order in which view-time queries run. Instead, data cache dependency determines the execution order of view-time queries.

**How to force a data set to refresh at view time**

- 1** Right-click a data set name, then choose Properties.
- 2** On Data Set Properties, select Force this data set to refresh at view-time, then choose OK.

BIRT Spreadsheet Designer refreshes the data set at view time, and at run time.



# Connecting to a database or JDBC data source

This chapter contains the following topics:

- Accessing a database
- Connecting to a database or other JDBC data source
- Accessing data using a SQL query
- Creating a SQL query using BIRT Spreadsheet Designer
- Creating a computed field using the query editor
- Summarizing row data in the query editor
- Filtering query results
- Modifying a query
- Accessing a stored procedure
- Creating a data set that uses a stored procedure call

---

## Accessing a database

Using BIRT Spreadsheet Designer, you can access information that is stored in a relational database, such as Oracle, Informix, or Microsoft Access. You also can access data from other JDBC data sources.

To access data from databases or other JDBC data sources, create a BIRT Spreadsheet Designer JDBC data source that provides connection information. After creating the JDBC data source, you can use either a SQL query or stored procedure to query the data.

---

## Connecting to a database or other JDBC data source

Java Database Connectivity (JDBC) provides an interface for BIRT Spreadsheet Designer to access data from a database using a JDBC driver. BIRT Spreadsheet Designer supports JDBC 2.0 compliant type 2, 3, and 4 drivers, and provides drivers for the following commonly used database types:

- Generic
- DB2
- Informix
- Oracle
- SQL Server
- Sybase

BIRT Spreadsheet Designer can also access other databases for which you provide connection classes. Before connecting to a database, confirm that BIRT Spreadsheet Designer can access the database driver. BIRT Spreadsheet Designer indicates when a driver is not available.

To use a JDBC type 4 database driver from BIRT Spreadsheet Designer, copy the driver file to the `\spreadsheet\extensions\` folder. For example, to use the Oracle JDBC driver, `odbc14.jar`, you must copy `odbc14.jar` to `\spreadsheet\extensions\`.

JDBC type 2 and 3 drivers require additional configuration. For information about using a JDBC type 2 or 3 driver, refer to your database driver documentation. Starting BIRT Spreadsheet Designer adds all files in the extensions folder to the class path in alphabetical order.

To access the database driver from Actuate BIRT iServer, place the driver archive (.jar) file in `C:\Program Files\Actuate11\iServer\reportengines\engines\ess\lib`. To access the database driver from Actuate BIRT Spreadsheet Deployment Kit, place the driver JAR file in `<context root>\WEB-INF\reportengines`

\engines\ess\lib in the web archive (WAR) file. After placing the JAR file in this folder, you must restart the application server.

## Creating a JDBC data source connection

When you choose a JDBC database as a data source, you supply a name for the source and connectivity information, such as the name of the driver class and the connection URL. Depending on the JDBC database you select, you may also need to specify host, port, database names, and other information.

As well as these standard database connection properties, JDBC databases can support additional properties. The number and kind of additional properties available for your JDBC data source vary according to the data source type and structure. You can view and change additional properties as you create the connection. For information about editing a JDBC connection, see “Modifying a JDBC data source connection,” later in this chapter.

Table 4-1 lists the driver classes and database URL patterns for the JDBC drivers provided by Actuate Corporation. An asterisk indicates a required field.

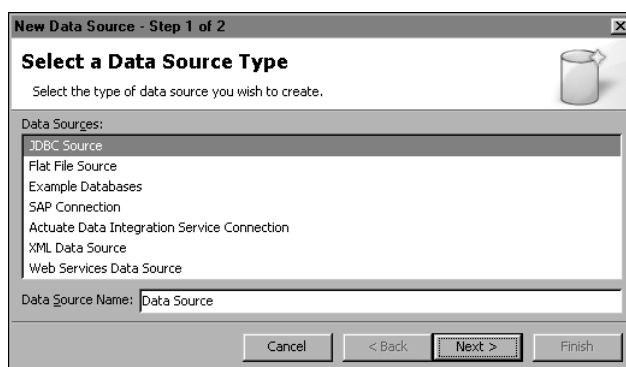
**Table 4-1** Driver classes and their URL patterns

Type	Driver class	URL pattern
DB2	com.actuate.actuatedd.jdbc.db2.DB2Driver	jdbc:actuate:db2:// {[host*]}{:[#port=50000]} {;databaseName=[database*]}
Informix	com.actuate.actuatedd.jdbc.informix.InformixDriver	jdbc:actuate:informix:// {[host*]}{:[#port=1533]} {;databaseName=[database*]} {;INFORMIXSERVER=[servername*]}
Oracle	com.actuate.actuatedd.jdbc.oracle.OracleDriver	jdbc:actuate:oracle:// {[host*]}{:[#port=1521]}{;SID=[sid*]}
SQL Server	com.actuate.actuatedd.jdbc.sqlserver.SQLServerDriver	jdbc:actuate:sqlserver:// {[host*]}{:[#port=1433]} {;databaseName=[database*]}
Sybase	com.actuate.actuatedd.jdbc.sybase.SybaseDriver	jdbc:actuate:sybase:// {[host*]}{:[#port=5000]} {;databaseName=[database*]}

### How to create a JDBC data source

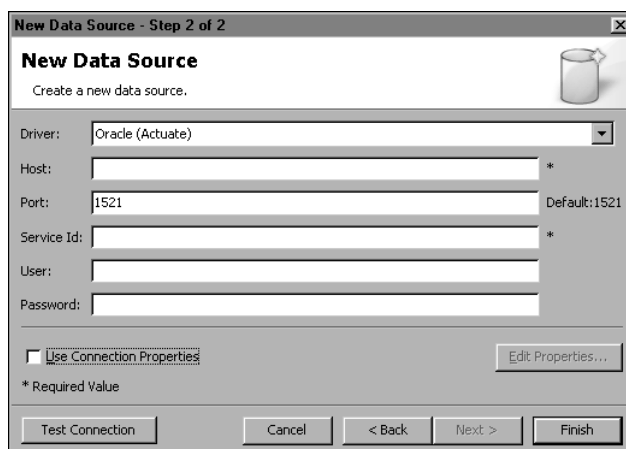
- 1 In Actuate BIRT Spreadsheet Designer, choose Report→Create Data Source.
- 2 On New Data Source—Step 1 of 2, take the following steps, as shown in Figure 4-1. Then choose Next.
  - 1 In Data Sources, select JDBC Source.

- 2 In Data Source Name, type a name for the data source.



**Figure 4-1** Selecting JDBC Source for your data source

- 3 On New Data Source—Step 2 of 2, take the following steps:
  - 1 In Driver, select the type of JDBC source to which to connect. In Figure 4-2, an Oracle database is selected. The other fields on the Step 2 of 2 dialog vary according to the type of JDBC source you select.

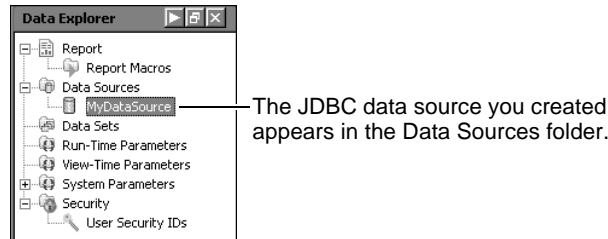


**Figure 4-2** Specifying a new JDBC data source

- 2 Complete the other fields as appropriate. Fields that typically appear on New Data Source—Step 2 of 2 include Host, Port, and database URL.
- 3 To view and set additional properties, select Use Connection Properties. Edit Properties becomes active. When you choose Edit Properties, JDBC Connection Properties appears. Change the available fields as needed, then choose OK to return to New Data Source—Step 2 of 2.
- 4 Choose Test Connection to confirm a connection to the source you are configuring.



- 4 Choose Finish. The data source appears in Data Explorer, as shown in Figure 4-3.

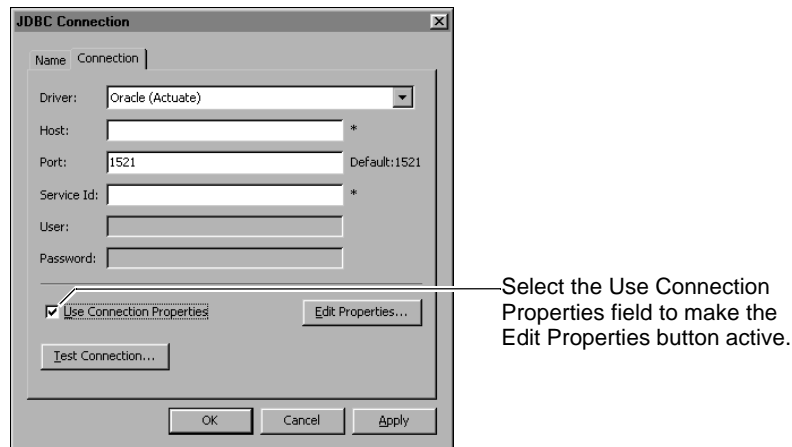


**Figure 4-3** JDBC data source appears in Data Explorer

You can now create a data set that uses a SQL query or a stored procedure to access data from the JDBC data source. For information on creating a JDBC query, see Chapter 6, “Creating a database or JDBC query.”

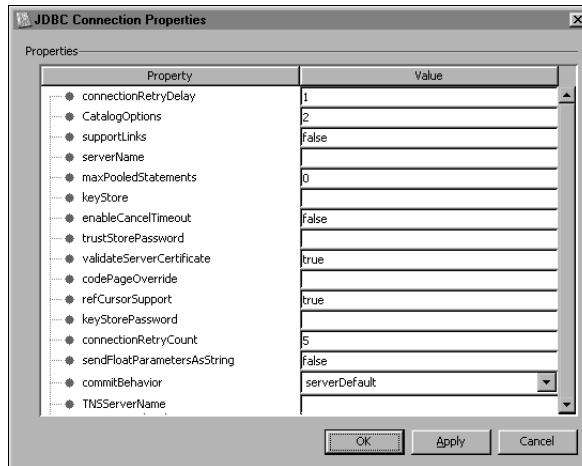
## Modifying a JDBC data source connection

To edit data source driver information, right-click the data source’s name and choose Edit from the context menu. JDBC Connection appears. The fields available vary according to the JDBC source type you use. Figure 4-4 shows the fields on JDBC Connection for an Oracle data source.



**Figure 4-4** Editing a JDBC connection for an Oracle database

To view, edit, and apply additional properties, select Use Connection Properties, as shown in Figure 4-4. Edit Properties becomes active. When you choose Edit Properties, JDBC Connection Properties appears. The available fields vary according to source type. Figure 4-5 displays the properties for an Oracle connection. Change the fields as needed, and choose OK to return to JDBC Connection, where you again choose OK to apply your changes.



**Figure 4-5** Viewing additional properties for an Oracle connection

To rename a data source, right-click the data source's name in Data Explorer and choose Rename from the context menu.

To delete a data source, right-click the data source's name in Data Explorer and choose Delete from the context menu.

## Adding a custom JDBC driver

To add a custom JDBC driver to BIRT Spreadsheet Designer, you place a driver JAR file into the spreadsheet\extensions folder. By default, to access this driver when you create or modify a data source, you select Generic JDBC for the driver and type in a driver class.

BIRT Spreadsheet Designer supports prompting the report developer for information about a custom JDBC data source when creating or modifying that source. To set up the parameters for the custom JDBC data source, you provide a template in the drivers.xml file in the BIRT Spreadsheet Designer installation directory.

In drivers.xml, for each new driver, add a new driver.template element within the jdbc.templates element. The driver.template element has the following syntax:

```
<driver.template
  driver="driver class"
  name="driver name"
  template="URL template">
  <prompt
    name="variable name"
    prompt="prompt text"
  />
</driver.template>
```

Table 4-2 lists and describes the attributes for driver.template. Attribute names and values are case-sensitive.

**Table 4-2** JDBC attributes

Attribute	Description
driver	The fully qualified class name. This class must be available in a driver JAR file in spreadsheet\extensions.
name	The name of the JDBC database type. This name appears in the list of drivers in the user interface for creating or modifying a JDBC data source.
template	A template for the URL to connect to the database and the fields to display on the data source user interface.

The template attribute builds the URL to access the database. The template can contain variables for which the report developer provides values. BIRT Spreadsheet Designer generates a custom user interface for these variables when the developer selects the custom driver.

When specifying a template attribute, use the following syntax:

```
<URL text>{<prefix>[<variable name>=<default> <required>]suffix}
```

where

- URL text is the static part of the URL to access the database.
- Prefix is optional text to include in the URL if the report developer types a value for the variable.
- Variable name is the name of the variable in the URL. This name appears as prompt in the user interface. Prefix the variable name with a # character to indicate an integer value.
- Default is the optional default value for the variable. This value appears on the user interface.
- Required is an optional field. To indicate the report developer must type a value for the variable, type an asterisk (\*) here.
- Suffix is optional text to include in the URL if the report developer types a value for the variable.

For example, the following code is the template attribute for the DB2 driver that Actuate provides:

```
template="jdbc:actuate:db2://{ [host*] } { : [#port=50000] }  
{ ;databaseName=[database*] }"
```

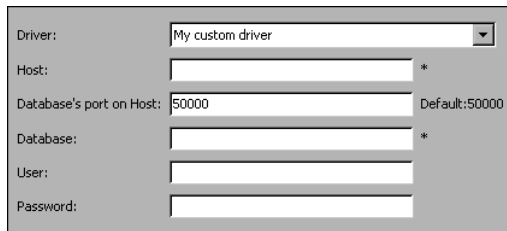
This template contains the variables host, port, and database. Host and database are required, as indicated by the asterisk. Port is an optional integer variable.

By default, the prompt text for each variable in the URL template is the name of the variable. To display different prompt text to the report developer, add a prompt element for the variable, within the driver.template element. The prompt element has the following syntax:

```
<prompt name="variable name" prompt="prompt text" />
```

For example, the following code provides a prompt for the variable, port, as shown in Figure 4-6:

```
<prompt  
  name="port "  
  prompt="Database's port on Host "  
>
```

A screenshot of a user interface for a custom JDBC driver. It features several input fields: a dropdown menu for 'Driver:' with 'My custom driver' selected; a text field for 'Host:' with an asterisk to its right; a text field for 'Database's port on Host:' with the value '50000' and 'Default:50000' to its right; a text field for 'Database:' with an asterisk to its right; a text field for 'User:'; and a text field for 'Password:'.

**Figure 4-6** User interface for custom JDBC driver

You must provide a prompt element for every variable in the URL template that requires a custom prompt.

---

## Accessing data using a SQL query

A SQL query is a SQL SELECT statement that specifies which data to retrieve from data set elements in a data source. You can create a data set that uses a SQL query to extract data from:

- A JDBC data source, including databases and other data sources accessed using a JDBC driver
- A BIRT Spreadsheet Designer example database
- A data set object

When using a JDBC source to access data, column names must not contain question marks (?). To include a column that has a name that contains a question mark in a SQL query, you must change the column name in the database. For information about changing a column name in your database, see your database documentation.

# Creating a SQL query using BIRT Spreadsheet Designer

To build a SQL query using the BIRT Spreadsheet Designer query editor, create a data set that uses a JDBC data source, example database, or a data set object as its data source. Next, select data elements for the data set. If the database supports schemas, select the appropriate schema before choosing the data elements for the query. To further refine the query, add filters, summarize aggregate data, and change the sort order of the columns, using the query editor.

You can use the query editor in two views:

- Design View provides a graphical interface in which to refine a data set and query. Design View is the default view.
- SQL View enables experienced developers to build and modify a query using SQL (Structured Query Language). You can view the SQL syntax of a query and directly modify its textual elements using SQL View.

As shown in Figure 4-7, the query editor Design View consists of two panes. Data elements selected for a data set appear in the upper pane. Double-click a field name listed in each table to add it to the query. To refine the data it extracts, add filters and parameters, and aggregate column values, if necessary.

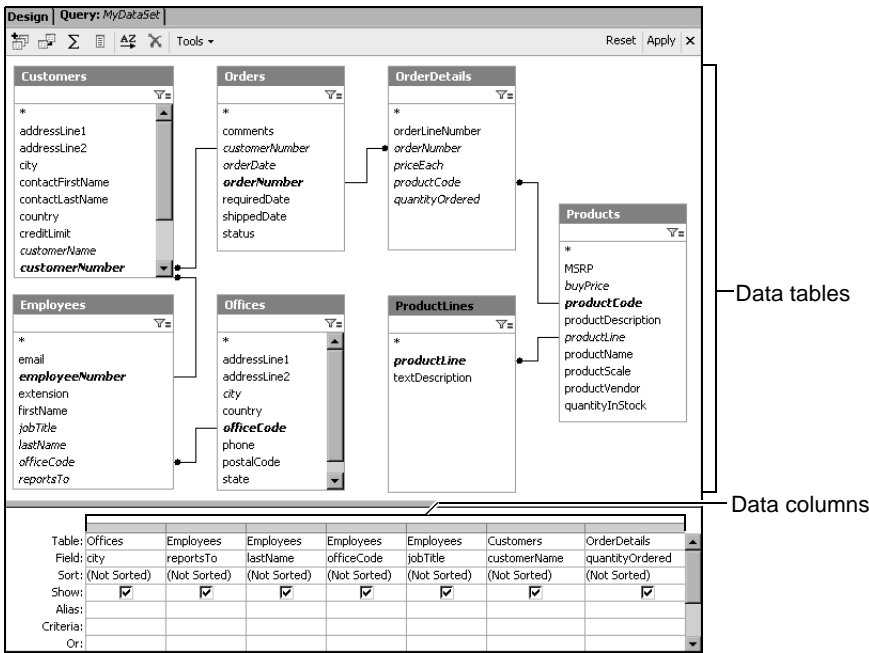
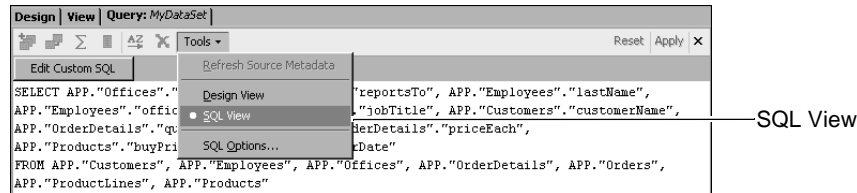


Figure 4-7 The query editor in Design View

Figure 4-8 shows how the same query appears in SQL View. You can use SQL View to review and modify the selections you make in Design View. Alternatively, if you know SQL, you can create queries on data sources with which you are familiar using only SQL View.



**Figure 4-8** The query editor in SQL View

You do not need to know SQL to use the query editor or any other BIRT Spreadsheet Designer feature. BIRT Spreadsheet Designer translates the data elements and selections you make in Design View into standard SQL syntax. When you select a data element from a data table, BIRT Spreadsheet Designer writes a specific SELECT clause of the SQL SELECT statement. When a data element displayed in the upper pane in Design View appears as a column in the lower pane, you can choose SQL View to see how your selections and changes appear as SQL, then return to Design View. Because BIRT Spreadsheet Designer does not reparse SQL, however, you cannot return to Design View after you save a change in SQL View.

To save any changes you make to query text displayed in SQL View, choose Apply. If you wish to return to Design View, select Reset to clear any changes you have not yet applied. Then, select Tools and choose Design View.

## About table joins

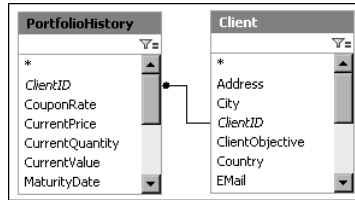
A join is a SQL query operation that combines records from two tables and returns them in a result set that is based on the values in the joined columns. For example, in the Sfddata example database, the customer and order tables join on the common field, custID. When the customer IDs match, the result set contains combined customer and order records. Joins ensure that a query returns the correct set of data rows. In a SQL query, every table with more than one column must use a join to connect to at least one other table.

In addition to creating joins between tables, you can join data set objects when working with multiple data sets. For information about joined data sets, see Chapter 3, “Accessing a data source.”

There are two types of joins:

- **Inner join**  
BIRT Spreadsheet Designer automatically joins fields that have the same name and data type when one of the fields is indexed as a primary key. This type of join is known as an inner join because the result set contains data rows with

matching values from both tables. For example, Figure 4-9 shows an inner join between the ClientID columns in the PortfolioHistory and Client tables. This inner join assures that the report displays portfolio data only for the clients whose IDs match in each table. If you remove this join, the report returns results for all of the clientIDs in both tables.

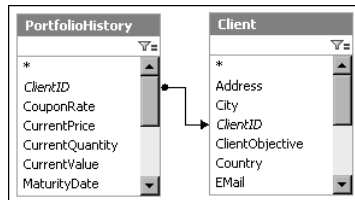


**Figure 4-9** Inner join

- Outer join

An outer join returns records from one table even when no matching values exist in the other table. For example, to display customers and orders for a period during which some customers have placed no orders, use an outer join.

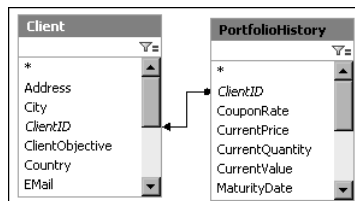
Figure 4-10 shows an outer join between PortfolioHistory.ClientID and Client.ClientID. Notice that an outer join is distinguished from an inner join by an arrowhead. You must create an outer join manually. You can create left and right outer joins. The join in Figure 4-10 is a left outer join. To learn more about using outer joins, see “Adding and removing joins,” later in this chapter.



**Figure 4-10** Left outer join

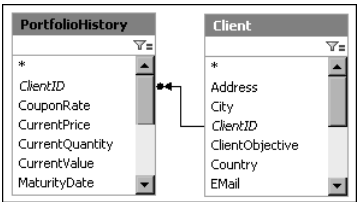
The node displayed at the join line end always indicates the left side of the join, independent from the relative position of tables in Design View.

Figure 4-11 shows the same outer join, with the PortfolioHistory table repositioned to the right of the Client table.



**Figure 4-11** Left outer join, repositioned

An arrowhead pointing at the node distinguishes a right outer join. Figure 4-12 shows a right outer join, with the PortfolioHistory table positioned to the left of the Client table.



**Figure 4-12** Right outer join

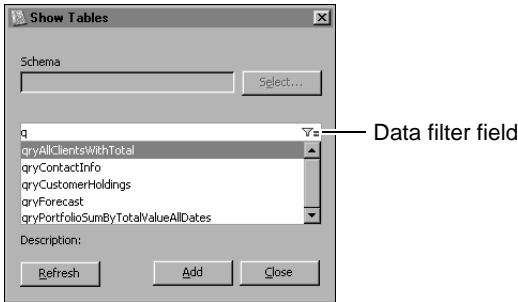
As you create or modify a join, BIRT Spreadsheet Designer writes the corresponding WHERE clause of the SQL SELECT statement.

## Selecting data elements using the query editor

A data element can include any column in the data source. A data source can contain predefined data elements known as views. Application or system administrators typically design specific views to simplify data access for other database users. A view can also contain computed fields that an administrator creates specifically for a column.

BIRT Spreadsheet Designer supports both tables and views as data elements. The query editor treats tables and views in the same way and does not distinguish between them. For brevity, all further references to tables include views. You can use the query editor to find and select any tables in a data source as you would data fields from a single table.

When you create a data set using the query editor, you use Show Tables to specify tables to include in a data set. Figure 4-13 shows a filtered list of tables displayed in Show Tables.



**Figure 4-13** Filtered list of tables

When connected to a database that supports catalogs and schemas, Show Tables displays Catalog and Schema fields active. When Show Tables displays the



Schema field, and Select is active, you must type or select a schema name before selecting data elements.

In the preceding example, several data element names that start with *q* exist in the data source. You can type a character, such as *q*, in the data filter field, to list only those data element names that start with *q*. After displaying such a filtered list, select Refresh to display all data element names in the data source. The data filter field supports two other simple wildcard search filters. For example, type *?or* in the data filter field to display a list of data element names that contain either *o* or *r*. For another example, type *\*or* in the data filter field to find field names that contain any combination of *o* and *r*.

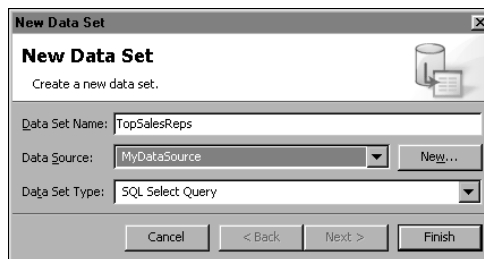
Any data element that you select using Show Tables appears in the query editor just like a data table.

## Creating a query using Design View

To create a SQL query without typing SQL syntax manually, use Design View.

### How to create a data set using Design View

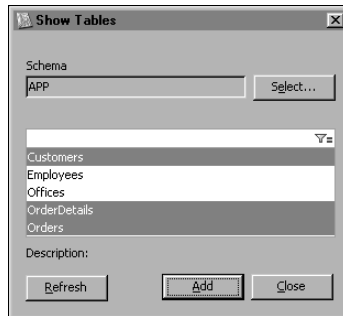
- 1 In BIRT Spreadsheet Designer, choose Report→Create Data Set.
- 2 On New Data Set, in Data Set Name, type a name for your data set.
- 3 In Data Source, select one of the following items:
  - Data Set Objects
  - The name of a JDBC data source
  - The name of an example database data source
- 4 In Data Set Type, select SQL Select Query, as shown in Figure 4-14, then Choose Finish.



**Figure 4-14** Creating a data set using a SQL query

- 5 On Show Tables, take the following actions:
  - 1 In Schema, type the schema name that contains the data to use in the report. This sample report uses the APP schema. Alternatively, you can choose Select to display and then choose from available schemas.

- 2 Select from the list of data elements those you wish to include in the data set. Figure 4-15 shows several tables selected for this example.

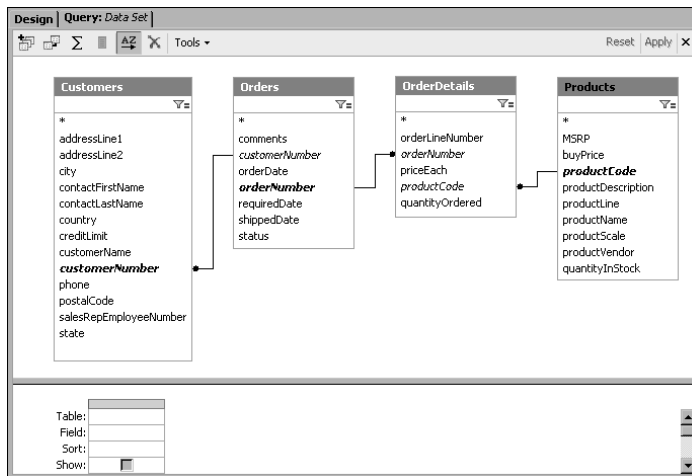


**Figure 4-15** Selecting data elements

- 3 After making your selections, choose Add, then Close.

The data elements you selected appear in the upper pane of the query editor.

When you select more than one table, BIRT Spreadsheet Designer creates joins that link related tables, as shown in Figure 4-16. You can also create additional joins between tables.



**Figure 4-16** Tables in the query editor

- 6 To save the data set, choose Apply.

In Data Explorer, a new element displays in Data Sets.

- 7 To add fields to the query, double-click field names in the tables in the upper pane. The selections appear as columns in the lower pane, as shown in Figure 4-17. To add all fields from a table, double-click the asterisk (\*) at the top of the table.

Table:	Employees	Employees	Employees	Employees	Customers	OrderDetails
Field:	reportsTo	lastName	officeCode	jobTitle	customerName	quantityOrd...
Sort:	(Not Sorted)	(Not Sorted)	(Not Sorted)	(Not Sorted)	(Not Sorted)	(Not Sorted)
Show:	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Alias:						
Criteria:						
Or:						

**Figure 4-17** Columns in the query editor

Apply

- 8 In the query editor toolbar, choose Apply.

In Data Explorer, a plus sign (+) appears next to the new data set to indicate that it contains fields. Select the plus sign to see the fields contained in the new data set. You can further refine the query by adding filters or parameters, and by aggregating column data and changing column sort order.

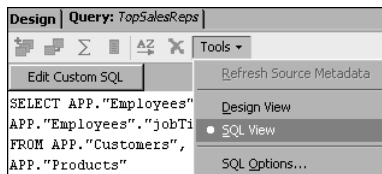
For instruction on how to preview query results, see Chapter 3, “Accessing a data source.”

## Creating a query using SQL View

If you know SQL, you can type and edit the SQL SELECT query directly into the text editor pane in SQL View. After you edit a SQL query in SQL View, you cannot view or edit it in Design View.

### How to view and modify a SQL SELECT statement

- 1 In the query editor, select Tools then choose SQL View from the drop down menu, as shown in Figure 4-18.
- 2 Select Edit Custom SQL.



**Figure 4-18** Opening Query Editor in SQL View

Apply

- 3 Type the SQL SELECT statement in the SQL View text pane. You can also copy a SELECT statement from another source and paste it in the text pane.
- 4 Choose Apply.

In Data Explorer, a plus sign (+) appears next to the data set specified in the query to indicate that it contains fields. Select the plus sign if you want to see the list of fields contained in the data set.

## Suppressing the catalog, schema, or table name in a SQL query

You can suppress the table name that appears in each column reference in the generated SQL for a SQL SELECT or stored procedure query. If your database supports catalogs and schemas, you can also suppress the schema and catalog name in the generated SQL. This capability is useful if you use different schemas in your development and deployment databases. In the BIRT Spreadsheet Designer query editor, choose Tools→SQL Options. In SQL Options, select the appropriate option.

If you select no options, the generated SQL refers to a table in the following format:

```
<catalog-name>.<schema-name>.<table-name>
```

If you select no options, the generated SQL refers to a column in the following format:

```
<catalog-name>.<schema-name>.<table-name>.<column-name>
```

You can suppress parts of the table and column identification:

- To suppress the catalog name when you refer to a table or column, select Suppress Catalog Name.
- To suppress the catalog and schema name when you refer to a table or column, select Suppress Schema Name.

To suppress the table name when you refer to a column, select Suppress Table Name.

---

## Creating a computed field using the query editor

A computed field is a data field you create to perform a calculation or action that cannot be performed by a field already existing in the data source. A computed field is also known as a calculated field.

You can use the following elements to create a computed field:

- Any field in the data source
- Any mathematical or logical expression and operator that the database engine can interpret

You can also create a computed field using Data Explorer. Use Data Explorer to create a computed field if your database does not support the calculations that you need. This section explains how to create a computed field using the query editor. For instruction about using Data Explorer to do so, see Chapter 3, “Accessing a data source.”

**How to create a computed field using the query editor**

- 1 If all the fields you use in the calculation for the computed field are in the same table, select the table name in the table cell.
- 2 In the lower pane of the query editor, select the Field cell in an empty column, as shown in Figure 4-19.

Table:	items	items	items	
Field:	itemcode	pricequote	quantity	
Sort:	(Not Sorted)	(Not Sorted)	(Not Sorted)	
Show:	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Alias:				
Criteria:				
Or:				

— Type an expression in the Field cell

**Figure 4-19** Creating a computed field in the query editor

- 3 Type the expression and operator in the Field cell.
- When you create a computed field, you must use syntax appropriate for the database in which the query runs. For example, to multiply items.pricequote by items.quantity using fields from the Sfddata example database, type:

"pricequote" \* "quantity"

If the fields in the calculation are from different tables, or if the same field names occur in more than one table, prefix the field names with the table names.

- 4 To finish adding the computed field and confirm that the computed field appears in the query editor, select Show for the computed field. Figure 4-20 shows the newly created computed field.

Table:	items	items	items	items	
Field:	itemcode	pricequote	quantity	orderID	"pricequote"*"quantity"
Sort:	(Not Sorted)	(Not Sorted)	(Not Sorted)	(Not Sorted)	Ascending
Show:	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Alias:					exp1
Criteria:					
Or:					

**Figure 4-20** Confirming the addition of a computed field

- 5 In Alias, type a name for the computed field.



Save the query. The computed field appears in Data Explorer, under Data Sets, along with the other data fields. A plus and minus sign differentiates a computed field from the other fields in the data set.

**Summarizing row data in the query editor**

To summarize retrieved data, use the lower pane of the query editor to create an aggregate row. An aggregate row is a single row that combines the data from a group of rows when those rows share one or more column values. BIRT

Spreadsheet Designer uses a SQL aggregate function to aggregate row data. The SQL aggregate function displays only the total value for that data field. For example, you can group all rows for a particular customer and apply a SQL aggregate function to a field to display the total value of that customer's orders. As you create an aggregate expression, BIRT Spreadsheet Designer adds it to the SELECT clause of the SQL SELECT statement.

Table 4-3 describes available functions for aggregate expressions. The data type of a field determines which aggregate expressions the field supports. For example, you cannot use avg for a text field. You cannot use an aggregate expression if the query uses an asterisk (\*) to import all the fields from a table.

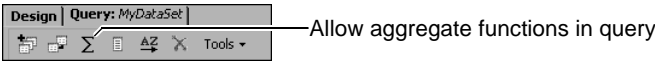
**Table 4-3** Aggregate functions and their actions

Function	Action
Sum	Totals the column
Avg	Finds the average value
Min	Provides the lowest (minimum) value in the column
Max	Provides the highest (maximum) value in the column
Count	Provides the total number of non-null values in the column

### How to aggregate query results



- 1 Choose aggregate function on the query editor, as shown in Figure 4-21.



**Figure 4-21** Choosing the SQL aggregate function

Choosing the aggregate function inserts a Total row into the data columns in the lower pane. Each cell in the new Total row contains a Group By clause. All fields in a query that uses SQL aggregation and are not themselves aggregated must be grouped.

- 2 To aggregate the data in a column, use the arrow in the Total field to replace Group By with an aggregate calculation.

The example in Figure 4-22 shows orders.status and orders.orderID set to Group by, items.itemcode set to Count, and items.pricequote set to Sum.

Table: orders	orders	items	items
Field: status	orderID	itemcode	pricequote
Total: Group By	Group By	Count	Sum
Sort: (Not Sorted)	(Not Sorted)	(Not Sorted)	(Not Sorted)
Show: <input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Alias:		COUNTofitemcode	SUMofpricequote

The aggregate function inserts the Total row

**Figure 4-22** Groupings created from an aggregate query

Figure 4-23 shows how the generated report looks using the aggregate query created in step 2.

status	orderID	COUNTofitemcode	SUMofpricequote
Cancelled	1100	8	954
Cancelled	1385	2	332
Cancelled	1595	5	898
Cancelled	1685	13	1359

**Figure 4-23** A sample aggregate report

---

## Filtering query results

To filter a query, you set a condition that the database row must meet. You can also use a parameter to request a value at run time or view time, then use that value as the condition. When you put a condition on row retrieval or use a parameter to filter a query, BIRT Spreadsheet Designer adds the condition or parameter to the WHERE clause of the SQL SELECT statement. You use the Criteria field in the lower pane of the query editor to create the filter.

### Using a condition to filter a query

A condition contains a comparison (<, >, <=, >=, or <>) or an arithmetic symbol and the value to which to compare the field content. For example, to return rows for sales representative identification numbers 10 and higher, use the following criteria expression in the sales representative ID column:

```
>=10
```

If you supply a string for comparison, enclose the string with single quotation marks. For example, adding the following criteria to the customers.state column returns only customers from New York:

```
'NY'
```

To filter by a date condition, supply the date value in the format that your database requires. For example, to return records for all dates after May 21, 2000, you use the following condition:

```
>{d '2000-05-21'}
```

For more information about the condition syntax your database accepts, refer to your database documentation.

### How to create a comparison filter

- 1 To add a filter, in the query editor, select the Criteria row for the column to filter. Then type the condition expression. For example, to return only customers from New York in a query that includes a state column, type:

```
'NY'
```

Figure 4-24 shows how the query editor looks after you add the filter.

Table:	customers	customers	customers	customers
Field:	state	city	customName	phone
Sort:	(Not Sorted)	(Not Sorted)	(Not Sorted)	(Not Sorted)
Show:	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Alias:				
Criteria:	'NY'			
Or:				

**Figure 4-24** Adding a filter



2 Choose Apply.



3 Choose Run to preview the report to see how the filter tailors query results. The spreadsheet report in Figure 4-25 shows the results of applying the New York filter.

state	city	customName	phone	
NY	Albany	Advanced Solutions Inc.	5185559644	
NY	Albany	CompuEngineering	5185553942	
NY	Albany	Signal MicroSystems	5185554154	
NY	Albany	Technical Systems Corp.	5185554378	
NY	New Rochelle	Advanced Design Corp.	9145556707	
NY	New Rochelle	Computer Engineering	9145557064	
NY	New Rochelle	Design Boards Co.	9145557468	
NY	New Rochelle	Design Systems	9145553870	
NY	New Rochelle	TekniSystems	9145554928	
NY	NYC	Advanced Design Inc.	2125558493	
NY	NYC	Design Solutions Corp.	2125553675	

**Figure 4-25** Results of adding a filter

### How to add a date filter to a query

The Classic Models database contains orders from January, 2003 through May, 2005. The following procedure shows you how to filter the orderDate column to narrow the order window. You can add a date filter to any field that contains date information.

1 In the lower pane of the query editor, in the orderDate field, type the following condition in the Criteria field, as shown in Figure 4-26.

"Between 01/01/2004 and 12/31/2004"

Table:	CustomerOrderDetails	Products	Orders
Field:	priceEach	buyPrice	orderDate
Sort:	(Not Sorted)	(Not Sorted)	(Not Sorted)
Show:	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Alias:			
Criteria:			"Between 01/01/2004 and 12/31/2004"
Or:			

**Figure 4-26** Creating a date filter



2 Choose Apply from the query editor toolbar to save the filter. The generated report displays information only for orders placed in 2004.



## Using a parameter to filter a query

A parameter passes a value to a spreadsheet report when you run the report. The information that you supply to the parameter customizes the report. For example, if you have a global database, you can supply a parameter to select only those customers from a particular country when you run the report. You can create any number of BIRT Spreadsheet Designer parameters for a particular workbook and set default parameter values.

The following sections provide information on how to use the query editor to create a parameter.

### About creating a parameter in a query

As with a conditional filter, you use the Criteria field in the lower pane of the query editor to create a parameter. When creating a parameter, consider the following guidelines:

- You declare a parameter as static or ad hoc. A user must supply a value for a static parameter when the report runs. A user can omit the value for an ad hoc parameter.
- BIRT Spreadsheet Designer can use a parameter name as a defined name. To use a parameter as a defined name, you must use the same naming guidelines you use for a defined name when you create the parameter.

### Using a static parameter

A static parameter acts as a placeholder in the report definition until a user runs the report. The user who runs the report must accept the default parameter value or provide a new value for the parameter. The parameter value changes the WHERE clause in the SQL statement.

When you add a static parameter to a query, you can use the parameter to filter a query in the same way in which you use query criteria. You can use an expression such as AND, or LIKE, or use a wildcard. For example, to return customer records in which the customer name begins with a letter that the user provides, use the following expression in the criteria field of the customer name column:

```
LIKE :Letter%
```

where Letter is a static parameter name.

#### How to add a static parameter

- 1 In the query editor, select Criteria for the column that you want to filter, and type a name for the parameter. Prefix the name with a colon (:), for example, :state, as shown in Figure 4-27.

Table:	Customers	Customers	Customers	Customers
Field:	"state"	"city"	"customerName"	"phone"
Sort:		Ascending	Ascending	
Show:	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Alias:				
Criteria:	:state			
Or:				

**Figure 4-27** Adding a static parameter

- 2 To view the parameter in the SELECT statement, choose SQL, as shown in Figure 4-28.

<pre>SELECT "APP"."Customers"."state", "APP"."Customers"."city", "APP"."Customers"."customerName", "APP"."Customers"."phone" FROM "APP"."Customers" WHERE ("APP"."Customers"."state"=:state) ORDER BY "APP"."Customers"."city", "APP"."Customers"."customerName"</pre>	Static parameter
--	------------------

**Figure 4-28** Examining a static parameter located in a WHERE clause

### Using an ad hoc parameter

An ad hoc parameter acts as a placeholder in the report definition until the user runs the report. The user who runs the report can provide a value for the parameter. When the report user supplies a value for an ad hoc parameter, the WHERE clause in the SQL statement changes.

When you use an ad hoc parameter in BIRT Spreadsheet Designer, the following guidelines and conditions apply:

- BIRT Spreadsheet Designer uses an ad hoc parameter as part of an simple equal condition in the query.
- You can use query-by-example (QBE) syntax with an ad hoc parameter.
- If you add more than one ad hoc parameter to a column in a query, BIRT Spreadsheet Designer uses the parameters in an OR statement in the query. The query returns data that matches any of the parameter values that you provide for that column.

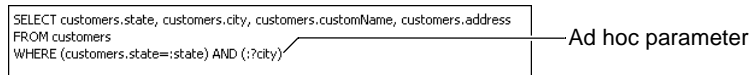
#### How to add an ad hoc parameter

- 1 In the query editor, choose the fields for the report.
- 2 To set the ad hoc parameter, in Criteria, type a parameter name that is preceded by a colon (: ) and a question mark (?). For example, :?city, as shown in Figure 4-29.

Table:	customers	customers	customers	customers
Field:	state	city	customName	address
Sort:				
Show:	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Alias:				
Criteria:	:state	:?city		
Or:				

**Figure 4-29** Defining an ad hoc parameter

- 3 To view the parameter in the SELECT statement, choose SQL, as shown in Figure 4-30.



```
SELECT customers.state, customers.city, customers.customName, customers.address
FROM customers
WHERE (customers.state=:state) AND (:?city)
```

Ad hoc parameter

**Figure 4-30** Examining an ad hoc parameter located in a WHERE clause

---

## Modifying a query

This section provides instruction on how to modify a query in the following ways:

- Adding and removing joins
- Adding, moving, reordering, and hiding columns

### Adding and removing joins

You can manually create an outer join in two ways:

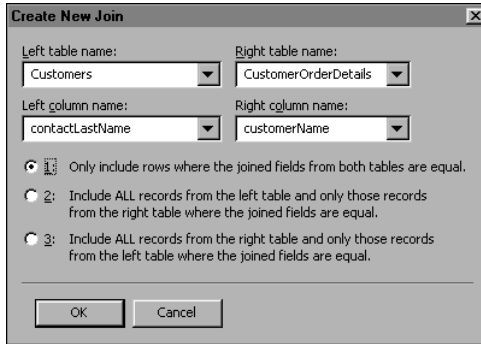
- By dragging a column from one table or data set object and dropping it onto a column in a different table or object. A join appears that links the columns in each table or data set object.
- Using the Create New Join dialog to choose the tables and columns to join.

To remove a join, right-click the join and choose Remove Join from the context menu. Do not run a report with a query that contains multiple tables but no joins.

#### How to add a join using the Create New Join dialog



- 1 In the query editor, choose the Add a join between two tables icon from the query editor toolbar.
- 2 On Create New Join, select the left and right table names, and the left and right column names, as shown in Figure 4-31.



**Figure 4-31** Using the query editor to create a join

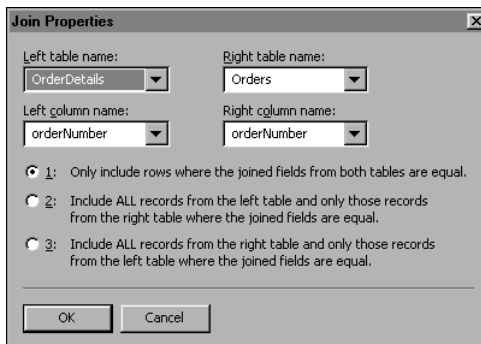
Choose OK. The join appears between the tables and columns in the upper pane of the query editor.

## Modifying join properties

Use Join Properties, shown in Figure 4-32, to modify a join in the following ways:

- Change the tables or columns in the selected join.
- Choose option 1, which includes only rows where the joined fields from both tables are equal. Option 1 creates an inner join.
- Choose option 2, which includes all records from the table on the left and only those records from the table on the right where joined fields match. Option 2 creates an outer join.
- Choose option 3, which includes all records from the table on the right and only those records from the table on the left where joined fields match. Option 3 creates an outer join.

The default setting for Join Properties is option 1. You can switch from one join option to another by choosing a different radio button.



**Figure 4-32** Using the query editor to modify a join

To access the Join Properties dialog, double-click on a join between two tables in the query editor. Join Properties appears.

## Modifying columns using the query editor

Using the query editor, you can add, move, reorder, and hide columns that appear in a report. You can also change a column’s data type and name, sort and filter data, create a computed field, and summarize column data. To narrow report output, you can use report parameters.

### Adding a table or column



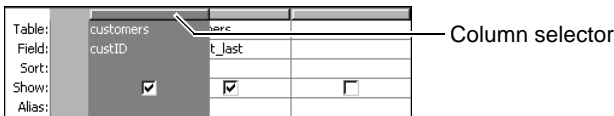
You can add a column from an existing query table to a query, or you can add a column from a new table. To add a table, choose the Display show tables dialog icon on the query toolbar. On Show Tables, select the table to add, and choose Add. To add a column to the query, double-click the column name in the upper pane of the query editor.

### Moving a column

You can move a selected column to the left or right. You can also reorder multiple columns simultaneously. Moving columns in the query editor changes the order of the fields in the SELECT clause of the query. The following procedures show you how to move and rearrange columns in a query.

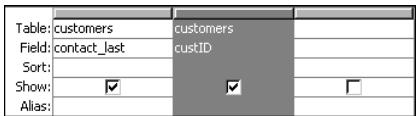
#### How to move a single column

- 1 In the query editor, select the column selector for the column you want to move, as illustrated in Figure 4-33.



**Figure 4-33** Selecting a column

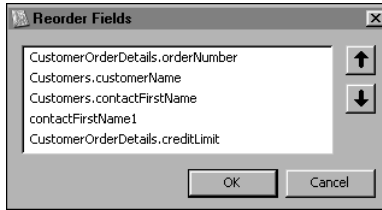
- 2 Drag the column to another position in the grid. Figure 4-34 shows the result of dragging the customers.custID column to the right.



**Figure 4-34** Dragging a column to a different grid position

#### How to rearrange multiple columns

- 1 Right-click the query editor, and choose Reorder Fields. Reorder Fields appears as shown in Figure 4-35.



**Figure 4-35** Reordering columns



- 2 On Reorder Fields, select a field name, and choose the appropriate arrow to change the position of the selected column. You can select additional field names and move them as needed.
- 3 When you finish, choose OK.

## Sorting data rows

You can change the display order of the rows that a query returns by selecting one or more sort keys. Sorted data appears in ascending or descending order. As you select columns for sorting, BIRT Spreadsheet Designer adds them to the ORDER BY clause of the SQL SELECT statement from left to right. To change the order of columns in this clause, move the columns as shown earlier in this chapter. You can also specify the sort order of data values when you add data fields to a data range.

### How to sort data in the query editor

In the Sort row of the query editor, for the column to sort, select Ascending or Descending. The query in Figure 4-36 sorts order records by state, then city.

Table:	Customers	Customers	Customers	Orders	Orders
Field:	state	city	customerName	orderNumber	status
Sort:	Ascending	Ascending	(Not Sorted)	(Not Sorted)	(Not Sorted)
Show:	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Alias:					

**Figure 4-36** Sorting data in the query editor

To remove a sort order from a column, choose (Not Sorted) in the Sort row for the column. The default Column sort order is (Not Sorted).

## Hiding a column

You can hide a column in the query so it does not appear on the published report. For example, you can use a column in a calculation without displaying it in the report. To hide a column, in the lower pane of the query editor, deselect Show in the column you wish to hide.

## Deleting a field from the query

You can use the query editor to delete a field from a query. In the lower pane of the query editor, right-click the column that you want to delete, and choose Remove Field from the context menu.

## About refreshing metadata

BIRT Spreadsheet Designer uses two kinds of metadata to show query results for a spreadsheet report. The query editor uses source metadata to display a list of available tables and views, and, for each table or view, to display a list of fields. Result set metadata is the metadata associated with the result set obtained from running a query. You can update, or refresh, source and result set metadata manually as you develop a query.

### Refreshing source metadata

The BIRT Spreadsheet Designer query editor displays refreshed source metadata in the following cases:

- When you open query editor and connect to a new database or select a database object such as a table, view, schema or stored procedure
- When you select query editor, then choose Tools→Refresh source metadata

For example, consider a change such as a database schema update. To review an existing query, open query editor, then choose Tools→Refresh source metadata. If the change affects a single database table or view, the report developer can select a table and then refresh the source metadata for that one table only.

After refreshing source metadata, query elements that do not match the current database appear red in the query editor. You must edit the query to yield correct query results.

### Refreshing result set metadata

Refreshing result set metadata discards result set metadata for external queries, then retrieves it after running the query on the current database. To refresh result set metadata, do either of the following:

- In the query editor, save the query.
- Choose Report→Refresh Metadata.

A data set that does not match current database conditions appears red in Data Explorer. For example, if a database server is unavailable when you run a query, the data set that the query returns appears red in Data Explorer. When you know that the server is available, choose Report→Refresh Metadata.

---

## Running multiple data set queries simultaneously

A spreadsheet report containing multiple data sets may hang when data set queries run simultaneously on a Windows system. This behavior occurs when connecting to a data source using a Type 2 JDBC driver.

To ensure that multiple data set queries run successfully, set a priority level for each query. Using Data Set Properties, set the Execution Priority for each query to a unique value. Alternatively, connect to each data source using a JDBC Type 4 driver.

---

## Accessing a stored procedure

A stored procedure is a named set of one or more SQL statements stored in a relational database for reuse. You can create a data set that uses a stored procedure to extract data from a JDBC data source or an example database data source.

---

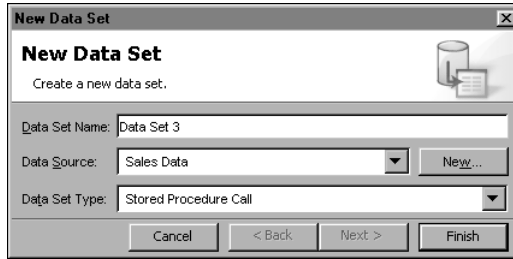
## Creating a data set that uses a stored procedure call

If your database supports Oracle stored procedures or any stored procedures that use standard JDBC syntax, you can create and modify a data set that uses a stored procedure. To define the stored procedure call, you use a stored procedure editor to choose the stored procedure and set the values of the parameters. If you open a report that was saved in a prior release and open the stored procedure editor, you must save the stored procedure call before you run the report.

### How to create a data set that uses a stored procedure call

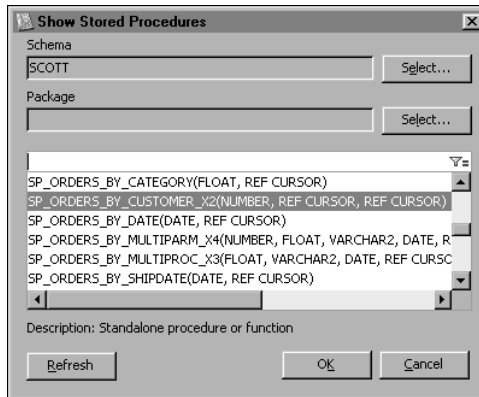
- 1 In BIRT Spreadsheet Designer, choose Report➤Create Data Set.
- 2 On New Data Set, in Data Set Name, type a name for your data set.
- 3 In Data Source, select one of the following items:
  - The name of a JDBC data source
  - The name of an example database data source
- 4 In Data Set Type, select Stored Procedure Call, as shown in Figure 4-37, then choose Finish.





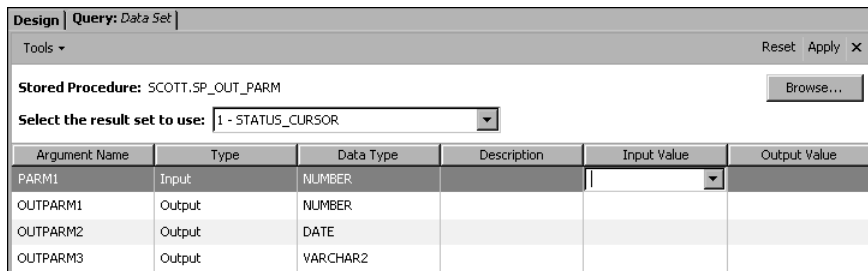
**Figure 4-37** Choosing to create a data set using a stored procedure call  
Stored Procedure appears on Query: Data Set so that you can specify the stored procedure and result set to retrieve.

- 5 On Query, select Browse.
- 6 On Show Stored Procedures, select a stored procedure, as shown in Figure 4-38, then choose OK.



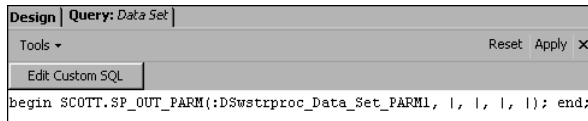
**Figure 4-38** Selecting a stored procedure

Query displays the stored procedure name. Arguments appear in the grid, as shown in Figure 4-39.



**Figure 4-39** Viewing the arguments for the stored procedure

- 7 Provide a value for each argument type using one of the following methods:
  - Type a static value.
  - Type a parameter name that is preceded by a colon (:).
  - Use the drop-down list to select an existing parameter.
  - Choose Create Parameter and then, on External Parameter:
    - In Name, type characters.
    - In Data Type select a type.
    - In Default Value, type characters or select No Default Value.
    - Choose OK.
- 8 Choose Apply. BIRT Spreadsheet Designer creates a data set and a corresponding data set object. The new data set appears under Data Sets and the new data set object appears under Data Set Objects in Data Explorer.
- 9 To view the stored procedure call, as shown in Figure 4-40, choose Tools, then choose SQL View.



**Figure 4-40** Viewing the stored procedure call

Do not modify the stored procedure call unless you plan to only use the SQL editor to modify the stored procedure call in the future. If you modify the stored procedure call in the SQL editor, you cannot use Design View to further modify the stored procedure call.

- 10 To view the first rows returned by the stored procedure query, right-click the data set and choose Preview.

# Accessing an Actuate information object

This chapter contains the following topics:

- About information objects
- Connecting to an Actuate information object
- Examining Information Object Query Builder
- Creating an information object query in the Basic Design interface
- Creating a customized graphical information object query
- Creating a textual information object query
- Displaying information object query output

---

## About information objects

An information object is an encapsulated SQL query that you use as a basis for creating queries in report designers such as BIRT Designer Professional, Actuate BIRT Spreadsheet Designer, and Actuate e.Report Designer Professional. Other Actuate products also use information objects. Information objects enable access to data from diverse data sources, such as database tables and views, other information objects, and result sets from stored procedures and open data access (ODA) data source queries. The encapsulation of these queries supports efficient report development by:

- Providing reusability across multiple reports and Actuate products. Report developers can use information objects to create reports in a report designer. Each report can use different columns, sorting rules, parameters, and filters, but the reports are based on the same data.
- Hiding the complexity of data access. Using information objects separates the data access logic from the report development process. Report developers can create queries without code, even for complex data from multiple sources. Information objects enhance report development by making it possible to create and modify queries without connecting to a data source or having knowledge of the data.

When you create a data set using a query that is based on an information object, you can use all the data in the information object, or you can use a subset of the data in the information object. You also can join multiple information objects.

Information objects are created using Actuate Information Object Designer and stored in an Actuate BIRT iServer Encyclopedia volume. To work with an information object, you must have an account on an Encyclopedia volume with the privileges required to access and run the information object.

## Working with an information object

To use an information object as a data source, you complete the following tasks:

- Connect to an Encyclopedia volume.
- Create a data set using a query based on one or more information objects.
- Set up the spreadsheet report design to display the data.
- Run the report to view the data.

## Preparing to access information object data

Before you create an information object query, you need the following information:

- The name of the Actuate BIRT iServer and the Encyclopedia volume that contains the information object
- A user name and password for the volume
- The name of the information object and its location in the Encyclopedia volume

You need the following privileges to use the information object:

- Read privilege on the information object to create a query using the information object
- Execute or trusted execute privilege on the information object to run the query

---

## Connecting to an Actuate information object

You access data from an Actuate information object using Actuate Data Integration Service. You access an information object using Actuate BIRT iServer System. To access data from an information object, you set up an Actuate Data Integration Service connection.

To access information object data, you connect to an Encyclopedia volume on an Actuate BIRT iServer machine. To enable BIRT Spreadsheet Designer to locate an information object, you specify values for the connection properties.

Table 5-1 describes the connection properties for an information object.

**Table 5-1** Information object connection properties

Property	Description
ServerUri	<p>The URL to the BIRT iServer machine that contains the Encyclopedia volume in which the information objects reside.</p> <p>Use the following syntax:</p> <p><code>http://&lt;server&gt;:&lt;port&gt;</code></p> <p>where</p> <ul style="list-style-type: none"> <li>■ &lt;server&gt; is the machine name for the BIRT iServer machine.</li> <li>■ &lt;port&gt; is the port number of the BIRT iServer machine.</li> </ul>
Volume	The BIRT iServer Encyclopedia volume that contains the information objects.
UserName	The name of the Encyclopedia volume account that has access to the information objects.
Password	The password for the Encyclopedia volume account that has access to the information objects.

*(continues)*

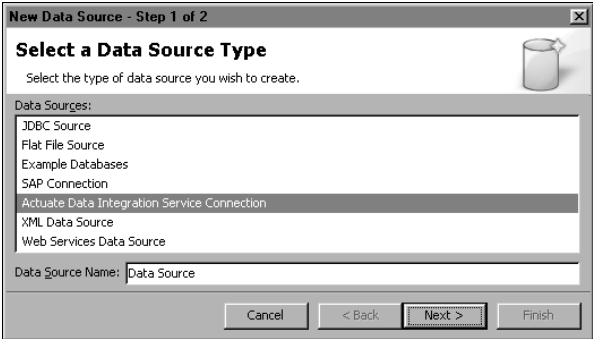
**Table 5-1** Information object connection properties (continued)

Property	Description
UseLoggedInUser CredentialsOnServer	Set this property to True to use the credentials of the user running the report when the report is run on the BIRT iServer. Set this property to False to use the ServerUri, Volume, UserName, and Password specified in the other properties on this page when running a report on a BIRT iServer. The ServerUri, Volume, UserName, and Password specified in the other properties on this page are always used when a user runs a report from BIRT Spreadsheet Designer.

You can change the BIRT iServer System connection properties when you create an information object query. The property values that you specify in the connection, however, remain the default property values. In BIRT Spreadsheet Designer, you specify values for the BIRT iServer System connection properties when you create an Actuate Data Integration Service Connection data source.

**How to create an Actuate Data Integration Service Connection data source**

- 1 In Actuate BIRT Spreadsheet Designer, choose Report➤Create Data Source.
- 2 On New Data Source—Select Data Source Type, in Data Sources, select Actuate Data Integration Service Connection, as shown in Figure 5-1.



**Figure 5-1** Selecting Actuate Data Integration Service Connection

- 3 In Data Source Name, type a name for the data source and choose Next. New Data Source—New Data Source appears, displaying the fields for an Actuate Data Integration Service Connection data source. Type the property values for your Actuate BIRT iServer system.

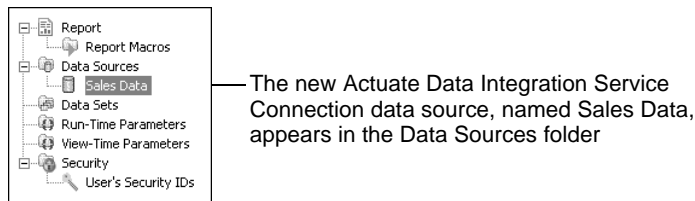
In Figure 5-2, the value of True for UseLoggedInUserCredentialsOnServer specifies that when a user runs the report on a BIRT iServer, BIRT Spreadsheet Designer uses that person’s user name and password on the current Actuate BIRT iServer and volume. When a user runs the report from BIRT Spreadsheet Designer, the connection uses tester as the user name, its associated password,

the TestSystem Actuate BIRT iServer system, and the Headquarters Encyclopedia volume specified in the other properties on this page.

Name	Value
ServerUri	http://TestSystem:8000
Volume	Headquarters
UserName	tester
Password	*****
UseLoggedInUserCredentialsOnServer	Yes

**Figure 5-2** Specifying a new Actuate Data Integration Service Connection data source

- 4 Choose Finish. The data source that you defined appears in Data Explorer. Figure 5-3 shows a new data source named Sales Data in Data Explorer.



**Figure 5-3** Examining an Actuate Data Integration Service Connection data source in Data Explorer

Now that you have an Actuate Data Integration Service Connection data source that specifies the connection information for an Actuate BIRT iServer system, you can create a data set that queries data from an information object.

## Examining Information Object Query Builder

Information Object Query Builder supports defining a query on information objects created using Actuate Information Object Designer. Use the query builder to specify the data to include, the sort order, and parameters for filtering the data. Information Object Query Builder produces a query that obtains data from the information object data source. A report developer uses this query builder to create an information object query or change an existing information object query.

BIRT Spreadsheet Designer uses Information Object Query Builder. Information Object Query Builder runs as an application separate from the designer

application. You must exit Information Object Query Builder before returning to work in the designer application. When you close Information Object Query Builder, Windows sometimes does not display the designer application as the top window. In that event, select the designer application from the Windows task bar.

## Opening Information Object Query Builder

After creating a connection to an information object data source, the next step is to open Information Object Query Builder to create a query. You can use this step to reopen an existing information object query.

### How to open Information Object Query Builder using BIRT Spreadsheet Designer

- 1 In BIRT Spreadsheet Designer, choose Report➤Create Data Set.
- 2 On New Data Set, in Data Set Name, type a name for your data set.
- 3 In Data Source, select an Actuate Data Integration Service Connection data source. The Data Set Type is already set to Actuate Data Integration Service Connection Data Source, as shown in Figure 5-4.



**Figure 5-4** Creating an Actuate Data Integration Service Connection data set  
Choose Finish.

- 4 In Actuate Data Integration Service Data Source Properties, type the name of the data set and press Edit. Connection Properties appears.
- 5 Type the appropriate values for the fields that are described in Table 5-2. Then, choose OK.

Table 5-2 Property values for accessing Information Object Query Builder	
Field	Description
BIRT iServer	The URL to the iServer that manages the Encyclopedia volume with the information objects
Port number	The port number to access the iServer
Volume	The Encyclopedia volume containing the information objects



**Table 5-2** Property values for accessing Information Object Query Builder

Field	Description
User name	The name of the Encyclopedia volume account with access to the information objects
Password	The password for the Encyclopedia volume account with access to the information objects

Information Object Query Builder Basic Design appears. You can then choose how you want to develop your information object query.

## Choosing an information object query editor

You can create an information object query in three ways, depending on the number of information objects you access in your query and your familiarity with SQL concepts and the SQL language. Table 5-3 provides a brief description of the differences between the interfaces. When you open Information Object Query Builder, you view the Basic Design interface by default.

**Table 5-3** Differences between the three Information Object Query Builder interfaces

Interface capability	Basic Design (graphical)	Advanced Design (graphical)	Advanced Design (textual)
Can access more than one information object in the query	No	Yes	Yes
Provides the ability to specify sorting options, filtering, and parameters	Yes	Yes	Yes
Provides the ability to create a complex query	No	Yes	Yes
Provides a graphical interface	Yes	Yes	No
Can access the expression builder	Yes	Yes	No
Requires understanding of SQL concepts such as joins, grouping, distinct rows, and filtering on an aggregate column	No	Yes	Yes
Requires the ability to write Actuate SQL code	No	No	Yes
Can open an existing query created or modified in the Basic Design interface	Yes	Yes	Yes
Can open an existing query created or modified in the Advanced Design interface	No	Yes	Yes
Can open an existing query created or modified in the textual editor	No	No	Yes



To access the Advanced Design graphical interface, open Information Object Query Builder, and choose Advanced Perspective.



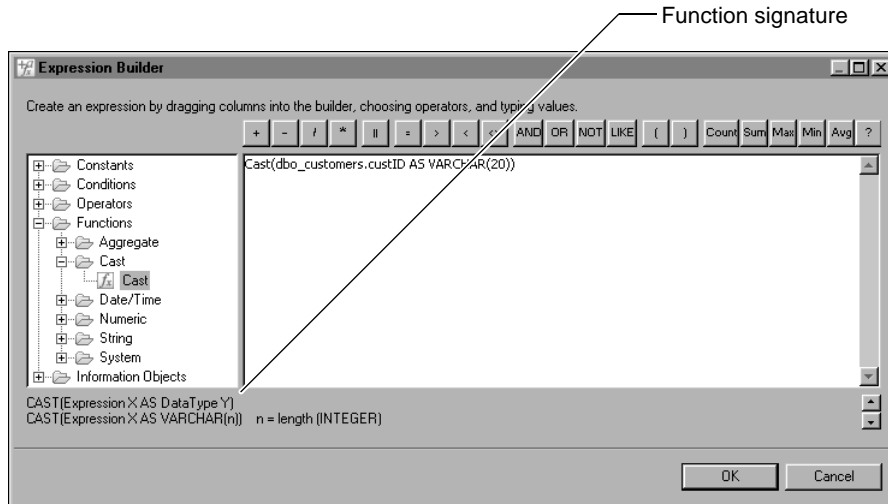
To access the textual editor, open Information Object Query Builder, and choose Advanced Perspective. On Query Design, choose SQL Editor.

## Using the expression builder

When designing a query, Actuate SQL expressions support specifying filters or joins, creating aggregate data, and so on. For example, the expression, `officeID = 101`, specifies that data returned by the query must have 101 in the `officeID` column. You can type these expressions in any of the three editors. In the textual editor, the expressions are part of the SQL `SELECT` statement. In the Basic Design and Advanced Design graphical query editors, you can type the expressions or use the expression builder to develop expressions. Expression Builder helps you create expressions by providing a graphical interface for selecting column names, constants, functions, operators, and so on, from lists.

Use the expression builder to create Actuate SQL expressions on the filter page of Information Object Query Builder Basic Design and many pages in Information Object Query Builder Advanced Design.

Figure 5-5 shows Expression Builder. Drag items from the left pane to the right pane or insert items by choosing the appropriate icon. If you select a function in the left pane, the function signature appears in the bottom pane.



**Figure 5-5** Using Expression Builder to create expressions

---

## Creating an information object query in the Basic Design interface

You can create a query on a single information object using Information Object Query Builder Basic Design. This query editor supports specifying sorting options, filtering, and parameters. If you need to work with more than one information object or require more customization than this query editor supports, use Information Object Query Builder Advanced Design.

You can browse an iServer System Encyclopedia volume using Information Object Query Builder to find and select the information object for the query. If you have already chosen your information object and are re-entering Information Object Query Builder, the editor displays the previously selected information object. If an information object that is used by an existing query no longer exists, you must specify a new information object and its columns, parameters, and filters.

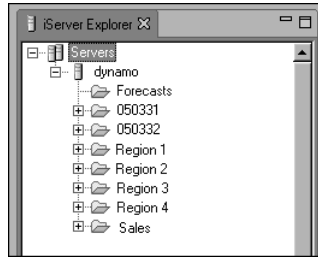
An expanded folder does not reflect changes to the volume. If you or someone else loads a new information object to the volume after you expand a folder, you must collapse and expand the folder to see the changes in the information object.

To specify a basic information object query, perform these tasks:

- Start Information Object Query Builder.
- Select an information object for this query.
- Specify the columns that you want to include in this query.
- Specify any additional information that you want in the query:
  - Specify how you want to sort the data.
  - Specify whether users can provide parameter values when they run the report.
  - Specify the default values of information object parameters.
  - Specify any filtering that you want to restrict the data returned by the information object query.

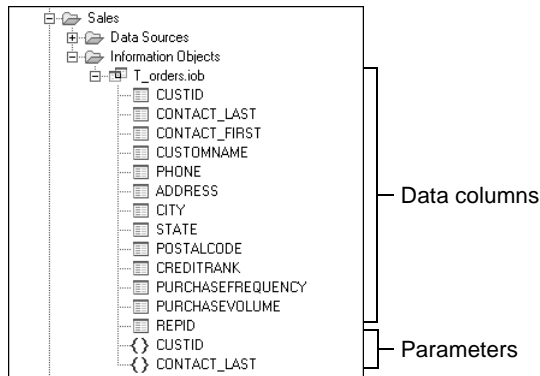
### How to create a basic information object query

- 1 Start Information Object Query Builder. Information Object Query Builder displays the Basic Design interface by default.
- 2 Select an information object and columns by following these steps:
  - 1 In iServer Explorer, expand the Servers node and the Encyclopedia volume node, as shown in Figure 5-6. Expand folders as necessary to view the information objects.



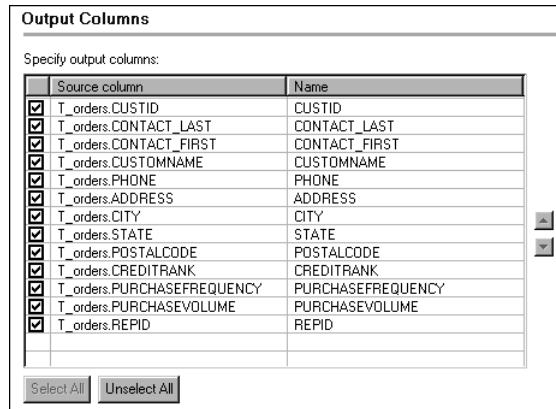
**Figure 5-6** iServer Explorer

- 2 Expand the information object to see the columns and parameters, as shown in Figure 5-7.



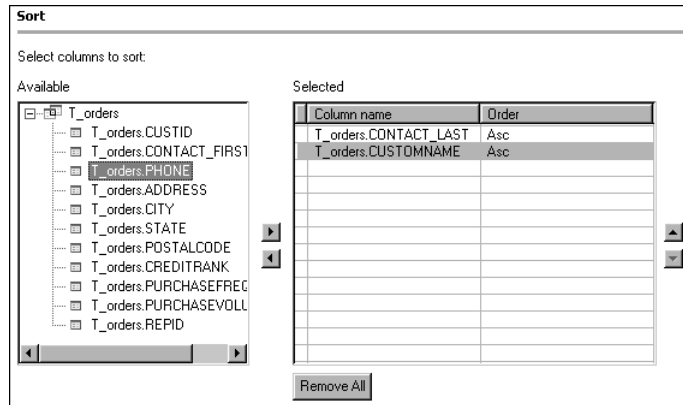
**Figure 5-7** Expanding an information object

- 3 In iServer Explorer, double-click the information object to use in the query. All columns in the information object appear in Output Columns. Beside each column, a check mark indicates that the query uses the column, as shown in Figure 5-8.



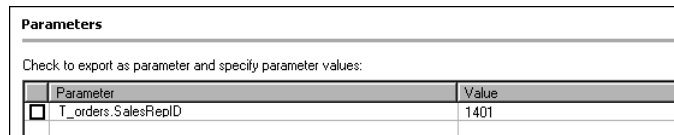
**Figure 5-8** Specifying columns to use in the query

- 4 Deselect any columns that you do not want to include in the output.
- 5 To move a column in the list, select the column, and choose the up or down arrow until the column is in the right position.
- 3 Specify the order in which to sort the data:
  - 1 On Query Design, choose Select Sort Order.
  - 2 On Sort, in Available, expand the information object to see the columns.
  - 3 Double-click a column on which to sort. The column appears in Selected.
  - 4 Under Order, as shown in Figure 5-9, click the field beside the first column on which you want to sort, and use the drop-down list to specify the sort direction for the column:
    - For ascending order, select Asc.
    - For descending order, select Desc.



**Figure 5-9** Specifying data sort order

- 5 Repeat substeps 3 and 4 for any additional columns on which to sort.
- 6 To change the position of a column in the sorting hierarchy, select the column and choose the up or down arrow key. The data returned by the query is sorted first by the column at the top of the list, then by each subsequent column in the list.
- 4 For each available parameter, specify any default parameter values and whether users can specify the values of those parameters.
  - 1 On Query Design, choose Define Parameters. Parameters appears, as shown in Figure 5-10.



**Figure 5-10** Specifying parameters to export

- 2 To enable report users to specify single parameter values, export the parameters. In the left column, select each available parameter that you want to export. Parameters appear on the Requester page.
- 3 Under Value, specify a for each available parameter. If you export the parameter, specifying a default value is optional.
- 5 Specify filters to limit the data returned from the query by following these steps:
  - 1 On Query Design, choose Define Filters.
  - 2 Define the filters. The procedures for adding, editing, and removing filter conditions in Information Object Query Builder Basic Design are the same

as the procedures for adding, editing, and removing filter conditions in Information Object Query Builder Advanced Design.

Choose OK to save the query and return to the report design.

---

## Creating a customized graphical information object query

To specify a graphical information object query using Information Object Query Builder Advanced Design, perform the following tasks:

- Start Information Object Query Builder and enter the Advanced Design perspective.
- Define the query:
  - Select one or more information objects for this query.
  - Define the columns that you want to include in this query.
  - If you have more than one information object, specify the joins to use in this query.
  - Specify any filtering that you want on individual columns.
  - Specify the sort order for the data.
  - Select the columns that you want to group in the query.
  - Specify any aggregate columns that you want to filter in the query.
  - Specify parameters that report users can provide values for when they run the report.
  - Choose OK to save the query and return to the report design.

While developing your query, you can use the following tools:

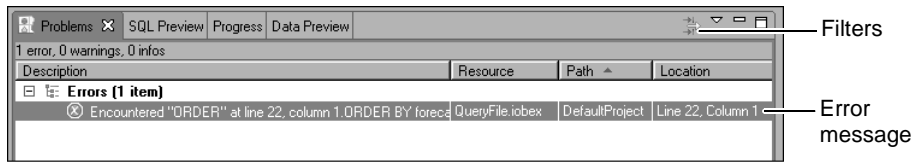
- Problems pane



As you design your queries using Information Object Query Builder Advanced Design, error messages appear in Problems, as shown in Figure 5-11. If an error description is truncated, select the error message. Information Object Query Builder Advanced Design displays an ellipsis button at the end of the description column for that error message. To view the complete description for that error message, choose ellipsis. Line numbers refer to the standard Actuate SQL query, not the extended Actuate SQL query.



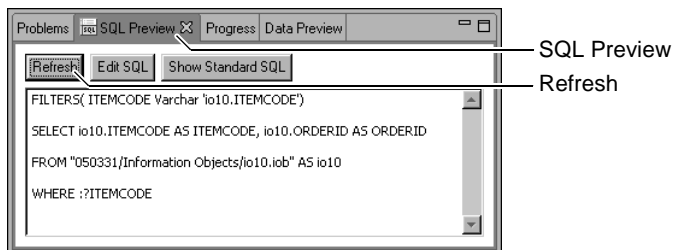
To filter the error messages, choose Filters. For more information about using Problems or Filters, see the *Workbench User Guide* in the Information Object Query Builder Advanced Design online help.



**Figure 5-11** Locating errors and filtering error messages

- **SQL Preview pane**

While using Information Object Query Builder Advanced Design to create a query, you can view the resulting Actuate SQL query statement. To display the query, choose SQL Preview, as shown in Figure 5-12. If you modify the query, choose Refresh to update the display.



**Figure 5-12** Displaying the SQL for an information object query

If you use a dynamic filter in your query, the Actuate SQL query includes a FILTERS clause and :? syntax. The FILTERS clause and :? syntax are part of extended Actuate SQL. The corresponding standard Actuate SQL statements substitute dynamic filters with WHERE clause conditions of the correct data type. To see the standard Actuate SQL query, choose Show Standard SQL in the SQL Preview pane.

Information Object Query Builder uses the standard Actuate SQL statement to validate the syntax of the query. If Information Object Query Builder reports a syntax error, the line number in the error message refers to the syntax that appears in standard SQL. If the query contains syntax errors, use Show Standard SQL to locate and identify the syntax errors. You can then return to viewing the extended Actuate SQL syntax by choosing Show Extended SQL.

- **Progress pane**

Choose Progress to view the progress of long-running tasks. Typically, tasks do not run long enough for Progress to be used. For more information about using Progress, see the *Workbench User Guide* in the Information Object Query Builder Advanced Design online help.

- **Data Preview pane**

Choose Data Preview to view the data rows returned by the query.



## Selecting one or more information objects

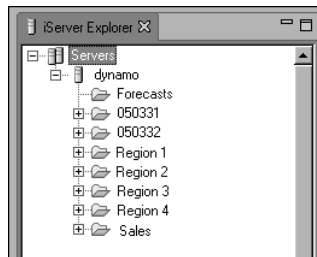
You can browse an iServer Encyclopedia volume using Information Object Query Builder to find and select the information objects for the query.

If you have already chosen your information objects and are re-entering Information Object Query Builder, the editor displays the previously selected information objects. If an information object used by an existing query no longer exists, you must specify a new information object and specify its columns, parameters, and filters.

An expanded folder does not reflect changes to the Encyclopedia volume. If you or someone else loads a new information object in the volume after you expand a folder, you must collapse and expand the folder to see the information object.

### How to select an information object

- 1 In iServer Explorer, expand the Servers node and the Encyclopedia volume node, as shown in Figure 5-13.



**Figure 5-13** Viewing information objects in iServer Explorer

- 2 To view your information objects, expand the appropriate folders.
- 3 Drag the appropriate information objects from iServer Explorer to the upper pane of Query Design. The items appear in the upper pane of Query Design, as shown in Figure 5-14. By default, Information Object Query Builder selects all columns in each information object.



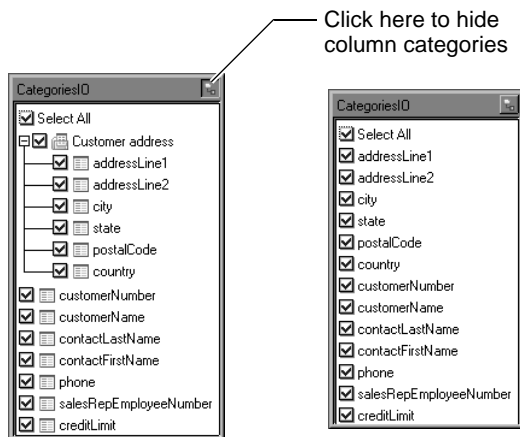
**Figure 5-14** Selected information objects as they appear in Query Design

## Hiding column categories

To help you locate information object columns, the data modeler can organize them into categories. For example, for an information object that returns customer data, the data modeler can create a Customer address category that contains the columns StreetAddress, City, State, and PostalCode.

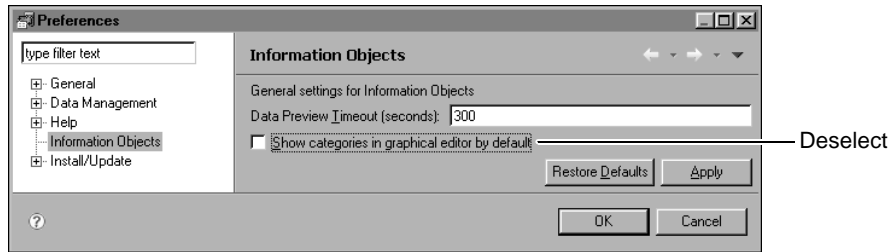


Column categories appear in iServer Explorer, the upper pane of Query Design, and expression builder. To hide column categories in Query Design, select Toggle categories view in the upper right corner of the information object, as shown in Figure 5-15. The information object on the left displays the Customer address category. The information object on the right does not display column categories.



**Figure 5-15** Information object with and without categories displayed

To hide column categories by default, choose Window→Preferences and deselect Show categories in graphical editor by default in Preferences—Information Objects, as shown in Figure 5-16.



**Figure 5-16** Preferences—Information Objects

## Defining output columns

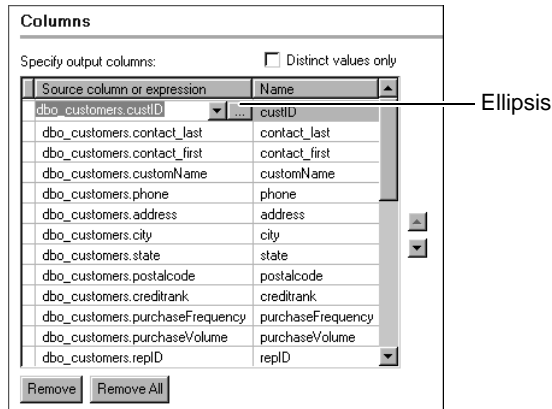
To define the output columns for an information object query, use Query Design—Columns. For example, you can create the following SQL fragment:

```
SELECT ename AS employee, (salary * 12) AS annual_comp
FROM Employees
```

### How to define output columns

- 1 In Query Design, choose Columns.
- 2 In the upper pane of Query Design, select the columns to include and deselect the columns to exclude from the query. To select all columns, select Select All at the top of the listing for that information object. By default, the query includes all columns in an information object. Selected columns appear in Columns.
- 3 In Query Design—Columns:
  - To return only distinct rows, select Distinct values only. Some queries return duplicate rows. In a group of duplicate rows, each selected column contains the same value for all the rows in the group. To return only one row for each group of duplicate rows, select Distinct values only. This setting affects only rows in which all column values match. The query returns rows in which only some of the column values match.
  - To change a column alias, type the new alias in Name. If a column alias contains a special character, such as a period (.) or a space, enclose the alias in quotation marks ("). Do not use column aliases that are identical except for case. For example, do not use both status and STATUS as column aliases.
  - To enter an expression, select the source column, and type the expression or choose ellipsis, as shown in Figure 5-17. Choosing ellipsis opens the expression builder.
  - To change the order of the columns, use the up and down arrows.





**Figure 5-17** Defining output columns

- 4 To define column properties, such as the display name, select the column in Columns, and define the properties in Properties.

#### How to delete output columns

To delete an output column, select the column in Query Design—Columns, and choose Remove. To delete all output columns, choose Remove All.

## Setting column properties

Table 5-4 lists column properties and a description of each property.

**Table 5-4** Column properties

Column property	Can set?	Description
Aggregate Type	Not used	Not used.
Analysis Type	Yes	Analysis type in e.Analysis. If the column contains numeric values or the data type is unknown, the default is Measure. If the column contains data of type TIMESTAMP, the default is Dimension. If the column contains data of type VARCHAR or BOOLEAN, the default is Attribute. If the column is a primary key, a foreign key, or an indexed column in the database, the default is Dimension even if the column contains numeric values.
Category Path	No	Path for column category and subcategories.
Data Type	No	Actuate SQL data type. If the data type is unknown, choose the Synchronize button.
Description	Not used	Not used.

**Table 5-4** Column properties

Column property	Can set?	Description
Display Format	Not used	Not used.
Display Length	Yes	Number of characters to allow for display of column values in report output.
Display Name	Not used	Not used.
Expression	Yes, on the Columns tab	Expression for a computed field.
Has Null	Yes	If column contains NULLs, set to True. Otherwise, set to False.
Heading	Not used	Not used.
Help Text	Not used	Not used.
Horizontal Alignment	Not used	Not used.
Indexed	No	Indicates whether the column is indexed in the data source. True indicates that the column is indexed. False indicates that it is not indexed.
Name	Yes, on the Columns tab	The alias for the column in the information object query.
Text Format	Not used	Not used.
Word Wrap	Not used	Not used.

## Specifying a join

To define the joins for an information object query, use Query Design—Joins. For example, the following SQL fragment specifies that the value of the custID column in the Customers table must match the value of the custID column in the Orders table. The query returns no rows for Customer records having no matching Order records.

```
FROM Customers  
INNER JOIN Orders ON (Customers.custID = Orders.custID)
```

## About joins

A join specifies how to combine data from two information objects. The information objects do not have to be based on the same data source. A join consists of one or more conditions that must all be true. In the resulting SQL SELECT statement, join conditions are linked with AND.

A join can consist of multiple conditions in the following form:

columnA = columnB

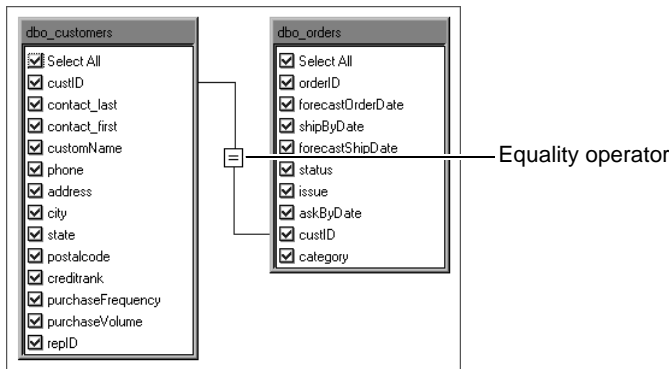
A join can have only one condition that uses an operator other than equality (=) or an expression, for example:

columnA < columnB

Information Object Query Builder Advanced Design does not support right outer joins or full outer joins.

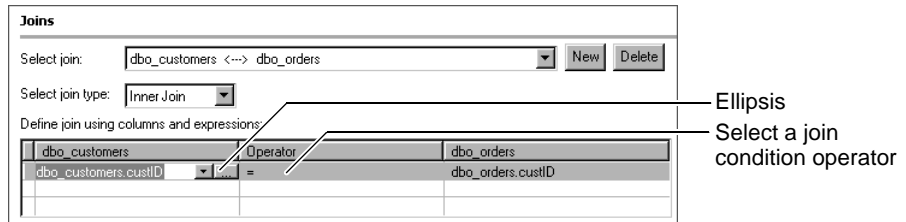
### How to define a join condition

- 1 In Query Design, choose Joins.
- 2 In the upper pane, drag the join column from the first information object, and drop it on the join column in the second information object. The upper pane shows the join condition, like the one in Figure 5-18, and the join columns and operator are listed in Query Design—Joins.



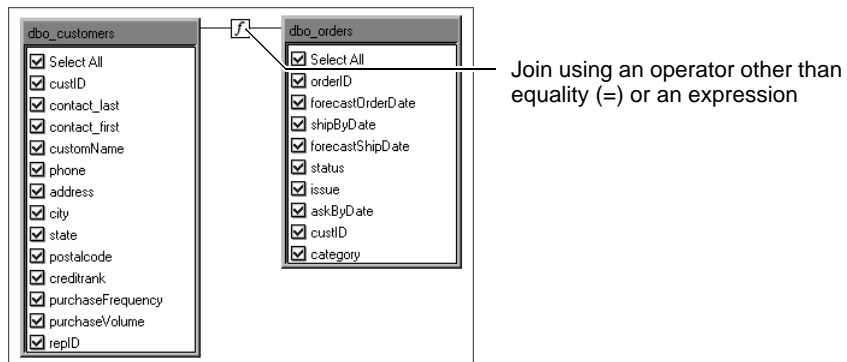
**Figure 5-18** Joined columns from two information objects

- 3 In Query Design—Joins, select the row that describes the new join condition.
- 4 If necessary, select a different join condition operator from the drop-down list. By default, Information Object Query Builder Advanced Design uses the equality operator (=) to relate two columns.
- 5 To change a column name to an expression, select the column name, and type the expression, or choose ellipsis to display the expression builder, as shown in Figure 5-19.



**Figure 5-19** Defining a join condition

If the join has a condition that uses an operator other than equality (=) or an expression, the symbol shown in Figure 5-20 appears in the upper pane of Query Design.



**Figure 5-20** A join that uses an expression or an operator other than equality

- 6 If the join consists of more than one condition, repeat this procedure for the other conditions.
- 7 Choose one of the following join types:
  - Inner join
  - Left outer join
- 8 Optimize the join.

#### How to delete a join condition

To delete a join condition, select the join condition in the upper pane of Query Design, and press Delete.

### Optimizing joins

You can improve a query's performance by optimizing the joins. To optimize a join, you can specify the cardinality of the join. Specifying the cardinality of the join adds the **CARDINALITY** keyword to the Actuate SQL query. If your query is based on two or more information objects that are based on different data sources,

you can also optimize the joins by specifying join algorithms. Figure 5-21 shows how to specify the cardinality of a join and how to specify a join algorithm in Query Design—Joins.

The screenshot shows a dialog box titled "Query Design—Joins". It contains two sections. The first section, "Specify relationship:", has two lines. The first line is "For each value in dbo\_CUSTOMERS, the number of values in dbo\_ORDERS is:" followed by a dropdown menu showing "0 or more". The second line is "For each value in dbo\_ORDERS, the number of values in dbo\_CUSTOMERS is:" followed by a dropdown menu showing "1". The second section, "Specify join algorithm:", has a dropdown menu showing "Dependent". To the right of the dialog box, there are two callout lines. The first line points to the "0 or more" dropdown and is labeled "Applies CARDINALITY keyword". The second line points to the "Dependent" dropdown and is labeled "Specifies join algorithm".

**Figure 5-21** Optimizing a join

When joining information objects built from different data sources, the Actuate SQL compiler chooses a join algorithm. If you have a good understanding of the size and distribution of the data, however, you can specify the join algorithm. Choosing the correct join algorithm can significantly reduce information object query execution time. Actuate SQL supports three join algorithms:

- Dependent
- Merge
- Nested loop

When you join information objects that are built from the same data source, specifying a join algorithm has no effect. The join is processed by the data source.

### About dependent joins

A dependent join is processed in the following way:

- The left side of the join statement is executed, retrieving all the results. The results are then processed one at a time (pipelined).
- For each left side result, the right side of the join is executed, parameterized by the values provided by the current left side row.

A dependent join is advantageous when the cardinality of the left side is small, and the selectivity of the join criteria is both high and can be delegated to the data source. When the cardinality of the left side is high, a dependent join is relatively slow because it repeatedly executes the right side of the join.

Dependent joins can be used for any join criteria, but only join expressions that can be delegated to the right side's data source result in improved selectivity performance.

### About merge joins

A merge join is processed in the following way:

- The left side of the join statement executes, retrieving all the results sorted by the left side data source. The results are then processed one at a time (pipelined).



- The right side of the join statement executes, retrieving all the results sorted by the right side data source. The results are then processed one at a time (pipelined).

A merge join supports only an equijoin. A merge join has much lower memory requirements than a nested loop join and can be much faster. A merge join is especially efficient if the data sources sort the rows.

## About nested loop joins

A nested loop join is processed in the following way:

- The left side of the join statement is executed, retrieving all the results. The results are then processed one at a time (pipelined).
- The right side of the join statement is executed. The results are materialized in memory. For each row on the left side, the materialized results are scanned to find matches for the join criteria.

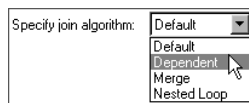
A nested loop join is advantageous when the cardinality of the right side is small. A nested loop join performs well when the join expression cannot be delegated to the data source. A nested loop join supports any join criteria, not just an equijoin.

A nested loop join is a poor choice when the cardinality of the right side is large or unknown, because it may encounter memory limitations. Increasing the memory available to the Integration service removes this limitation. The Integration service parameter Max memory per query specifies the maximum amount of memory to use for an Integration service query. For more information about this parameter, see *Configuring BIRT iServer*.

## How to specify a join algorithm

In Query Design—Joins, select the appropriate join and choose one of the following from the Specify join algorithm drop-down list shown in Figure 5-22:

- Dependent
- Merge
- Nested loop



**Figure 5-22** Specifying the join algorithm

## Filtering data

If an information object query returns more data rows than you need, restrict the number of data rows by using a filter. For example, rather than list all customer

sales, create a filter to select only the sales data for a particular week or only the sales data for a particular region.

Filtering data helps you work effectively with large amounts of data. It enables you to find the necessary pieces of information to answer specific business questions, such as which sales representatives generated the top ten sales accounts, which products generated the highest profit in the last quarter, which customers have not made a purchase in the past 90 days, and so on.

Filtering data can also have a positive effect on processing speed. Limiting the number of data rows can reduce the load on the databases because the information object query does not need to return all the rows every time it is run.

## Creating a filter condition

When you create a filter, you define a condition that specifies which data rows to return. A filter condition is an If expression that must evaluate to true in order for a data row to be returned. For example:

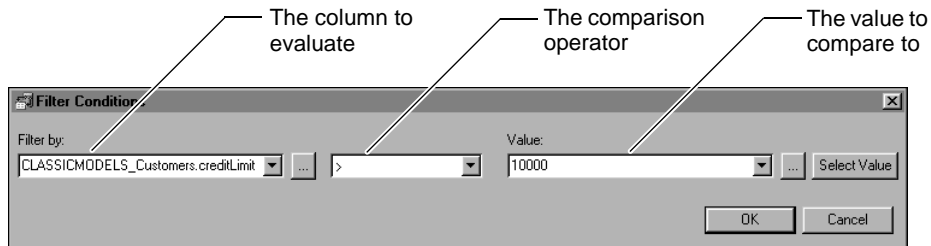
```
If the order total is greater than 10000
If the sales office is San Francisco
If the order date is between 4/1/2008 and 6/30/2008
```

Filter conditions are appended to the information object query's WHERE clause, for example:

```
WHERE OrderTotal > 10000 AND SalesOffice LIKE 'San Francisco%' AND
      OrderDate BETWEEN TIMESTAMP '2008-04-01 00:00:00' AND TIMESTAMP
      '2008-06-30 00:00:00'
```

Figure 5-23 shows an example of a condition defined in Filter Conditions. Filter Conditions helps you define the condition by breaking it down into the following parts:

- The column to evaluate, such as credit limit
- The comparison operator that specifies the type of comparison test, such as > (greater than)
- The value to which all values in the column are compared, such as 10000



**Figure 5-23** Filter Conditions displaying a filter condition

Table 5-5 lists the operators you can use when you create expressions for filter conditions.

**Table 5-5** Operators in filter condition expressions

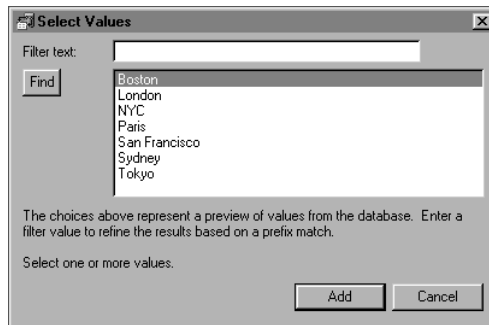
Operator	Use to test whether	Example
BETWEEN	A column value is between two specified values.	Profit BETWEEN 1000 AND 2000
= (Equal to)	A column value is equal to a specified value.	CreditLimit = 100000
> (Greater than)	A column value is greater than a specified value.	Total > 5000
>= (Greater than or equal to)	A column value is greater than or equal to a specified value.	Total >= 5000
IN	A column value is in the specified set of values.	Country IN ('USA', 'Canada', 'Mexico')
IS NOT NULL	A column value is not a null value. A null value means that no value is supplied.	CreditLimit IS NOT NULL
IS NULL	A column value is a null value.	CreditLimit IS NULL
< (Less than)	A column value is less than a specified value.	Total < 5000
<= (Less than or equal to)	A column value is less than or equal to a specified value.	Total <= 5000
LIKE	A column value matches a string pattern.	ProductName LIKE 'Ford%'
NOT BETWEEN	A column value is not between two specified values.	Profit NOT BETWEEN 1000 AND 2000
<> (Not equal to)	A column value is not equal to a specified value.	CreditLimit <> 100000
NOT IN	A column value is not in the specified set of values.	Country NOT IN ('USA', 'Canada', 'Mexico')
NOT LIKE	A column value does not match a string pattern.	ProductName NOT LIKE 'Ford%'

#### How to create a filter condition

- 1 In Query Design, choose Filters.
- 2 On Query Design—Filters, choose New.
- 3 In Filter Conditions, in Filter by, do one of the following:



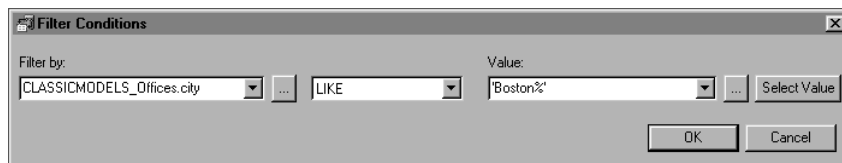
- Select a column from the drop-down list. The drop-down list contains the non-aggregate columns that you defined on Query Design—Columns. To create a filter for an aggregate column, use Query Design—Having.
  - Type an expression.
  - Choose ellipsis to create an expression.
- 4 Select the comparison test, or operator, to apply to the selected column or expression. Depending on the operator you select, Filter Conditions displays one or two additional fields, or a completed filter condition.
  - 5 If you selected an operator that requires a comparison value, specify the value in one of the following ways:
    - Type the value or expression.
    - If you selected a column in Filter by, choose Select Value to select from a list of values. Figure 5-24 shows the selection of Boston from a list of possible sales office values.



**Figure 5-24** Select Values showing the values in the selected column

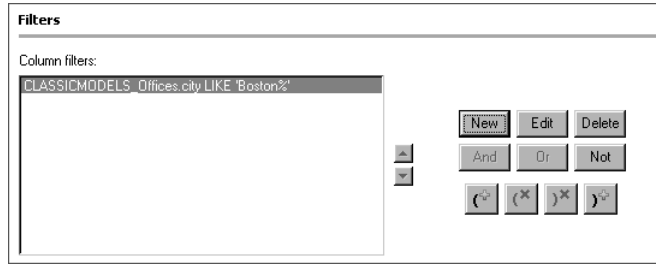


- Select a parameter or column from the drop-down list. You create parameters on Query Design—Parameters.
  - Choose ellipsis to create an expression.
- Figure 5-25 shows the completed filter condition.



**Figure 5-25** Filter Conditions displaying a completed filter condition

Choose OK. The filter condition appears on Query Design—Filters, as shown in Figure 5-26.



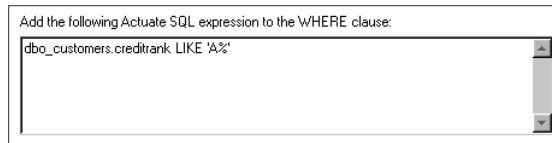
**Figure 5-26** Query Design—Filters displaying a filter condition

- 6 Display the Actuate SQL query. Verify that the filter condition is appended to the WHERE clause and that the syntax is correct, for example:

```
WHERE SalesOffice LIKE 'Boston%'
```

### How to create a filter condition using Actuate SQL

- 1 In Query Design, choose Filters.
- 2 On Query Design—Filters, complete the following tasks:
  - Click in the text box.
  - Type the filter condition using Actuate SQL, as shown in Figure 5-27. If a table or column identifier contains a special character, such as a space, enclose the identifier in double quotation marks (").

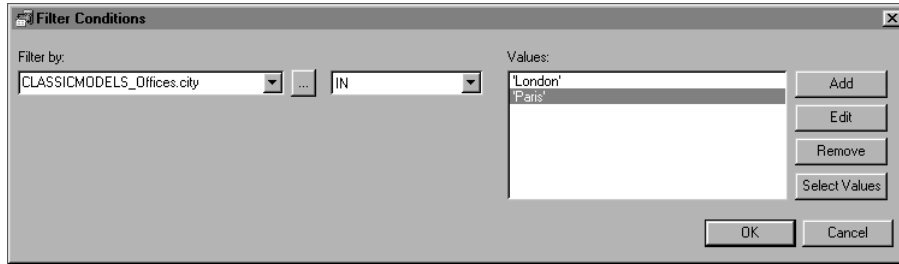


**Figure 5-27** Using Actuate SQL to create a filter condition

### Selecting multiple values for a filter condition

So far, the filter examples specify one comparison value. Sometimes you need to view more data, for example, sales details for several sales offices, not for only one office. To select more than one comparison value, select the IN operator, choose Select Values, then select the values. To select multiple values, press Ctrl as you select each value. To select contiguous values, select the first value, press Shift, and select the last value. This action selects the first and last values and all the values in between.

Figure 5-28 shows the selection of London and Paris from a list of sales office values.



**Figure 5-28** Filter Conditions showing the selection of multiple comparison values

## Excluding data

You use comparison operators, such as = (equal to), > (greater than), or < (less than), to evaluate the filter condition to determine which data to include. Sometimes it is more efficient to specify a condition that excludes a small set of data. For example, you need sales data for all countries except USA. Instead of selecting all the available countries and listing them in the filter condition, simply use the NOT LIKE operator. Similarly, use NOT BETWEEN to exclude data in a specific range, and <> (not equal to) to exclude data that equals a particular value.

For example, the following filter condition excludes orders with amounts between 1000 and 5000:

```
OrderAmount NOT BETWEEN 1000 AND 5000
```

The filter condition in the next example excludes products with codes that start with MS:

```
ProductCode NOT LIKE 'MS%'
```

## Filtering empty or blank values

Some rows display nothing for a particular column. For example, suppose a customer database table contains an e-mail field. Some customers, however, do not supply an e-mail address. The e-mail field for such a customer contains an empty value or a blank value. An empty value, also called a null value, means no value is supplied. A blank value is entered as " (two single quotes without spaces) in the database table field. Blank values apply to string fields only. Null values apply to all data types.

You can create a filter to exclude data rows where a particular column has null or blank values. You use different operators to filter null and blank values.

When filtering to exclude null values, use the IS NOT NULL operator. If you want to view only rows that have null values in a particular column, use IS NULL. For example, the following filter condition excludes customer data where the e-mail column contains null values:

```
email IS NOT NULL
```

The following filter condition displays only rows where the e-mail column contains null values:

```
email IS NULL
```

When filtering blank values, use the NOT LIKE operator with " (two single quotes without spaces) as the operand. For example, to exclude rows with blank values in an e-mail column, specify the following filter condition:

```
email NOT LIKE ''
```

Conversely, to display only rows where the e-mail column contains blank values, create the following condition:

```
email LIKE ''
```

In a report, you cannot distinguish between an empty value and a blank value in a string column. Both appear as missing values. If you want to filter all missing values whether they are null or blank, specify both filter conditions as follows:

```
email IS NOT NULL AND email NOT LIKE ''
```

### **Specifying a date as a comparison value**

When you create a filter condition that compares the date-and-time values in a column to a specific date, the date value you supply must be in the following format regardless of your locale:

```
TIMESTAMP '2008-04-01 12:34:56'
```

Do not use locale-dependent formats such as 4/1/2008.

### **Specifying a number as a comparison value**

When you create a filter condition that compares the numeric values in a column to a specific number, use a period (.) as the decimal separator regardless of your locale, for example:

```
123456.78
```

Do not use a comma (,).

### **Comparing to a string pattern**

For a column that contains string data, you can create a filter condition that compares each value to a string pattern instead of to a specific value. For example, to display only customers whose names start with M, use the LIKE operator and specify the string pattern, M%, as shown in the following filter condition:

```
Customer LIKE 'M%'
```

You can also use the % character to ensure that the string pattern in the filter condition matches the string in the column even if the string in the column has trailing spaces. For example, use the filter condition:

```
Country LIKE 'USA%'
```

instead of the filter condition:

```
Country = 'USA'
```

The filter condition Country LIKE 'USA%' matches the following values:

```
'USA'  
'USA   '  
'USA      '
```

The filter condition Country = 'USA' matches only one value:

```
'USA'
```

You can use the following special characters in a string pattern:

- % matches zero or more characters. For example, %ace% matches any value that contains the string ace, such as Ace Corporation, Facebook, Kennedy Space Center, and MySpace.
- \_ matches exactly one character. For example, t\_n matches tan, ten, tin, and ton. It does not match teen or tn.

To match the percent sign (%) or the underscore character (\_) in a string, precede those characters with a backslash character (\). For example, to match S\_10, use the following string pattern:

```
S\_10
```

To match 50%, use the following string pattern:

```
50\%
```

## Comparing to a value in another column

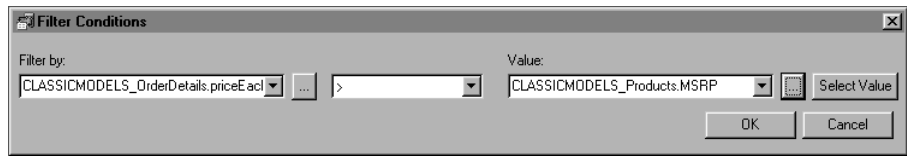
Use a filter condition to compare the values in one column with the values in another column. For example, in a report that displays products, sale prices, and manufacturer suggested retail price (MSRP), you can create a filter condition to compare the sale price and MSRP of each product, and display only rows where the sale price is greater than the MSRP.

### How to compare to a value in another column

- 1 In Query Design, choose Filters.
- 2 On Query Design—Filters, choose New.
- 3 In Filter Conditions, in Filter by, select a column from the drop-down list.
- 4 Select the comparison test, or operator, to apply to the selected column.



- 5 In Value, select a column from the drop-down list. Figure 5-29 shows an example of a filter condition that compares the values in the priceEach column with the values in the MSRP column.



**Figure 5-29** Comparing the values in priceEach with the values in MSRP  
Choose OK.

### Using an expression in a filter condition

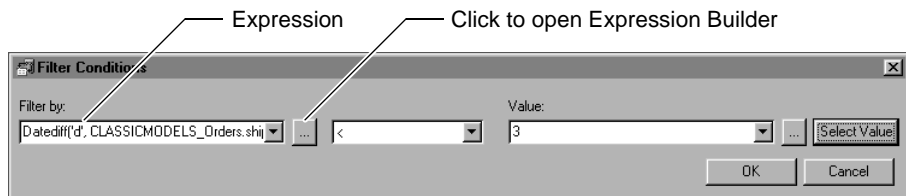


An expression is any combination of Actuate SQL constants, operators, functions, and information object columns. When you create a filter condition, you can use an expression in Filter by, Value, or both. You create an expression with the expression builder.

For example, in an information object query that returns customer and order data, you want to see which orders shipped less than three days before the customer required them. You can use the DATEDIFF function to calculate the difference between the ship date and the required date:

```
DATEDIFF('d', shippedDate, requiredDate) < 3
```

Figure 5-30 shows this condition in Filter Conditions.

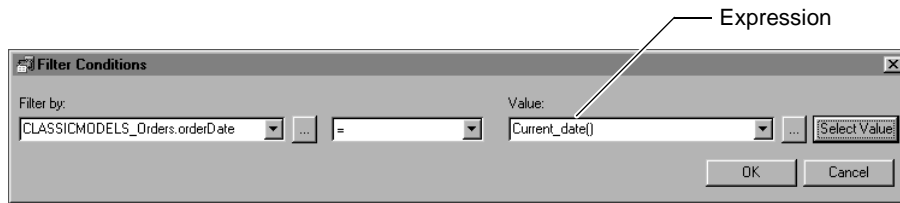


**Figure 5-30** Filter Conditions with expression in Filter by

In an information object query that returns order data, you want to see which orders were placed today. You can use the CURRENT\_DATE function to return today's date:

```
orderDate = CURRENT_DATE( )
```

Figure 5-31 shows this condition in Filter Conditions.

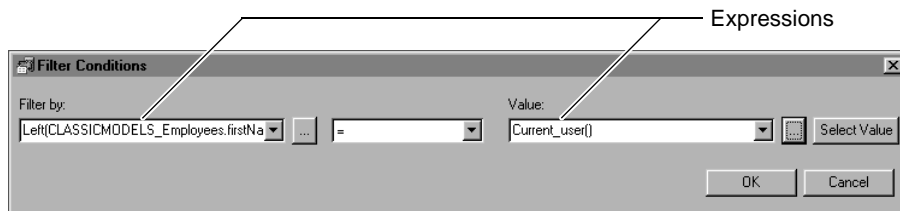


**Figure 5-31** Filter Conditions with expression in Value

In an information object query that returns employee data, you want the information object query to return only data for the user who is currently logged in to the Encyclopedia volume. Use the LEFT function and the concatenation operator ( || ) to construct the employee's user name, and the CURRENT\_USER function to return the name of the user who is currently logged in:

```
LEFT(firstName, 1) || lastName = CURRENT_USER ( )
```

Figure 5-32 shows this condition in Filter Conditions.



**Figure 5-32** Filter Conditions with expressions in Filter by and Value

## Creating multiple filter conditions

When you create a filter, you can define one or more filter conditions. Each condition you add narrows the scope of data further. For example, you can create a filter that returns rows where the customer's credit rank is either A or B and whose open orders total between \$250,000 and \$500,000. Each condition adds complexity to the filter. Design and test filters with multiple conditions carefully. If you create too many filter conditions, the information object query returns no data.

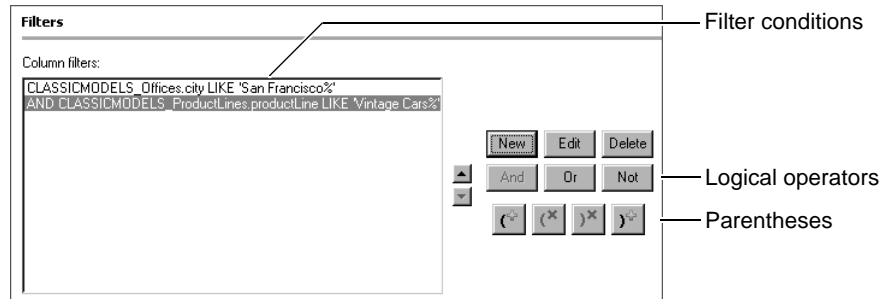
## Adding a condition

You use Query Design—Filters, shown in Figure 5-26, to create one or more filter conditions. To create a filter condition, you choose New and complete the Filter Conditions dialog, shown in Figure 5-25. When you create multiple filter conditions, Information Object Query Builder precedes the second and subsequent conditions with the logical operator AND, for example:

```
SalesOffice LIKE 'San Francisco%' AND  
ProductLine LIKE 'Vintage Cars%'
```

This filter returns only data rows that meet both conditions. Sometimes, you want to create a filter to return data rows when either condition is true, or you want to create a more complex filter. To accomplish either task, use the buttons on the right side of Query Design—Filters, shown in Figure 5-33.

If you create more than two filter conditions and use different logical operators, use the parentheses buttons to group conditions to determine the order in which they are evaluated. Display the query output to verify the results.



**Figure 5-33** Query Design—Filters displaying two conditions

### Selecting a logical operator

As you add each filter condition, the logical operator AND is inserted between each filter condition. You can change the operator to OR. The AND operator means both filter conditions must be true for a data row to be included in the report. The OR operator means only one condition has to be true for a data row to be included. You can also add the NOT operator to either the AND or OR operators to exclude a small set of data. For example, the following filter conditions return only sales data for classic car items sold by the Boston office:

```
SalesOffice LIKE 'Boston%' AND ProductLine LIKE 'Classic Cars%'
```

The following filter conditions return all sales data for the San Francisco and Boston offices:

```
SalesOffice LIKE 'San Francisco%' OR SalesOffice LIKE 'Boston%'
```

The following filter conditions return sales data for all product lines, except classic cars, sold by the San Francisco office:

```
SalesOffice LIKE 'San Francisco%' AND NOT (Product Line LIKE 'Classic Cars%')
```



### Specifying the evaluation order

Information Object Query Builder evaluates filter conditions in the order in which they appear. You can change the order by selecting a filter condition in Query Design—Filters, shown in Figure 5-26, and moving it up or down using the arrow buttons. Filter conditions that you type in the Actuate SQL text box, shown in

Figure 5-27, are preceded by AND and are evaluated last.

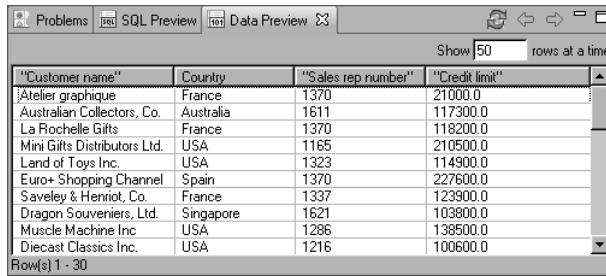
If you define more than two conditions, you can use parentheses to group conditions. For example, A AND B OR C is evaluated in that order, so A and B must be true or C must be true for a data row to be included. In A AND (B OR C), B OR C is evaluated first, so A must be true and B or C must be true for a data row to be included.

The following examples illustrate the difference a pair of parentheses makes.

The following filter contains three ungrouped conditions:

```
Country IN ('Australia', 'France', 'USA') AND  
SalesRepNumber = 1370 OR CreditLimit >= 100000
```

Figure 5-34 shows the first 10 data rows returned by the query. Although the filter specifies the countries Australia, France, and USA and sales rep 1370, the data rows display data for other countries and sales reps. Without any grouped conditions, the filter includes rows that meet either conditions 1 and 2 or just condition 3.



"Customer name"	Country	"Sales rep number"	"Credit limit"
Atelier graphique	France	1370	21000.0
Australian Collectors, Co.	Australia	1611	117300.0
La Rochelle Gifts	France	1370	118200.0
Mini Gifts Distributors Ltd.	USA	1165	210500.0
Land of Toys Inc.	USA	1323	114900.0
Euro+ Shopping Channel	Spain	1370	227600.0
Saveley & Henriot, Co.	France	1337	123900.0
Dragon Souvenirs, Ltd.	Singapore	1621	103800.0
Muscle Machine Inc.	USA	1286	138500.0
Diecast Classics Inc.	USA	1216	100600.0

**Figure 5-34** Results of a complex filter without parentheses grouping

The following filter contains the same three conditions, but this time the second and third conditions are grouped:

```
Country IN ('Australia', 'France', 'USA') AND  
(SalesRepNumber = 1370 OR CreditLimit >= 100000)
```

Figure 5-35 shows the first 10 data rows returned by the query. The Country IN ('Australia', 'France', 'USA') condition must be true, then either the SalesRepNumber = 1370 condition or the CreditLimit >= 100000 condition is true.

"Customer name"	Country	"Sales rep number"	"Credit limit"
Atelier graphique	France	1370	21000.0
Australian Collectors, Co.	Australia	1611	117300.0
La Rochelle Gifts	France	1370	118200.0
Mini Gifts Distributors Ltd.	USA	1165	210500.0
Land of Toys Inc.	USA	1323	114900.0
Saveley & Henriot, Co.	France	1337	123900.0
Muscle Machine Inc.	USA	1286	138500.0
Diecast Classics Inc.	USA	1216	100600.0
Daedalus Designs Imports	France	1370	82900.0
Mini Caravy	France	1370	53800.0

Row(s) 1 - 18

**Figure 5-35** Results of a complex filter with parentheses grouping

## Changing a condition

You can change any of the conditions in Query Design—Filters.

### How to change a filter condition

- 1 In Query Design—Filters, shown in Figure 5-26, select the filter condition. Choose Edit.
- 2 In Filter Conditions, shown in Figure 5-25, modify the condition by changing the values in Filter by, Operator, or Value. Choose OK.

## Deleting a condition

To delete a filter condition, in Query Design—Filters, select the condition. Then, choose Delete. Verify that the remaining filter conditions still make sense.

## Prompting for filter values

You can prompt the report user for a single filter value or for multiple filter values. To prompt for a single value, create an Actuate SQL parameter. To prompt for multiple values, create a dynamic filter.

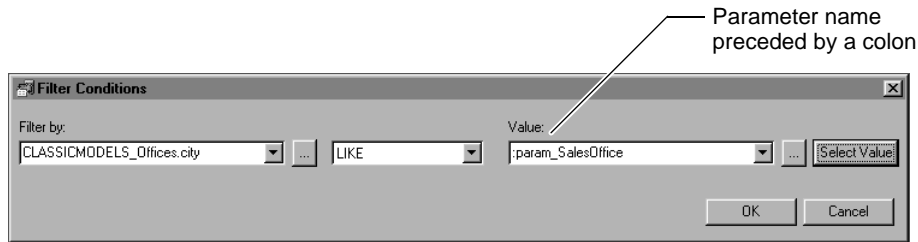
### Prompting for a single value

Use an Actuate SQL parameter to prompt the report user for a single filter value. An Actuate SQL parameter enables the report user to restrict the data rows returned by the information object query without having to modify the WHERE clause. For example, for an information object query that returns sales data by sales office, instead of creating a filter that returns data for a specific office, you can create an Actuate SQL parameter called `param_SalesOffice` to prompt the report user to select an office. The WHERE clause is modified as follows:

```
WHERE SalesOffice LIKE :param_SalesOffice
```

You create Actuate SQL parameters and define their prompt properties on Query Design—Parameters. Prompt properties include the parameter's default value, a list of values for the user to choose from, and whether the parameter is required or optional. Parameters appear in the Value drop-down list in Filter Conditions

with a : (colon) preceding the parameter name, as shown in Figure 5-36.



**Figure 5-36** Filter Conditions with a parameter in the Value field

Do not use an Actuate SQL parameter in a filter condition with the IN operator, for example:

```
Country IN :param_Country
```

Actuate SQL parameters accept only a single value, but the IN operator takes multiple values. Instead, create a dynamic filter.

## Prompting for multiple values

Use a dynamic filter to prompt the report user for multiple values. A dynamic filter can accept a single value, a list of values, or a range of values. For example, you want to prompt the report user for the location of one or more sales offices. You create a dynamic filter for the SalesOffice column. When the report user runs the report, they type the expression Boston | NYC to display data for the Boston and New York sales offices.

Data modelers can create one or more predefined filters when they design an information object. If an information object used in a query has a predefined filter, the predefined filter appears in the Information Object Query Builder as a dynamic filter.

Dynamic filters are also called ad hoc parameters. The syntax that the report user employs to provide a list of values or a range of values is called QBE syntax. For more information about ad hoc parameters and QBE syntax, see *Using Information Console*.

### How to create a dynamic filter

- 1 In Query Design, choose Filters.
- 2 On Query Design—Filters, scroll down to see Select dynamic filters, as shown in Figure 5-37.

Select dynamic filters:

Column or expression	Data type	Prompt editor
io_dynamic_par_dynamic_picklist.CATEGORY_ORDER	Double	
io_dynamic_par_dynamic_picklist.PRICEQUOTE	Integer	
io_dynamic_par_dynamic_picklist.CATEGORY	Varchar	
io_dynamic_par_dynamic_picklist.SHIPBYDATE	Timestamp	
io_dynamic_par_dynamic_picklist.CATEGORY_1	Double	

Remove Remove All

**Figure 5-37** Selecting a dynamic filter



**3** To create a new dynamic filter, click in the first blank row in the Column or expression column. Use the drop-down menu to add a column or choose ellipsis to create an expression.

**4** For each dynamic filter, you can perform the following steps:

**1** To set or change the data type, click the filter's row under Data type, and select a data type from the drop-down list.



**2** To specify how the user is prompted when running the report, click the filter's row under Prompt editor. This action displays Prompt editor, where you can change the prompt properties for the filter.

#### How to remove a dynamic filter

**1** In Query Design, choose Filters.

**2** On Query Design—Filters, complete the appropriate task:

- To delete a single dynamic filter, select the filter in Select dynamic filters, and choose Remove.
- To delete all dynamic filters, choose Remove All.

### Setting dynamic filter prompt properties

When you create a dynamic filter, use Prompt editor to specify the filter's display control type, list of values, and default value. You create a list of values by specifying the values or by typing an Actuate SQL query that retrieves the values. Information Object Query Builder does not validate the query. You can specify the filter values as well as the values displayed to the report user. If you type a query, the query must meet the following requirements:

- The query must retrieve one or two columns from an information object or map, for example:

```
SELECT DISTINCT CAST(custID AS VARCHAR(50)), customName
FROM "MyInformationObject.iob"
ORDER BY 2
```

The first column contains the filter values and must be of string data type. The second column contains the values displayed to the report user. The

information object or map must reside in the same volume as the report executable. You must use an absolute path to reference the information object or map. If the information object or map defines a parameter, you must provide a value for the parameter, for example:

```
SELECT DISTINCT CAST(custID AS VARCHAR(50)), customName
FROM "MyInformationObject.iob" ['CA']
ORDER BY 2
```

- The query must not contain a WITH clause.

The filter values are interpreted as QBE expressions. Certain characters, for example, the comma (,) and the pipe sign (|), are interpreted as operators in a QBE expression. For example, the QBE expression:

```
16M x 1 Dynamic Ram, 3.3 volts
```

is interpreted as:

```
WHERE description LIKE '16M x 1 Dynamic Ram%'
OR description LIKE '3.3 volts%'
```

If you want these characters to be interpreted literally, enclose the strings in four single quotation marks (''''') as shown in the following Actuate SQL queries:

- To match a string exactly:

```
SELECT '''' || description || ''''
FROM "MyInformationObject.iob"
```

- To match a string using the LIKE operator:

```
SELECT '''' || description || '%''''
FROM "MyInformationObject.iob"
```

|| is the concatenation operator.

The values returned by the query appear when a report user specifies a value for the dynamic filter when running the report in the Encyclopedia volume. The values do not appear when running the report in the report designer. In this environment, the prompt is a text box.

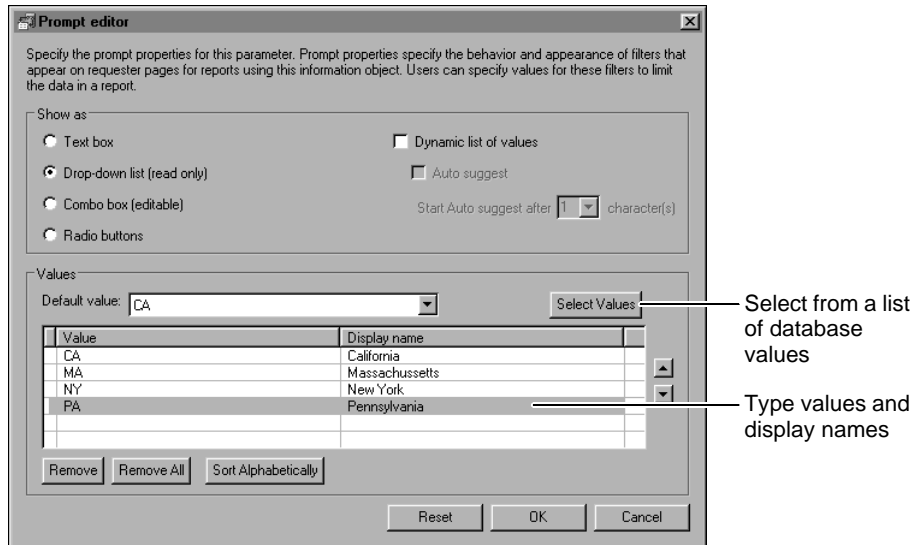
### How to set the prompt properties of a dynamic filter

Setting the prompt properties of a dynamic filter affects how the report user sees the filter when running the report.

- 1 In Query Design, choose Filters.
- 2 In Query Design—Filters, scroll down to Select Dynamic Filters.
- 3 In the row for the filter, choose Prompt editor.
- 4 On Prompt editor, complete the following tasks, as shown in Figure 5-38:



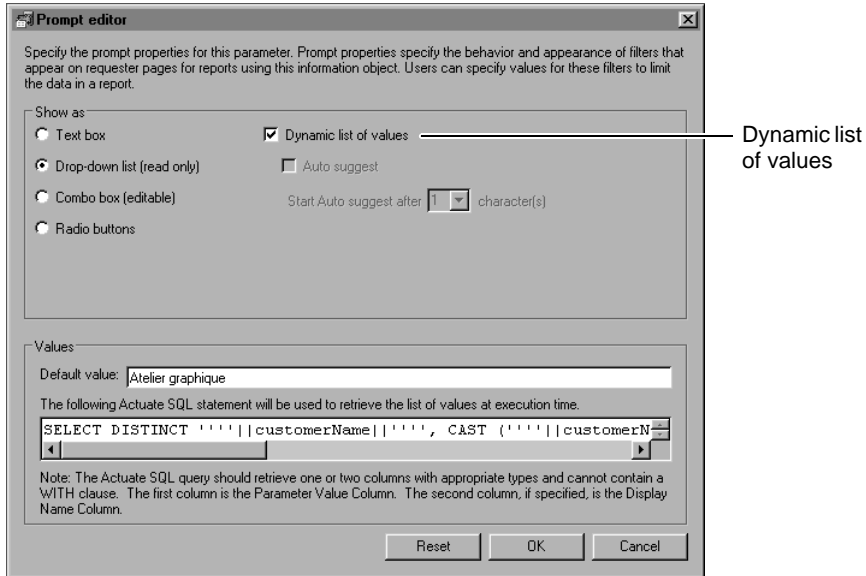




**Figure 5-38** Typing values and display names for a filter

- In Show as, select the display control type. The choices available and appearance of the page depend on the display control type you select.
- If you choose a display control type other than Text box, you can specify a list of values for the user to choose from by typing the values and, optionally, the display names. Alternatively, you can select from a list of database values by choosing Select Values.
- In Default value, specify the default value. The default value can be a QBE expression.

To create an Actuate SQL query that retrieves the values, select Dynamic list of values, as shown in Figure 5-39, and type the query.



**Figure 5-39** Creating an Actuate SQL query to generate values for a filter

If you select Combo box (editable), Dynamic list of values, and Auto suggest, a list appears after the report user types the number of characters specified in Start Auto suggest after N character(s). The list contains values that begin with the characters the user typed. For example, if the user typed 'Atel and N=4, the list contains the value 'Atelier graphique'. In this case, the query that retrieves the values must select two columns, a value column and a display name column.

Choose OK.

## Grouping data

A GROUP BY clause groups data by column value. For example, consider the following query:

```
SELECT orderNumber
FROM OrderDetails
```

The first 10 data rows returned by this query are as follows:

```
orderNumber
10100
10100
10100
10100
10101
10101
10101
10101
10101
10102
10102
```

Each order number appears more than once. For example, order number 10100 appears four times. If you add a GROUP BY clause to the query, you can group the data by order number so that each order number appears only once:

```
SELECT orderNumber
FROM OrderDetails
GROUP BY orderNumber
```

The first 10 data rows returned by this query are as follows:

```
orderNumber
10100
10101
10102
10103
10104
10105
10106
10107
10108
10109
```

Typically, you use a GROUP BY clause to perform an aggregation. For example, the following query returns order numbers and order totals. The Total column is an aggregate column. An aggregate column is a computed column that uses an aggregate function such as AVG, COUNT, MAX, MIN, or SUM.

```
SELECT orderNumber, (SUM(quantityOrdered*priceEach)) AS Total
FROM OrderDetails
GROUP BY orderNumber
```

Figure 5-43 shows the first 10 data rows returned by the information object query. The data is grouped by order number and the total for each order appears.

## Creating a GROUP BY clause

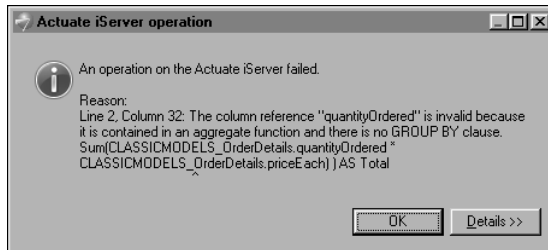
By default, Information Object Query Builder creates a GROUP BY clause automatically. If you prefer, you can create a GROUP BY clause manually.

### Creating a GROUP BY clause automatically

When an information object query's SELECT clause includes an aggregate column and one or more non-aggregate columns, the non-aggregate columns must appear in the GROUP BY clause. If the non-aggregate columns do not appear in the GROUP BY clause, Information Object Query Builder displays an error message. For example, consider the following information object query:

```
SELECT orderNumber, (SUM(quantityOrdered * priceEach)) AS Total
FROM OrderDetails
```

When you attempt to compile the information object query, the error message shown in Figure 5-40 appears in the Problems view.

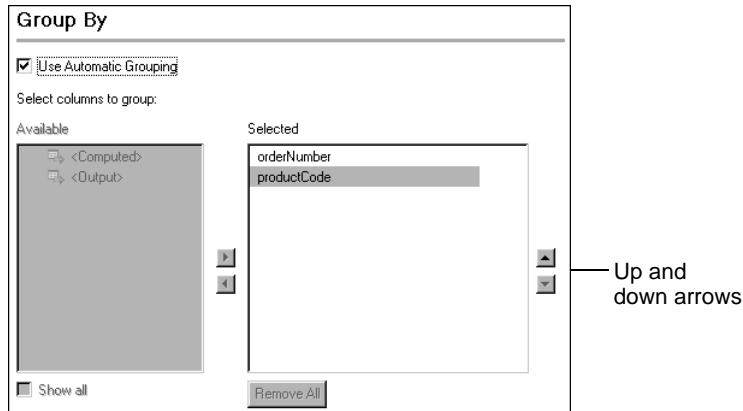


**Figure 5-40** Information object query requires a GROUP BY clause

To avoid this problem, Information Object Query Builder automatically creates a GROUP BY clause:

```
SELECT orderNumber, (SUM(quantityOrdered * priceEach)) AS Total
FROM OrderDetails
GROUP BY orderNumber
```

If more than one column appears in the GROUP BY clause, you can change the order of the columns using the up and down arrows in Group By, as shown in Figure 5-41.



**Figure 5-41** Changing the order of GROUP BY columns

## Creating a GROUP BY clause manually

If automatic grouping does not generate the desired SQL query, create the GROUP BY clause manually. Create the GROUP BY clause manually if you want to group on a column that does not appear in the SELECT clause, for example:

```
SELECT (SUM(quantityOrdered * priceEach)) AS Total
FROM OrderDetails
GROUP BY orderNumber
```

### How to create a GROUP BY clause manually

- 1 In Query Design, choose Group By.
- 2 In Query Design—Group By, deselect Use Automatic Grouping.
- 3 In Available, expand the Computed and Output nodes to view the available columns.

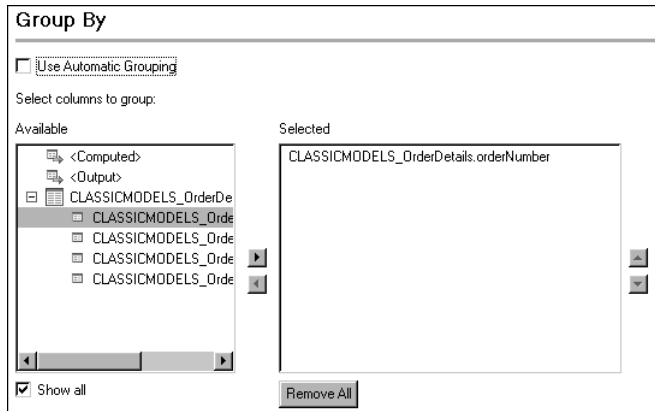
By default, Information Object Query Builder displays only output columns and non-aggregate computed fields. To group on a column that is not an output column, choose Show all.



- 4 In Available, select the appropriate column, and choose Select. This action moves the column name to Selected, as shown in Figure 5-42.



- 5 Repeat the previous step for each GROUP BY column.
- 6 To change the order of the GROUP BY columns, select a column in Selected, and use the up or down arrow.



**Figure 5-42** Selecting a GROUP BY column

## Removing a column from the GROUP BY clause

By default, Information Object Query Builder removes GROUP BY columns automatically. If you disable automatic grouping, you must remove GROUP BY columns manually.

### Removing a GROUP BY column automatically

Information Object Query Builder automatically removes a column from the GROUP BY clause when:

- You remove the column from the SELECT clause. For example, consider the following information object query:

```
SELECT orderNumber, productCode, (SUM(quantityOrdered *
    priceEach)) AS Total
FROM OrderDetails
GROUP BY orderNumber, productCode
```

You remove the productCode column from the SELECT clause. Information Object Query Builder automatically removes productCode from the GROUP BY clause:

```
SELECT orderNumber, (SUM(quantityOrdered * priceEach)) AS Total
FROM OrderDetails
GROUP BY orderNumber
```

- You manually add a column to the GROUP BY clause that does not appear in the SELECT clause and then enable automatic grouping. For example, consider the following information object query:

```
SELECT orderNumber, (SUM(quantityOrdered * priceEach)) AS Total
FROM OrderDetails
GROUP BY orderNumber, productCode
```

The `productCode` column appears in the `GROUP BY` clause but not in the `SELECT` clause. You enable automatic grouping. Information Object Query Builder automatically removes `productCode` from the `GROUP BY` clause:

```
SELECT orderNumber, (SUM(quantityOrdered * priceEach)) AS Total
FROM OrderDetails
GROUP BY orderNumber
```

Information Object Query Builder automatically removes the `GROUP BY` clause when:

- You remove all aggregate columns from the `SELECT` clause. For example, consider the following information object query:

```
SELECT orderNumber, (SUM(quantityOrdered * priceEach)) AS Total
FROM OrderDetails
GROUP BY orderNumber
```

You remove the aggregate column `SUM(quantityOrdered * priceEach)` from the `SELECT` clause. Information Object Query Builder automatically removes the `GROUP BY` clause:

```
SELECT orderNumber
FROM OrderDetails
```

- You remove all non-aggregate columns from the `SELECT` clause. For example, consider the following information object query:

```
SELECT orderNumber, (SUM(quantityOrdered * priceEach)) AS Total
FROM OrderDetails
GROUP BY orderNumber
```

You remove the `orderNumber` column from the `SELECT` clause. Information Object Query Builder automatically removes the `GROUP BY` clause:

```
SELECT (SUM(quantityOrdered * priceEach)) AS Total
FROM OrderDetails
```

## Removing a `GROUP BY` column manually

If you disable automatic grouping, you must remove `GROUP BY` columns manually.

### How to remove a `GROUP BY` column manually

- 1 In Query Design, choose Group By.
- 2 In Query Design—Group By, complete one of the following tasks:



- Select the column in Selected, and choose Deselect.
- To remove all Group By columns, choose Remove All.

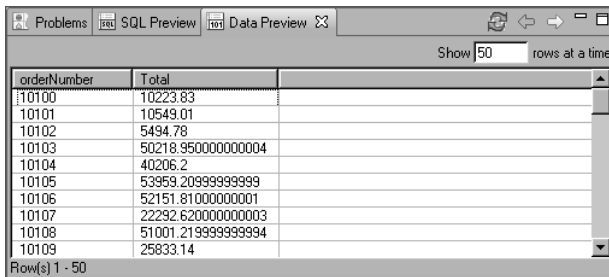
## Filtering on an aggregate column

If an information object query includes a GROUP BY clause, you can restrict the data rows the query returns by adding a HAVING clause. The HAVING clause places a filter condition on one or more aggregate columns. An aggregate column is a computed column that uses an aggregate function such as AVG, COUNT, MAX, MIN, or SUM, for example SUM(quantityOrdered \* priceEach).

For example, the following query returns order numbers and order totals. The Total column is an aggregate column. The data is grouped by order number and no filter condition is placed on the Total column.

```
SELECT orderNumber, (SUM(quantityOrdered * priceEach)) AS Total
FROM OrderDetails
GROUP BY orderNumber
```

Figure 5-43 shows the first 10 data rows returned by this information object query.



orderNumber	Total
10100	10223.83
10101	10549.01
10102	5494.78
10103	50218.950000000004
10104	40206.2
10105	53959.209999999999
10106	52151.810000000001
10107	22292.620000000003
10108	51001.219999999994
10109	25833.14

**Figure 5-43** Data rows returned by query with GROUP BY clause

You can add a HAVING clause to this information object query to place a filter condition on the Total column. The following information object query returns only rows for which the order total is greater than or equal to 50000:

```
SELECT orderNumber, (SUM(quantityOrdered * priceEach)) AS Total
FROM OrderDetails
GROUP BY orderNumber
HAVING SUM(quantityOrdered * priceEach) >= 50000
```

Figure 5-44 shows the first 10 data rows returned by this information object query.

The procedures for creating filter conditions for aggregate columns are identical to the procedures for creating filter conditions for other columns, except that you use Query Design—Having instead of Query Design—Filters. Filter conditions that you create using Query Design—Filters are evaluated before filter conditions that you create using Query Design—Having. In other words, filter conditions in the WHERE clause are applied before filter conditions in the HAVING clause.



orderNumber	Total
10103	50218.950000000004
10105	53959.209999999999
10106	52151.810000000001
10108	51001.219999999994
10122	50824.659999999996
10126	57131.92
10127	58841.35
10135	55601.840000000004
10142	56052.560000000001
10145	50342.74

**Figure 5-44** Data rows returned by query with GROUP BY and HAVING clauses

## Defining parameters

An Actuate SQL parameter is a variable that is used in an information object query. When a report developer runs the report on the desktop, they provide a value for this variable. When a user runs the report in an Encyclopedia volume, the user provides a value for this variable on the Requester page in Information Console.

For example, the following Actuate SQL query uses the parameters lastname and firstname in the WHERE clause:

```
WITH ( lastname VARCHAR, firstname VARCHAR )
SELECT lname, fname, address, city, state, zip
FROM customerstable
WHERE (lname = :lastname) AND (fname = :firstname)
```

If an Actuate SQL query defines a parameter in a WITH clause but does not use the parameter, the query does not return any rows if no value is provided for the parameter when the report runs. For example, the following query does not return any rows if no values are provided for the lastname and firstname parameters when the report runs:

```
WITH ( lastname VARCHAR, firstname VARCHAR )
SELECT lname, fname, address, city, state, zip
FROM customerstable
```

### How to define a parameter

- 1 In Query Design, choose Parameters.
- 2 In Query Design—Parameters, click the top empty line, and complete the following tasks:
  - In Parameter, type the name of the parameter. If a parameter name contains a special character, such as a period (.) or a space, enclose the name in double quotation marks (").
  - In Data type, select a data type from the drop-down list.

- 

- To change the order of the parameters, use the up or down arrow.
- To use Prompt editor to specify the parameter's prompt properties, choose Prompt editor, as shown in Figure 5-45.

**Figure 5-45** Choosing Prompt editor to specify a parameter's prompt properties

- ## How to delete a parameter

- 146 Designing Spreadsheets

## Specifying a parameter's prompt properties

Use Prompt editor to specify a parameter prompt's properties, including display control type, list of values, and default value. You can specify the parameter values and, if desired, a corresponding set of display values that report users choose from. You create a list of values by typing the values or by typing an Actuate SQL query that retrieves the values.

The query must meet the following requirements:

- The query must retrieve one or two columns from an information object or map, as shown in the following example:

```
SELECT DISTINCT custID, customName
FROM "MyInformationObject.iob"
ORDER BY 2
```

The first column contains the parameter values. The second column contains the values that are displayed to the report user. The information object or map must reside in the same volume as the report executable. You must use an absolute path to reference the information object or map. If the information object or map defines a parameter, you must provide a value for the parameter, as shown in the following example:

```
SELECT DISTINCT custID, customName
FROM "MyInformationObject.iob" ['CA']
ORDER BY 2
```

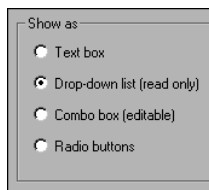
- The first column's data type must match the parameter's data type.
- The query must not contain a WITH clause.

The query editor does not validate the query. The values returned by the query appear when a user specifies a value for the parameter. The values do not appear when a report developer specifies a value for the parameter on the desktop.

### How to specify a parameter's prompt properties

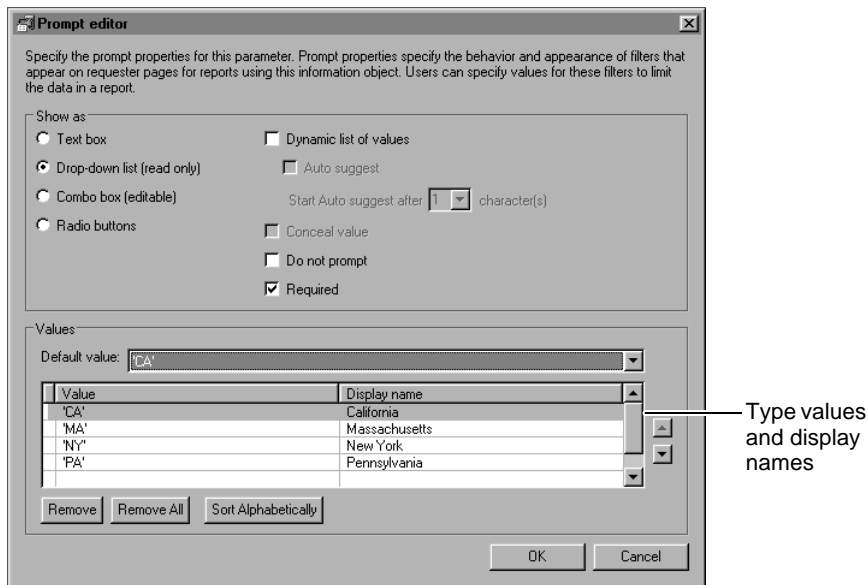


- 1 Locate the appropriate parameter in Query Design—Parameters, and choose Prompt editor.
- 2 On Prompt editor, in Show as, select the method of prompting the user, as shown in Figure 5-46. If you use a type of display other than text box, you can specify a list of values for the user to choose from.



**Figure 5-46** Selecting the method of prompting the user

You can create a list of values by typing the parameter values and, optionally, the display names, as shown in Figure 5-47. If you do not provide display names, the parameter values are displayed to the user.



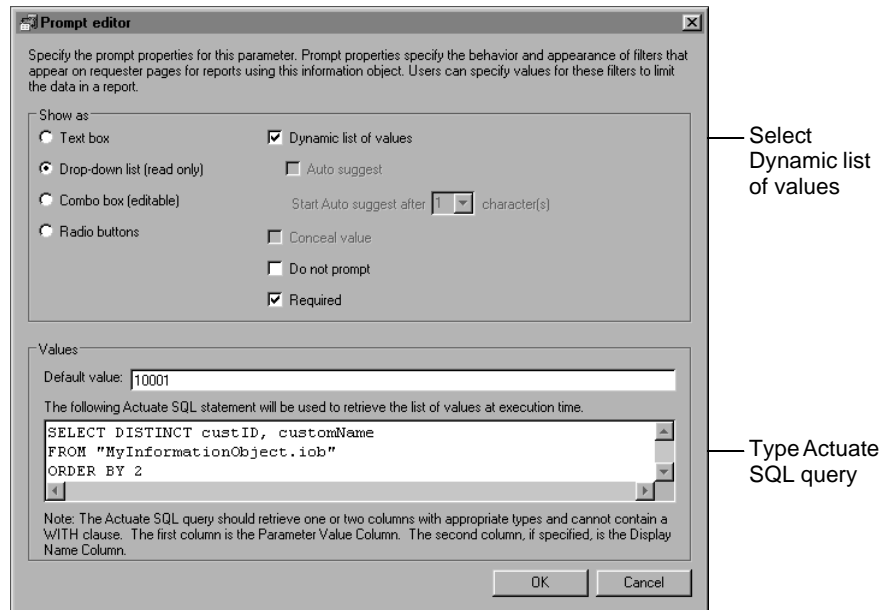
**Figure 5-47** Typing a list of values and display names

You can create an Actuate SQL query that retrieves the parameter values or both the values and the corresponding display names. If the query has two columns, the values in the second column are used as the display names. To use a query to create the list of values, select Dynamic list of values, as shown in Figure 5-48, and type the query.

If you select Combo box (editable), Dynamic list of values, and Auto suggest, a drop-down list appears after the report user types the number of characters specified in Start Auto suggest after N character(s). The list contains values that begin with the characters the user typed. For example, if the user typed 'Atel and N=4, the list contains the value 'Atelier graphique'. In this case, the query that retrieves the values must select two columns, a value column and a display name column.

- 3 In Default value, specify the default value.
- 4 You also can specify values for the following additional properties:
  - Conceal value
  - Do not prompt
  - Required

When you finish specifying the property values for the prompt, choose OK.



**Figure 5-48** Specifying an Actuate SQL query to provide a dynamic list of values

## Setting parameter properties

Table 5-6 lists parameter properties and provides a description of each property.

**Table 5-6** Parameter properties

Parameter property	Can set?	Description
Conceal Value	Yes, in Prompt editor	Visibility of the value that the user provides for this parameter. To conceal the value, set to True. To display the value, set to False. This parameter property applies only to parameters with the varchar data type and the text box display type.
Data Type	Yes, on the Parameters tab	Parameter's data type.
Default Value	Yes, in Prompt editor	Parameter's default value. If a parameter does not have a default value, and the Required property is set to False, the parameter takes one of the following values if the user does not provide a value:

(continues)

**Table 5-6** Parameter properties (continued)

Parameter property	Can set?	Description
Default Value (continued)	Yes, in Prompt editor (continued)	<ul style="list-style-type: none"> <li>■ 0 if the parameter is of type decimal, double, or integer</li> <li>■ Empty string if the parameter is of the varchar data type</li> <li>■ Current date and time if the parameter is of the timestamp data type</li> </ul>
Description	Not used	Not used.
Display Control Type	Yes, in Prompt editor	Control type for this parameter. The options are text box, read-only drop-down list, editable drop-down list, or radio buttons.
Display Format	Not used	Not used.
Display Length	Not used	Not used.
Display Name	Not used	Not used.
Do Not Prompt	Yes, in Prompt editor	Visibility of the parameter to the user. To hide this parameter, set to True. To display the parameter, set to False.
Heading	Not used	Not used.
Help Text	Not used	Not used.
Horizontal Alignment	Not used	Not used.
Name	Yes, on the Parameters tab	Parameter name.
Parameter Mode	Yes	Setting for parameters that are in stored procedures and ODA data source queries to specify the input or output type of the parameter. The options are Input, Output, InputAndOutput, or ReturnValue. ReturnValue is used only for stored procedures and is equivalent to Output.
Required	Yes, in Prompt editor	Indicator of whether the parameter is required. To require a value for this parameter, set to True. Otherwise, set to False.
Size	Yes	The size of the parameter if the parameter data type is varchar. Otherwise, not used. Must be set if size is greater than 1300.

## Setting source parameters

A source parameter is a parameter that is defined in an information object from which you are building an information object query. If a query contains an information object with a parameter, Query Design—Parameters has a Source parameter field.

You can set a source parameter to one of the following types of values:

- A single scalar value
- A local parameter in the information object query that you are creating

You cannot set a source parameter to a column reference, such as `ORDERS.ORDERID`, or an Actuate SQL expression.

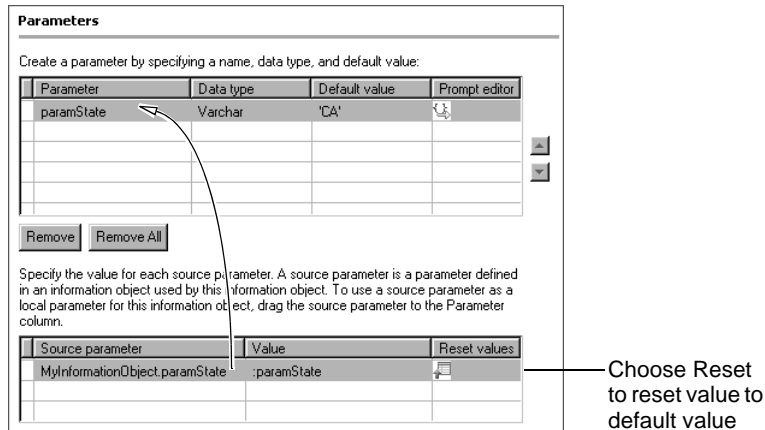
When you set a source parameter to a local parameter, you can indicate that the local parameter inherits the values of its prompt properties from the source parameter. The available prompt properties are Conceal Value, Default Value, Display Control Type, Do Not Prompt, and Required. If you specify that the local parameter inherits its prompt property values from the source parameter, and prompt property values for the source parameter change, the changes are propagated to the local parameter. For example, if the display control type for the source parameter changes from text box to read-only drop-down list, the display control type for the local parameter also changes from text box to read-only drop-down list.

If you change a prompt property value for a local parameter, its prompt property values are no longer inherited from the source parameter. For example, if you change the display control type for the local parameter to editable drop-down list, and the display control type for the source parameter later changes to text box, the change is not propagated to the local parameter. To reinstate inheritance, choose Reset in Prompt editor. Choosing Reset returns all property values in the local parameter to inherited values, and the local parameter inherits any future changes to property values in the source parameter.

To set source parameters, use Query Design—Parameters. To define a local parameter and set a source parameter to the local parameter in one step, drag the source parameter from Source parameter, and drop it in Parameter, as shown in Figure 5-49.

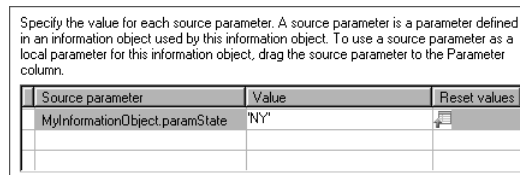
### How to set a source parameter

- 1 In Query Design, choose Parameters.
- 2 On Parameters, complete the following tasks:
  - In Source parameter, select the appropriate parameter.
  - In Value, complete one of the following tasks:



**Figure 5-49** Setting a source parameter to a local parameter

- ❑ Choose a parameter from the drop-down list. The drop-down list contains the local parameters for the information object query you are building.
- ❑ Type a value, as shown in Figure 5-50:
  - ❑ If Value is a string, enclose the string in single quotation marks, as shown in the following example:  
   'New York City'
  - ❑ If Value is a timestamp, it must be in the following form:  
   TIMESTAMP '2001-02-03 12:11:10'
  - ❑ If Value is a number, use a period (.) as the decimal separator, as shown in the following example:  
   123456.78
  - ❑ If Value is a parameter, precede the parameter name with a colon (:).



**Figure 5-50** Providing a value for a source parameter

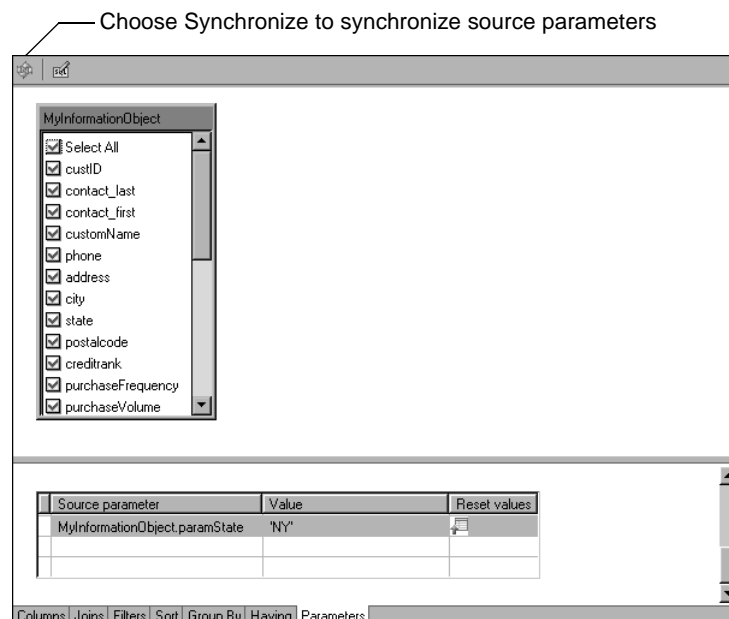
## Synchronizing source parameters



You must synchronize source parameters when parameters in a source information object are added, removed, or reordered, or their data types or other



properties change. To synchronize source parameters, choose Synchronize in the upper pane of Query Design, as shown in Figure 5-51. Synchronizing source parameters refreshes the list of source parameters on Query Design—Parameters.



**Figure 5-51** Synchronizing source parameters

## Creating a textual information object query

Use the Actuate SQL text editor if either of the following conditions is true:

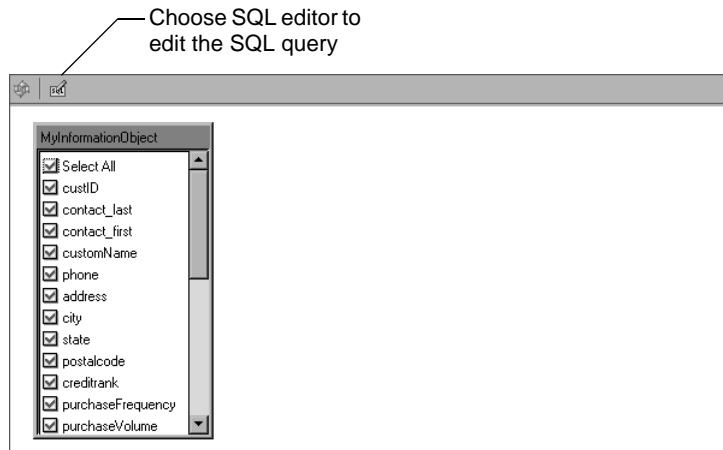
- Information Object Query Builder Advanced Design does not generate the desired Actuate SQL query, so you must edit the query. For example, if the query includes OR or UNION, you must use the Actuate SQL text editor to edit the query.
- You want to type or paste an Actuate SQL query instead of creating it graphically.

If you save a query in the Actuate SQL text editor, you cannot modify the query in the graphical editor.

To display the Actuate SQL text editor, complete one of the following tasks:



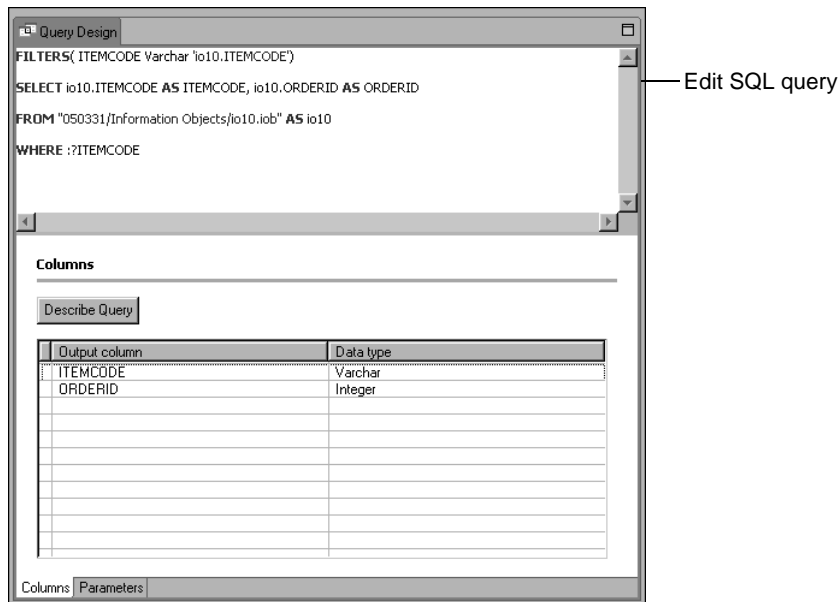
- In Query Design, choose SQL editor, as shown in Figure 5-52.



**Figure 5-52** Choosing SQL editor to edit an Actuate SQL query

- In SQL Preview, choose Edit SQL.

Figure 5-53 shows the Actuate SQL text editor.



**Figure 5-53** Editing the Actuate SQL query in the text editor

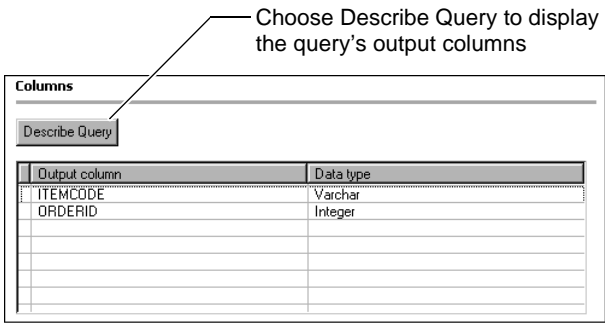
You edit the query in the upper pane of the Actuate SQL text editor. The lower pane displays output columns or parameters.

When you edit a query in the SQL text editor, do not use table and column aliases that are identical except for case. For example, do not use both status and STATUS as column aliases.

Use absolute paths in the Information Object Query Builder because the report executable and information object may not be in the same Encyclopedia volume.

## Displaying output columns

In SQL Text Editor—Columns, to display the query’s output columns and the data type for each column, choose Describe Query, as shown in Figure 5-54.

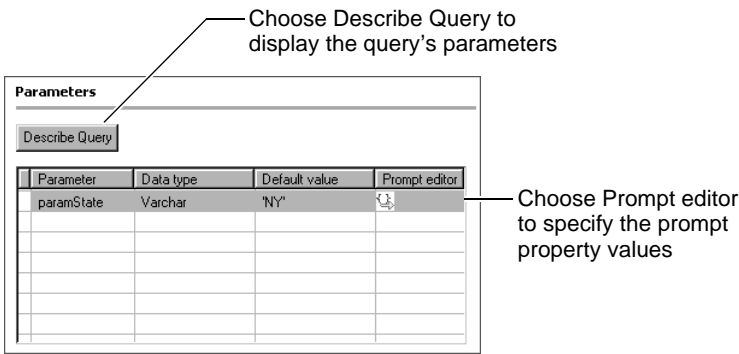


**Figure 5-54** Using Describe Query to display the query’s output columns

To specify column property values, select the column and specify the property values in Properties.

## Displaying parameters

In SQL Text Editor—Parameters, choose Describe Query to display the query’s parameters and the data type for each parameter. You can type a default value for a parameter in Default value, as shown in Figure 5-55.



**Figure 5-55** Using Describe Query to display the query’s parameters

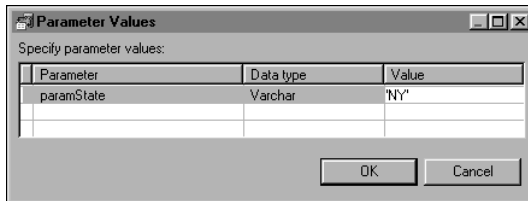
You can choose Prompt editor to set the prompt property values.

## Displaying information object query output

You can display information object query output in Data Preview.

### How to display information object query output

- 1 Choose Data Preview.
- 2 In Data Preview, choose Refresh. Parameter Values, shown in Figure 5-56, appears if the information object query defines parameters.



**Figure 5-56** Specifying parameter values

- 3 On Parameter Values, type the parameter values. A parameter value must be a single value, not a list of values. Choose OK. Information object query output appears, as shown in Figure 5-57.



**Figure 5-57** Viewing the information object query's output

- 4 Use the scroll bars to view all columns and displayed rows. Use the Page Size list box to change the number of rows displayed on each page. Use the Next Page and Previous Page icons to navigate through the data preview one page at a time.

# Accessing data in a text file

This chapter contains the following topics:

- Accessing data from a text file
- Connecting to a text file
- Creating a data set for a delimited text file
- Creating a data set for a fixed-width text file

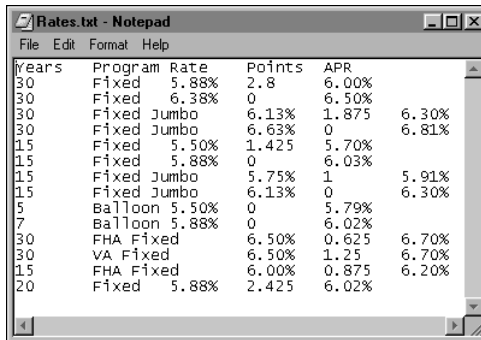
---

## Accessing data from a text file

BIRT Spreadsheet Designer can access data from delimited text files and fixed-width text files. Create a connection and query for the type of text file that you use:

- A delimited text file uses a character to signify the beginning of a new column. BIRT Spreadsheet Designer recognizes tabs, semicolons, commas, and spaces as delimiters. You can also supply another character for BIRT Spreadsheet Designer to recognize as the delimiter.

Figure 6-1 shows an example of a tab-delimited text file.

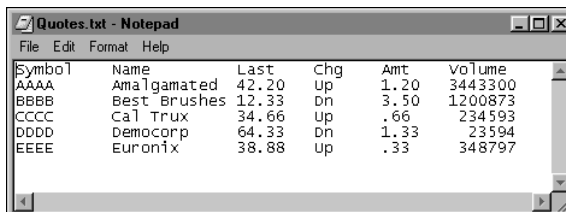


Years	Program	Rate	Points	APR
30	Fixed	5.88%	2.8	6.00%
30	Fixed	6.38%	0	6.50%
30	Fixed Jumbo	6.13%	1.875	6.30%
30	Fixed Jumbo	6.63%	0	6.81%
15	Fixed	5.50%	1.425	5.70%
15	Fixed	5.88%	0	6.03%
15	Fixed Jumbo	5.75%	1	5.91%
15	Fixed Jumbo	6.13%	0	6.30%
5	Balloon	5.50%	0	5.79%
7	Balloon	5.88%	0	6.02%
30	FHA Fixed	6.50%	0.625	6.70%
30	VA Fixed	6.50%	1.25	6.70%
15	FHA Fixed	6.00%	0.875	6.20%
20	Fixed	5.88%	2.425	6.02%

**Figure 6-1** A tab-delimited text file

To access a delimited text file, you first create a flat file data source to a delimited text file. After you create the data source, you must create a data set that uses a delimited text file query to extract data from the file.

- A fixed-width text file contains the same number of characters or spaces for each field in a column. For example, in the sample file that appears in Figure 6-2, the first column spans 8 characters, and the second column spans 13 characters.



Symbol	Name	Last	Chg	Amt	Volume
AAAA	Amalgamated	42.20	Up	1.20	3443300
BBBB	Best Brushes	12.33	Dn	3.50	1200873
CCCC	Cal Trux	34.66	Up	.66	234593
DDDD	Democorp	64.33	Dn	1.33	23594
EEEE	Euronix	38.88	Up	.33	348797

**Figure 6-2** A fixed-width text file

To access a fixed-width text file, you first create a flat file data source to a fixed-width text file. After you create the data source, you must create a data set that uses a fixed-width text file query to extract data from the file.

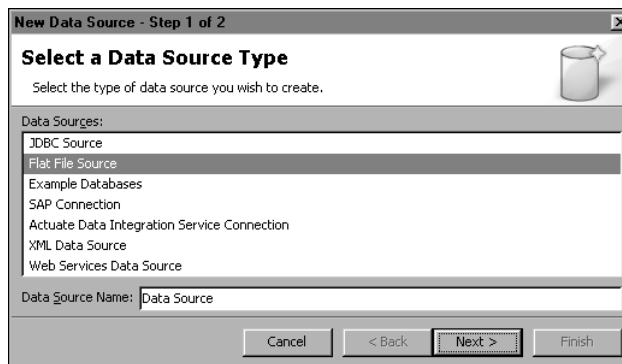
---

## Connecting to a text file

To create a connection to a text file, create a flat file data source that specifies a file name and location. Next, create a data set that retrieves text data from the flat file source. Specify data set properties that match the organization of text in the data file.

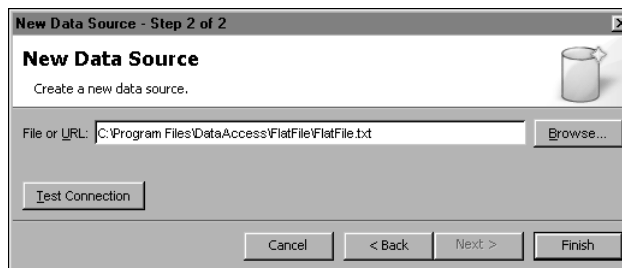
### How to create a flat file data source

- 1 In Actuate BIRT Spreadsheet Designer, choose Report→Create Data Source.
- 2 On New Data Source—Select Data Source Type, in Data Sources, select Flat File Source, as shown in Figure 6-3.



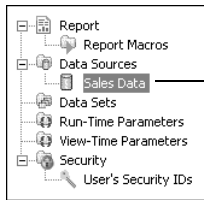
**Figure 6-3** Selecting Flat File Source for your data source

- 3 In Data Source Name, type a name for the data source and choose Next. New Data Source—New Data Source appears, displaying the fields for a flat file data source, as shown in Figure 6-4. Type the name and location of your file. To navigate to and select your file from directory listings, choose Browse. You also can type the URL of a data stream that provides your text data.



**Figure 6-4** Specifying a new flat file data source

- 4 Choose Finish. The data source that you defined appears in Data Explorer. Figure 6-5 shows a new data source in Data Explorer.



The new flat file data source, named Sales Data, appears in the Data Sources folder

**Figure 6-5** Flat file data source appears in Data Explorer

Now that you have a flat file data source that specifies the connection information for your data, you can create a data set that accesses data from that file.

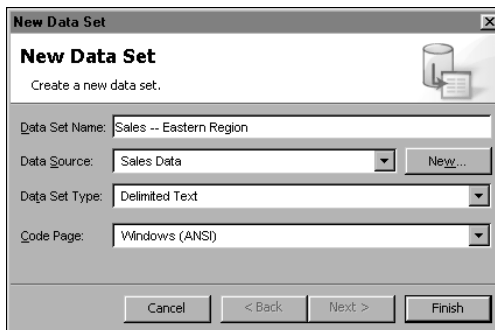
## Creating a data set for a delimited text file

To access the data in a delimited text file, create a data set and define a text-file query. When you define the query, you specify various information, including:

- Which rows to select from the file
- Whether the first row contains column labels
- Type of delimiter that marks the beginning of each column in the file
- The data type and the name to use for each column
- The type of encoding used for the text characters in the file, such as ANSI or Unicode

### How to define a data set for a delimited text file

- 1 In BIRT Spreadsheet Designer, choose Report→Create Data Set.
- 2 On New Data Set, perform the following steps, as shown in Figure 6-6.

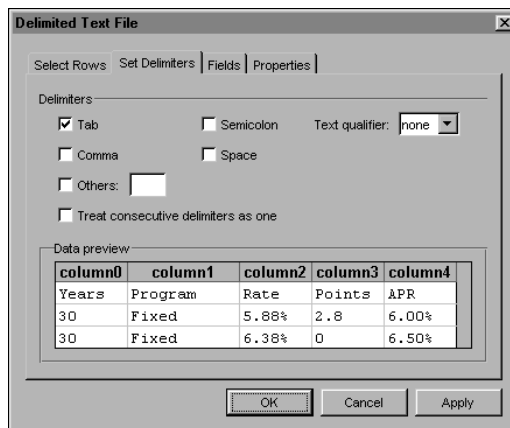


**Figure 6-6** Creating a new delimited text data set

- 1 In Data Set Name, type a name for your data set.

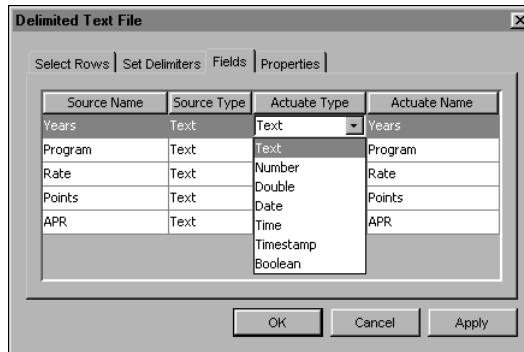


- 2 In Data Source, select a flat file data source.
- 3 In Data Set Type, choose Delimited Text.
- 4 In Code Page, select the encoding used for the text file and then choose Finish.
- 3 On Delimited Text File—Select Rows, perform the following steps to specify how to use the data row information:
  - To use the first row of data as labels for the columns, select Use first row to label columns.
  - To start at a row other than the first data row, in Start import at row, select the row number at which to begin.
  - To set a maximum number of rows to return, select Limit number of rows to import and select the number of rows.
- 4 To specify how the file is delimited, choose Set Delimiters and select the delimiter that separates the columns in the file, as shown in Figure 6-7.



**Figure 6-7** Creating a delimited text file query

- 5 To verify or modify column names and types, choose Fields, and set up the data fields on Delimited Text File—Fields:
  - To change the data type for a column, double-click Actuate Type for the column, then select an Actuate data type, as shown in Figure 6-8.
  - To change the name of a column, select Actuate Name for the column. Then, type the new name in the cell.



**Figure 6-8** Modifying field types

- 6 To set query properties, choose Properties, and set the property values:
  - To change the name of the data set, type a new name in Name for query.
  - To modify your selection identifying the encoding used for the file, select the new encoding in Code page.

Choose OK. Data Explorer displays the new data set in the Data Sets folder and the corresponding data set object in the Data Set Objects folder.

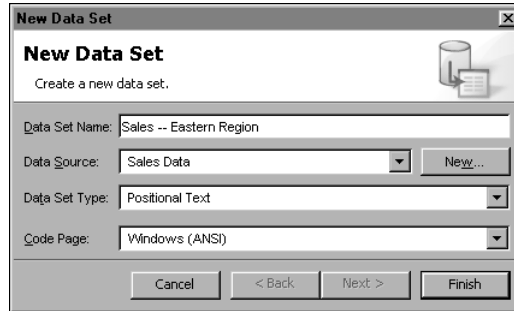
## Creating a data set for a fixed-width text file

To access the data in a fixed-width text file, create a data set and define a text-file query. When you define the query, you specify various information, including:

- Which rows to select from the file
- Whether the first row contains column labels
- The location of the beginning of each column
- The data type and the name to use for each column
- The type of encoding used for the text characters in the file, such as ANSI or Unicode

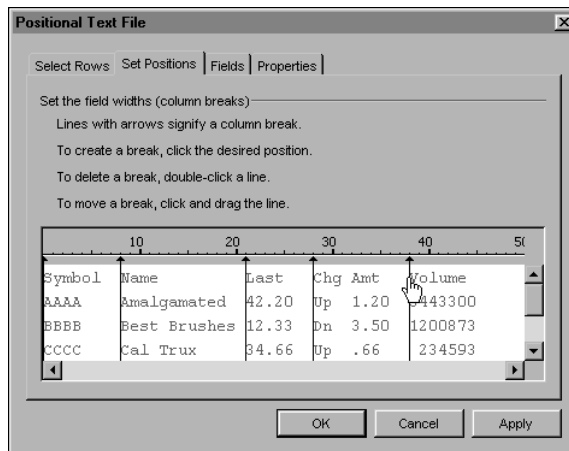
### How to define a data set for a fixed-width text file

- 1 In BIRT Spreadsheet Designer, choose Report → Create Data Set.
- 2 On New Data Set, perform the following steps:
  - 1 In Data Set Name, type a name for your data set.
  - 2 In Data Source, select a flat file data source.
  - 3 In Data Set Type, choose Positional Text, as shown in Figure 6-9.



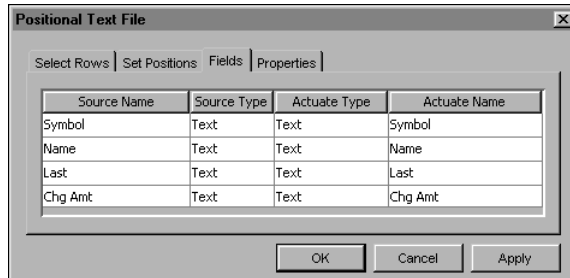
**Figure 6-9** Creating a new positional text data set

- 4 In Code Page, select the encoding used for the text file and then choose Finish.
- 3 On Positional Text File—Select Rows, specify how to use the data row information:
  - To use the first row of data as labels for the columns, select Use first row to label columns.
  - To start at a row other than the first data row, in Start import at row, select the row number at which to begin.
  - To set a maximum number of rows to return, select Limit number of rows to import and select the number of rows.
- 4 Choose Set Positions and set the column breaks by selecting the beginning position of each column. Figure 6-10 shows the results of selecting several column breaks.



**Figure 6-10** Defining positions for a fixed-width text file query

- 5 Choose Fields, as shown in Figure 6-11, and set up the data fields:
  - To change the data type for a column, double-click Actuate Type for the column, and select an Actuate data type from the drop-down list.
  - To change the name of a column, select Actuate Name for the column, and type the new name in the cell.



**Figure 6-11** Positional Text File—Fields

- 6 To set query properties, choose Properties and set the property values:
  - To change the name of the data set, type a new name in Name for query.
  - To modify your selection identifying the encoding used for the file, select the new encoding in Code page.

Choose OK. Data Explorer displays the new data set in the Data Sets folder and the corresponding data set object in the Data Set Objects folder.

# Part Three

---

**Developing a spreadsheet report**



# Laying out a report

This chapter contains the following topics:

- About layout components
- Comparing layout options
- Constructing a data range
- Modifying data range components
- Using multiple data ranges in a report design

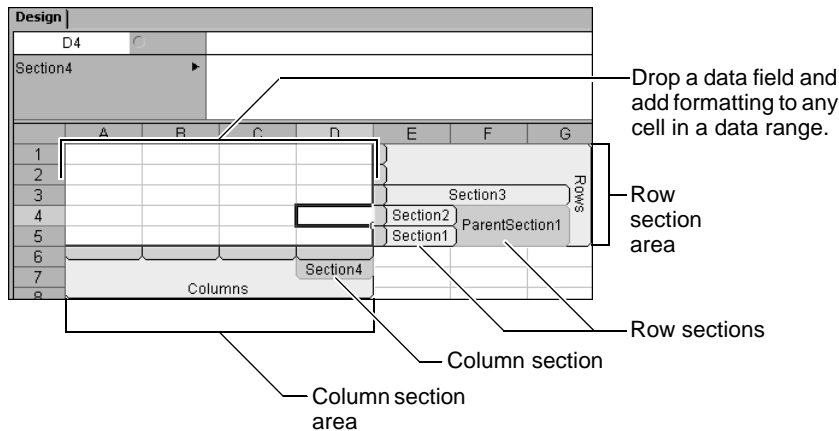
## About layout components

In BIRT Spreadsheet Designer, you use a range to structure the report layout. A range typically consists of row and column areas, which contain sections and groups that determine how data appears on the generated report. You can use the sections and groups to create a nested organizational structure that consists of parent and child sections.

You can create two types of ranges in BIRT Spreadsheet Designer:

- A data range, which uses row and column sections to specify the report layout. You can use BIRT Spreadsheet Designer's Listing Wizard to create a simple data range, or you can construct one using the Create Data Range option.
- A pivot range, which uses row and columns fields to specify the report layout. The BIRT Spreadsheet Designer PivotRange Wizard helps you design a pivot range.

Figure 7-1 shows a sample data range that includes four row sections and one column section. Notice that the row section area contains a nested structure that consists of one parent section and two child sections.



**Figure 7-1** A blank data range

You can preview the report layout at various stages in the development process to see how the structure looks on the generated report, and can return to the range to modify it as needed.

After creating a range, you add data fields from the data set and use report script functions to specify how data values from the data fields display on the published report. For information about adding data fields and report script functions, see Chapter 9, "Adding content to a data range."



# Comparing layout options

This section describes the methods you can use to design a data or pivot range. Use these descriptions to help determine the most effective layout for the report you want to deliver.

## About the listing wizard

The listing wizard uses data fields from your data set to generate a simple data range. Tutorial 1 in Chapter 1, “Building your first spreadsheet reports,” used the listing wizard to create a data range for a simple list report.

A data range constructed by a listing wizard typically contains one detail row and, if you indicate a total or subtotal, one summary row. A detail row contains data fields from the data set. A summary row contains a total or subtotal calculation for the data fields you selected.

When you use the listing wizard, BIRT Spreadsheet Designer creates a section for the detail row. It does not create a section for the summary row. You can modify the data range by adding additional detail rows, summary rows, and sections. You can also group a section, and add or remove data fields and report script functions, just as you would a data range created from scratch.

Figure 7-2 shows how a typical data range generated by a listing wizard looks before the formatting has been modified.

Design		View				
A1		Name		Cannot enter Report Functions on leaf section.		
		Cannot enter Report Functions on leaf section.				
	A	B	C	D	E	F
1	Name	Last	Origin	Current	Detail Range	Rows
2	#Name	#Last	#OriginalV	#CurrentV		
3	#form		#form	#form	Summary row	
4	Columns					
5						
6						

**Figure 7-2** Data range generated by a listing wizard

Figure 7-3 shows how the generated listing wizard report appears. Because the listing wizard automatically locates the summary row at the end of the detail rows, you need to scroll down to the end of the report to see it. You can change this arrangement in the data range so that the summary row appears above the detail row.

Design		View			
A2		3M			
	A	B	C	D	
1	<b>Name</b>	<b>Last</b>	<b>OriginalValue</b>	<b>CurrentValue</b>	
643	Walt Disney	Benitez	\$7,630	\$9,825	
644	Walt Disney	Bolick	\$15,260	\$19,650	
645	Walt Disney	Carlson	\$22,890	\$29,475	
646	Walt Disney	Christians	\$22,890	\$29,475	
647	Walt Disney	Davis	\$22,890	\$29,475	
648	Walt Disney	Edmonds	\$15,260	\$19,650	
649	Walt Disney	Fong	\$15,260	\$19,650	
650	Walt Disney	Franco	\$7,630	\$9,825	
651	Walt Disney	Franco	\$22,890	\$29,475	
652	Walt Disney	Frank	\$22,890	\$29,475	
653	Walt Disney	Gadea	\$22,890	\$29,475	
654	Walt Disney	Howell	\$22,890	\$29,475	
655	Walt Disney	Jung	\$7,630	\$9,825	
656	Walt Disney	Müller	\$7,630	\$9,825	
657	Walt Disney	Picard	\$15,260	\$19,650	
658	Walt Disney	Planson	\$22,890	\$29,475	
659	Walt Disney	Sanford	\$7,630	\$9,825	
660			<b>\$103,206,740</b>	<b>\$127,556,388</b>	
661					

**Figure 7-3** Generated Listing Wizard report

## Creating a data range manually

The Create Data Range menu option inserts a blank data range with row and column section areas into the worksheet location that you specify. You must then define individual row and column sections and groups, and add data and formatting. Tutorial 1 in Chapter 1, “Building your first spreadsheet reports,” demonstrates using the Create Data Range option to create a data range.

Figure 7-4 shows how a sample data range looks after row and column sections have been created, and after data and formatting have been added. In this example, the row sections are grouped by security name and by client name. The parent row section Client contains two sub-sections, ClientTotals and Security. Groups are created from sections and are not visible on their own.

In the column section area data is grouped by month. In this sample data range, row 6 is the detail row, and row 5 is the summary row.

	A	B	C	D	E	F	
1	<b>Portfolio Totals</b>						
2							
3			#write(PortfolioDate)				
4	<b>Client Name</b>	<b>Security Name</b>	<b>Value</b>	<b>Shares</b>			Rows
5	#write(Client.Last)		#sum(CurrentValue)	#sum(CurrentQuant	ClientTotals	Client	
6		#write(Name	#sum(CurrentValue)	#sum(CurrentQuant	Security		
7							
8			Value				
9				Month			
10			Section1				
11			Columns				

**Figure 7-4** A populated data range

Figure 7-5 shows how the report output generated by the report design shown in the previous example looks after running the report. Notice that the rows display data grouped first by client name, then by security name. Also, notice that quantities and values contained in the Month group, are displayed by month.

Design View						
D24		1000				
	A	B	C	D	E	F
1	<b>Portfolio Totals</b>					
2						
3			Jan-06		Feb-06	
4	<b>Client Name</b>	<b>Security Name</b>	<b>Value</b>	<b>Shares</b>	<b>Value</b>	<b>Shares</b>
5	Benitez		\$1,121,810	24000	\$2,445,270	69000
6		Alcoa	\$112,290	3000	\$241,710	9000
7		Allied Signal	\$76,460	2000	\$172,420	6000
8		Boeing	\$93,700	2000	\$228,620	6000
9		Caterpillar	\$140,730	3000	\$327,690	9000
10		Dupont	\$110,180	2000	\$238,020	6000
11		General Motors	\$167,850	3000	\$276,480	6000
12		Hewlett Packard	\$118,700	2000	\$272,840	6000
13		Sears Roebuck	\$44,990	1000	\$115,150	3000
14		Texaco	\$150,360	3000	\$350,130	9000
15		Walt Disney	\$106,350	3000	\$222,210	9000
16	Boivent		\$329,930	7000	\$955,200	27000
17		Allied Signal	\$38,230	1000	\$86,210	3000
18		American Express	\$0	0	\$201,810	6000

**Figure 7-5** Report generated from a user-constructed data range

For instructions about how to construct a data range from scratch, see “Constructing a data range,” later in this chapter.

## About a pivot range

Use a pivot range to combine and display data in a format that is similar to that of an Excel pivot table. In a pivot range, data is organized by row, column, and page fields rather than by sections, as in a data range. You can use the field areas in many of the same ways as you use the section areas of a data range. A pivot range supports interactive filtering of report data.

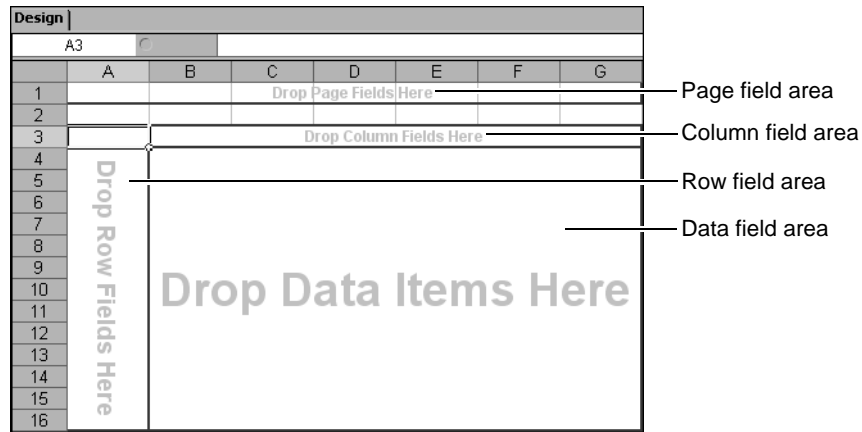
After constructing a data set and query, you use the PivotRange Wizard to create a blank pivot range. A blank pivot range supports dragging data fields from Data Explorer and dropping them in a pivot range in the following areas:

- **Column field area**  
Drop a data field in the column field area to display data in vertical columns.
- **Data field area**  
Drop a data field in the data row area to display summary data values.
- **Page field area**  
Drop a data field in the page field area to filter large groups of data.

- Row field area

Drop a data field in the row field area to display data in horizontal rows.

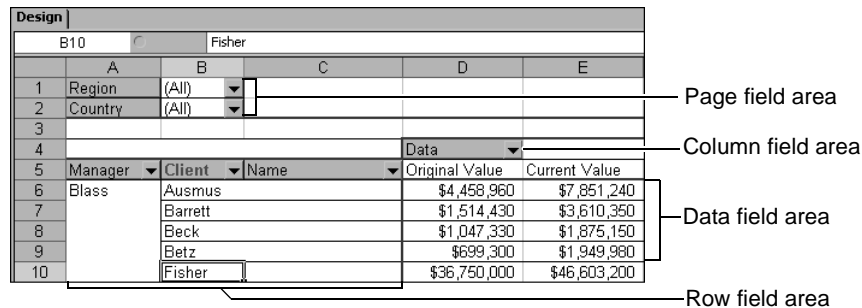
Figure 7-6 shows how each area appears initially in a blank pivot range.



**Figure 7-6** Viewing areas in a blank pivot range

After generating a report design that includes a pivot range, the example shown in Figure 7-7 shows how data values appear in the following pivot range areas:

- The page field area enables users to filter report data by region or country.
- The column field area organizes data values in vertical columns.
- The data field areas display the original and current value of the stocks for each client.
- The row field areas display data values, grouped first by client manager, then by the manager's client names.



**Figure 7-7** Viewing a populated pivot range

For more detailed information about building a pivot range, see Chapter 13, "Working with a pivot range."

---

## Constructing a data range

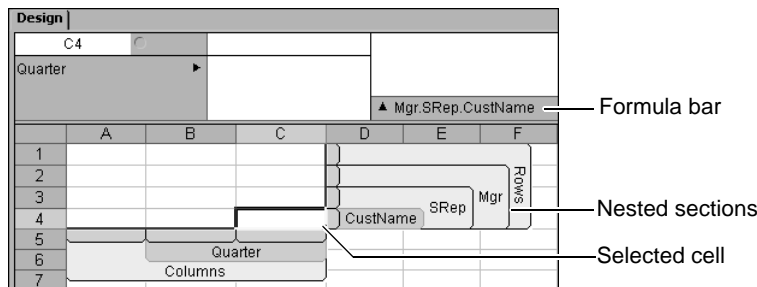
To construct a data range from scratch, you perform the following tasks:

- Use the Create Data Range option to insert a blank data range into a location you specify, such as cells A1 through F10.
- Create row and column sections.
- Create section groups.

This section provides conceptual information about row and column sections and groups. For step-by-step instructions on how to create sections and groups in a data range, see Tutorial 2 in Chapter 1, “Building your first spreadsheet reports.”

### Understanding row and column sections

A data range typically contains at least one row section and one column section. As shown in Figure 7-8, section names appear in the row and column section tabs of the design editor. When you select a cell in a data range, the formula bar displays the names of the row and column sections to which the selected cell belongs, including the names of any parent sections. The nesting structure of the section area defines the full name of the section.



**Figure 7-8** Viewing a data range under construction

You can create a section in the following ways:

- If you use the Create Parent method to create a section, you provide the section name. To use this method, right-click on a section stub, select Create Parent from the context menu, and type a name for the section.
- If you drag a data field from Data Explorer and drop it into a section stub, BIRT Spreadsheet Designer creates a section that groups on the data field and names the section after the data field. For instructions on how to use this method, see “Creating a grouped section from a data field,” later in this chapter.

A section is used primarily to structure related data fields. You can also use a section name in a report script function to retrieve data and to create totals and calculations. For more information about using a section name in a report script function, see Chapter 9, “Adding content to a data range.”

## About data range groups

In a data range, section grouping facilitates the retrieval and display of multiple data rows. If you do not create at least one section group, your report displays only a single row of data. For example, consider a report that displays order information for different customers. If you create a section group by customer name in the report design, then the report output displays order data for each customer by name. If you do not create a customer name group, the report output summarizes all the order data in a single output row for all customers.

You use a data field to set up a group’s content. When used in this way, a data field is called a group key. After you select the field to group on, you can choose a sort order to determine how grouped data values appear in the generated report. For more about changing the sort order for a section group, see “About sorting and grouping,” later in this chapter.

To create a grouped section, right-click the section name, then choose Group from the context menu. In Section Filtering and Grouping, use the Group tab to select options for the section group, as shown in Figure 7-9.

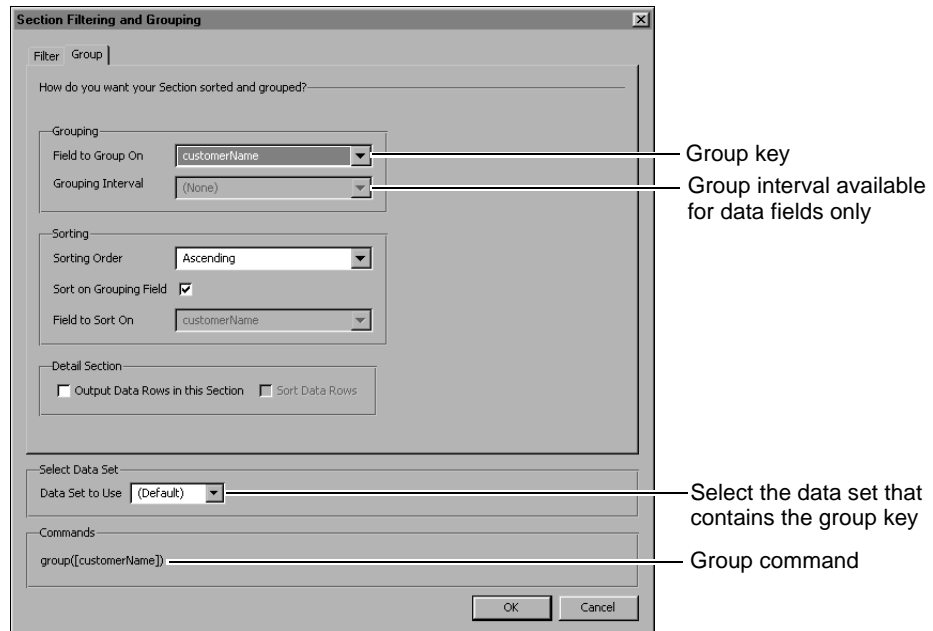
Though groups are not visibly identifiable in a data range, you can verify the groups you created by right-clicking on a section group, and choosing Group from the context menu. The Section Filtering and Grouping dialog shows the grouping that you specified. You can also verify a group by selecting a section stub. If a group was created for that section, the group key appears in the formula bar, as shown in Figure 7-10.

You can create groups from both row and column sections. You can also create sheet-level groups, which instruct BIRT Spreadsheet Designer to disperse data across multiple sheets in the same workbook.

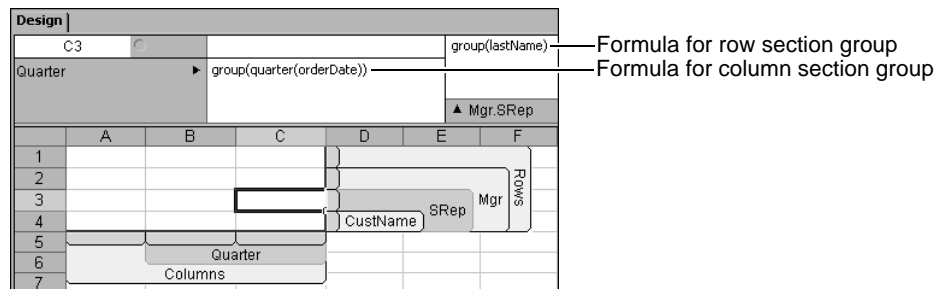
## Creating a sheet-level group

Use a sheet-level group to display data on multiple sheets in the generated report. A sheet-level group uses a data field in conjunction with the row and column groups you create.

You use the group report script function to burst data onto separate sheets in the workbook. You use the sheet report script function to name the sheets.



**Figure 7-9** Specifying the customerName group



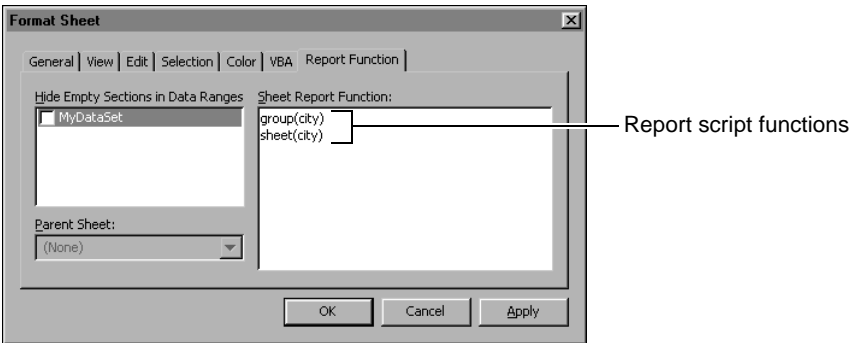
**Figure 7-10** Examining the group keys for row and column sections

### How to create a sheet-level group

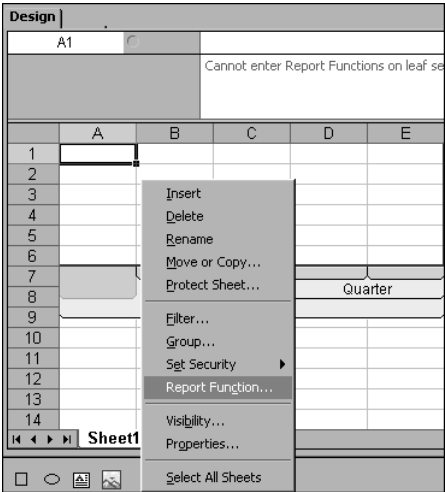
- 1 In Design Editor, right-click on a sheet tab and choose Report Script Function from the context menu, as shown in Figure 7-12.
- 2 On Format Sheet—Report Script Function, in Sheet Report Script Function, type group and sheet functions, as shown in Figure 7-11. In this example, the group function uses the city data field to group data. The sheet function uses values from the city data field to name each sheet:

```
group(city)
sheet(city)
```

Using both functions groups data for each city on a different sheet.



**Figure 7-11** Using group and sheet report script functions  
Choose OK.



**Figure 7-12** Selecting Report Script Function for Sheet 1

- 3 Preview the report to see how using the group function on the sheet disperses data across multiple sheets. Figure 7-13 shows how the sheet function uses the name of one city to name each sheet.

## About sorting and grouping

By default, BIRT Spreadsheet Designer sorts on the grouping field and displays grouped values in ascending order. You can change the sort order to descending or to unsorted. You can also choose to sort on a different field. For example, you could group section data by customer name, but choose to display data for that section by a different data field, such as order price.



		Q1/2004				Q2/2004	
		Revenue	Profit	Revenue	Profit		
6	Bott	Totals	\$32,306	\$11,842	\$90,051	\$35,714	
7		AV Stores, Co.	\$0	\$0	\$0	\$0	
8		Double Decker Gift Stores, Ltd	\$7,310	\$2,845	\$0	\$0	
9		giftsbymail.co.uk	\$24,996	\$8,997	\$0	\$0	
10		Oulu Toy Supplies, Inc.	\$0	\$0	\$16,213	\$5,663	
11		Stylish Desk Decors, Co.	\$0	\$0	\$0	\$0	
12		Suominen Souvenirs	\$0	\$0	\$28,395	\$12,400	
13		Toys of Finland, Co.	\$0	\$0	\$0	\$0	
14		UK Collectables, Ltd.	\$0	\$0	\$45,444	\$17,651	
15	Jones	Totals	\$66,359	\$24,043	\$0	\$0	
16		Baane Mini Imports	\$0	\$0	\$0	\$0	
17		Bavarian Collectables Imports, Co	\$0	\$0	\$0	\$0	
18		Blauer See Auto, Co.	\$33,821	\$11,685	\$0	\$0	
19		Clover Collections, Co.	\$32,539	\$12,358	\$0	\$0	

One sheet per city

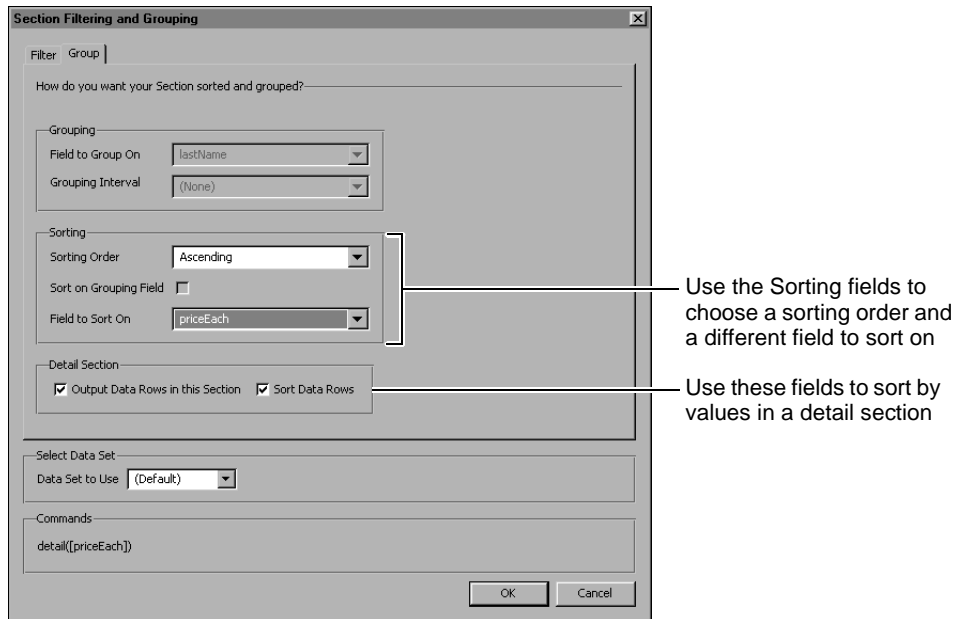
**Figure 7-13** Data grouped and displayed by sheet

You can also choose to group on a detail section. When you group on a detail section, you can choose to sort on values in that detail section, or by values from a different data field. When you choose Output Data Rows in this Section, which is selected in Figure 7-14, the Sort Data Rows field becomes active. When you choose Sort Data Rows, the Sorting fields become active, as shown in Figure 7-14.

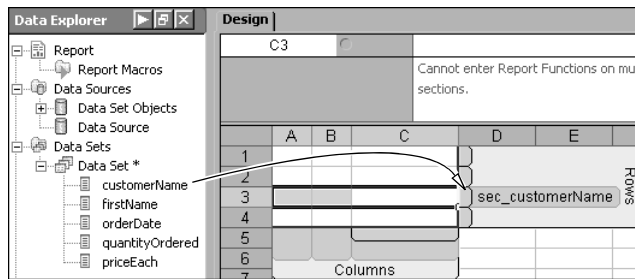
In some situations, the sorting by detail section options provide more control and flexibility over how data appears on the published report. Sorting by detail section may also enable more efficient output of data rows. For example, use detail section sorting options when a report connects to a data source, such as a stored procedure, that does not allow you to change the order of rows in a data set.

## Creating a grouped section from a data field

A quick way to create a grouped section is to drag a data field from the data explorer and drop it into a section stub, as shown in Figure 7-15. When you use this method, the section name takes the name of the data field, to which BIRT Spreadsheet Designer adds the prefix sec. To change this default name, right-click on the section name and choose Rename from the context menu.



**Figure 7-14** Viewing the sorting options



**Figure 7-15** Creating a section and group from a data field

## Modifying data range components

You can modify data range components in the following ways:

- Edit a section.
- Insert a row or column.
- Delete a cell, row, or column.
- Hide empty sections in a generated report.

- Hide a row or column.
- Set row height and column width.

The instructions that follow are specific to modifying elements of a data range. You can modify rows and columns outside a data range just as you would rows and columns in an Excel spreadsheet.

## Editing a section

You can edit a section in the following ways:

- Delete.
- Rename.
- Change the group key.

## Deleting a section

Deleting a section removes the group from the data range, but does not delete the data fields and functions from the cells. Deleting a parent section does not delete subordinate sections. If the section name is used in a formula, be sure to delete the name there as well. If you do not, the formula returns the #REF! error, indicating that BIRT Spreadsheet Designer cannot find the section name.

### How to delete a section

Right-click the section to delete, then choose Delete from the context menu. BIRT Spreadsheet Designer removes the section from the section area.

## Renaming a section

You can rename a section at any time. If the section name is used in an expression, BIRT Spreadsheet Designer updates the expression accordingly.

### How to rename a section

- 1 Right-click the selection to rename, then choose Rename from the context menu.
- 2 On Section Name, in Name, type a new name and choose OK.

## Changing a group key

The group key is the data field you use to group the data rows in a section. For more information about group keys, see “About data range groups,” earlier in this chapter.

### How to select a different group key

- 1 Right-click the section, then choose Group from the context menu. Section Filtering and Grouping appears.

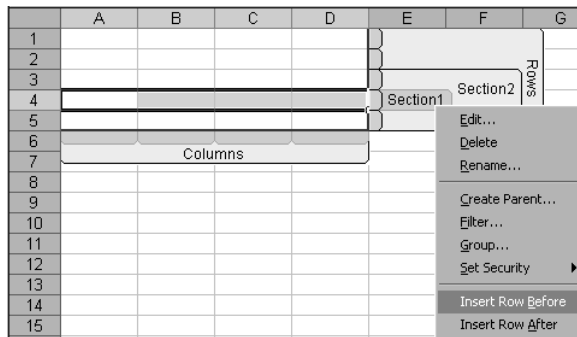
- 2 In Field to Group On, choose a different field.
- 3 Change Grouping Interval and the Sorting fields as needed.
- 4 Choose OK.

## Inserting a row or column

You can insert a row or column inside or outside a section in a data range. You can also insert a row above or below a selected row.

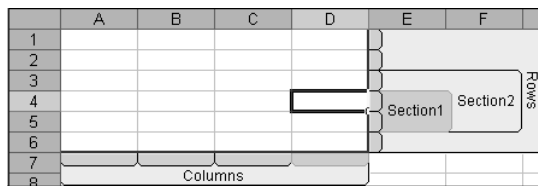
### How to insert a row into a section

To insert a row into an existing section, right-click on the row stub, and choose Insert Row Before or Insert Row After, as shown in Figure 7-16. In Figure 7-16, the starting section, Section1, consists of a single row, row 4.



**Figure 7-16** Inserting a row or column

The new row appears above the starting row. Notice that Section1 now includes two rows, 4 and 5, as shown in Figure 7-17.



**Figure 7-17** Inserting a row or column

## Deleting a cell, row, or column

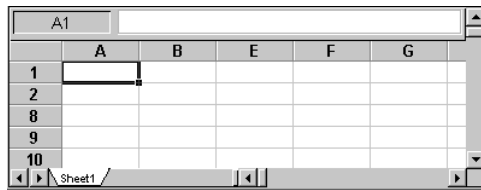
In most cases, deleting a cell's content, or deleting a row or column, does not cause a problem in your report design. If the cell, row, or column contains a cell reference or defined name that is used in an expression elsewhere in the range, the expression returns the #REF! error. This error indicates that BIRT Spreadsheet Designer cannot find the reference or defined name.

You cannot delete a cell, but you can clear its contents or shift its location. Clearing content removes data, text, and formatting. Shifting moves both contents and formatting to the specified location.

If you delete a row or column that contains data fields, the data fields are also deleted. They do not shift to an adjacent row or column.

## Hiding a row or column

To hide a row or column, select it, then choose **Format**→**Column**→**Hide** or **Format**→**Row**→**Hide**. When you hide a row or column, the visible sequence of row numbers and column letters skips the hidden ones. In Figure 7-18, notice that the sequence of rows and columns do not display the numbers of hidden rows or the letters of hidden columns. In this example, columns C and D are hidden. Rows 3 through 7 are also hidden.



**Figure 7-18** Hiding rows and columns

To redisplay a hidden row, select the rows above and below the hidden row and choose **Format**→**Row**→**Unhide**. To redisplay a hidden column, select the two columns to the left and right of the hidden column and choose **Format**→**Column**→**Unhide**.

If you hide rows or columns in a data range, ensure that the data range does not use the size report script function to adjust the height or width of the row or column. The size function overrides the hide feature and the row or column will appear in the report.

## Hiding empty sections in a report

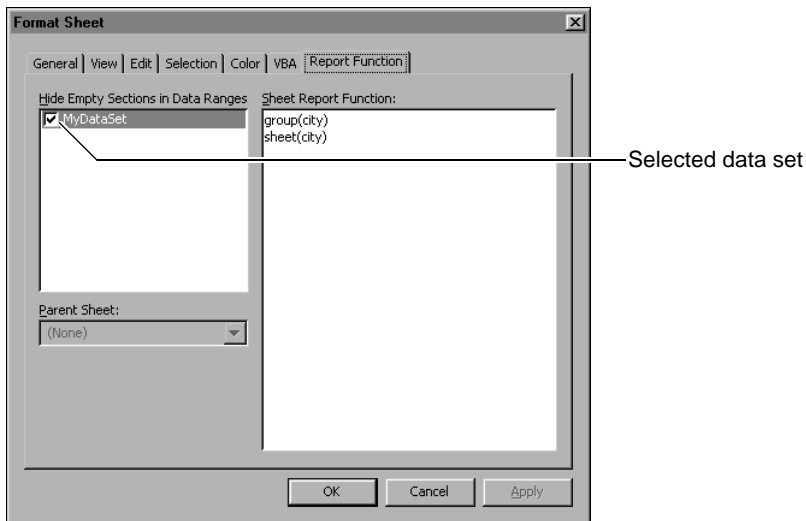
By default, BIRT Spreadsheet Designer shows all sections a report generates, even if those sections return no data. You can specify that a report hide empty sections. For example, if you apply the hide empty section option to a report that shows expenses by department, only departments that have expenses display on the generated report.

When you specify that a report hide empty sections, you do so for an entire data set. You cannot choose which sections within a data set should hide data.

### How to hide empty sections in a report

- 1 Right-click on a sheet tab and choose **Report Script Function** from the context menu.

- 2 On Format Sheet—Report Script Function, in Hide Empty Sections in Data Ranges, select the data set for which to hide empty sections, as shown in Figure 7-19.



**Figure 7-19** Hiding empty report sections

Choose OK. The generated report hides all sections that do not contain data to display.

## Setting row height and column width

When you change the height of a row or the width of a column, graphics and charts placed on those rows or columns reduce or enlarge accordingly because BIRT Spreadsheet Designer anchors objects to the cells behind them.

You can use the size report script function in the following ways:

- To size the rows or columns in a section to match the size of the largest data cell in the row or column. The auto size function performs this task.
- To size the rows or columns in a section to match a specific size you determine.
- To hide a column based on a parameter condition you define.

You can also manually resize a column or row as you would in Excel, by dragging a row or column border.

## Using the size report script function

The auto size report script function:

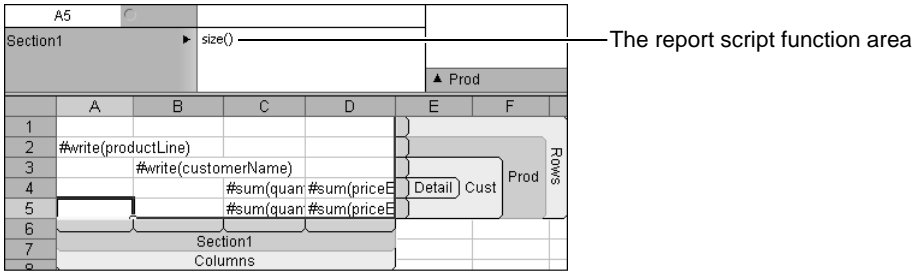
```
size()
```

instructs BIRT Spreadsheet Designer to size the rows or columns in a section to match the contents of each data cell.

**How to use the size report script function to resize a row or column**

To apply the size report script function to a section, select the section, then type the size report script function in the report script function area. The following procedure shows you how to apply the auto size function to a column section. The procedure for applying the size report script function to a row section is the same, except you start by selecting a row section rather than a column section.

- 1 Select the section to size and type the size function in the report script function area for that section, as shown in Figure 7-20.



**Figure 7-20** Adding a size function to a section

Press Enter.



- 2 Choose Run to preview the report. In this example, we specified auto size. In Figure 7-21, you can see that BIRT Spreadsheet Designer sized the columns to match the size of the largest data cell in the column.

**Setting a specific row or column size**

You can also use the size report script function to set the row height or column width of a section to a specific number of twips. There are 1440 twips in an inch. For example, to set the height of all the rows in a section to 1000 twips, use the following report script function:

```
size(1000)
```

**Using the size function to hide a section**

You can also use the size function with a zero to hide a section when a certain condition is met. This use of size option is typically employed in conjunction with a parameter or a defined name. For example, the following expression:

```
if (:portMgrID=8, size(0))
```

in the formula bar of a column section, instructs BIRT Spreadsheet Designer to hide the columns in that section if the value of the portMgrID parameter is 8. For more information about creating and using parameters, see Chapter 12,

“Designing a customizable report.”

	A	B	C	D
2	Classic Cars			
3		Alpha Cognac		
4			\$25	\$117
5			\$36	\$121
6			\$41	\$169
7			\$24	\$132
8			\$42	\$128
9			\$41	\$71
10			\$48	\$105
11			\$22	\$92
12			\$45	\$68
13			\$45	\$42
14			\$32	\$69
15			\$22	\$63
16			\$48	\$86
17			\$36	\$80
18			\$34	\$53
19			\$687	\$1,749
20		Amica Models & Co.		
21			\$34	\$206
22			\$24	\$98
23			\$50	\$87
24			\$27	\$47
25			\$26	\$162

**Figure 7-21** Result after you apply the auto size function

---

## Using multiple data ranges in a report design

You can create and use multiple data ranges on the same sheet in a workbook, or on multiple sheets. You can use a report script function to extract data from both ranges, or you can use the data ranges independently of one another. For example, you can create a data range that displays summary information on one sheet in a workbook, and detail information on the other sheets. Alternatively, you can use two or more data ranges to display summary and detail information on the same worksheet.

### Using multiple data ranges on a single worksheet

The worksheet in Figure 7-22 contains two data ranges. Notice that row and column section areas are displayed for the top range, but not the bottom range. Because you can work in only one range at a time, BIRT Spreadsheet Designer displays the section areas of only the selected range, which, in this case, is the top one. You can modify the selected data range as you normally would, by adding and removing data fields, changing sections and groups, and adding and removing formatting. To work in another range on the same sheet, click on any cell in the range.



	A	B	C	D	E	L	M
3	<b>Security Category Name</b>		<b>Total Value by Category</b>				
4	#SummaryData:write	<b>Security Name</b>					
5		#SummaryData:write(Nam	#SummaryData:sum(SUMofCurrentVa			SecName	
6		<b>Category Total</b>	#SummaryData:sum(SUMofCurrentVa			SecCat	
7							
8	Columns						
9							
10							
11	<b>Security Name</b>	<b>Client Name</b>	<b>Price</b>	<b>Qty</b>	<b>Value</b>		
12	#write(Name)						
13		#write(Client_Last)	#sum(CurrentPri	#sum(Cui	#sum(CurrentValue)		

**Figure 7-22** Two data ranges on the same sheet

Figure 7-23 shows how data from the two ranges in Figure 7-22 appear on the generated report. When you scroll down to the end of the data for the first range, you can see data for the second range. As shown in Figure 7-23, the first data range displays total values for each security by security category, whereas the second range displays more detailed security information by client name.

	A	B	C	D	E
1	<b>Stock Value and Gain</b>				
3	<b>Security Category Name</b>		<b>Total Value by Category</b>		
54	Small Cap	<b>Security Name</b>			
55		Coca Cola Company	\$524,154,859		
56		Tech-Sym Corporation	\$145,680,480		
57		Tesco	\$44,325,000		
58		<b>Category Total</b>	\$714,160,339		
59					
60	Tax-exempt Bonds	<b>Security Name</b>			
61		US Treasury Bond	\$630,079,739		
62		<b>Category Total</b>	\$630,079,739		
63					
64	Taxable Bonds	<b>Security Name</b>			
65		Coca Cola Company Bond	\$57,550,000		
66		Fidelity Municipal Bond F	\$241,815,000		
67		<b>Category Total</b>	\$299,365,000		
70					
71					
72	<b>Security Name</b>	<b>Client Name</b>	<b>Price</b>	<b>Current Qty</b>	<b>Value</b>
73	3M				
74		Abner	\$0	\$500,000	\$38,000
75		Anderson	\$7,422	\$280,500	\$11,821,070
76		Arendt	\$2,388	\$49,500	\$2,171,240
77		Ashby	\$2,128	\$110,000	\$4,293,440
78		Ashley	\$4,439	\$99,000	\$3,966,660
79		Ausmus	\$1,452	\$100,000	\$3,862,640
80		Avery	\$6,128	\$121,000	\$6,489,230

**Figure 7-23** Two data ranges on a generated report

When using multiple data ranges on any worksheet, consider the following guidelines:

- When you place a data range below another data range, the data range must start at the same column. In addition, the data range on top must include more columns than the data ranges below it or the same number of columns.

- When you place data ranges side-by-side in a worksheet, the data ranges must start at the same row. In addition, the range on the left must include more rows than the data ranges on the right or the same number of rows.

## Using multiple data ranges to display data on different worksheets

The report design used in this example contains two worksheets, each with its own data range, as shown in Figure 7-24. The Summary data range displays summary information for each sales representative managed by the report's viewer. The Detail data range shows more detailed information for each of the sales reps listed on the Summary sheet. This example uses sheet grouping to disperse data on the Detail sheet across multiple sheets in the generated report.

To create a report design that uses this or a similar scenario, create and populate data ranges on separate sheets. You then use group and sheet report script functions to disperse the appropriate data across multiple sheets in the workbook. The following procedure uses the Detail worksheet to illustrate this technique, which is known as sheet-bursting.

	A	B	C	D	E	F
1	Sales Rep					
2	#write(lastName)	Customer Name	Qty Ordered	Profit		
3	#write(customer)	#sum(quantity)	#sum(Profit)		Cust	SalesRep
4	Sales Rep Totals	#sum(quantity)	#sum(Profit)			
5						
6		Section1				
7		Columns				

	A	B	C	D	E	F
1	Customer Name					
2	#write(customer)	Product Name	Qty Ordered	Profit		
3	#write(product)	#sum(quantity)	#sum(Profit)		Prod	Cust
4	Customer Totals	#sum(quantity)	#sum(Profit)			
5						
6		Section1				
7		Columns				

**Figure 7-24** Data ranges on separate sheets

- 1 Right-click on the Detail sheet tab, then choose Report Script Function from the context menu.
- 2 On Format Sheet—Report Script Function, type the following commands:

```
group(lastName)
sheet(lastName & "details")
```

The group function instructs BIRT Spreadsheet Designer to group the detail information by last name. By adding the sheet function to the Details sheet, you instruct BIRT Spreadsheet Designer to use the last name of the sales rep in the sheet name. Choose OK.



- 3 Run the report to see how the group report script function uses the lastName field to group the detail data on separate sheets. Figure 7-25 shows how data from both data ranges appear on the generated report. Notice that detail sheets for each sales rep follow the Summary sheet. Use the Summary sheet to view grand totals for all sales reps. Choose a detail sheet to view detail information for an individual sales rep.

Figure 7-26 shows how the detail data range displays detail data for sales rep, Bondur.

	A	B	C	D
1	Sales Rep			
2	Bondur	Customer Name	Qty Ordered	Profit
3		Auto Canal+ Petit	1001	\$38,756
4		La Corne D'abondance, Co.	836	\$34,650
5		Lyon Souvenirs	684	\$28,391
6		Marseille Mini Autos	804	\$27,185
7		Reims Collectables	1433	\$52,699
8		Saveley & Henriot, Co.	1428	\$53,211
9		Sales Rep Totals	6186	\$234,891
11	Bott	Customer Name	Qty Ordered	Profit
12		AV Stores, Co.	1778	\$60,096
13		Double Decker Gift Stores, Ltd	357	\$10,868
14		giftsbymail.co.uk	895	\$26,809
15		Oulu Toy Supplies, Inc.	1110	\$38,335
16		Stylish Desk Decors, Co.	937	\$31,668
17		Suominen Souvenirs	1031	\$40,096
18		Toys of Finland, Co.	1051	\$38,809
19		UK Collectables, Ltd.	1046	\$43,524
20		Sales Rep Totals	8205	\$290,204
Summary Bondurdetails Bottdetails Castilodetails				

Detail sheets for each sales rep follow the Summary sheet

Figure 7-25 Viewing the summary report

	A	B	C	D
1	Customer Name	Product Name	Qty	Profit
2	Auto Canal+ Petit	1936 Harley Davidson El Knucklehead	55	\$1,866
3		1957 Vespa GS150	77	\$1,798
5		1960 BSA Gold Star DBD34	59	\$1,438
6		1961 Chevrolet Impala	60	\$2,480
7		1968 Ford Mustang	41	\$3,111
8		1969 Harley Davidson Ultimate Chopper	41	\$1,727
10		1970 Plymouth Hemi Cuda	28	\$1,341
11		1971 Alpine Renault 1600s	48	\$676
12		1974 Ducati 350 Mk3 Desmo	62	\$2,379
13		1982 Ducati 900 Monster	68	\$1,069
15		1996 Moto Guzzi 1100i	45	\$2,248
17		1997 BMW R 1100 S	25	\$1,212
18		2002 Suzuki XREO	36	\$2,169
19		2002 Yamaha YZR M1	88	\$3,300
20		Customer Totals	1001	\$38,756
21	La Corne D'abondance, Co.	Product Name	Qty	Profit
23		1903 Ford Model A	48	\$3,212
24		1926 Ford Fire Engine	45	\$1,285
25				
Summary Bondurdetails Bottdetails Castilodetails Firr				

Notice that the total in cell D21 matches the total in cell D3 on Summary sheet

Figure 7-26 Viewing a generated detail report



# Formatting a report

This chapter contains the following topics:

- About BIRT Spreadsheet Designer formatting options
- Formatting numbers
- Merging cells
- Formatting styles
- Using conditional formatting
- Creating a report outline
- Working with workbooks and worksheets

---

## About BIRT Spreadsheet Designer formatting options

You can use the following BIRT Spreadsheet Designer options to format data and pivot ranges:

- **Number and style formatting**  
You can format numeric data to display in multiple currencies, such as the yen or Euro, or as fractions, percents, and dates. In addition, you can apply a variety of style formats to numeric and textual data, including borders, colors, and fonts. You can also merge cells and use report script functions in them just as you would in unmerged cells.
- **Conditional formatting and formulas**  
You can create a condition that displays values with matching conditions in different colors. For example, you can create a condition that highlights values over \$1,000,000 in blue, and values under \$500,000 in red.  
  
You can also create a conditional formula which instructs BIRT Spreadsheet Designer to highlight certain data when formula conditions are met.
- **Outlining a report**  
The outlining feature enables you to display different levels of summary and detail information in the generated report. An outline is a panel to the top and left of the worksheet's row and column headings that indicates which rows and columns show details and which show summary data.
- **Create and format charts**  
You can use a chart to create a graphic representation of report data. BIRT Spreadsheet Designer supports the use of standard chart types such as bar, graph, line, pie, and bubble. Connecting to a data source keeps chart data, like all report data, accurate and up-to-date.
- **Working with multiple worksheets**  
You can add multiple worksheets to a workbook, and can use data fields, report script functions, and other expression elements to link the sheets together.

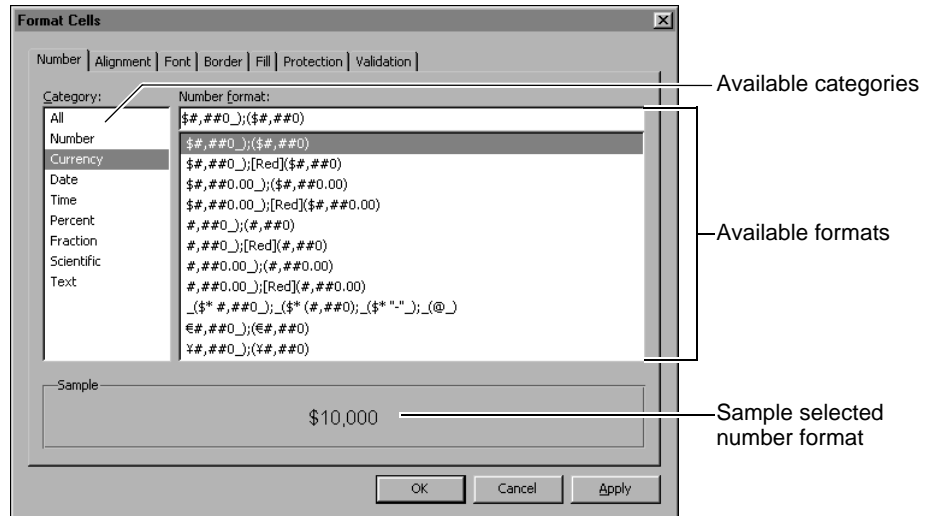
---

## Formatting numbers

BIRT Spreadsheet Designer provides a variety of ways to work with numbers and number formats. You can use pre-defined number formats for currencies, dates, percentages, and scientific formulas, and you can create custom number formats as well.

## Accessing number formatting options

To access BIRT Spreadsheet Designer's standard number formats, choose **Format**→**Cells** from the main menu. **Format Cells**—**Number** appears, as shown in Figure 8-1. The **Category** pane displays all the types of number formats available. When you select a category, such as **currency**, all the pre-defined currency formats appear in the **Number Format** pane. Similarly, when you select the **date** category, standard date formats appear in the right pane under **Number Formats**. The **All** category displays all the formats for all categories in the **Number Format** pane. Any custom number formats that you create appear under **All**.



**Figure 8-1** Number format options

To view a sample number format in the **Sample** section of **Format Cells**, first type a number in a cell in a data range. Right-click in the cell. Then, choose **Format Cells** from the context menu. On **Format Cells**, select the **Category** and **Number Format** to sample. To view the selected format, look in the **Sample** area.

## Creating a custom currency format

To create and apply a custom currency format to a data cell, you perform the following actions:

- Add the appropriate currency data fields to the report range. Typically, you perform this step by dragging a data field from the data set into a cell in the data range.
- Copy a currency symbol into BIRT Spreadsheet Designer. You use the Microsoft Character Map tool to select and copy the currency symbol.

- Create and apply a custom currency format to a data cell. You use the Format Cells—Number dialog to replace the dollar symbol in a custom currency format.

BIRT Spreadsheet Designer saves the custom formats you create so you can reuse them. You can also create custom date and other number formats.

---

## Merging cells

You can combine two or more cells within one cell border by merging the cells. For example, you can merge cells to create a title or subtitle, such as 2nd Quarter, that spans multiple columns. Merged cells function as one cell on a worksheet, whose row and column reference appears in the upper left corner of the range. For example, if you merge cells A1:B5, the resulting cell is A1. A reference to any other cell in the merged range returns the same value as a blank cell. If you merge cells with different values, the merged cell contains the value of the upper left cell.

You can merge cells in the following ways:

- To merge a range of adjacent cells, use the merge report script function without a key value.
- To merge cells with repeating values, use the merge report script function with a key value.
- To merge cells in a heading, use the merge menu command.

### About the merge report script function

You can use the merge report script function in data cells, and in row and column sections. You can also use the merge report script function with or without a key value. To apply the merge report script function with a key value, use the following syntax:

```
#merge(key_value)
```

where `key_value` is a field name, text value, numeric value, or Boolean expression. For example, to merge all the cells in a range that contains equal city values, use:

```
#merge([city])
```

where `city` is the field name.

### Merging a range of adjacent cells

To merge text over a group of dynamically generated cells, use merge without a key value. For example, to display By Month as a single-cell heading for a report



section that expands to include several column sections, use the following expression:

```
#"By Month" merge()
```

To merge cells in a section that contains multiple rows or columns, you must ensure that the cells are contiguous. To do so, place the merge function in each column or row to be merged. In Figure 8-2, the Month column section spans two columns, C and D. To ensure that the merged heading in row 1 displays across all the section's columns, place the merge function in both columns.

	A	B	C	D	E	F	
1			#"By Month" merge(#merge)		Header		
2	#write(LastName)		#(PortfolioDate)		ClientHeader		
3			Current Value	Original Value		Client	ROWS
4			#sum(SUMofCurrentV	#sum(SUMofOrigina	ClientTotals		
5		#write(Name	#sum(SUMofCurrentV	#sum(SUMofOrigina	Security		
6		#sum(SUMof	#sum(SUMofCurrentV	#sum(SUMofOrigina	Totals		
7							
8	Title	ColTotals	Month				
9			Columns				

**Figure 8-2** Placing the merge function in contiguous cells

Figure 8-3 shows how the merged heading spans across the cells in row 1.

	A	B	C	D	E	F
1			By Month			
2			Jan		Apr	
3			Current Value	Original Value	Current Value	Original Value
4			\$137,195	\$137,195	\$131,230	\$137,195
5	Reed	3M	\$69,270	\$69,270	\$63,825	\$69,270
6		Alcoa	\$9,085	\$9,085	\$7,805	\$9,085
7		Boeing	\$19,960	\$19,960	\$20,890	\$19,960
8		Caterpillar	\$23,620	\$23,620	\$24,960	\$23,620
9		Walt Disney	\$15,260	\$15,260	\$13,750	\$15,260
10			\$816,520	\$137,195	\$131,230	\$137,195

**Figure 8-3** Viewing merged cells in the generated report

When you use the merge function without a key value in a section area, all cells in that section area appear merged.

## Merging cells with repeating values

This section shows you how to use a key value to merge cells with repeating values. For example, the report in Figure 8-4 shows orders by date and order status. Identical headers in adjacent cells, such as 2006 in row 1, appear multiple times. In row 2, the same month appears in adjacent cells.

	A	B	C	D	E	F	G	H	I	J
1			2006	2006	2006	2006	2006	2006	2006	2006
2			Apr	Apr	Apr	Apr	May	May	May	May
3		Order Status	Cancelled	Closed	In Evaluation	Open	Cancelled	Closed	In Evaluation	Open
4	CustID	101	0	0	0	0	0	1	0	0
5		102	0	0	0	0	1	0	0	0
6		106	0	1	0	0	0	0	0	0
7		109	0	1	0	0	1	0	0	0

**Figure 8-4** Report with recurring heading text

Figure 8-5 shows the same report with merged date headings. Row 1 merges cells with a common year. Row 2 merges cells with common month values.

	A	B	C	D	E	F	G	H	I	J
1			2006							
2			Apr				May			
3	CustID	Order Status	Cancelled	Closed	In Evaluation	Open	Cancelled	Closed	In Evaluation	Open
4	101		0	0	0	0	0	1	0	0
5	102		0	0	0	0	1	0	0	0
6	106		0	1	0	0	0	0	0	0
7	109		0	1	0	0	1	0	0	0

**Figure 8-5** Report with merged cells and headings

To merge the year cells shown in Figure 8-5, the following merge expression was appended to the orderDate data field in the report design:

```
merge (year(orderDate))
```

To merge the month cells shown in Figure 8-5, the following merge expression was appended to the orderDate data field in the report design:

```
merge (month(orderDate))
```

When you use merge with a key value in a section area, all the cells in the section area that share the same key value appear merged.

## Merging cells in a heading

To create a merged header above or to the left of a section that expands to show multiple rows or columns, you create a section that applies to the heading cell only. Then you right-click the section header and choose Merge Cells from the context menu. BIRT Spreadsheet Designer merges the cells in the intersection of those sections.

For example, the information shown in Figure 8-6 is grouped by customer under the heading, Reed, in cell A2.

	A	B	C	D	E	F
1			By Month			
2	Reed		Jan		Apr	
3			Current Value	Original Value	Current Value	Original Value
4			\$137,195	\$137,195	\$131,230	\$137,195
5		3M	\$69,270	\$69,270	\$63,825	\$69,270
6		Alcoa	\$9,085	\$9,085	\$7,805	\$9,085
7		Boeing	\$19,960	\$19,960	\$20,890	\$19,960
8		Caterpillar	\$23,620	\$23,620	\$24,960	\$23,620
9		Walt Disney	\$15,260	\$15,260	\$13,750	\$15,260
10		\$816,520		\$137,195		\$137,195

**Figure 8-6** Before merging the customer name heading

Figure 8-7 shows the same report after merging the cells in the Reed section. Notice that Reed's name now spans cells A2-A9, all of the cells in that customer section.

	A	B	C	D	E	F
1			<b>By Month</b>			
2			Jan		Apr	
3			<b>Current Value</b>	<b>Original Value</b>	<b>Current Value</b>	<b>Original Value</b>
4			\$137,195	\$137,195	\$131,230	\$137,195
5	<b>Reed</b>	3M	\$69,270	\$69,270	\$63,825	\$69,270
6		Alcoa	\$9,085	\$9,085	\$7,805	\$9,085
7		Boeing	\$19,960	\$19,960	\$20,890	\$19,960
8		Caterpillar	\$23,620	\$23,620	\$24,960	\$23,620
9		Walt Disney	\$15,260	\$15,260	\$13,750	\$15,260
10		\$816,520		\$137,195		\$137,195

**Figure 8-7** After merging the customer name heading

To facilitate this merge, a column section, Title, shown in Figure 8-8, was created. Notice that the Title section and the Client row section are highlighted. This means that they are both selected.

	A	B	C	D	E	F
1			#"By Month" merge0	#merge0		Header
2	#write(LastName)		#PortfolioDate			ClientHeader
3			<b>Current Value</b>	<b>Original Value</b>		Client
4			#sum(SUMofCurrentValui	#sum(SUMofOriginal		
5		#write(Name)	#sum(SUMofCurrentValui	#sum(SUMofOriginal		
6		#sum(SUMofCui		#sum(SUMofOriginal		
7						Totals
8	Title	ColTotals	Month			

**Figure 8-8** Using the merge menu command

To apply the merge, you select the two sections at which the heading intersects. In this sample design, cell A2 is where both sections intersect. You then choose Merge Cells from the context menu, shown in Figure 8-9. In the generated report, the cells in the Client section appear merged, as shown in Figure 8-7.

	A	B	C	D	E	F
1			By Month" merge0	#merge0		Header
2	#writ		#PortfolioDate			ClientHeader
3			urrent Value	Original Value		Client
4			um(SUMofCurrentValui	#sum(SUMofOriginal		
5			um(SUMofCurrentValui	#sum(SUMofOriginal		
6				#sum(SUMofOriginal		
7						Totals
8			Month			
9			Columns			
10						
11						

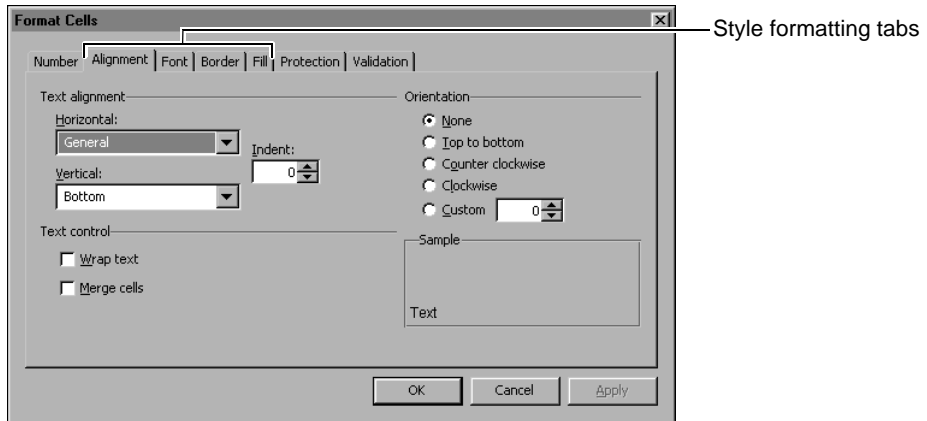
**Figure 8-9** Selecting the merge menu command

## Formatting styles

BIRT Spreadsheet Designer offers many of the same style formatting options as Excel. You can apply different font sizes, types, and styles, as well as borders and background colors to an entire worksheet or section, or to a single cell or group of cells.

To access BIRT Spreadsheet Designer style options, choose Format→Cells from the main menu. The Format Cells dialog contains seven tabs, four of which contain style formatting options:

- **Alignment.** Use this tab, shown in Figure 8-10, to change the alignment and orientation of text values.



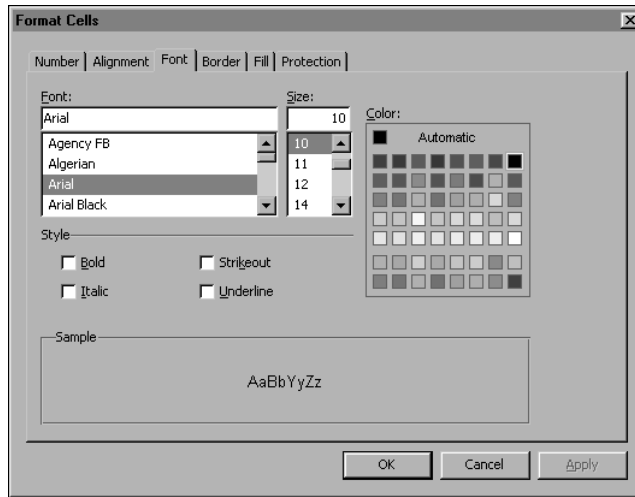
**Figure 8-10** Examining style formatting tabs

- **Font.** Choose the Font tab to change the font size, type, and style of cell values. You can also change font color from this tab.
- **Border.** Choose Border to add a border to the selected cell, range of cells, or worksheet. You can also add border color from this tab.
- **Fill.** Use the Fill tab to apply a solid or patterned background to a cell, range of cells, or worksheet. You can also change fill color from this tab.

## Applying a theme

A theme is a set of formatting options. BIRT Spreadsheet Designer provides 20 themes, each of which includes a different font. Each theme also includes a set of color choices. Theme colors appear on the color palette on each of the style formatting tabs.

For example, when you choose Office theme, Format—Font displays the Arial font. In Color, the palette displays the selection of colors from Office theme, as shown, rendered in grayscale, in Figure 8-11.



**Figure 8-11** Font style and colors in the Office theme

For more information about interactions between theme colors and custom colors, see “Modifying the color palette,” in Chapter 17, “Personalizing BIRT Spreadsheet Designer.”

### How to apply a theme

- 1 Open a new workbook, then select a cell, a group of cells or a data range.
- 2 Select **Format**→**Choose Theme**, or select Themes from the standard toolbar.
- 3 On **Choose Theme**, select a theme name, then choose **OK**.



## Using conditional formatting

You can use BIRT Spreadsheet Designer to create and apply formatting to cell values based on conditions you set. For example, you can instruct BIRT Spreadsheet Designer to apply a red background color to a cell when its value falls below 100, and a blue background color to the cell when its value exceeds 100. You can create up to three conditions for each cell.

You can base conditional formatting on:

- The value in a cell  
To base the condition on a cell value, you supply one or two numeric values, text values, or cell references that BIRT Spreadsheet Designer uses to compare the values. If the data meets the criteria, BIRT Spreadsheet Designer applies the format.

- A formula

You can base a condition on any formula that returns a TRUE or FALSE value. If the formula returns TRUE, BIRT Spreadsheet Designer applies the conditional format. If the formula returns FALSE, BIRT Spreadsheet Designer does not change the format.

You can use the following types of formatting to highlight a condition:

- Font colors and styles
- Border colors and styles
- Fill colors and patterns

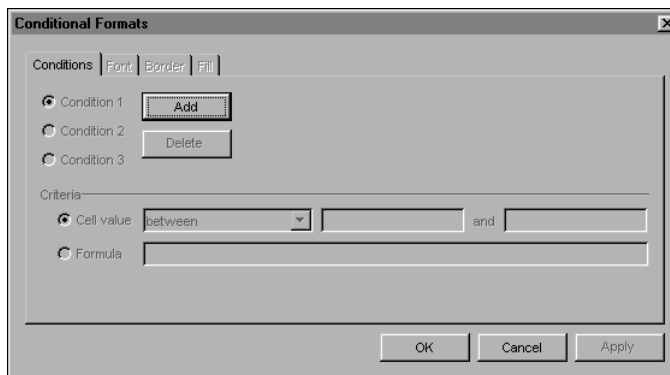
## Creating a conditional format

You can use the Conditional Formats dialog to create up to three conditional formats. Using Conditional Formats, you first create the condition. Then you choose the formatting you want BIRT Spreadsheet Designer to apply when the cell values meet the condition.

### How to create a conditional format

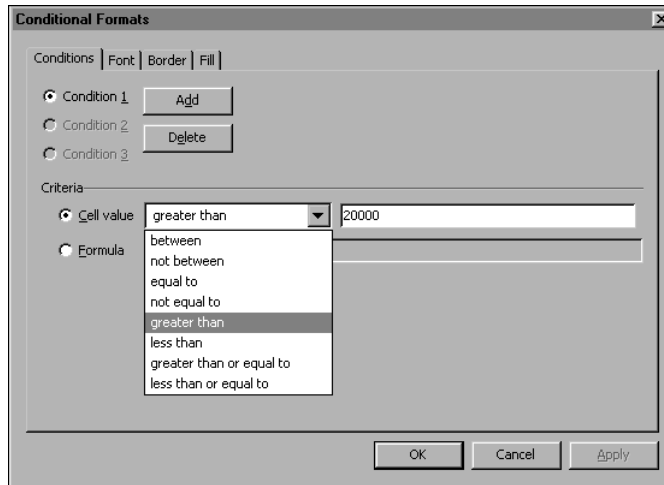
The following procedure creates a conditional format that applies a fill color to quarterly profits that exceed \$20,000.

- 1 Select the cell or range to which to apply the condition, then choose **Format**→**Conditional**. Conditional Formats appears, as shown in Figure 8-12.



**Figure 8-12** The Conditional Formats dialog

- 2 Choose Add. This choice makes the fields in the Criteria section available.
- 3 To define a condition, perform the following actions, as shown in Figure 8-13:
  - In Cell value, select a comparison operator.
  - In the next field, type a value.



**Figure 8-13** Setting conditional criteria

Choose Apply.

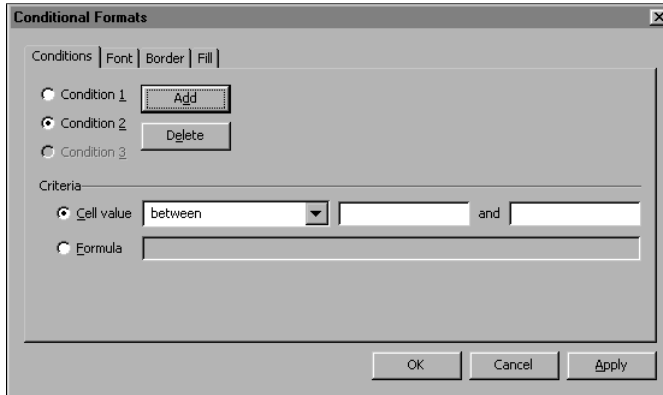
- 4 Select Fill.
- 5 On Fill, select Solid for the type of fill, then choose a fill color. Then choose OK.
- 6 Choose Run to preview the results, shown in Figure 8-14. In this example, profit totals over \$20,000 are highlighted in a solid color.



3				Q1/2004		Q2/2004	
4	Sales Rep	Customer Name		Revenue	Profit	Revenue	Profit
12	Fixter		Totals	\$44,895	\$18,348	\$0	\$0
13		Australian Collectables, Ltd		\$0	\$0	\$0	\$0
14		Australian Collectors, Co.		\$44,895	\$18,348	\$0	\$0
15		Souvenirs And Things Co.		\$0	\$0	\$0	\$0
16	Marsh		Totals	\$54,145	\$20,374	\$119,212	\$47,513
17		Down Under Souvenirs, Inc		\$0	\$0	\$37,281	\$14,006
18		Extreme Desk Decorations, Ltd		\$31,670	\$12,560	\$0	\$0
19		GiftsForHim.com		\$0	\$0	\$37,769	\$16,242
20		Handji Gifts & Co		\$22,474	\$7,814	\$44,161	\$17,264
21		Kelly's Gift Shop		\$0	\$0	\$0	\$0
22	1102						
23	Bondur		Totals	\$139,758	\$59,129	\$25,081	\$10,893
24		Auto Canal+ Petit		\$49,165	\$22,311	\$25,081	\$10,893
25		La Come D'abondance, Co.		\$0	\$0	\$0	\$0
26		Lyon Souvenirs		\$0	\$0	\$0	\$0
27		Marseille Mini Autos		\$0	\$0	\$0	\$0
28		Reims Collectables		\$0	\$0	\$0	\$0
29		Saveley & Henriot, Co.		\$90,593	\$36,818	\$0	\$0
30	Bott		Totals	\$32,306	\$11,842	\$90,051	\$35,714

**Figure 8-14** Viewing conditional results

- 7 To create another condition for the same cell, ensure the cell is selected, then choose Format>Conditional.
- 8 On Conditional Formats, choose Add. Condition 2 appears selected, as shown in Figure 8-15. Now you can set the condition and format for Condition 2.



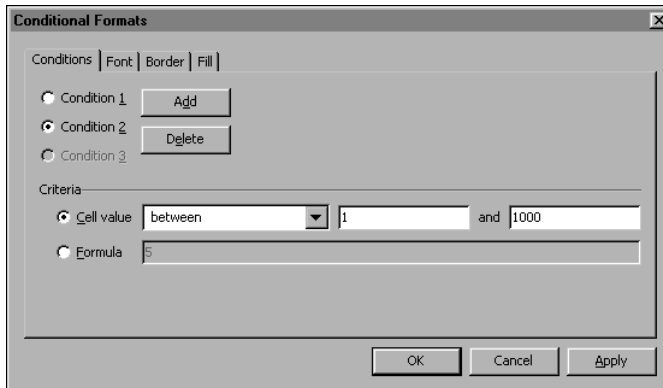
**Figure 8-15** Adding condition 2

## Removing a conditional format

To remove a conditional format from a cell, use the procedure described in the next section.

### How to remove a conditional format

- 1 Select the cell or range that contains the condition to delete, then choose Format→Conditional.
- 2 On Conditional Formats, select the condition to delete. In Figure 8-16, Condition 2 is selected.



**Figure 8-16** Removing a conditional format

- 3 Choose Delete, then choose OK.
- 4 Preview the report to ensure that the condition was removed.





## Creating a conditional format with an expression

Another way to apply a conditional format is to use the format report script function in an expression. You can include data fields, parameter values, and additional report script functions and operators in the expression. You type the conditional expression directly into the formula bar for a cell, just as you would any other expression.

To apply the format report script function, use the following syntax:

```
format (value_format)
```

where value\_format identifies the format. For example, if you type:

```
format (" [red] ")
```

in a cell, BIRT Spreadsheet Designer applies the color red to the values in that cell.

You can also use the if report script function with the format function to create a Boolean expression that evaluates the cell's value. For example, the following formula instructs BIRT Spreadsheet Designer to display a cell's values in green when the current value is greater than 500:

```
if ([CurrentValue]>500, format (" [green] ") )
```

---

## Creating a report outline

You can use an outline to display different levels of summary and detail information in a spreadsheet report. An outline is a panel at the top and left of a worksheet's row and column headings that indicates which rows and columns show details and which show summary data. End users can collapse and expand the rows and columns to show more or less detail, as they prefer. Outlining is particularly helpful when a generated report contains numerous columns or rows.

Figure 8-17 shows a generated report with an outline panel at the top, to expand and collapse columns, and another on the left, to expand and collapse rows. In Figure 8-17, the detail rows for the two Porsche models are collapsed. For the Jaguar, the detail rows are expanded. When you design the outline, you decide which rows and columns to include in the outline.

Figure 8-17 illustrates a spreadsheet with row and column outlines. The spreadsheet is divided into three sections: '2003' (rows 2-10), '2004' (rows 11-19), and '2005' (rows 20-27). The '2003' section is expanded, showing the 'Product Name' column. The '2004' section is collapsed, showing only the 'Totals' row. The '2005' section is expanded, showing the 'Product Name' column. The spreadsheet is divided into three sections: '2003' (rows 2-10), '2004' (rows 11-19), and '2005' (rows 20-27). The '2003' section is expanded, showing the 'Product Name' column. The '2004' section is collapsed, showing only the 'Totals' row. The '2005' section is expanded, showing the 'Product Name' column.

Product Line	Product Name	Customer Name	Totals	MSRP	Purchase Price	Customer Savings
Classic Cars	1948 Porsche 356-A			\$770	\$539	\$231
	1948 Porsche Type			\$1,413	\$622	\$791
	1949 Jaguar XK 120			\$909	\$473	\$436
	Anna's Decorations, Ltd			\$91	\$47	\$44
	AV Stores, Co.			\$91	\$47	\$44
	Down Under Souvenirs, Inc			\$0	\$0	\$0
	Euro+ Shopping Channel			\$0	\$0	\$0
	FunGiftIdeas.com			\$0	\$0	\$0

**Figure 8-17** Row and column outlines in a worksheet

## About outline components

You create row and column outlines separately. Each outline consists of the following components:

- Detail bracket**  
 A detail bracket spans the columns and rows included in an individual outline level. You can have single or multiple brackets on the same level, as shown in Figure 8-17.
- Detail buttons**  
 A detail button indicates a summary row or column. When you expand an outline level, a minus sign appears on the detail button. When you collapse an outline level, a plus sign appears on the detail button.
- Detail dots**  
 Rows or columns marked by a dot are typically detail rows and are hidden when you collapse the bracket.
- Outline buttons**  
 Outline buttons are numbered to indicate the number of levels contained in the outline. To collapse or expand all the columns or rows in a level, click the level's outline button.

- **Outline level**

Each row and column outline contains at least two levels: a summary level and a detail level. A report design can have up to eight levels of outlined rows and columns.

## Creating a report outline

To create a report outline, you perform the following tasks:

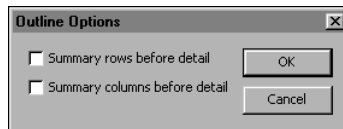
- Specify the summary data location.
- Define outline levels for each row and column outline.

The procedures that follow use a modified version of the ComparePrices.sod to illustrate how to create report outlines. Instructions on how to create the ComparePrices SOD file are provided in Chapter 1, “Building your first spreadsheet reports.”

### How to specify the location of summary data

In this task, you choose whether to display total or subtotal information before or after detail information in the outline.

- 1 Select the rows or columns to which to add the outline.
- 2 Choose Data→Group and Outline→Outline Options from the main menu. Outline Options appears, as shown in Figure 8-18.



**Figure 8-18** Outline Options

- 3 On Outline Options, select one of the following display options:
  - To move row detail buttons to the top edge of the bracket, select Summary Rows Before Detail.
  - To move column detail buttons to the left edge of the bracket, select Summary Columns Before Detail.

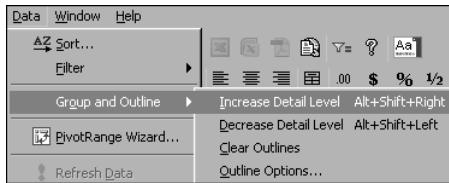
Choose OK.

### How to create a row outline

This How-to shows you how to create a row outline with a single level. This outline enables end users to hide the detail rows and display only summary rows.

- 1 In Design Editor, select the row or rows to include in the outline. In this example, a single row, 7, is selected.

- 2 Choose Data→Group and Outline→Increase Detail Level, as shown in Figure 8-19.



**Figure 8-19** Increasing the detail level

A row detail bracket and detail button appear to the left of row 7, as shown in Figure 8-20. Notice that the detail button is in expand mode, and that row 7 is visible. Click the detail button to see row 7 disappear from view.

1	2	A	B	C	D	E	F	G	H
	2	Product Line				#ComparePrices:write(orderDate)			
	3	#write(productLine)							
	4	Product Name							
	5	#write(proc	Customer Name		MSRP	Purchase	Customer		
	6				Totals	um(MSRP)	#sum(buyF	#sum(Savin	
	7		#write(customerN	um(MSRP)	#sum(buyF	#sum(Savin		CustName	
	8								
	9					Year			

**Figure 8-20** Viewing an expanded detail level



- 3 Preview the report to test how the row outline looks on the generated report, which is shown in Figure 8-21. Notice that row 7 is hidden, and that the detail button is in collapse mode.

#### How to create a multi-level column outline

- 1 In the design editor, select the column or columns to include in the outline. In this example, two columns, E and F, are selected.
- 2 Choose Data→Group and Outline→Increase Detail Level.

A column detail bracket with detail button appears above columns E-G, as shown in Figure 8-22. Notice that the detail button is in expand mode, and that columns E and F are visible. To collapse the columns, click the level 1 button or the detail button.

1	2	A	B	C	D	E	F	G	H
	2	Product Line				#ComparePrices:write(orderDate)			
	3	#write(productLine)							
	4	Product Name							
	5	#write(proc	Customer Name		MSRP	Purchase	Customer		
	6				Totals	um(MSRP)	#sum(buyF	#sum(Savin	
	8								
	9					Year			

**Figure 8-21** Viewing a collapsed row outline

	1						
	2						
1	2	A	B	C	D	E	F
	2	Product Line				ComparePrices:write(orderDate	
	3	#write(productLine)					
	4	Product Name					
	5		Customer Name			Purchase Price	Customer Savings
	6	#write(proc	Name		MSRP		
	7			Totals	um(MSRP)	#sum(buyF	#sum(Savi
	8			#write(customerN	um(MSRP)	#sum(buyF	#sum(Savi
	9					Year	

**Figure 8-22** Viewing an expanded column outline

- 3 To create an additional column outline with multiple levels, do the following steps:
  - 1 Select the column or columns to include in the outline. In this example, columns A and B are selected.
  - 2 Choose Data>Group and Outline>Increase Detail Level.

This choice inserts a detail bracket above columns A, B, and C, as shown in Figure 8-23. To collapse columns A and B, click the outline button labeled 1 or the detail button.

	1						
	2						
1	2	A	B	C	D	E	F
	2	Product Line				ComparePrices:write(orderDate	
	3	#write(productLine)					
	4	Product Name					
	5		#write(product	Customer Name		MSRP	Purchase Price
	6				Totals	um(MSRP)	#sum(buyF
	7			#write(customerName	um(MSRP)	#sum(buyF	#sum(Savi

**Figure 8-23** Viewing multiple expanded column outlines

- 3 To create a third level for this outline, select column A and choose Data>Group and Outline>Increase Detail Level.

This choice inserts a detail bracket above columns A and B, as shown in Figure 8-24. Notice that the column outline panel now contains two outlines, and that the second outline contains 3 levels. To hide column A, click level 2 or the detail button.

	1						
	2						
	3						
1	2	A	B	C	D	E	F
	2	Product Line				ComparePrices:write(orderDate	
	3	#write(productLine)					
	4	Product Name					
	5		#write(product	Customer Name		MSRP	Purchase Price
	6				Totals	um(MSRP)	#sum(buyF
	7			#write(customerName	um(MSRP)	#sum(buyF	#sum(Savi

**Figure 8-24** Viewing nested expanded column outlines



- 4 Preview the report.

On the newly generated report, all of the outlines appear in expanded mode. To test the outlines, take the following steps:

- 1 Click the detail button above column G to collapse columns E and F. The report now appears as shown in Figure 8-25. Notice that the MSRP and Purchase Price columns are no longer visible.

	1 2 3					
1/2		A	B	C	D	G
	2	<b>Product Line</b>				2003
	3	Classic Cars				
	4	<b>Product Name</b>				
	5		1948 Porsche 356-	<b>Customer Name</b>		<b>Customer Savings</b>
	6				<b>Totals</b>	\$231
	7			Australian Collectors, Co.		\$0
	8			Blauer See Auto, Co.		\$23
	9			Collectables For Less Inc.		\$23
	10			Cruz & Sons Co.		\$23

**Figure 8-25** Viewing a collapsed column outline

- 2 Click the detail button over columns C and J to display only customer savings by name for the years 2003 and 2004. The report now appears as shown in Figure 8-26.

	1 2 3				
1/2		C	D	G	J
	2			2003	2004
	3				
	4				
	5	<b>Customer Name</b>		<b>Customer Savings</b>	<b>Customer Savings</b>
	6		<b>Totals</b>	\$231	\$277
	7	Australian Collectors, Co.		\$0	\$23
	8	Blauer See Auto, Co.		\$23	\$0
	9	Collectables For Less Inc.		\$23	\$0
	10	Cruz & Sons Co.		\$23	\$0
	11	Down Under Souvenirs, Inc		\$0	\$23
	12	Euro+ Shopping Channel		\$23	\$23

**Figure 8-26** Viewing a completely collapsed column outline

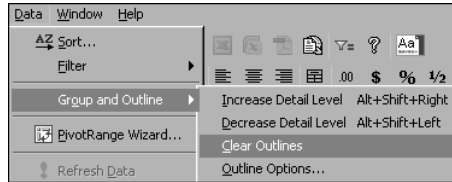
- 3 Click the level 3 button to expand both column outlines.

## Removing an outline or an outline level

You can clear an outline, which removes all detail brackets and buttons from the data range. When an outline contains multiple levels, you can remove the levels individually.

### How to clear an outline

- 1 In the design editor, select the rows or columns that are included in the outline.
- 2 Choose Data>Group and Outline>Clear Outlines, as shown in Figure 8-27.

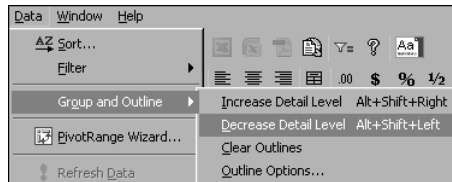


**Figure 8-27** Clearing an outline

Clear Outlines removes the outline from the data range.

### How to remove an outline level

- 1 Select the row or column to remove from the bracket.
- 2 Choose Data→Group and Outline→Decrease Detail Level, as shown in Figure 8-28.



**Figure 8-28** Removing an outline level

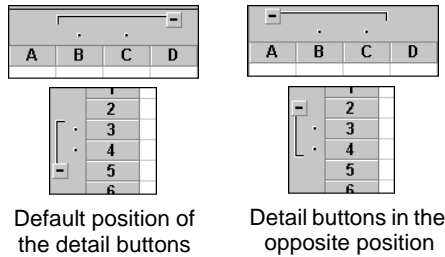
This procedure removes the row or column from the outline. If the selected row or column appears at the end of a detail bracket, the bracket shrinks to include the remaining columns or rows in the outline. If it is the only row or column in a bracket, that bracket disappears. If the row or column appears in the middle of a detail bracket, that bracket splits into two brackets.

When you can remove detail from two or more detail levels, choosing Data→Group and Outline→Decrease Detail Level always removes rows or columns from the lowest detail level of the outline.

To remove multiple levels at once, select multiple rows or columns before choosing Data→Group and Outline→Decrease Detail Level.

## Setting outline display options

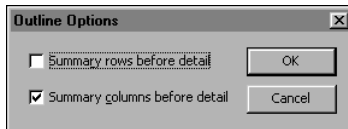
By default, BIRT Spreadsheet Designer places the detail button to the right of column outline brackets and below row outline brackets. You can also display the buttons in the opposite position. Figure 8-29 shows how the outlines look in the default position, right and bottom, and in the opposite positions, left and top. The rows or columns in the detail bracket do not change when you change the position of a detail button.



**Figure 8-29** Outline button positions

#### How to display a detail button in the opposite position

- 1 Select the rows or columns for which to change the detail button position.
- 2 Choose Data→Group and Outline→Outline Options from the main menu.
- 3 On Outline Options, select Summary columns before detail, as shown in Figure 8-30.



**Figure 8-30** The Outline Options dialog

Choose OK.

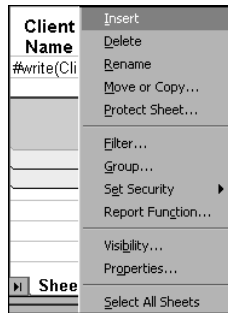
## Working with workbooks and worksheets

In BIRT Spreadsheet Designer, a report design file consists of a single workbook. By default, a new workbook contains one worksheet, and you can add others. Worksheets in a workbook can contain related information and be connected through cell references or hyperlinks, or can be completely independent of one another. Thus, the report design can vary for each sheet. For example, you can show sales data on one worksheet and operations data on another. You can create a sales report that uses different worksheets to arrange and display data by region, sales representative, and product line.

A worksheet can contain a single or multiple data ranges. For information on using multiple data ranges in a report design, see Chapter 7, “Laying out a report.”

To access the worksheet menu, right-click on a sheet tab. Worksheet menu options appear, as shown in Figure 8-31.





**Figure 8-31** Worksheet menu

You can perform the following actions from the worksheet menu:

- Insert a worksheet.
- Rename a worksheet.
- Delete a worksheet.
- Choose security options.
- Access the report script function builder.
- View or change sheet properties.



# Adding content to a data range

This chapter contains the following topics:

- About adding content
- Adding data fields to a data range
- Adding detail to a data range
- Setting up data filters
- Creating totals and subtotals
- Summarizing data in hierarchies
- Using defined names and other reference tools
- Working with data that contains null values
- Working with static content

---

## About adding content

BIRT Spreadsheet Designer requires two elements to generate data for a report: data fields from a data set, and a report script function assigned to the data fields. You add data fields to a report design after constructing the layout. When you add a data field to a cell, you specify a report script function for it. The report script function tells BIRT Spreadsheet Designer how to display the values associated with the data field in the generated report. For example, the sum function for a data field called Profit calculates and displays the sum of all Profit values. The write function for a CustomerName data field locates customer names in the data source and displays them as text.

BIRT Spreadsheet Designer enables you to calculate, manipulate, and display data values in a number of ways. You can create expressions that retrieve different levels of data, and that filter through the available data using criteria or conditions you set. You can also create expressions that display different types of totals and subtotals.

---

## Adding data fields to a data range

You can add data to a range in two ways:

- By dragging and dropping data fields from the data explorer into the data range.
- By typing an expression that uses a data field and a report script function directly into a cell's formula bar.

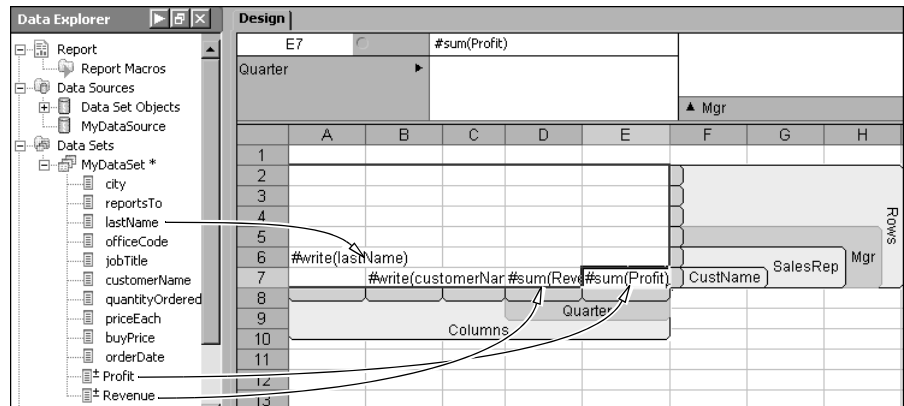
As you add data fields to a range, you can preview the report to see how the generated data appears in the layout. You can then return to the report design to modify the layout and data fields as needed.

### About dragging and dropping data fields

In BIRT Spreadsheet Designer, you can add data to a report design by dragging a field from the data set and dropping it into a cell in a data range. You can drag any field in a data set, including computed fields you create.

When you drop the data field into a cell, BIRT Spreadsheet Designer prompts you to add a report script function, such as sum, average, count, or write. After you specify the report script function, the data field appears in the cell, preceded by the function name. Figure 9-1 shows how a data range looks after data fields have been dragged from the Data Explorer and dropped into cells in a range.

For step-by-step instructions about dragging and dropping fields, see Tutorial 2, in Chapter 1, "Building your first spreadsheet reports."



**Figure 9-1** Dragging and dropping data fields

## Working with report script functions

BIRT Spreadsheet Designer uses a report script function to determine how to display values in a data field. A report script function can retrieve and filter data values and use mathematical operators to perform a calculation. Report script functions can affect the display and calculation of data in individual cells, sections, and sheets.

A cell, section, and sheet can contain single or multiple report script functions. The following expression contains one report script function:

```
#sum(priceEach*quantityOrdered)
```

This sample expression instructs BIRT Spreadsheet Designer to multiply and display the combined value of two data fields, priceEach and quantityOrdered.

More complex expressions can use a combination of report script functions, operators, and references to retrieve more tailored results. For example, the following expression in a section:

```
select ([Region]="Northwest") exclude ([State]="Idaho" and
[qtyOrdered] < 50)
```

instructs BIRT Spreadsheet Designer to retrieve order data for the Northwest region and to exclude data from Idaho if the quantity ordered there falls below 50. Type expressions that use more than one report script function directly into a cell or section's formula bar. To assign a single report script function to a cell, use the report script function builder, which is discussed later in this chapter.

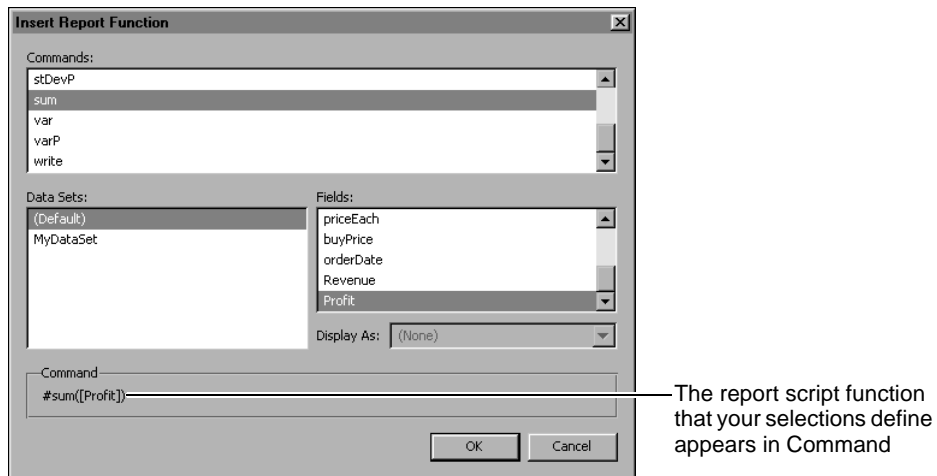
In addition to creating and applying report script functions at the cell and section levels, you can use report script functions at the sheet level to display data across multiple sheets in a workbook. This technique is known as sheet bursting and is explained in Chapter 7, "Laying out a report."

When you use a report script function in a cell, you must precede the report script function with #. When you use a report script function in a section or on the sheet level, do not precede the report script function with #. Sample report script functions and expressions are provided throughout this chapter. For a more complete description of each report script function, see Chapter 18, “Report script reference guide.”

## Using the report script function builder

When you drop a data field into a cell, BIRT Spreadsheet Designer prompts you to specify a report script function for that field. This prompt is the Insert Report Script Function dialog, also known as the report script function builder.

Figure 9-2 shows how Insert Report Script Function appears after you drop the Profit field into the data range shown in Figure 9-1.



**Figure 9-2** Inserting a report script function

Insert Report Script Function includes the following three panes:

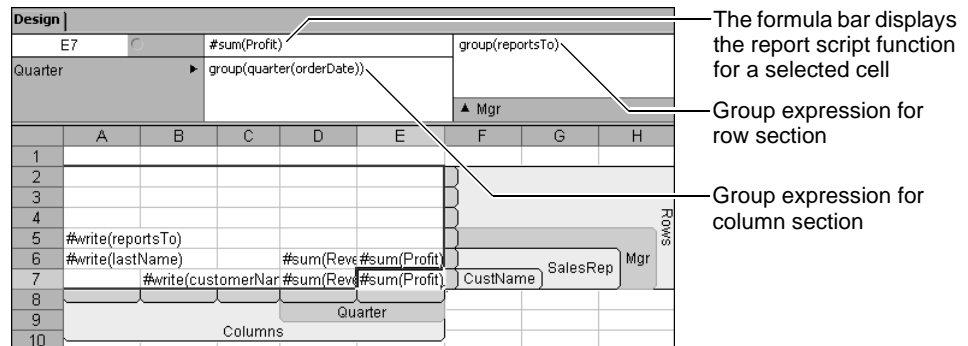
- **Commands**  
In Commands, you select the command, or function, you want BIRT Spreadsheet Designer to perform on the data field. In Figure 9-3, the selected command is sum.
- **Data Sets**  
In Data Sets, you choose the data set that contains the data field that you drag and drop in the data range cell. When only one data set exists, that data set appears as the default data set. Select either Default or the data set name.

## ■ Fields

In Fields you choose from the list of data fields in the data range. The field that you last dropped in the data range appears selected.

After you select items in each pane, Command shows the report script function that your selections define.

When you choose OK, the report script function appears in the cell, and, because the cell is selected, in the formula bar also, as illustrated in Figure 9-3.



**Figure 9-3** The report script function and formula bar areas

## Adding detail to a data range

Use detail retrieval functionality to return the most detailed data available for a section. This functionality displays one row in the report for every data row that the query returns. You display detail rows by setting a section's grouping properties. You can further refine results by modifying the detail report script function on the section.

For example, the generated report in Figure 9-4 shows stock values and share quantities for securities owned by two clients over a three-month period. In this report, the security section lists the names of the securities, along with the total monthly value and number of shares for each security. No additional detail is provided. Notice that client Benitez owned 1000 shares of Sears stock in January, for a total value of \$44,990, and 3000 shares of Sears stock in February, for a total of \$115,150.

Figure 9-5 shows the same report design modified by adding a detail section. This example shows details of three, 1000-share purchases, each valued at a different amount, that sum to the total value of Benitez' Sears stock in February. You can also see that Benitez' January Sears total came from a single, 1000-share purchase.

Design   View		Client Name						
A4	C							
3	A	B	C	D	E	F	G	H
4	Client Name	Security Name	Value of stock	Number of shares	Value of stock	Number of shares	Value of stock	Number of shares
5	Benitez							
6		Sears Roebuck	\$44,990	1000	\$115,150	3000	\$121,610	3000
7		Walt Disney	\$106,350	3000	\$222,210	9000	\$209,640	9000
8		Caterpillar	\$140,730	3000	\$327,690	9000	\$333,630	9000
9		General Motors	\$167,850	3000	\$276,480	6000	\$468,690	9000
10		Alcoa	\$112,290	3000	\$241,710	9000	\$240,720	9000
11		Allied Signal	\$76,460	2000	\$172,420	6000	\$173,120	6000
12		Texaco	\$150,360	3000	\$350,130	9000	\$369,030	9000
13		Dupont	\$110,180	2000	\$238,020	6000	\$256,960	6000
14		Boeing	\$93,700	2000	\$228,620	6000	\$227,640	6000
15		Hewlett Packard	\$118,700	2000	\$272,840	6000	\$255,800	6000
16	Boivent							
17		Allied Signal	\$38,230	1000	\$86,210	3000	\$86,560	3000
18		Walt Disney	\$70,900	2000	\$148,140	6000	\$139,760	6000
19		Union Carbide	\$42,750	1000	\$109,780	3000	\$116,530	3000
20		American Expre	\$0	0	\$201,810	6000	\$208,320	6000
21		Hewlett Packard	\$178,050	3000	\$409,260	9000	\$383,700	9000

**Figure 9-4** Sample report before applying detail function

Design   View		Client Name						
B22	C							
3	A	B	C	D	E	F	G	H
4	Client Name	Security Name	Value of stock	Number of shares	Value of stock	Number of shares	Value of stock	Number of shares
5	Benitez							
6		Sears Roebuck	\$44,990	1000	\$115,150	3000	\$121,610	3000
7			\$44,990	1000	\$0	0	\$0	0
8			\$0	0	\$20,120	1000	\$0	0
9			\$0	0	\$52,050	1000	\$0	0
10			\$0	0	\$42,980	1000	\$0	0
11			\$0	0	\$0	0	\$19,020	1000
12			\$0	0	\$0	0	\$56,410	1000
13			\$0	0	\$0	0	\$46,180	1000
14			\$0	0	\$0	0	\$0	0
15			\$0	0	\$0	0	\$0	0
16			\$0	0	\$0	0	\$0	0
17			\$0	0	\$0	0	\$0	0
18		Walt Disney	\$106,350	3000	\$222,210	9000	\$209,640	9000
19			\$106,350	3000	\$0	0	\$0	0
20			\$0	0	\$46,500	3000	\$0	0
21			\$0	0	\$111,390	3000	\$0	0
22			\$0	0	\$64,320	3000	\$0	0
23			\$0	0	\$0	0	\$106,230	3000
24			\$0	0	\$0	0	\$62,730	3000
25			\$0	0	\$0	0	\$40,680	3000

**Figure 9-5** Sample report after applying detail function

## Creating a detail report script function

You can create a detail report script function in the following ways:

- Use Section Filtering and Grouping to select the sort order in which detail rows appear on the generated report.
- Type characters that comprise a report script function in the formula bar. Use this option when an expression requires multiple report script functions or numerous data fields.



**How to use Section Filtering and Grouping to create a detail report script function**

- 1 Add a new row to the Security section. To do so, right-click the row 6 section stub, and choose Insert Row After. As shown in Figure 9-6, two rows, 6 and 7, now appear in Security.

Design   View							
D7						Cannot enter Report Func	
		Cannot enter Report Functions on leaf section.					
	A	B	C	D	E	F	
3	#write(PortfolioDate) merge(Pmerge(PortfolioDate)				Rows		
4	Client Name	Security Name	Value of stock	Number of shares			
5	#write(Client.Last)						
6		#write(Name	#sum(CurrentValu	#sum(CurrentQua			
7					ClientTotals	Security	Client
8					New row		
9			Value				
10			Month				
11		Section1					
12		Columns					

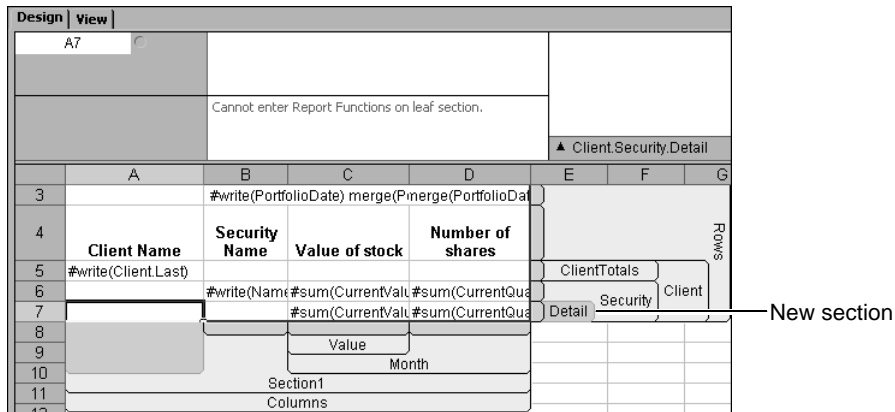
**Figure 9-6** After adding a new row

- 2 Drag data fields from Data Explorer and drop them in cells in the new row. Select the same fields that appear as totals in the parent section.
- 3 Right-click the row 7 stub and choose Create Parent, as shown in Figure 9-7.

Design   View							
D7		#sum(CurrentQuantity)				Cannot enter Report Fu	
		Cannot enter Report Functions on multiple sections.					
	A	B	C	D	E	F	
3	#write(PortfolioDate) merge(Pmerge(PortfolioDate)						
4	Client Name	Security Name	Value of stock	Number of shares			
5	#write(Client.Last)						
6		#write(Name	#sum(CurrentVal	#sum(CurrentQua			
7			#sum(CurrentVal	#sum(CurrentQua	ClientTotals		Selected row stub
8					Security	Client	
9			Value		Edit...		
10			Month		Delete		
11		Section1			Rename...		
12		Columns			Create Parent...		
13					Filter...		

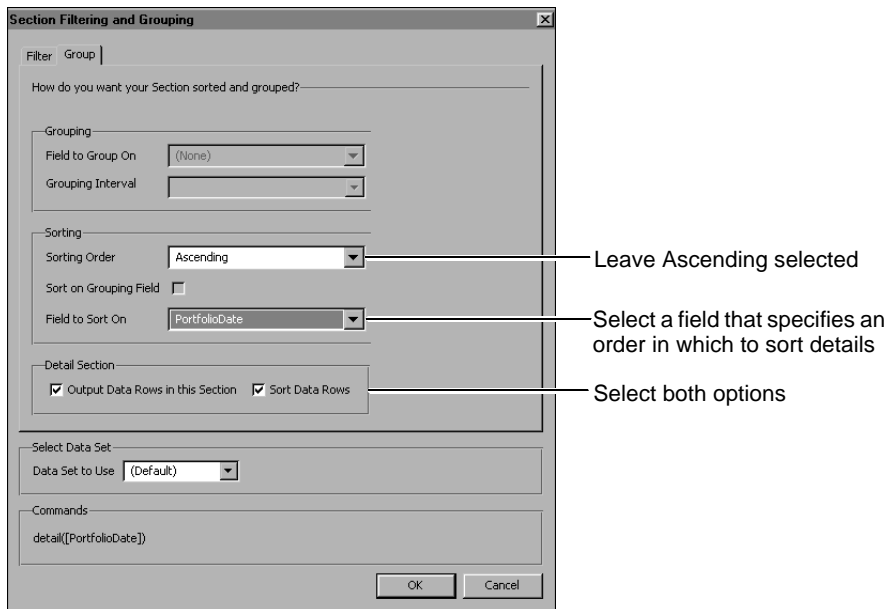
**Figure 9-7** Naming a detail subsection

- 4 On Create Section, in Name, type the following name, then choose OK:  
Detail  
The Detail subsection appears in the Security section, as shown in Figure 9-8.



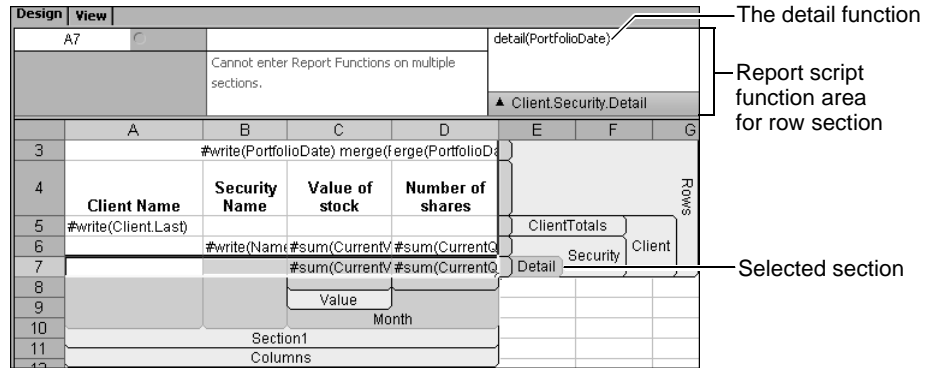
**Figure 9-8** Viewing the detail subsection

- 5 Right-click the Detail section and choose Group from the context menu.
- 6 On Section Filtering and Grouping, make the selections shown in Figure 9-9, then choose OK.



**Figure 9-9** Defining the detail function

The detail function you created appears in the detail section's report script function area, as shown in Figure 9-10.



**Figure 9-10** Viewing the detail function

7 Format the cells in the detail section as desired.



8 Choose Run to see how the generated report looks after applying the detail function to a section.

Review Figure 9-5 in the previous section to see how the sample report used in this procedure looks after adding the detail report script function to the Security section.

## Setting up data filters

A result set contains all the data rows that a data set can provide to a data range. You can filter a result set to return different subsets of data. Later, you can refine the subset by adding or removing data rows.

For example, a result set from the grocery example database can include meat, dairy, produce, bakery, and household cleaning subsets. Using this result set, you could create a filter to retrieve just the bakery subset, which contains pies, cakes, and cookies. To further refine the subset, you could exclude cookies or cakes.

## Defining a data subset

Use the select function to retrieve a subset of data rows from a result set. You can use select to create a simple filter, or to create a more complex Boolean expression that uses multiple report script functions and data fields. Typically, you use a select report script function only once in a nested set of sections. If you use the select function on an inner section, you should include the same conditions as on an outer section's select function, then refine the filter with further conditions. Consider using the include or exclude functions, which are discussed later in this chapter, rather than nested select functions. Select filters the entire result set, whereas include and exclude filter the set of rows that an outer section passes to an inner section.

In the following example, select filters the prodName data field:

```
select ([prodName]="apples")
```

In this case, the select function limits report output to apples rather than all produce names. The next expression uses two report script functions and two data fields to narrow report output to apples ordered in May:

```
select ([prodName]="apples" and month([orderDate]) = 5)
```

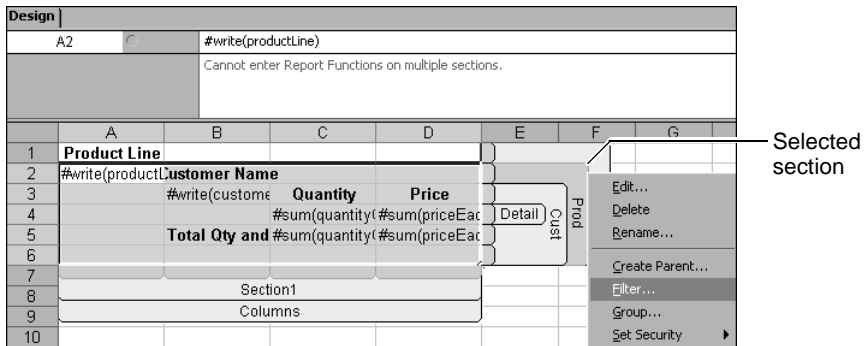
You can create a select report script function in the following ways:

- Use Section Filtering and Grouping to create a filter that uses one or more simple Boolean expressions.
- Type characters that comprise a report script function in the formula bar. Use this option when your filter requires multiple report script functions or numerous data fields.

### How to filter report data using Section Filtering and Grouping

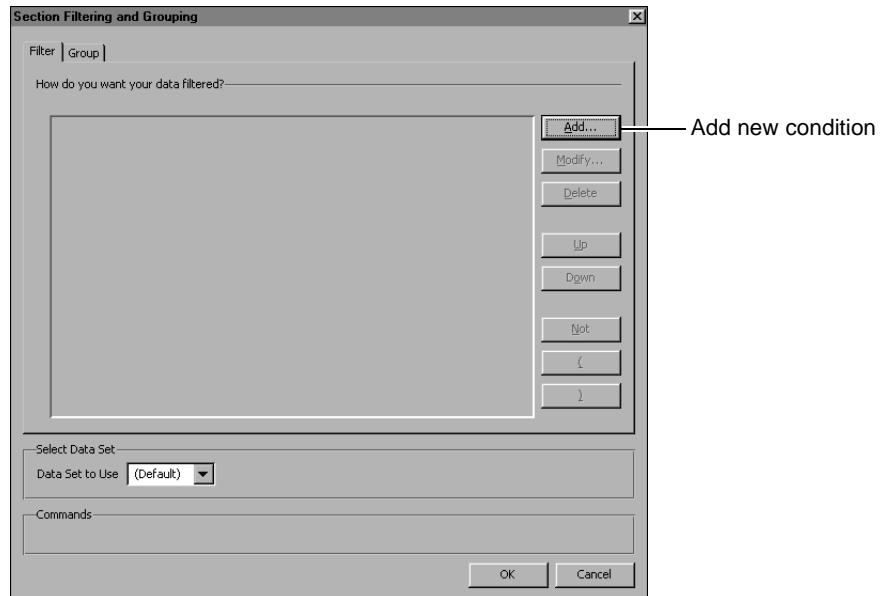
In this procedure, you add a filter to a parent-level section group. You can filter data at any section level.

- 1 In a data range, right-click a parent-level section tab, then choose Filter from the context menu, as shown in Figure 9-11.

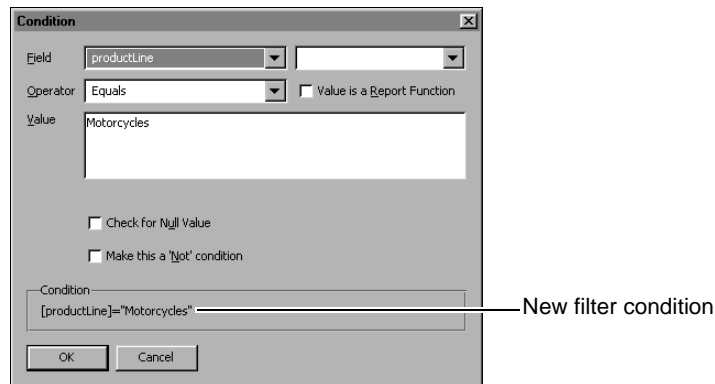


**Figure 9-11** Adding a filter to a section tab

- 2 In Section Filtering and Grouping, choose Add, as shown in Figure 9-12.
- 3 In Condition, do the following steps, as shown in Figure 9-13. Then, choose OK.
  - 1 In Field, select the field to filter.
  - 2 In Operator, select the condition to apply to the field.
  - 3 In Value, type a value or a report script function expression.
  - 4 If the field has a string data type, and if the value is a report script function expression, select Value is a report script function.



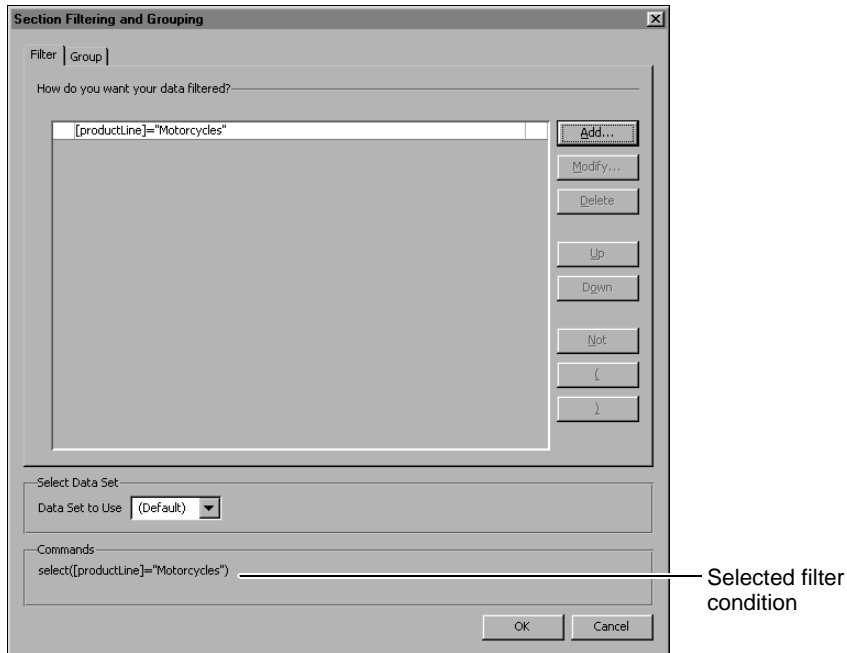
**Figure 9-12** Adding a new filter condition



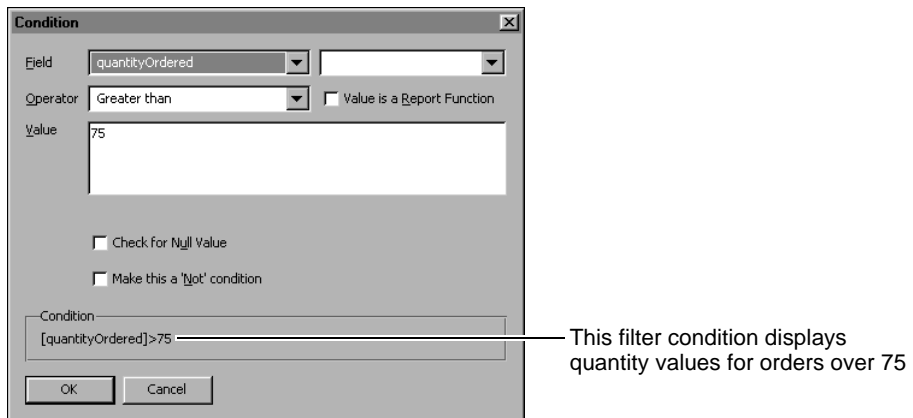
**Figure 9-13** Creating a filter condition

The condition that you added appears in the first row in the center pane of Section Filtering and Grouping, as shown in Figure 9-14.

- 4 To add another condition, choose Add.
- 5 On Condition, create a condition, as shown in Figure 9-15, then choose OK.



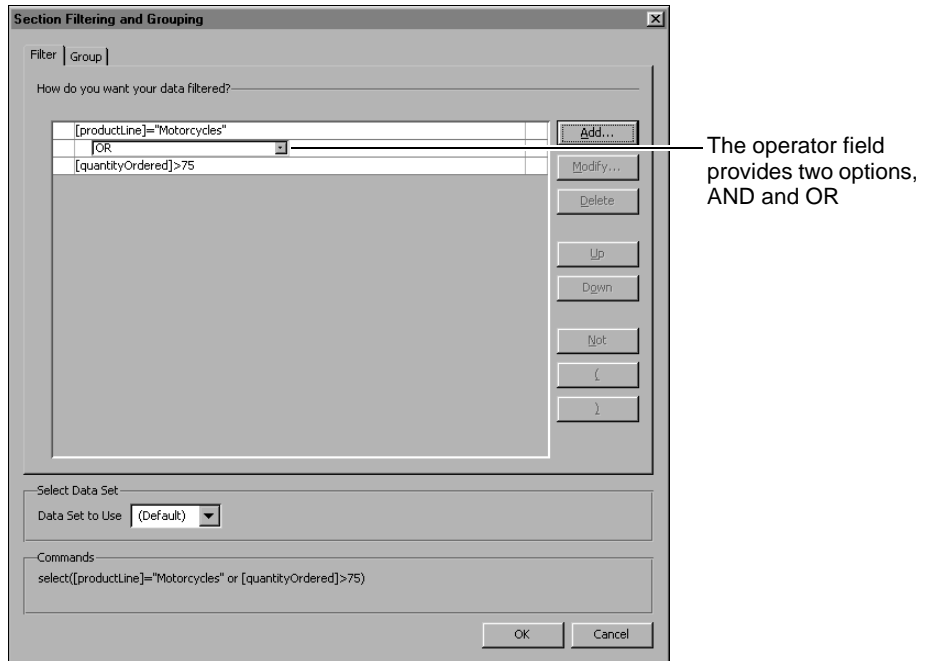
**Figure 9-14** Adding a second filter condition



**Figure 9-15** Creating a second filter condition

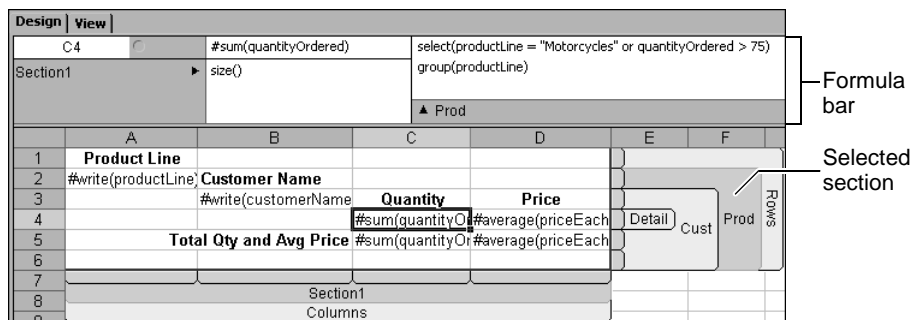
When a select expression contains more than one condition, an operator field appears between the two conditions in Section Filtering and Grouping.

- 6 Select an operator with which to join the two conditions, then choose OK. Figure 9-16, shows OR selected. The expression shown in Commands displays orders of over 75 for any product line, and all orders for motorcycles.



**Figure 9-16** Selecting the OR operator

The select expression appears in the formula bar, as shown in Figure 9-17.



**Figure 9-17** Viewing the select expression in the formula bar



- 7 Run the report to see how results returned by the filter condition appear on the generated report.

The generated report, shown in Figure 9-18, displays orders of any quantity for the motorcycle product line, and orders greater than 75 for all other product lines. Product lines appear in alphabetical order. Orders appear sorted by customer name in alphabetical order.

Design	View			
C4		77		
	A	B	C	D
1	<b>Product Line</b>			
2	Classic Cars	<b>Customer Name</b>		
3		Down Under Souvenirs, Inc	<b>Quantity</b>	<b>Price</b>
4			77	\$67
5			90	\$68
6		<b>Total Qty and Avg Price</b>	167	\$67
7				
8		Mini Caravy	<b>Quantity</b>	<b>Price</b>
9			97	\$115
10			76	\$128
11		<b>Total Qty and Avg Price</b>	173	\$121
12				
13		The Sharp Gifts Warehouse	<b>Quantity</b>	<b>Price</b>
14			76	\$82
15		<b>Total Qty and Avg Price</b>	76	\$82
16				
17	Motorcycles	<b>Customer Name</b>		
18		Anna's Decorations, Ltd	<b>Quantity</b>	<b>Price</b>
19			26	\$37
20			38	\$53
21			38	\$66
22			48	\$76
23			34	\$84
24			35	\$127
25		<b>Total Qty and Avg Price</b>	219	\$74
26				
27		Atelier graphique	<b>Quantity</b>	<b>Price</b>
28			32	\$61
29			39	\$106
30		<b>Total Qty and Avg Price</b>	71	\$83

**Figure 9-18** Viewing the generated report

## Changing condition order

When you use more than one Boolean expression to create a filter, you can use Section Filtering and Grouping to rearrange their order. Expression order is important when you use parentheses to group each expression.

To move an expression, select it, then choose Up or Down, shown in Figure 9-19.

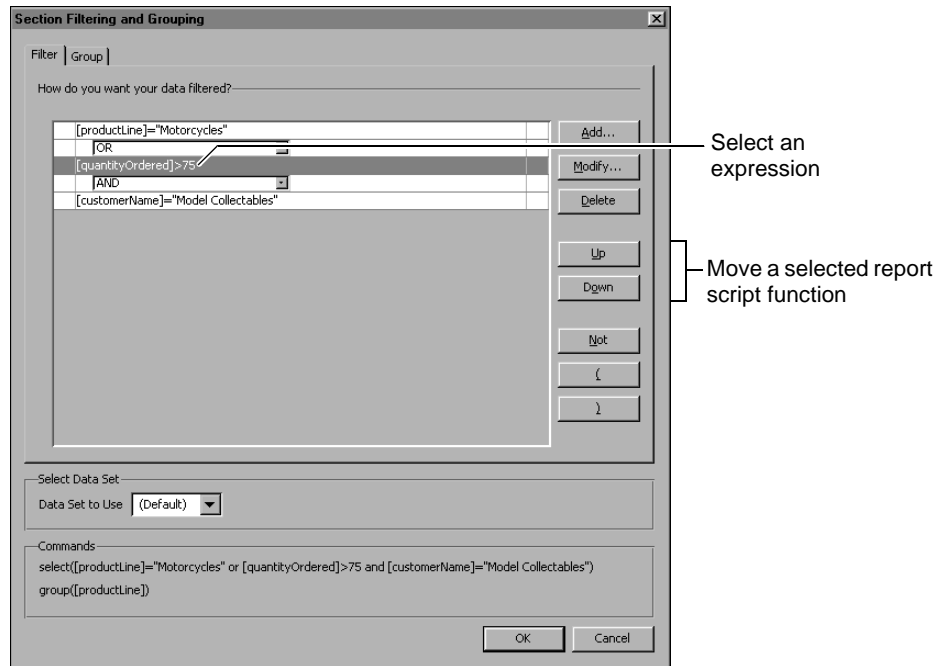
## Using include and exclude to filter data

Use the include and exclude report script functions to further refine a data subset. Typically, you use include and exclude after a select, group, or rollup report script function. Exclude removes data rows from a group or subset. Include adds rows from the result set to the subset.

For example, the following expression instructs BIRT Spreadsheet Designer to select data for the Motorcycle product line and to exclude one particular type of motorcycle from the results:

```
select ([productLine]="Motorcycles")
  exclude ([productName]="1969 Harley Davidson Ultimate Chopper")
```





**Figure 9-19** Changing select function order

In another example, the following expression uses `isnull` with a select report script function to retrieve data rows in which the Customer field is not null. The group report script function then creates groups based on State values. To display totals using all available data, the include report script function replaces the rows in which Customer is null:

```
select(not isnull([Customer])) group([State])
include(isnull([Customer]))
```

Using `include` and `exclude` can also improve performance in a data range. For example, when you select data from a set of six customers, using `exclude` to remove data with a Customer value of Simmons is more efficient than using `select` to retrieve data with Customer values of Moore, Rodriguez, Lyon, French, or Walter.

## Selecting data from multiple data sets

If a report design contains multiple data sets, you can use the select report script function to retrieve and arrange data from each set. For example, to retrieve country data from both `DataSet1` and `DataSet2`, use the following expression:

```
DataSet1:group([Country]) DataSet2:select([Country]
=DataSet1:[Country])
```

The first report script function creates group sections based on the Country values in DataSet1 and retrieves data for those groups. The second report script function retrieves country data from DataSet2 where the values in DataSet2 are the same as those in DataSet1.

Multiple data sets are connected by a join on one or more tables from each data set. For more information about working with multiple data sets, see Chapter 3, “Accessing a data source.”

---

## Creating totals and subtotals

In BIRT Spreadsheet Designer, a row that displays a total or a subtotal is known as a summary row. You can total or subtotal values for a range of contiguous cells, such as B4:B34, or for a number of non-contiguous cells, such as C4+C10+C16 or B6+D6+F8+G8. You can also summarize values for a single section or for multiple sections.

You can use the following options to generate summary data:

- Use the Excel SUM and SUBTOTAL functions.  
The SUM and SUBTOTAL options use Excel formulas to summarize data values. Both options enable end users to manipulate summary values on a generated report.
- Create a summary expression.  
Use any combination of report script functions, formulas, and data fields to retrieve summary data. Sample summary expressions are provided later in this chapter.
- Use external aggregation.  
You can use SQL aggregation or another aggregation tool to return a total or subtotal. Whereas the other options use the BIRT Spreadsheet engine to calculate totals and subtotals, an aggregate expression uses the native functionality of the report’s data source to summarize data values.

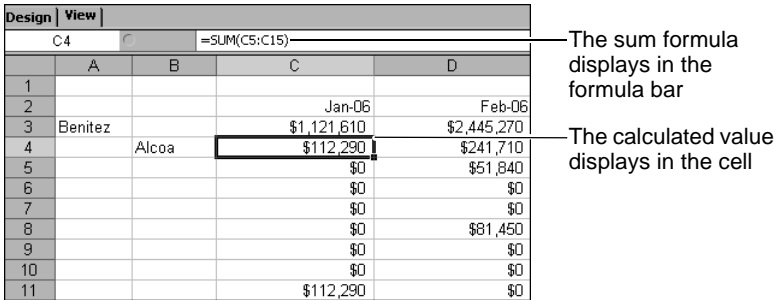
This section explains how to use the Sum and Subtotal options and how to add summary expressions to a data range. For instruction about adding totals and subtotals to a pivot range, see Chapter 13, “Working with a pivot range.” For instructions on creating a SQL aggregate expression, see “Summarizing row data in the query editor,” in Chapter 4, “Connecting to a database or JDBC data source.”

### Using Excel SUM and SUBTOTAL functions

The Excel SUM function and the BIRT Spreadsheet Designer sum report script function calculate data in the same way, but display it differently on the generated report. In addition to the summary value that both display, Excel

displays in the formula bar the formula used to calculate the value. Exposing the formula on the formula bar allows a report user to modify the formula on the generated report, then view the changed results.

Figure 9-20 shows how the Excel SUM function displays on a generated report. Notice that the editable SUM formula displays in the formula bar and the calculated value displays in the data cell. An end user can edit the characters in the formula bar to calculate a different result. The BIRT Spreadsheet Designer sum function, on the other hand, displays the calculated value in both the formula bar and in the cell.



**Figure 9-20** Viewing the subtotal formula

The Excel SUBTOTAL function also displays both the value and formula. BIRT Spreadsheet Designer does not have a single subtotal function, but provides a number of functions with which to create subtotals. Sample total and subtotal expressions are provided later in this section.

BIRT Spreadsheet Designer uses the formula report script function to call both Excel functions. For example, the following expression uses the formula report script function to call the Excel SUM function to sum the values in a nested section:

```
#formula("SUM("& cells(Section1.Section3)&")")
```

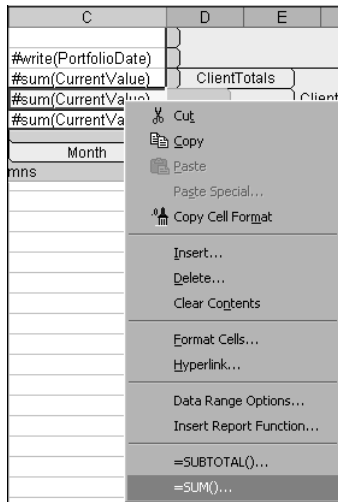
The Sum and SubTotal dialogs enable you to select from a list of parent and child sections in the data range.

## Using the Excel SUM function

The Sum dialog enables you to select the section that contains the values you want to sum.

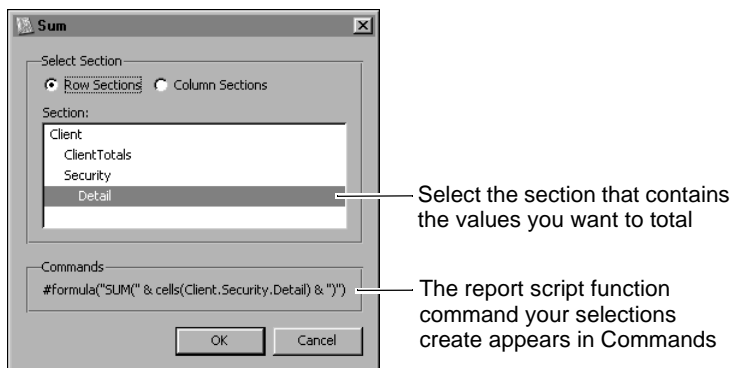
### How to use the Sum dialog to create a total in a row section

- 1 Right-click the cell in which to sum the total, and choose `=SUM()` from the context menu, as shown in Figure 9-21.



**Figure 9-21** Selecting the Sum option

- 2 On Sum, make the following selections, as shown in Figure 9-22. Then, choose OK.
  - In Select Section, select Row Section.
  - In Section, select the name of an inner section.



**Figure 9-22** Setting up a SUM expression

The expression includes both the formula report script function and the Excel Sum function. The expression appears in the formula bar for the selected cell, as shown in Figure 9-23.

Design		View	
C4		#formula("SUM(" & cells(Client.Security.Detail) & ")")	
Columns		section.	
1	A	B	C
2			#write(PortfolioDate)
3	#write(Client.Last)		#sum(CurrentValue)
4	#write(Name)		#formula("SUM(" & ce
5			#sum(CurrentValue)
6			Month
7			
8			

**Figure 9-23** Viewing the SUM formula in the formula bar



- 3 Preview the report. Figure 9-24 shows how the Excel SUM function appears in the generated report. In View, the editable formula appears in the formula bar, and the calculated value in the data cell.

Design		View	
C4		=SUM(C5:C15)	
1	A	B	C
2			Jan-06
3	Benitez		\$1,121,610
4		Alcoa	\$112,290
5			\$0
6			\$0
7			\$0
8			\$0
9			\$0
10			\$0
11			\$112,290

The editable sum formula

Calculated values

**Figure 9-24** Viewing the SUM formula on the generated report

## Using the Excel SUBTOTAL function

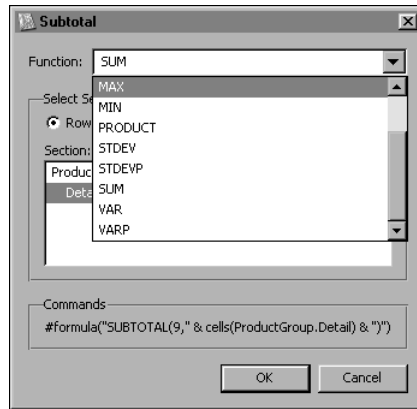
The Excel SUBTOTAL function is more versatile than the Excel SUM function. SUBTOTAL distinguishes between cells that contain a value and cells that contain other subtotal formulas, which means that it does not include subtotal values in other subtotals. The Excel SUM function, by contrast, calculates all the values in a range of cells without distinguishing between the different types of values.

The Subtotal dialog enables you to select from all the aggregate functions that Excel's SUBTOTAL function supports.

### How to use the Subtotal dialog to create a subtotal on a row section

- 1 Right-click the cell to which to add the subtotal, and choose =SUBTOTAL() from the context menu.
- 2 On Subtotal, make the following selections:
  - In Function, select the function by which to define the subtotal. This example uses MAX, as shown in Figure 9-25.
  - In Select Section, select Row Section.

- In Section, select the name of an inner section.



**Figure 9-25** Selecting a SUBTOTAL function

Choose OK. The subtotal expression you created appears in the formula bar for the selected cell. In this example, BIRT Spreadsheet Designer locates and displays the maximum value of all the cells in the section.

## Creating summary expressions

You can use a number of report script functions and operators to create total and subtotal expressions. For example, you can use the average function to compute the average value of cells in a section or range, or the var function to calculate the variance between the values in a range or section.

You can also create conditional expressions that return results based on conditions you define. For example:

```
#sum(if([priceEach] > 50, priceEach*quantity))
```

instructs BIRT Spreadsheet Designer to sum priceEach multiplied by quantity when priceEach is 50 or greater. For this example, BIRT Spreadsheet Designer would not include priceEach values of 50 or less in its calculation.

To create an expression, select a cell, then type the expression elements directly into the formula bar and press Enter.

## Using Excel functions in different locales

Typically, Excel functions, such as SUM and MAX, use different names and formats in different locales. For example, the function that calculates an average has a different name in U.S. English than it does in French. If you use text and report script functions to build a calculation in one locale, that calculation may not work when you deploy the report in other locales.

To create formulas that produce correct calculations in any locale, use the formula report script function with the following syntax:

```
#formula(formula_text)
```

where `formula_text` is the function or expression that builds the formula to use. For example, the following expression produces a sum calculation in any locale:

```
#formula("SUM(" & cells(Section1) & ")")
```

When you use the formula report script function, BIRT Spreadsheet Designer evaluates the formula in U.S. English even if the report runs in a different locale.

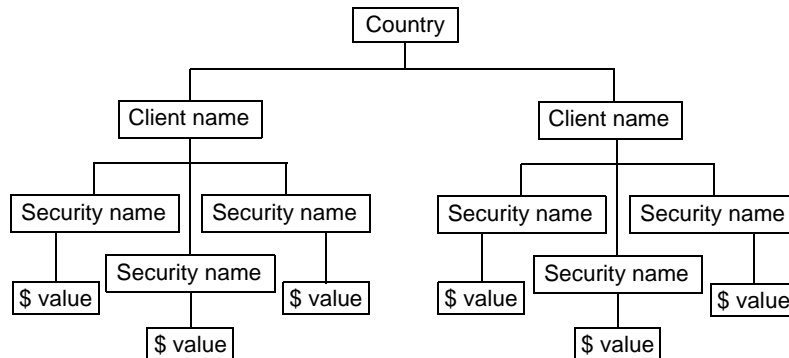
---

## Summarizing data in hierarchies

In a spreadsheet design you typically construct a data range that contains sections and groups to summarize data at multiple levels, in a hierarchy. A hierarchy consists of a top, or parent level and one or more subordinate, or child levels. A tree diagram shows multi-level, or hierarchical, structure.

### About an even data hierarchy

An even hierarchy exists when each branch includes the same number of child elements at each level. Lower levels in an even hierarchy appear parallel. For example, the tree diagram in Figure 9-26 shows an even data hierarchy. The top level called Country has one child level called Client name. Client name has one child level called Security name, and Security name has one child level called \$ value. Because each child level in this example has the same number of elements, the branches appear parallel.



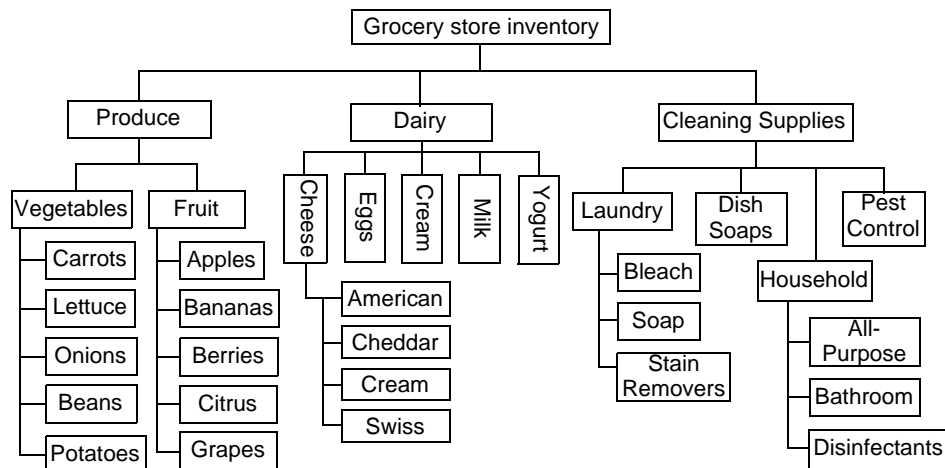
**Figure 9-26** Example of an even hierarchy

In a spreadsheet report design, you can construct a data range with sections so that each section represents a level in a data hierarchy. You can also group sections so that one group represents each branch. An even data hierarchy maps

especially well to nested, grouped sections in a data range. For simplicity, many examples in this book show summaries of an even data hierarchy.

## About an uneven data hierarchy

An uneven hierarchy exists when the number of child elements in each branch differs at one or more levels. Lower levels in an uneven hierarchy do not appear parallel. For example, the tree diagram in Figure 9-27 shows an uneven data hierarchy that represents a sample database called Grocery. In it, the Dairy branch has a child level with five elements. Of those five elements, only one, Cheese, has a child level. The Produce branch has one child level with two elements. Each of those child elements has a sublevel that contains five elements. The Cleaning Supplies branch has a child level with four elements. Of those elements, two have a child level and two do not.



**Figure 9-27** Example of an uneven data hierarchy

## Retrieving data from an uneven hierarchy

You can summarize data from an uneven hierarchy in two ways:

- Provide summaries or list details at the lowest child level in the hierarchy. For example, you can subtotal sales amounts for local sales representatives.
- Provide a summary for a parent level that includes subtotals from the child levels under it. For example, you can total sales amounts for each regional office.

To retrieve data from an uneven hierarchy, you must consolidate values from different levels of the hierarchy. You use the rollup function to search an uneven hierarchy. Rollup() evaluates two parameters to perform a specific search.



When creating an expression that uses a rollup function, use the following syntax:

`rollup(match_in_child_expression, match_in_parent_expression)`

where

- `match_in_child_expression` defines the value to use when searching the hierarchy for child-level data rows.
- `match_in_parent_expression` defines the value to use for parent-level data rows.

The BIRT Spreadsheet engine matches the value of `match_in_parent_expression` in a parent-level row to the `match_in_child_expression` value in a child-level row. The matching continues down the hierarchy until no further matches are available. You typically use the rollup function and the select function together in an expression to retrieve data from an uneven hierarchy.

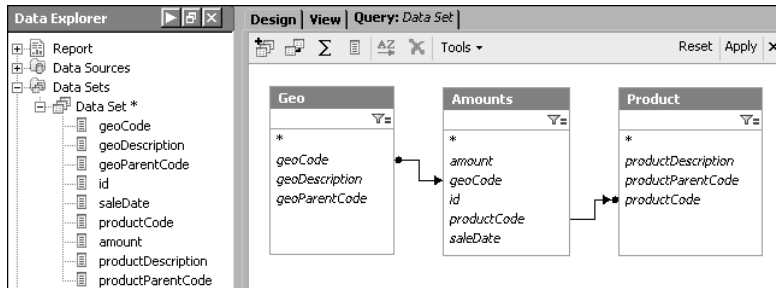
For example, suppose you need to develop a report that totals amounts of grocery products sold in various geographic markets. Data sets for both products and markets represent uneven hierarchies. The product data hierarchy was shown previously, in Figure 9-27. The following examples use the sample database, Grocery, to demonstrate reporting data from an uneven data hierarchy.

Figure 9-28 shows a view that lists the three market groups with geographic names: Europe, North America, and Japan. Europe and North America have subordinate markets, but Japan does not. Europe has one subordinate level. North America has three.

	A	B	C	D
	Parent Code	Parent Description	Geo Code	Geo Description
1				
2				
3			1	Europe
4			3	North America
5			100	Japan
6	1			
7		Europe	101	Germany
8		Europe	102	Italy
9		Europe	103	UK
10		Europe	104	Zurich
11	3			
12		North America	32	East US
13		North America	33	West US
14		North America	105	Canada
15	32			
16		East US	106	New England
17		East US	107	Atlantic Coast
18		East US	108	New York
19	33			
20		West US	109	Midwest
21		West US	110	Southwest
22		West US	111	West Coast
23				

**Figure 9-28** View that lists an uneven data hierarchy

To summarize the data from these uneven data hierarchies, you can first build a query that creates a single data set as shown in Figure 9-29.



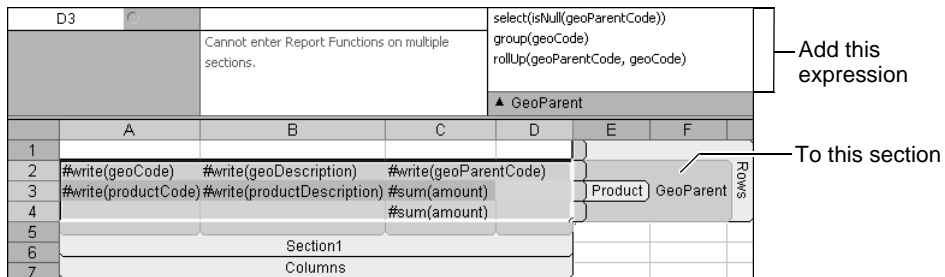
**Figure 9-29** Query that builds a data set from an uneven data hierarchy

Next, design a spreadsheet report that includes product and geographic market data in a data range section called GeoParent.

To find and fetch product amounts sold in each market, combine the select and rollup report script functions in the following expression:

```
select (isNull (geoParentCode)) group (geoCode)
      rollup (geoParentCode, geoCode)
```

Add this expression to the GeoParent section, as shown in Figure 9-30.



**Figure 9-30** Design that uses rollup to search an uneven data hierarchy

The report view shows the results calculated in the Geoparent section, and lists amounts and totals for each market. Figure 9-31 shows this view, with all products except the first and last in each market hidden for brevity.

In this example, the select function selects data for which GeoParentCode is null. These values are at the topmost, parent level of the hierarchy. The rollup report script function searches for data through all the descendent branches of Geoparent. Because the select expression only specifies the top level group, subordinate markets do not display in the report view.

To calculate and list amounts for lower level markets, you can create a run time parameter and add it to the select statement in the previous example. In the following, modified select statement the parameter's name is Area:

```
select (geoParentCode=:Area)
```

	A	B	C	D
1				
2		1 Europe	0	
4	2000	Beef	100250.16	
38	2038	Laundry Detergent	14087.3	
39			3512051.2	
40		3 North America	0	
42	2000	Beef	221954.12	
79	2038	Laundry Detergent	199378.82	
80			5343211.6	
81		100 Japan	0	
82	2001	Chicken	43580.62	
96	2038	Laundry Detergent	36162.3	
97			776293.1	

**Figure 9-31** View of rollup summary

The report view, when run with parameter value 3 for Europe, looks similar to the previous example but shows lower level data. The parameterized report view shown in Figure 9-32 lists amounts and totals for the regional markets at the child level subordinate to Europe. All product rows except the first and last in each market are hidden for brevity.

	A	B	C	D
1				
2		101 Germany	1	
3	2004	Smoked Meats	73062.19	
14	2035	Bathroom Cleaners	41791.31	
15			750678.47	
16		102 Italy	1	
17	2001	Chicken	25788.7	
38	2037	Bleach	62815.67	
39			1092524.8	
40		103 UK	1	
41	2000	Beef	100250.16	
60	2038	Laundry Detergent	14087.3	
61			1105324.2	

**Figure 9-32** View of parameterized rollup summary

For more information about parameters, see Chapter 12, “Designing a customizable report.”

## Using defined names and other reference tools

To refer to dynamically generated cells in a data range, you can use a combination of section names, defined names, individual cell references, and report script functions. Using a section reference ensures that the reference includes all the cells the section expands to include, whereas a defined name can make it easier to identify the data used in a formula. For example, unless you know that cells A1:A80 contain expense data, it is easier to understand the purpose of Sum(SalesExpenses), where SalesExpenses is the defined name, than it is to understand Sum(A1:A80).

## About defined names

To create a defined name, use the following syntax:

```
#name("SalesExpenses")
```

where name is the report script function used to create the defined name, SalesExpenses.

## Usage guidelines for defined names

When creating a defined name, use the following guidelines:

- Limit the name to 255 characters.
- Use no spaces or single quotation marks.
- Begin the name with a letter, `_`, or `\`. Supply alphabetic, numeric, `_`, `\`, `.`, or `?` characters for the remainder of the defined name.
- Do not use a name that is the same as any possible cell reference, such as CD45 or IS100. If the name matches a possible column reference, you must enclose the name in parentheses. This matching only occurs with a defined name up to 4 characters long that contains only letters. For example, you do not have to distinguish SAM\_1 from a column reference.
- Do not use a name that is the same as an existing defined name.
- Defined names are not case-sensitive.

## Creating a defined name

The examples in this section use a report design with summary and detail worksheets to show you how to create and use defined names. This design contains a sheet-level group that uses the Country data field. The Country group instructs BIRT Spreadsheet Designer to display client security data for each country on a separate sheet. For example, the Germany sheet lists stock data for German clients, while the France sheet displays stock information for French clients.

In addition, a separate Summary worksheet contains totals for each country, and grand totals for all countries combined, over a five-month period. Defined names are used to retrieve and calculate summary information from the individual sheets to display on the Summary sheet. For example, the following expression:

```
#sum(CurrentValue) name("Country" & Country &  
    month(PortfolioDate)) name("Country" & Country)
```

creates two defined names:

```
name("Country" & Country & month(PortfolioDate))
```

and:

```
name("Country" & Country)
```

On the generated Summary sheet, these defined names, together with the sum function and the CurrentValue data field, produce a value that constitutes the sum of values for each country and month, as shown in Figure 9-33. For example, cell D5 displays the sum total of USA data for month 3, March.

Design View								
D5		=SUM(CountryUSA3)						
	A	B	C	D	E	F	G	
1		Jan-2006	Feb-2006	Mar-2006	Apr-2006	May-2006	Totals	
2	France	\$26,957,870	\$63,001,840	\$63,937,950	\$27,430,395	\$61,584,340	\$242,912,395	
3	Germany	\$27,593,005	\$62,965,440	\$63,890,285	\$28,779,170	\$63,020,480	\$246,248,380	
4	Spain	\$26,118,860	\$59,796,665	\$60,498,870	\$26,985,735	\$59,370,035	\$232,770,165	
5	USA	\$95,099,765	\$226,839,080	\$229,756,275	\$100,327,755	\$224,642,070	\$876,664,945	
6	Grand Totals	\$175,769,500	\$412,603,025	\$418,083,380	\$183,523,055	\$408,616,925	\$1,598,595,885	
Summary		France	Germany	Spain	USA			

Figure 9-33 Focusing on a single cell in the Summary worksheet

Figure 9-34 displays the report design for the generated Summary sheet shown in Figure 9-33 and Figure 9-35. The formula displayed in cell B2 of Figure 9-34 creates the following defined name:

name("CountryTotal" & month(PortfolioDate))

The CountryTotal name is then used in a formula that instructs BIRT Spreadsheet Designer to sum the totals for each country for each month.

	A	B	C	D
1		#write(PortfolioDate)	Totals	
2	#write(Country)	#'=sum(Country" & Country & month(PortfolioDate) & ") name("CountryTotal" & month(PortfolioDate))	#'=sum(Country" & Country & ") name("CountryGrandTotal")	Country
3	Grand Totals	#'=sum(CountryTotal" & month(PortfolioDate) & ")	#'=sum(CountryGrandTotal")	
4		Month		

Figure 9-34 Creating and using multiple defined names

In Figure 9-35, the values displayed in row 6 are the sum of the individual country totals for each month. Notice that the formula for cell B6 explains that it contains the sum of all country totals for month 1, January.

B6		=SUM(CountryTotal1)						
	A	B	C	D	E	F	G	
1		Jan-2006	Feb-2006	Mar-2006	Apr-2006	May-2006	Totals	
2	France	\$26,957,870	\$63,001,840	\$63,937,950	\$27,430,395	\$61,584,340	\$242,912,395	
3	Germany	\$27,593,005	\$62,965,440	\$63,890,285	\$28,779,170	\$63,020,480	\$246,248,380	
4	Spain	\$26,118,860	\$59,796,665	\$60,498,870	\$26,985,735	\$59,370,035	\$232,770,165	
5	USA	\$95,099,765	\$226,839,080	\$229,756,275	\$100,327,755	\$224,642,070	\$876,664,945	
6	Grand Totals	\$175,769,500	\$412,603,025	\$418,083,380	\$183,523,055	\$408,616,925	\$1,598,595,885	
Summary		France	Germany	Spain	USA			

Figure 9-35 Viewing country totals for each month

The formula in cell C2, also shown in Figure 9-34, contains a formula that creates the following defined name:

name("CountryGrandTotal")

CountryGrandTotal is used to calculate the grand totals in column G, Totals, on the generated Summary page shown in Figure 9-36. CountryGrandTotal is also used to create the combined grand total that is displayed in cell G6.

G5		=SUM(Country:USA)					Cell formula
	B	C	D	E	F	G	
1	Jan-2006	Feb-2006	Mar-2006	Apr-2006	May-2006	Totals	
2	\$26,957,870	\$63,001,840	\$63,937,950	\$27,430,395	\$61,584,340	\$242,912,395	
3	\$27,593,005	\$62,965,440	\$63,890,285	\$28,779,170	\$63,020,480	\$246,248,380	
4	\$26,118,860	\$59,796,665	\$60,498,870	\$26,985,735	\$59,370,035	\$232,770,165	
5	\$95,099,765	\$226,839,080	\$229,756,275	\$100,327,755	\$224,642,070	\$876,664,945	Summary value
6	\$175,769,500	\$412,603,025	\$418,083,380	\$183,523,055	\$408,616,925	\$1,598,595,885	
Summary France Germany Spain USA							

**Figure 9-36** Viewing the Summary worksheet

## Using a conditionally defined name

You can instruct BIRT Spreadsheet Designer to create a defined name when a certain condition is met. For example, the following expression:

```
# [buyPrice] if(not isNull(buyPrice) and (quantityOrdered > 25),
    name("averageMe"))
```

assigns the defined name, averageMe, to buyPrice values only if the quantity associated with the buyPrice is 25 or greater, and if the buyPrice value is not null. The expression requires the not isNull element when you plan to use the defined name with an Excel function that does not discard null values. For example:

```
# "=AVERAGE (averageMe)"
```

uses an Excel function to average the buyPrice values specified by the defined name, averageMe.

## About cell references

Like Excel, BIRT Spreadsheet Designer supports the following types of cell references:

- A relative reference points to a cell based on its position relative to the current cell. When you move or copy a cell containing a relative reference, the reference adjusts to accommodate both the new location and the existing reference. For example, if you copy the formula =B1+B2 from cell A1 and paste it into cell A3, the formula adjusts down two rows to =B3+B4.
- An absolute reference points to a cell at an exact cell location. When you move or copy a cell containing an absolute reference, the reference does not change. A dollar sign (\$) precedes the column letter and row number in an absolute reference. For example, the absolute reference \$A\$1 points to cell A1.
- A mixed reference is part absolute and part relative. For example, \$A1 contains an absolute column reference and a relative row reference.

## Using the cells report script function

The cells function identifies the cells that occupy a specified section or range. You can use the cells function with another report script function to calculate or count the values in the cells identified by the cells function. The following example uses the cells report script function to identify and provide a subtotal for the cells that occupy the Region1 section:

```
#formula("sum(" & cells[Region1] & ")")
```

where

```
cells[Region1]
```

identifies the cells in region1 and the sum formula calculates a subtotal for those cells.

## About section references

To refer to a data range section, use the full section name, including any parent sections. For example, if a parent Client section contains a nested Security section, the full name of the security section is Client.Security. When referring to the name of a parent section, you can use the actual section name, or you can use the BIRT Spreadsheet Designer parent identifier, parent. To use the parent identifier, type parent and a colon, then type the report script function. For example, if Country is the parent section of Region, you can use either of the following identifiers to refer to Country from a Region cell:

- parent:
- Country:

You can use additional report script functions to refine data selection and to align child sections with the correct parent sections. For example, to retrieve data for Region sections in which the Amount value is greater than the average Amount value in the parent section, Country, you can use either of the following report script functions:

- `select([Amount]>parent:average([Amount]))`
- `select([Amount]>Country:average([Amount]))`

You can use the section identifier to use data from one section in a different section. To use data from a different section, precede the field reference or total calculation with the name of the section and a colon.

If you change the name of a section, BIRT Spreadsheet Designer automatically updates any report script functions that refer to it, to use the new name.

## About worksheet and workbook references

You can refer to cells on different sheets in the same workbook. You can also refer to cells in a separate workbook if both workbooks are open at the same time.

After you create a reference to a separate workbook, BIRT Spreadsheet Designer changes the format of the reference to show the absolute path to the workbook. For example, if you create the following reference to a workbook named September in the Payroll directory on your C drive:

```
[September] Payroll!C2:C420
```

BIRT Spreadsheet Designer changes it to:

```
'[C:\Payroll\September.sod] Payroll'!C2:C420
```

This absolute path appears in the worksheet cell. If you later move the September workbook, the external reference still works if you open September.sod before you open the workbook that refers to it.

The following rules apply to worksheet and workbook references:

- If the worksheet name contains spaces, enclose the worksheet name in single quotation marks:  
`'2003 Sales'!B17`
- If the workbook name contains spaces, enclose the workbook name in square brackets, and enclose the workbook and worksheet names in single quotation marks:  
`'[My Sheet.sod] Sheet1'!A1`
- To refer to cells that appear on two different worksheets, place a colon between the worksheet names. For example, Sheet1:Sheet2!A1 refers to cell A1 on Sheet1 and cell A1 on Sheet2. List the worksheets in the order in which they appear in the workbook. Figure 9-37 shows another example.



**Figure 9-37** Referring to cells on different worksheets

## Working with data that contains null values

In BIRT Spreadsheet Designer, a null value indicates that a variable or field contains no data. BIRT Spreadsheet Designer can leave cells that contain null values empty or display default values. In a numeric field, the default appearance for a null value is zero. In a date field in a report, the default appearance for a null value is:

1/0/1900

Depending on the design choices that you make, the calculation of values in the spreadsheet report can include or exclude null values. To avoid displaying



inaccurate data, you can use many different function expressions to work with data that contains null values. You can also change the way null values appear in report output.

## Handling null values with function expressions

You can choose to perform calculations on null and non-null values separately or together. Before choosing, consider both how null values affect calculated results and how null values appear in report output.

To separate null and non-null values, remove data rows with null values from sections, then add these rows to the result set later. For example, if the Customer field includes null values and you group row data by Customer, the row hierarchy returns some empty rows. To ensure that the data range uses only non-null values in a group hierarchy, use an expression that excludes null values, then creates group sections. Separating null from non-null values produces separate null and non-null results.

To select and group data where Customer is not null, use the select function with the group function:

```
#select(not isnull(Customer)) group(Customer)
```

To restore the rows with null Customer values to the result set, add an include report script function to the expression:

```
#select(not isnull(Customer)) group(Customer)
      include(isnull(Customer))
```

Calculating null and non-null values together may be more or less efficient than separating them, depending upon the data set and the required calculation.

You can ignore null values in some calculations that handle null and non-null values together. To ignore null values in aggregation calculations, use a report script function. For example:

```
#min(orderDate)
```

returns the earliest value in an order date field that is not null.

The following report design shows a more complex example that involves calculating null and non-null values together. The part of the data range shown in this example represents a product ordered by a range of customers during one year. Of course, not all customers who placed orders during this year purchased this particular product. Figure 9-38 shows several functions that handle the null values in this data range.

In column D, isNull returns a numeric result for non-null values or a string value that describes why no quantity exists. The sum function shown in columns E and F checks for null values by calculating the PriceEach data field as a numeric value. If no price exists, sum returns zero.

	D	E	F
1			
2	#ComparePrices:write(orderDate)		
3			
4	Quantity Ordered	Price	Price Rounded
5	#sNull(quantityOrdered, "Did not order	#sum(priceEach)	#sum(priceEach)
6	Average Purchase Price	#average(priceEach)	#formula("SUBTOTAL(1," & cells(ProdName.CustName)
8			
9	Year		
	Columns		

**Figure 9-38** Averaging data that contains null values

In the highlighted cell:

#average (PriceEach)

ignores any null values that represent no orders placed for a product during the period and returns an average price paid by only those customers who placed orders for this product.

Figure 9-39 shows the part of the report output generated from this design that represents one product. Comparing column F in both figures shows how null values affect an average calculated by using the Excel SUBTOTAL formula. SUBTOTAL includes the zeros that represent null values in the calculation. The result is less representative than the result calculated by using the average function shown in column E.

	D	E	F
	2003		
	Quantity Ordered	Price	Price Rounded
	Did not order this product this year	\$0.00	\$0
	Did not order this product this year	\$0.00	\$0
	50	\$115.52	\$116
	Did not order this product this year	\$0.00	\$0
	43	\$117.97	\$118
	41	\$119.20	\$119
	41	\$109.37	\$109
	40	\$111.83	\$112
	Did not order this product this year	\$0.00	\$0
	Did not order this product this year	\$0.00	\$0
	Did not order this product this year	\$0.00	\$0
	Did not order this product this year	\$0.00	\$0
	30	\$105.69	\$106
	33	\$114.29	\$114
	Did not order this product this year	\$0.00	\$0
	Did not order this product this year	\$0.00	\$0
	Did not order this product this year	\$0.00	\$0
	Did not order this product this year	\$0.00	\$0
	33	\$110.60	\$111
	45	\$113.06	\$113
	Did not order this product this year	\$0.00	\$0
	Average Purchase Price	\$113.06	\$48.45

The average in column E more accurately displays data that contains null values

**Figure 9-39** Displaying data that contains null values

Test your report design by checking how the report script functions you choose handle different parts of your data range. Verify that your report design displays data clearly.

## Handling null values in Excel formulas

The Excel SUM and SUBTOTAL formulas interpret a null value as zero. In BIRT Spreadsheet Designer, report script functions that call Excel formulas include null values as zeros when aggregating values. For example:

```
#formula("SUBTOTAL(2, (" & cells(ProdName.CustName) & ")) ")
```

includes both zero and non-zero values when calculating a value for the ProdName.CustName section.

For more information about how to use the SUM and SUBTOTAL functions, see “Using Excel SUM and SUBTOTAL functions,” earlier in this chapter.

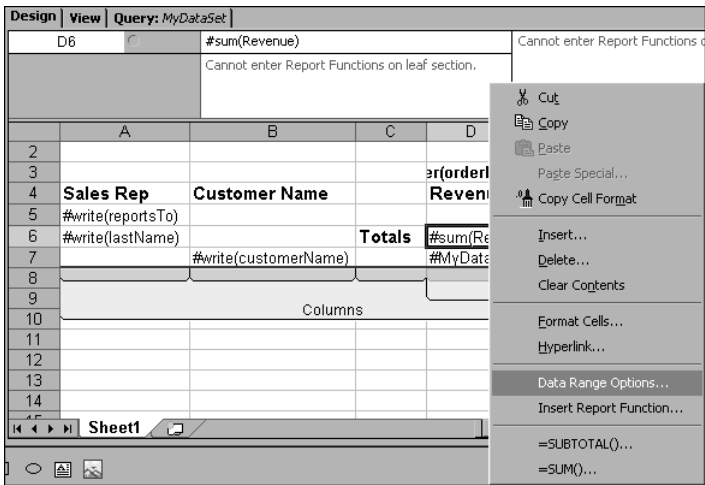
To exclude null values from calculations that call Excel formulas, use a conditionally defined name. For more information and examples, see “Using a conditionally defined name,” earlier in this chapter.

## Changing the appearance of null data values

In BIRT Spreadsheet Designer, the default appearance for null numeric values is zero. For null dates, the default appearance is 1/0/1900. You can change the behavior of BIRT Spreadsheet Designer so that null numeric and date values appear as empty cells.

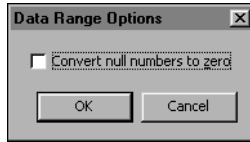
### How to change the appearance of null values

- 1 Select a data cell, then right-click and choose Data Range Options from the context menu, as shown in Figure 9-40.



**Figure 9-40** Choosing data range options

- 2 In Data Range Options, deselect Convert null numbers to zero, as shown in Figure 9-41.



**Figure 9-41** Deselecting Convert null numbers to zero  
Choose OK. Cells containing null values now appear empty.

## Working with static content

In addition to adding dynamic data fields to a report design, you can add static content. Static content typically refers to text or Excel formulas. Textual content can include report titles, and row, column, and section headings.

BIRT Spreadsheet Designer places the static content relative to other elements in the range, which means that the text could appear in a different cell or repeat in multiple cells. Dynamic content pushes the static content down in the spreadsheet, so that a value can appear in different places in a report depending on the number of values that precede it. For example, Figure 9-42 shows how text typed in cell B9 appears in cell B13 in the worksheet. In this example, BIRT Spreadsheet Designer pushed the static content down a few lines to display the dynamic content.

	A	B	C
1	<b>Rating and Order Status</b>		
2			
3			
4		#creditrank merge(c	
5			Quantity
6	#status		#sum(quantity) na
7	Totals		#"=sum(" & cells
8			
9	<i>This report brought to you by Smith</i>		

	A	B	C	D
1	<b>Rating and Order Status</b>			
2				
3				
4				A
5			Quantity	Ave Qty
6	Cancelled	23,944		\$855
7	Closed	171,813		\$365
8	In Evaluation	90,969		\$469
9	Open	98,205		\$349
10	Selected	10,540		\$527
11	Totals	395,471		\$513
12				
13	<i>This report brought to you by Smith</i>			

**Figure 9-42** Static content moved to accommodate dynamic content

## Using the setEntry function to locate static content

To keep text in a specific cell in a worksheet, use the setEntry report script function. SetEntry enables you to indicate exactly where text appears in the worksheet. If dynamic text intersects with static text you insert using setEntry, the

static text appears rather than the dynamic text. When you use `setEntry`, use the following syntax:

```
setEntry("sheet_name", row, column, "text")
```

where

- `Sheet_name` is the worksheet on which to place the supplied text.
- `Row` is the row number of the cell in which to place the supplied text. Row numbers begin at zero. To indicate row number 4, use 3.
- `Column` is the column number of the cell in which to place the supplied text. Column numbers begin at zero. To indicate column B, use 1.
- `Text` is the text string to place in the cell. If supplying a formula, preface the formula with an equal sign.

For example, the `setEntry` function places specific text in cell B9 in the `Sheet1` worksheet:

```
setEntry("Sheet1", 8, 1, "This report brought to you by Smith")
```

## Displaying static and dynamic text in the same cell

You can use quotation marks and ampersands to concatenate text with data values. When you add static text in a cell that also includes a report script function or a reference to a data field, you must enclose the text in quotation marks. For example, to create headings such as Eastern Region or Northern Japan Region, use the following expression. The expression displays the `GeoDescription` value, a space, and `Region`:

```
#GeoDescription & " Region"
```

When you display data from a text-type data set field, you can use the `extract` report script function to display just part of the text. You use wildcards to specify which part of the text field to return. For example, the following report script function returns the domain name from each value in an e-mail address field:

```
extract(Email, "*~@@")
```

## Using a report macro to simplify a custom report script function

You can use a report macro to replicate a report script function or expression you use frequently. A macro enables you to type an expression once, then use it multiple times. To use a report macro, you name and define the macro, then use the macro in the same way as you use other report script functions.

## How to add a report macro

- 1 In Data Explorer, right-click Report Macros, then choose Create Macro, as shown in Figure 9-43.



**Figure 9-43** Creating a new macro

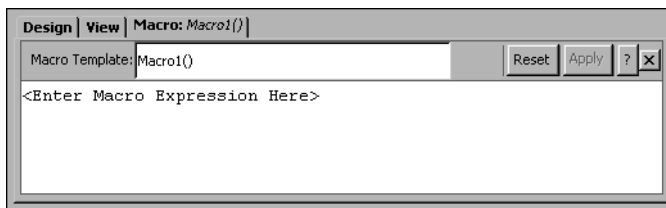
- 2 On Macro, in Macro Template, type a name for the macro, using the following syntax:

```
macro_name (macro_parameters)
```

where

- macro\_name is the name of the macro.
- macro\_parameters lists one or more parameters to use in the expression.

- 3 In the text box, provide an expression that is the formula or expression to insert or execute, as shown in Figure 9-44.



**Figure 9-44** Settings for a new macro

For example, to create a macro that creates a formula using a section name, use the following name:

```
SumSection (Section)
```

where

- SumSection is the name of the macro.
- Section is a variable that defines the section name.

Use the following expression for the macro:

```
"=sum(" & cells(Section) & ") "
```

To use the macro with a section called `geosection1`, type the following report script function:

```
#SumSection(geosection1)
```

If `geosection1` appears in cells `B5:B10` in the report, the macro creates the following expression:

```
=sum(B5:B10)
```

## Validating data entry

To limit data entry in a worksheet cell, use BIRT Spreadsheet Designer to create a validation rule. A validation rule consists of a formula against which to test the cell entry and text to display if the validation fails. The formula must return either `TRUE` or `FALSE`. If the formula returns `TRUE`, BIRT Spreadsheet Designer supplies the value. If the formula returns `FALSE`, BIRT Spreadsheet Designer does not supply the value and the validation text appears in an error message.

For example, you can limit the range of values a cell accepts by creating a rule that fails if the user types a number less than 100. If the user types 99 in that cell, BIRT Spreadsheet Designer displays the message you supply.

Microsoft Excel does not support BIRT Spreadsheet Designer validation rules. A validation rule you create using BIRT Spreadsheet Designer does not function in a spreadsheet report that you export to Excel. Use validation rules to check your own work while developing a report.

Valid examples of formulas for validation rules follow:

```
SUM(A6:A7) > A5
```

```
AND(A6 > 1, A6 < 100)
```

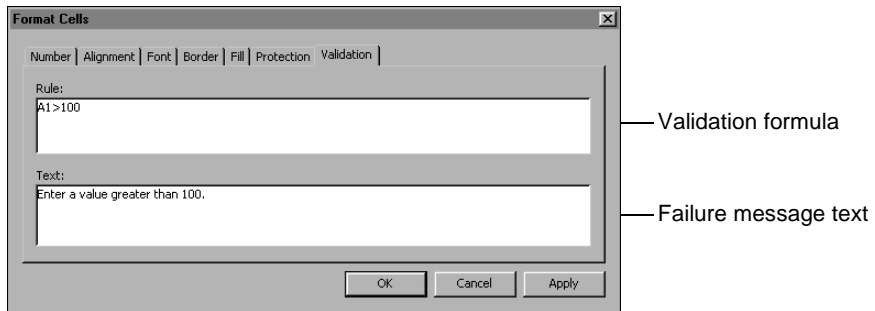
```
IF(A7 > 1, A7 < 100, A7 > 0)
```

```
OR(ISLOGICAL(A7), A7 = 1, A7 = 0)
```

BIRT Spreadsheet Designer checks the validation rule you apply to a cell only if you type data in that cell, then press Enter. Supplying data in any other way, such as selecting a value from a check box, bypasses validation.

### How to create a validation rule

- 1 Select a cell in a spreadsheet report design and choose **Format** ➤ **Cells**.
- 2 On **Format Cells**, select **Validation**.
- 3 In **Rule**, type a formula. Do not prepend the formula with `=`.
- 4 In **Text**, type text that displays if the rule formula returns `FALSE`, as shown in Figure 9-45.



**Figure 9-45**     Setting up a validation rule  
Choose OK.



# 10

## Using a chart

This chapter contains the following topics:

- About charts
- Placing a chart in a report design

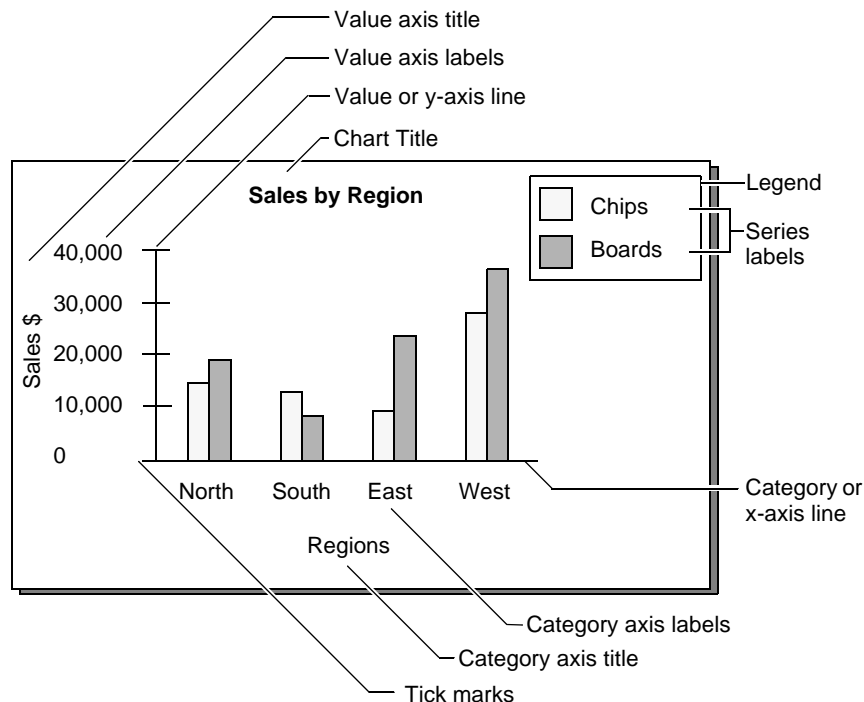
---

## About charts

Use a chart to graphically display data or relationships between data values. A chart displays graphic design elements that highlight, summarize, compare and contrast data values. As you design a chart, select elements that best emphasize relationships such as differences, comparisons or trends in a data range. A good chart adds a clear picture of complex information to a spreadsheet report.

### Understanding chart elements

Figure 10-1 shows how graphic design elements appear in a chart.



**Figure 10-1** Graphic design elements displayed in a chart

Table 10-1 lists additional chart elements that BIRT Spreadsheet Designer supports.

**Table 10-1** Additional chart elements

Chart element	Description
Data label	Identifies individual data points. Emphasizes or explains a particular piece of data on a chart.
Data point	Identifies a chart coordinate that joins an x-axis value and a y-axis value.
Drop line	Leads from a data point on a line chart to the category axis. On large or complex charts, drop lines show to which category a data point belongs.
Floor	In a three dimensional chart type, the background that provides contrast along the z axis
Grid line	Extends an axis tick horizontally or vertically.
Up or down bar	Used on a stock chart to show the stock's price at market opening and closing.
Plot area	Displays area bounded by the chart axes.
Study chart	Appears as a subchart below the main chart in the same plot area.
Tick	Marks a regular segment on an axis. On a value axis, an axis label appears on a tick. On most category axes, an axis label appears between ticks.
Trendline	Shows tendencies in data. The position and shape of the line is determined by an equation performed on the chart data. Appears as a line on a dual y-axis chart.
Walls	In a three dimensional chart type, the background that provides contrast for grid lines along the x-axis and the y-axis.

## Understanding chart types

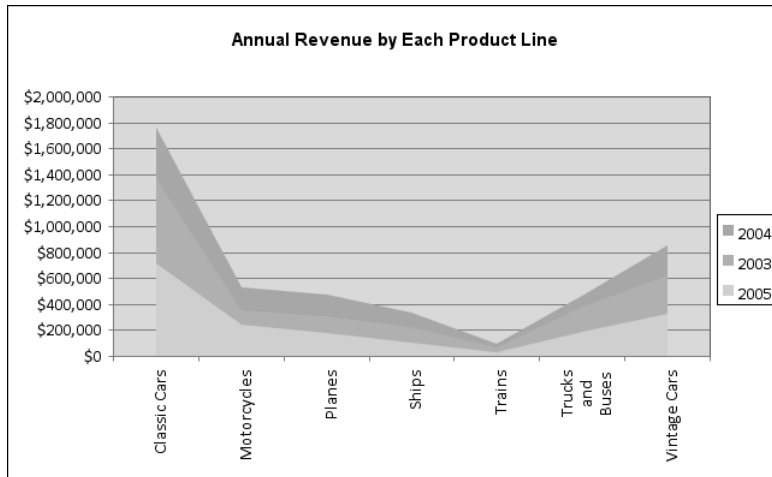
BIRT Spreadsheet Designer supports multiple chart types. Each chart type includes different elements that, when combined, represent data as a different picture. Within each chart type, BIRT Spreadsheet Designer provides templates. Each template provides a set of format selections that produces a slightly different chart image, but preserves the chart type image.

This section describes chart types that BIRT Spreadsheet Designer supports.

## About area charts

An area chart displays data values as set of points that are connected by a line. The line sets the upper boundary of an area. If you include several series on an area chart, the chart displays each series as a shaded area that overlaps another series.

The area chart shown in Figure 10-2 displays total revenue as a shaded area under a line. Each line shows the upper bound of an annual data series.



**Figure 10-2** Example area chart

The BIRT Spreadsheet Designer chart wizard provides several templates that format an area chart differently.

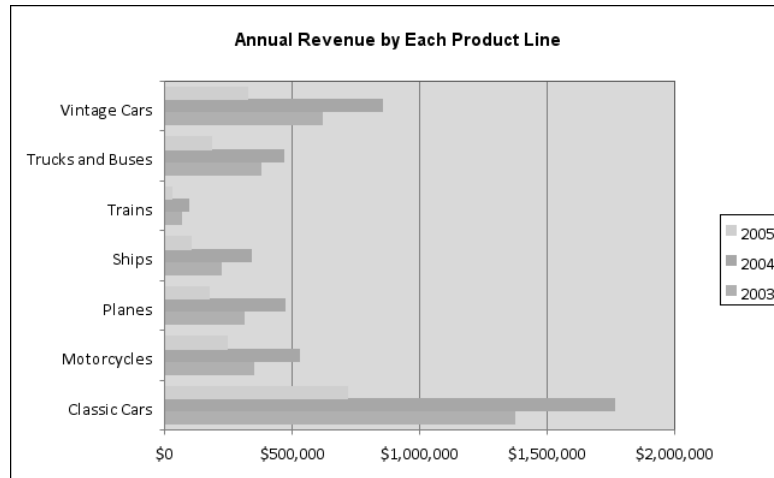
## About bar and column charts

A bar chart typically displays data values as a set of horizontal bars. A column chart displays a similar set of design elements, with the data values arranged vertically, so that data values appear as a set of vertical columns. Both bar and column charts can show how data changes with time, or differs by location. Adding a data series to a bar or column chart provides a clear comparison between data series.

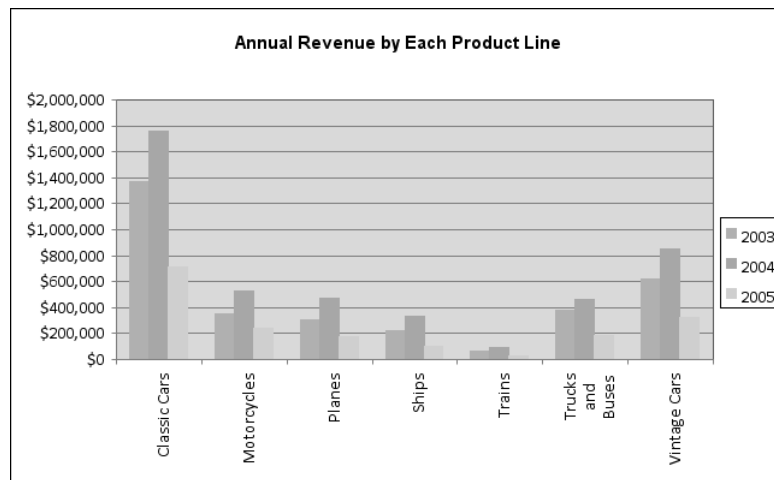
The bar chart shown in Figure 10-3 plots product lines as categories along the vertical axis. Bars represent annual revenue values plotted along the horizontal axis.

Shapes that represent annual revenue appear as vertical columns in Figure 10-4. Product lines appear as categories along the horizontal axis.

The BIRT Spreadsheet Designer chart wizard provides several templates that format bar and column charts differently.



**Figure 10-3** Example bar chart



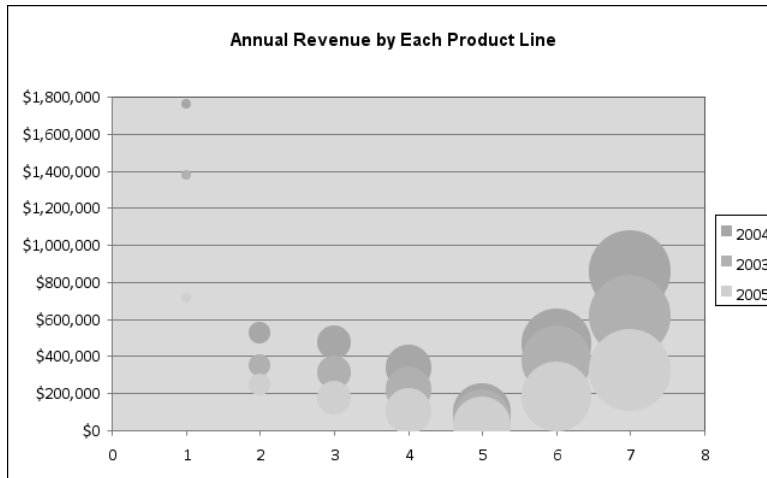
**Figure 10-4** Example column chart

## About bubble charts

A bubble chart shows three values that describe each data point. For example, if your data set includes a retail price, a wholesale price and difference between those two values for each item, you can use a bubble chart to highlight that data for each item.

The location of each bubble shape represents an annual revenue value in Figure 10-5. The size of each bubble represents the number of units sold. Product lines appear as categories along the horizontal axis. Because a bubble chart

typically sorts the x-axis in ascending order, categories appear sorted by increasing number of units sold.



**Figure 10-5** Example bubble chart

A bubble chart plots data in a layout similar to that in a scatter chart. A bubble chart includes one more value that describes each data point.

## About combination charts

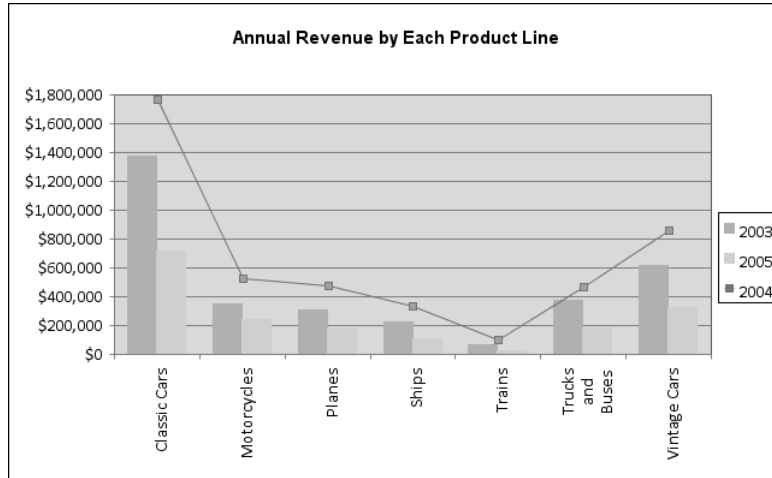
A combination chart can display each data series in a data range using up to four different chart type elements. Select a combination chart type when you want to graphically contrast different data series. Only after you insert a combination chart type can you change the chart type of each data series to area, column, line, or step.

In Figure 10-6, the data series for 2004 appears as a line, which stands out as the year of highest annual revenue.

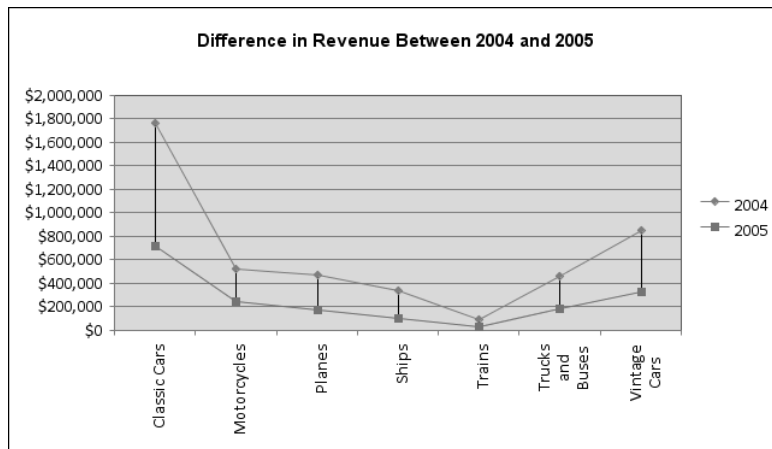
## About difference charts

A difference chart contrasts two data series by highlighting the differences between values in each series. To show differences between two data series, format the data series using the High-low lines option.

In Figure 10-7, a high-low line shows the difference between annual revenue values for each product line.



**Figure 10-6** Example combination chart

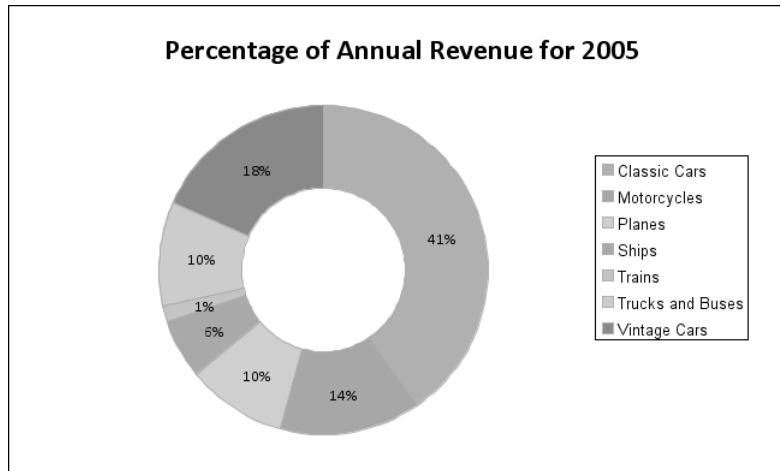


**Figure 10-7** Example difference chart

## About doughnut charts

A doughnut chart displays a data series as a circular area, or donut ring, within two, concentric circles. Data categories display as shaded sections of the donut. A doughnut chart displays additional data series that have a lower total value inside the larger circle. Use a doughnut chart to show relationships between each data series and the combined total of all data series.

The example doughnut chart shown in Figure 10-8 displays one series. To study a more complex doughnut chart example, see Figure 10-22.



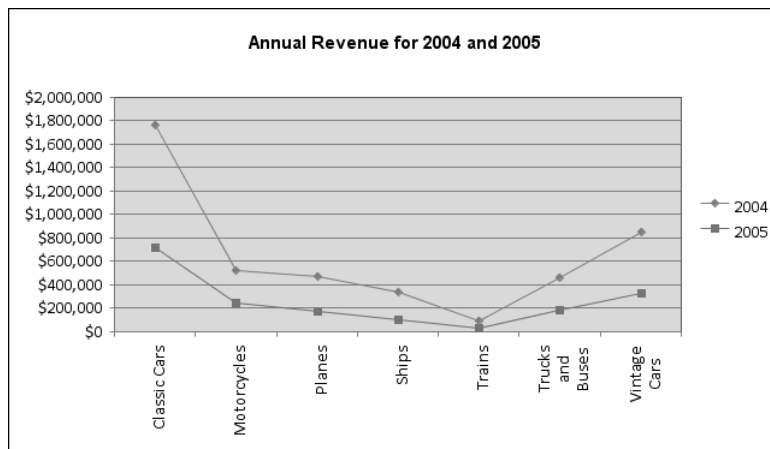
**Figure 10-8** Example donut chart

The BIRT Spreadsheet Designer chart wizard provides several templates that format a donut chart differently.

## About line charts

A line chart displays each data series as a sequence of data points connected by a line. Use a line chart to compare different data series. Use different shapes to represent data points in different data series.

In Figure 10-9 for example, a circle shape appears at each data point along the line that represents one data series. A square shape appears at each data point along the other data series.



**Figure 10-9** Example line chart

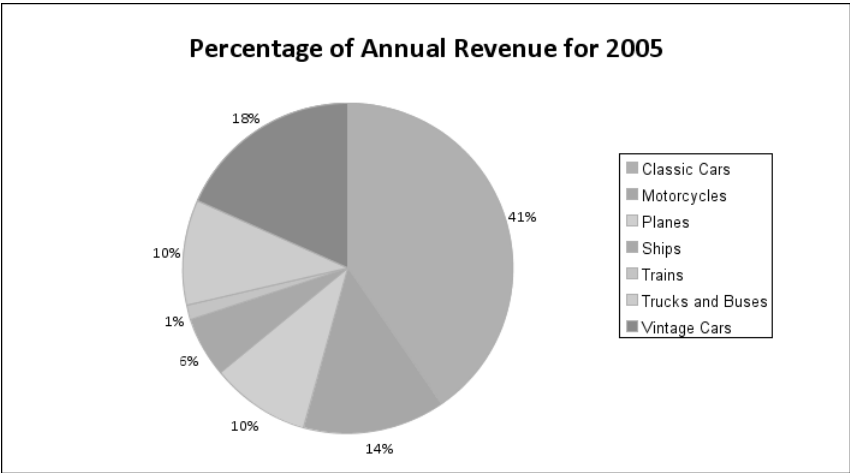


The BIRT Spreadsheet Designer chart wizard provides several templates that format a line chart differently.

**About pie charts**

A pie chart displays data series or categories as parts of an overall total. Use a pie chart to show how much each section, that looks like a slice, contributes to the whole pie. Typically, a pie chart also displays a label for each slice that shows a percentage that specifies the portion of the pie that each slice represents.

Figure 10-10 shows a typical pie chart.



**Figure 10-10** Example pie chart

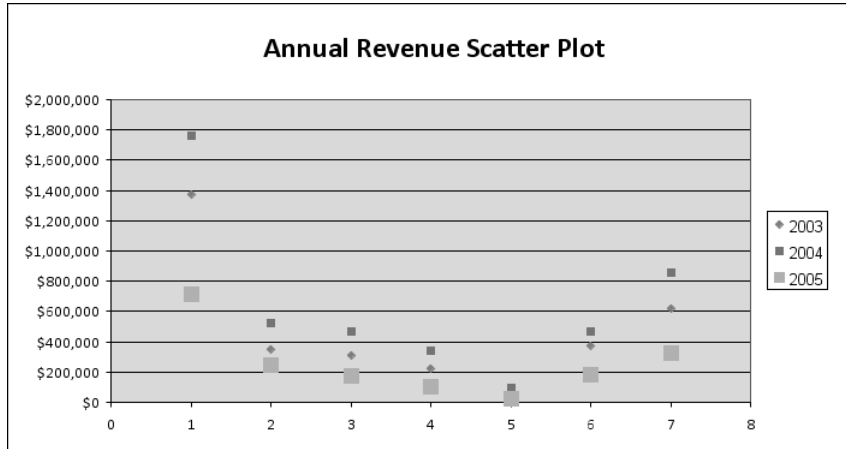
The BIRT Spreadsheet Designer chart wizard provides several templates that format a pie chart differently.

**About scatter charts**

A scatter chart shows paired data values as individual data points. Use the two axes of a scatter chart to show how the values that you plot along one axis compare to the values that you plot along the other axis. A scatter chart can show multiple series of data pairs. A typical scatter chart plots values along the x-axis and the y-axis in increasing order.

The BIRT Spreadsheet Designer chart wizard provides several templates that format a scatter chart differently.

The scatter chart example shown in Figure 10-11 plots the same data as the bubble chart example shown in Figure 10-5. The scatter chart type does not display graphically data for the number of units sold.

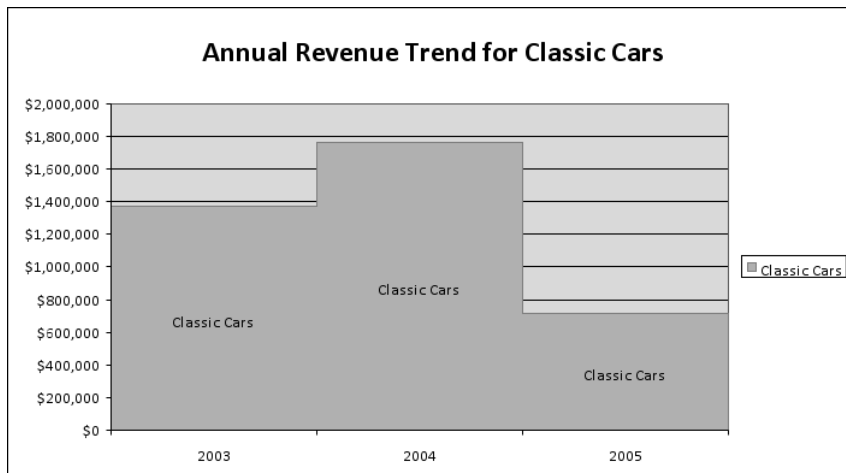


**Figure 10-11** Example scatter chart

## About step charts

A step chart displays data values along two axes, one of which typically represents time. Like in an area chart, a line connects each set of data points. Unlike an area chart, the line that bounds an area for each data series shows change as a distinct step between values instead of a point-to-point slope.

Figure 10-12 shows steps that represent annual revenue values for one data series.



**Figure 10-12** Example step chart

To use a step chart, you first insert an area chart in your spreadsheet report design, then change the chart type to step.

## About stock charts

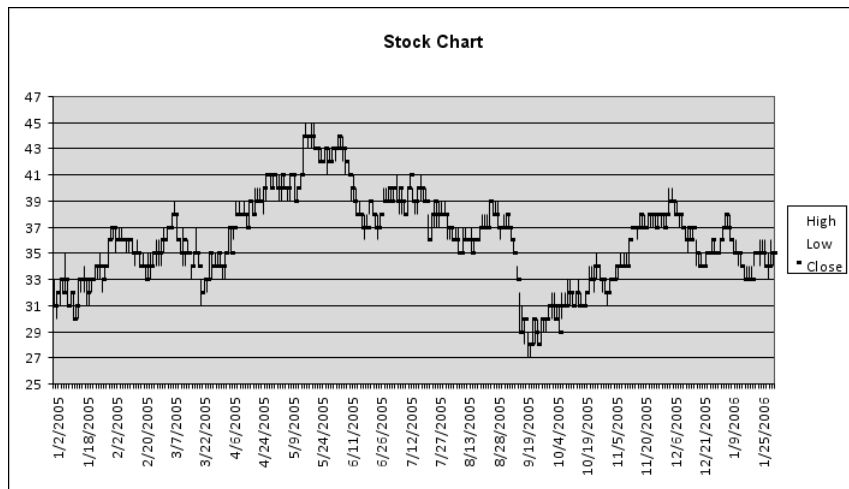
A stock chart shows the value of a stock on one axis and time on the other axis. Typically, the time categories represent days of a business week. Each data point for a stock must include a high and low value for each day, and a value at which the stock begins and a value at which the stock ends each day. These values are also called open and close values, respectively. Although you usually use a stock chart to display stock price data, you can also use a stock chart to display a scientific data set, such as one that represents daily temperature changes.

The example stock chart shown in Figure 10-13 uses a line to connect the closing price of a stock each day during a year.

The BIRT Spreadsheet Designer chart wizard provides two templates that format a stock chart differently.

## About study charts

A study chart shows values for comparison along a second y-axis, called a study axis. To create a study chart, add a second y-axis to any chart type with axes. Only a report output file generated as an Actuate workbook supports a study chart. For more information about setting report output preferences, see Chapter 17, “Personalizing BIRT Spreadsheet Designer.”



**Figure 10-13** Example stock chart

---

## Tutorial 3: Creating a chart in a spreadsheet report

Completing this tutorial demonstrates how to:

- Create a report design that includes a data range with grouped sections.
- Use Chart Wizard to create a chart object in a report design.
- Link data range sections to a chart.
- Format individual chart elements to emphasize certain aspects of the data range.

Complete all tasks in the following order:

- Create a report design with a data set.
- Create a data range.
- Create a chart object.
- Refine the chart appearance.
- Add descriptive information.
- Resize and relocate the chart.

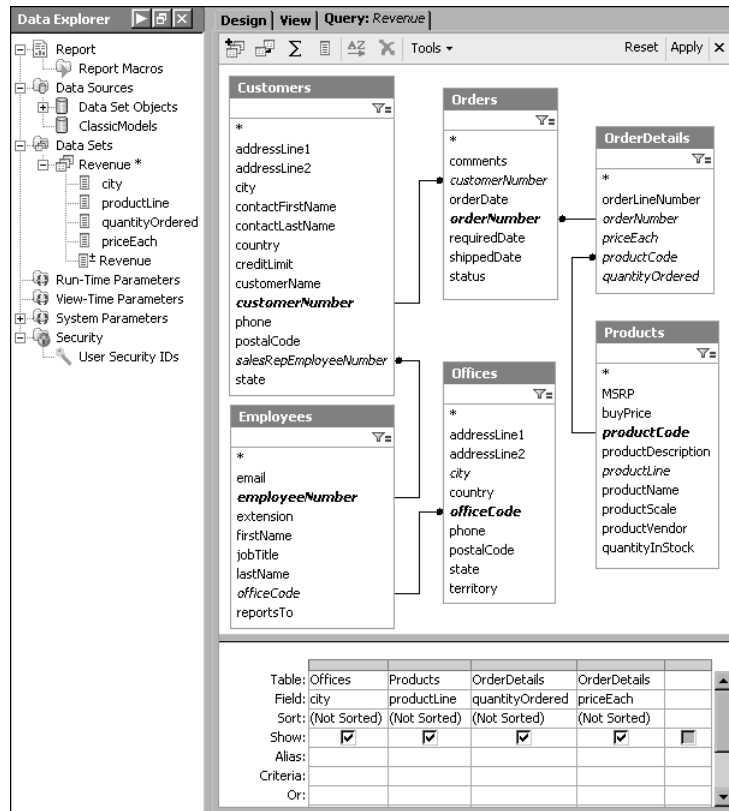
### Task 1: Create a report design

Before you create a chart, you must create a report design with a data range. In this task, you create a report design, a data source and data set using skills you learned from the tutorials in Chapter 1, “Building your first spreadsheet reports.”

- 1 Choose File→New Design, then name the report ChartDesign.sox.
- 2 Create a data source for the report design file that uses the sample database Classic Models.
- 3 Create a data set called Revenue that uses a SQL Select query and a calculated field, called Revenue, that uses the following field expression:

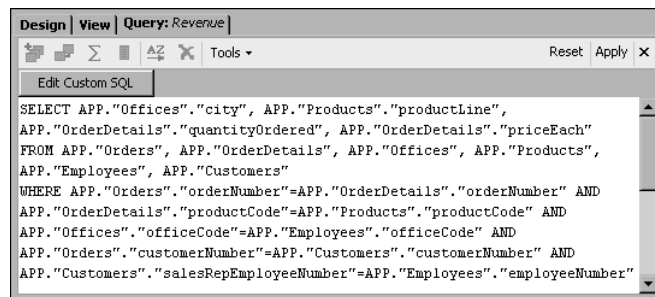
`quantityOrdered*priceEach`

In Design View, the query looks like the one shown in Figure 10-14.



**Figure 10-14** Query in Design View

In SQL View, the query looks like the one shown in Figure 10-15.



**Figure 10-15** Query in SQL View

## Task 2: Create a data range in the report design

When you design a chart that graphically describes a data range, layout the data range in a way that you can chart. Create sections and groups that organize the data using common values or fields. In this task, you create a data range, using skills you learned from the tutorials in Chapter 1, “Building your first spreadsheet reports.”

- 1 Create a data range that spans cells B2:C3.
- 2 To add sections and group the product data for product categories, perform the following actions:
  - Select the row stub in D2, right-click, then select Create Parent.
  - Type ProdRow.
  - Select the column stub in C4, right-click, then select Create Parent.
  - Type ProdCol.
  - Select ProdCol, right-click, choose Group, select productLine, then choose OK.

Your data range should look like the one shown in Figure 10-16.

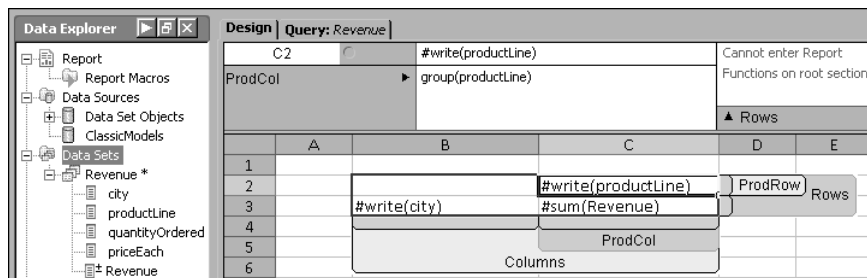


Figure 10-16 Data range

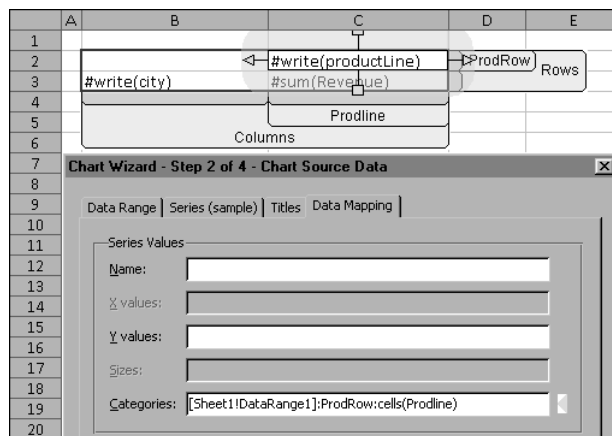
## Task 3: Create a chart object using chart wizard

In this task, you create a chart object in the report design and link the data range that you created in Task 2 with the new chart object. You also add sections and groups in the data range, using Chart Wizard.

- 1 Select cell F7, outside the data range, then choose Insert>Chart.
- 2 On Chart Wizard Step 1 of 4, select Column as the chart type, then choose Next.
- 3 On Chart Wizard Step 2 of 4, use Data Mapping to link the data range with the chart object:
  - 1 Map Categories to the data range section that contains productLine:

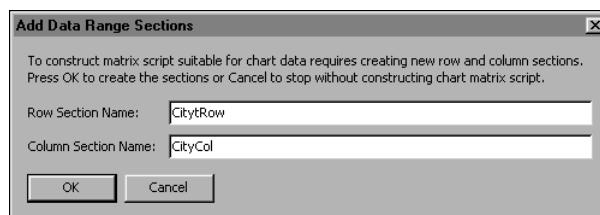
- 1 On Data Mapping, select Categories, then select cell C2 in the data range.
- 2 Click the “handles” around C2, so that the arrows point along row 2, that contains the Prodrange section.

A cell reference appears in Categories. The reference defines the range of series values for productLine, as shown in Figure 10-17.



**Figure 10-17** Categories mapped to the productLine range

- 2 In the data range, create row and column sections for the data series values:
  - 1 On Data Mapping, select Name, then select cell B3 in the data range.
  - 2 On Add Data Range Sections, create the following new data range section names, then choose OK, as shown in Figure 10-18:
    - In Row Section name, type City.
    - In Column Section name type CityCol.

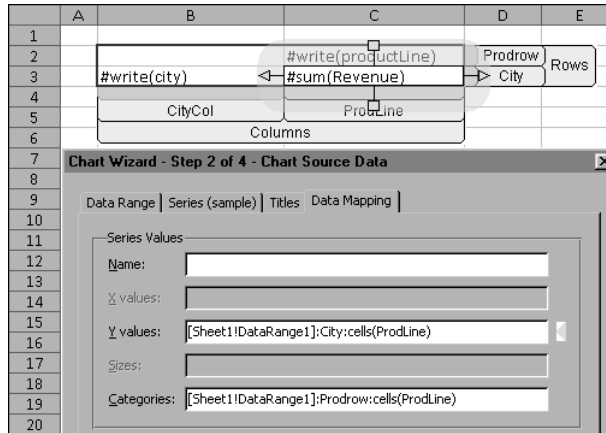


**Figure 10-18** Creating data range sections when using data mapping

- 3 Select City section, right-click, choose Group, and then select City.
- 3 Map y values to the revenue range:
  - 1 On Data Mapping, select Y values, then select cell C3 in the data range.

- 2 Click the “handles” around C3, so that the arrows point along row 3, that contains the City section.

A cell reference appears in Y values, as shown in Figure 10-19. The reference defines the range of series values for Revenue.



**Figure 10-19** Y values mapped to the revenue range

For information about using multiple series in charts, see “Showing many data series in one chart,” later in this section.

- 4 Map Name to the city range:
  - 1 On Data Mapping, select Name then select cell B3 in the data range.
  - 2 Click the “handles” around B3, so that the arrows point along row 3, that contains the City section.

A cell reference appears in Name. The reference defines the range of series values for Name.

- 4 On Chart Wizard Step 2 of 4, choose Next.
- 5 On Chart Wizard Step 3 of 4, keep default selections, then choose Next.
- 6 On Chart Wizard Step 4 of 4, select object in sheet, then choose Finish.

On Design, a chart outline that is based on sample data appears on the same sheet as the data range.

- 7 Choose Run.

On View, a chart that displays data from the data range appears.



## Task 4: Refine the chart appearance

In this task, you format the chart using graphic design elements that illustrate certain aspects of the data range. The chart elements you add or change should clearly focus the picture of the data that the chart provides.

**1** Add sample data to the chart design:

- 1 On Design, right-click the chart, then choose Source Data.
- 2 On Source Data, on Series (Sample), select Sample Data.
- 3 On Sample Data, type numbers that represent the sample size of the charted data, then choose OK.

On Source Data, a chart image appears under Sample. The image under Sample shows how each change affects the chart.

**2** Change the appearance of the data series:

- 1 Right-click the data series, then choose Format Data Series.
- 2 On Shape, select the cone shape icon.
- 3 Right-click the back wall in the plot area, choose Format Walls, then select None.

**3** Change the appearance of the value axis:

- 1 Right-click the vertical chart axis, then choose Format Axis.
- 2 On Format Axis, in Number, select a currency number format.

**4** Change the Chart Type:

- 1 Right-click the chart area, then choose Chart Type.
- 2 Select 2D, then choose OK.

**5** Choose Run to apply changes that appear in the design to the report view.

## Task 5: Add descriptive information to the chart

In this task, you add two elements to the chart. The title element uses text to describe chart data. The trendline element uses a line to show a general trend in the data.

**1** Add titles to the chart:

- 1 Right-click the chart area, choose Source Data, then select Titles.
- 2 In Chart, type a name that describes the charted data relationship, for example, Revenue per Product Line by Office Location:
  - In x-axis, type a title for the category axis.

- In y-axis, type a title for the value axis.
- 2 Add a trendline to the chart:
  - 1 Right-click a data series, choose Format Data Series, then select Add Trendline.
  - 2 On Options, select Linear, then choose OK.

## Task 6: Relocate and resize the chart

In this task, you move the chart, then change the chart size.

- 1 On Design, right-click the chart area, then drag the chart and drop it under the report range.
- 2 Select the chart area, then select a corner handle. Drag the handle to change the size of the chart object.
- 3 Choose Run to apply the changes that appear in the design to the report view.

On View, preview how the changes you make to the report design affect the report output.

---

## Placing a chart in a report design

Before you place a chart in a spreadsheet report design, consider how the location of a chart object in the report design affects the report output. This section shows how moving a chart to a new location in the design affects the report output. As the examples show, you should also consider chart type and data before you choose a location in a report design for a chart.

This section contrasts a report design that uses a single chart type with one that uses two. The last example shows how a multi-sheet report design can provide different charts that represent a single data range to different readers. Compare the examples to better understand how you can represent a data range using the chart design elements available in BIRT Spreadsheet Designer.

## Comparing and contrasting data using chart elements

To compare and contrast differences between data series and data values in a spreadsheet report, place charts or chart elements side-by-side. A chart provides elements that enhance differences between graphical images. Placing multiple, similar charts or elements in a report allows a reader to compare and contrast the images.

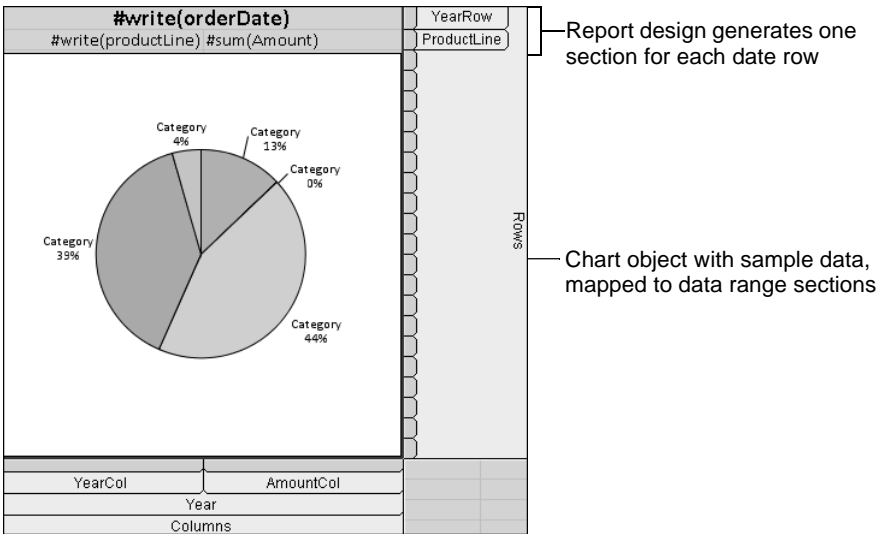
## Displaying one chart type many times

To compare data series that share common elements, such as sales values collected during regular time periods, use the same chart to represent the data for each time period. Comparing many charts of the same type, each of which shows a different data series, provides contrast between the series. To display one chart type many times, place a chart object within a data range section that bursts, or displays one report section for each row element in that data range section, when the report runs.

For example, suppose that you need to compare annual revenue from each of several product lines generated by sales that occurred during several different years. Use a pie chart to show revenue from each product line as a piece of the annual pie. Place the pie chart in a report section that bursts into several sections, one for each year.

Figure 10-20 shows a report design that:

- Bursts sections based on orderDate
- Includes a pie chart in each report section
- Shows productLine revenue as a percentage of revenue
- Displays a chart title linked to orderDate

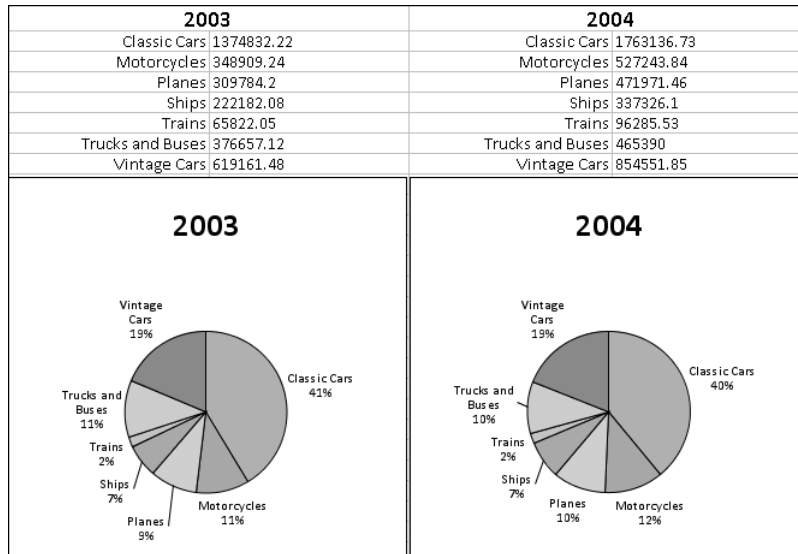


**Figure 10-20** Pie chart object inside a data range

Figure 10-21 shows two of the report sections generated by the preceding report design.

Notice that the pie chart design shows the percentage that each product line contributes to annual revenue. The side-by-side display allows the reader to

compare whether each percentage changes each year or not. The report sections list the different revenue values generated by each product line, each year. Repeating the pie chart allows comparison and contrast of the revenue differences as percentages each year.



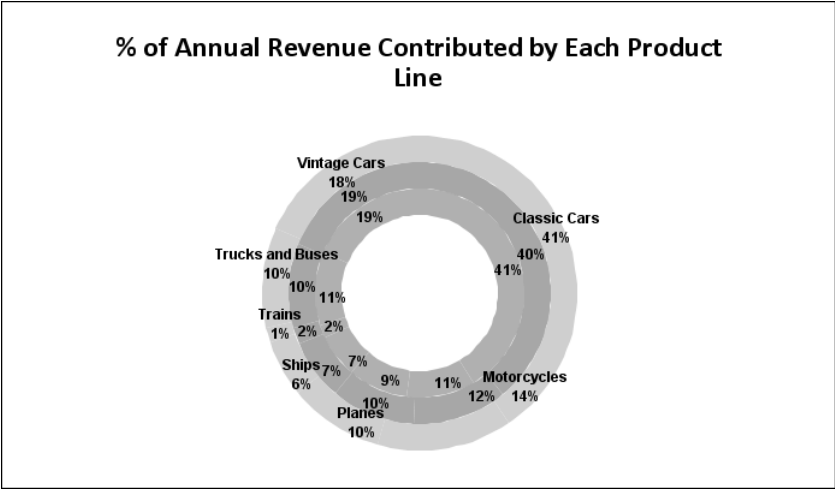
**Figure 10-21** Report view with pie chart in each section

## Showing many data series in one chart

To compare how changing the chart type and design location affects report output, consider the following actions, results, and design decisions in the following order:

- Starting with the design shown in the previous example, drag the chart object and drop it completely outside the data range, then choose Run. In View, a chart that displays all data series appears as one graphic.
- The pie chart type does not display multiple data series clearly. In Design, change the chart type to Doughnut and then choose Run. The chart now displays each data series as a ring that represents annual revenue. Each ring shows sections that represent product line categories.
- The doughnut chart does not show category values when you select percentage. Add a summary section to the report design that calculates the total annual revenue value for all product lines.
- Add a legend to the chart that shows data series labels. This element guides the reader to the ring that represents each data series.

Figure 10-22 displays a doughnut chart generated from a chart object located outside the data range. Like the pie chart type, the doughnut chart type shows the percentage that each category contributes to a data series. Each category appears as a segment of a doughnut. Each doughnut shows a data series of total annual revenue. A report design arranged in this way produces one, summary chart. This chart provides a side-by-side comparison of annual revenue, as percentages contributed by each product line.



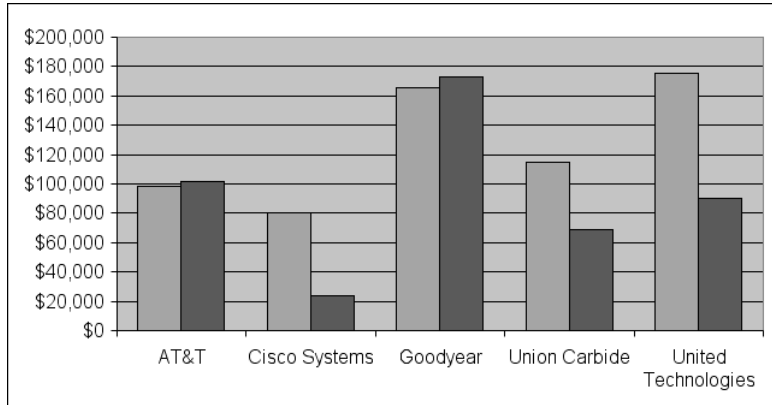
**Figure 10-22** Doughnut chart that shows several data series

To display multiple series in some types of charts, such as bar charts and column charts, you must use the following syntax to type the series reference manually in Data Mapping:

```
[Worksheet!Range] : ( {Section:cells(Group)} +  
                    {Section:cells(Group)} )
```

For example, the following y-series value specification in Data Mapping selects the values for CurrentValuesSection and OriginalValuesSection in DataRange1 on the ValuesByClient worksheet to create the column chart in Figure 10-23:

```
[ValuesByClient!DataRange1] :  
    ( {CurrentValuesSection:cells(SecurityGroup)} +  
      {OriginalValuesSection:cells(SecurityGroup)} )
```



**Figure 10-23** Using multiple series in a column chart

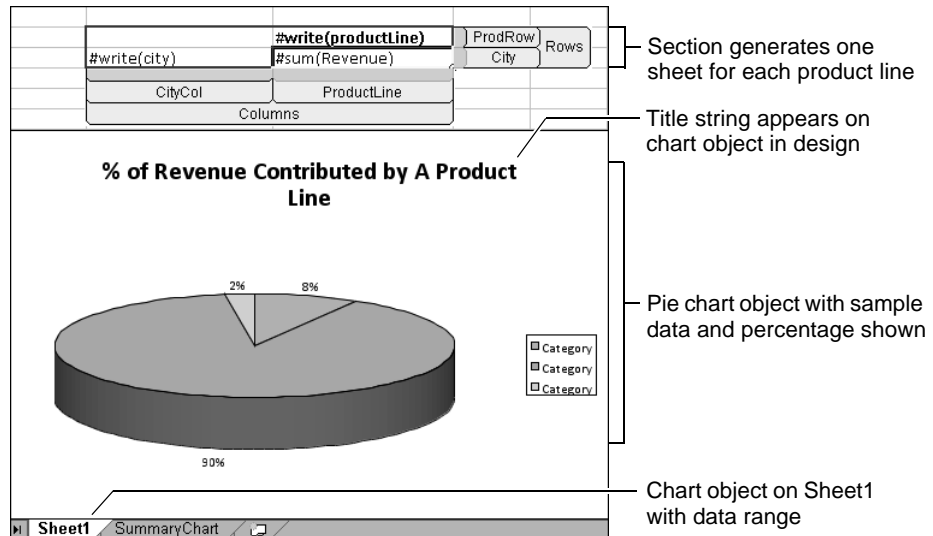
## Separating a chart from a data range

BIRT Spreadsheet Designer supports linking a data range on one worksheet with a chart on a different worksheet. Placing a data range and a chart that you link to that data range on different worksheets allows you to establish separate access permissions for the data and for the chart. The following sections describe an example report design that includes one data range and two chart types in a design that uses two worksheets.

### Using more than one worksheet

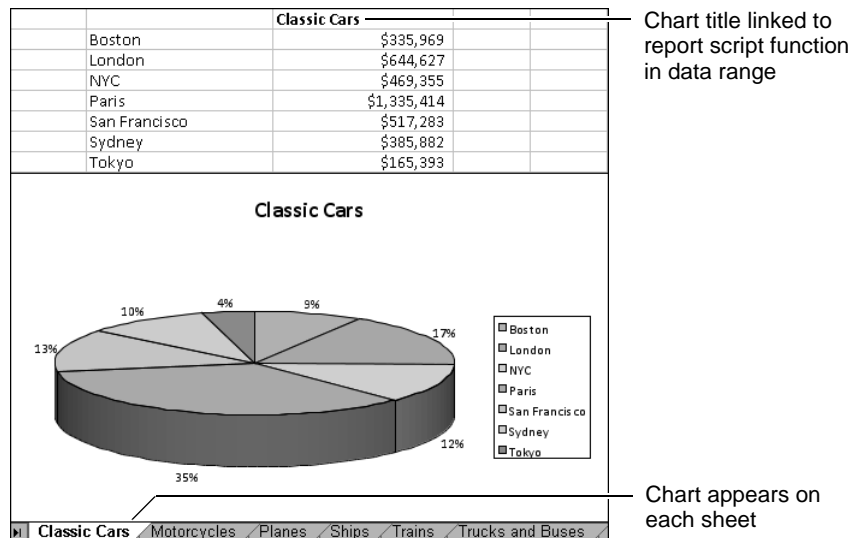
Consider another report design that uses the same data source, data set and query as the previous examples. In this example, Sheet1 contains a modified data range and a pie chart. Another worksheet, called SummaryChart, contains a column chart that links to the data range on Sheet1.

The data range for this example includes sections that burst onto separate report sheets; one for each productLine row. Like the previous pie chart example, this design produces multiple versions of a pie chart that shows percentages of annual sales revenue. Unlike the previous example, this report shows one pie for each product line. Each slice represents the part of product line revenue that each sales office generates. The data range and pie chart object both appear on Sheet1 of the report design, as shown in Figure 10-24.



**Figure 10-24** Sheet 1 of a two-sheet report design

Sheet1 generates one page per product line. Figure 10-25 shows the page that displays data for the Classic Cars product line.



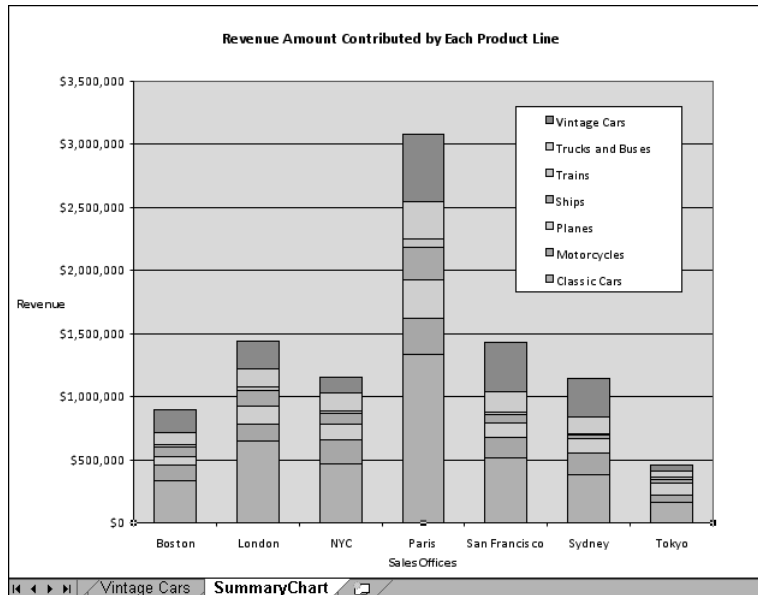
**Figure 10-25** One sheet of the multi-sheet report

## Using more than one chart type

One data range supports more than one chart. In a report design, you select the chart type that best displays one aspect or dimension of your data set. In the same

design, you can use another chart type that best displays a different aspect of the data; optionally, to a different audience.

Figure 10-26 shows a column chart that plots the data range located on Sheet1 of the report design discussed in the previous example. The column chart appears on another worksheet called SummaryChart.



**Figure 10-26** Summary chart of product line revenue

On SummaryChart, one two-dimensional column chart plots revenue values along the y-axis. Columns that represent sales offices appear along the category axis. Shaded sections depict data series for all product lines.

This design scales to as many offices and product lines as needed.



# 11

## Charting a data range

This chapter contains the following topics:

- Designing a chart
- Linking a chart to a data range
- Refining a chart

---

## Designing a chart

When you design a chart, consider:

- Data range layout
  - Row and column arrangement
  - Grouping
  - Sorting
- What aspects of the data you want the chart to display
- What chart type best fits the data range
- What design elements the chart uses to show information

Before you chart a data range, consider the layout and content of the data range that your chart will display, and any data relationships within the data that you want to emphasize. If the data under consideration suggests an available chart type, then begin by selecting a chart type that follows that suggestion.

To chart a data range, use the following process as a guideline:

- Create a basic chart object.
- Link a data range to the chart object.
- Refine the chart object.

Use the BIRT Spreadsheet Designer chart wizard to create a chart object and link the chart with source data.

Use formatting options to add and modify the chart object.

This chapter describes how to use chart wizard and the many formatting options to refine a chart object for deployment in a spreadsheet report design.

## Understanding the axes of a chart

Many charts display data along axes. An axis provides a direction, or dimension reference. A chart with axes may contain one, two or more axes.

### About the axes

Most charts with axes display text or date-and-time data series as categories along the x-axis. Most charts display numeric data series as values along the y-axis. Typically, the x-axis appears horizontally and the y-axis appears vertically. A category axis plots data at regular intervals. A value axis plots data at points that represent the data values. A data point plotted along a value axis appears at a

position that relates to a mark called a tick. Each tick marks a regular value interval along an axis.

### Defining the axes

While you map a data range to a chart, you define how the chart axes plot data values. You assign, or map a data range section to each axis of the chart.

For example, to show revenue amounts generated by sales offices in several cites, and categorize the amount by product line, you map a data range to elements of a column chart, as shown in Figure 11-1.

The following example shows:

- Revenue values in the data range section City mapped to Y values for a chart
- Product line names in a data range section called Prodrow mapped to Categories for a chart

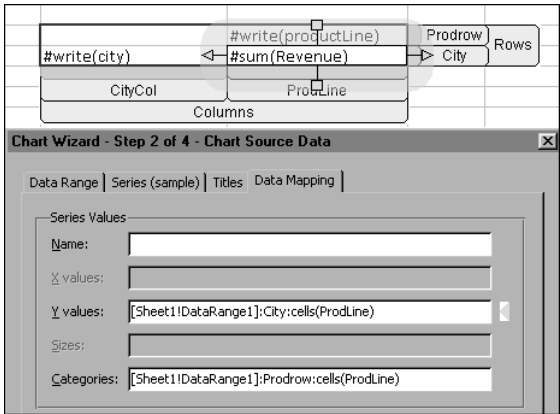


Figure 11-1 Defining axes for a column chart

### Plotting different chart types

Data Mapping displays options appropriate for a selected chart type. For example, if you select Column as the chart type, you map data series values as Y-values and Categories. If you select XY (Scatter) as the chart type, you map data series values as X-values and Y-values. Table 11-1 describes how the Series Value fields appear on Data Mapping and how each plots data for a selected chart type.

**Table 11-1** Differences in x- and y-axis plotting options for chart types

Chart type	Series values	Description
Area, column, cone, line, pyramid	Categories	Arranges data as shapes grouped on x-axis.
	Y values	Plots values on one or more y-axis and sets shape size.
Bar	Categories	Arranges data as shapes grouped on vertical axis.
	Y values	Plots values on horizontal axis and sets size of bar shape.
Bubble	X values	Plots values as points on x-axis.
	Y values	Plots value points on y-axis.
	Sizes	Defines size of a bubble shape located at each point defined by the (x,y) pair.
Combination	Categories	Arranges data as shapes unique for each series grouped on x-axis.
	Y values	Plots values on one or more y-axis and sets shape size.
Doughnut	Categories	Plots values as sections of each series ring.
	Y values	Arranges data series as concentric rings.
Pie	Categories	Arranges data as sections of a circle.
	Y values	Defines the size of each section .
XY (Scatter)	X values	Plots value points on x-axis.
	Y values	Plots value points on y-axis.
Step	Categories	Arranges data as platform steps grouped on x-axis.
	Y values	Plots values on one or more y-axis that sets step size.
Stock	Categories	Plots values along x-axis.
	Y values	Defines four levels of data: high, low, open and close. Data values determine candlestick shape size.

## Creating a chart object



To create a chart object, select a cell in a spreadsheet report, then choose Insert→Chart. Alternatively, select Insert Chart on the main toolbar.

### About Chart Wizard

The chart wizard provides default selections for chart type, format template, and sample data. Axes, major grid lines and series legend options appear visible by default. You can accept the default options in each Chart Wizard step and then choose Finish.

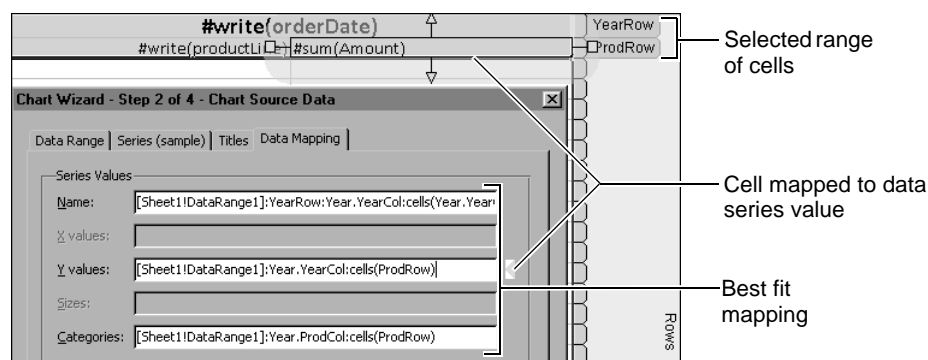
In this case, the chart wizard creates a chart object with basic column format on the same worksheet as the data range. Alternatively, you can select from a wide range of options at each step in the chart wizard.

### Selecting options for a new chart

In Step 1, Chart Wizard—Standard displays supported chart types and templates. You must select a chart type to initialize the chart object. Column is the default chart type. Optionally, select a different template icon from the group to the right of the chart type list. Choose Next to see Chart Source Data options.

In Step 2, Data Range displays a reference for the cell or range you selected for the chart. On Series, you can change the default sample data options, such as the number of sample data series. On Titles, you can add title text to the chart. Data Mapping provides a mapping tool that you can use to link a data range to the chart object.

If you select a range of cells that contains data series values for a new chart, a best fit of cell references appears in Data Mapping, as shown in Figure 11-2.



**Figure 11-2** Best fit cell references selected for Data Mapping

Choose Next to see Chart Options, or Back to see Chart Type options again.

In Step 3, Chart Options lets you choose to display a complete range of options for chart axes, legends and labels. Choose Next to see Chart Location, or Back to see Source Data Options again.

In Step 4, you may choose to locate the chart on a new worksheet. Select Back to revisit Chart Options. To build the chart object, choose Finish.

To follow a procedural example that demonstrates how to use the chart wizard, see Tutorial 3: “Creating a chart in a spreadsheet report,” in Chapter 10, “Using a chart.”

---

## Linking a chart to a data range

BIRT Spreadsheet Designer supports linking a chart to a data range. It is best practice to create a data source, data set, and data range using Data Explorer as described in Tutorial 2: “Creating a report from scratch,” in Chapter 1, “Building your first spreadsheet reports.” Use Data Mapping to link series values in a chart object to a cell in a data range.

Cells within a data range that you link to a chart may contain expressions that include report script functions and data range references. A section in the data range must contain these cells. That section may include grouped and sorted data. If you select a cell for which no section exists, Data Mapping helps you create a new section before you complete linking the chart to the data range.

## Working with Data Mapping

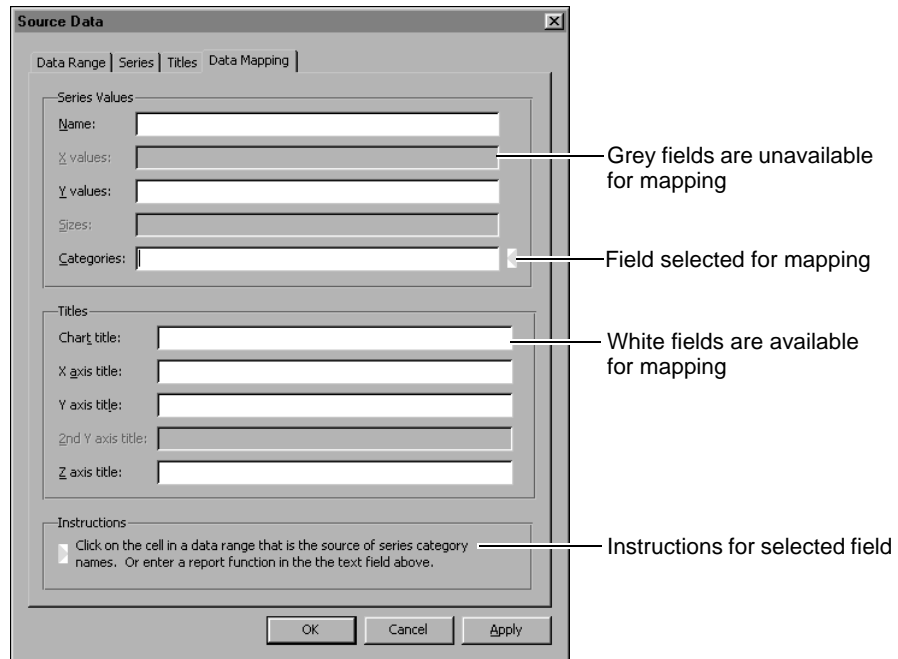
You open Data Mapping in the following ways:

- To create a link between a new chart and a data range, on Chart Wizard—Chart Source Data, select Data Mapping.
- To modify a link between an existing chart and a data range, right-click the chart area, choose Source Data, then select Data Mapping.

## Mapping a chart element to a data range

Data Mapping allows you to select a chart element and select a data range cell. When you select the cell, Data Mapping creates a cell reference expression and links it to the selected chart element. The reference expression provides data series values that the chart object plots along that element.

Values from a cell that you map to a field under Series Values appear in the chart as X values or Y values. Values that you map to a field under Titles appear in the chart as the title of the chart or of an axis. The selected chart type determines which fields accept values on Data Mapping. For example, Figure 11-3 displays elements available for mapping for a column chart. These elements include Y- and category values.

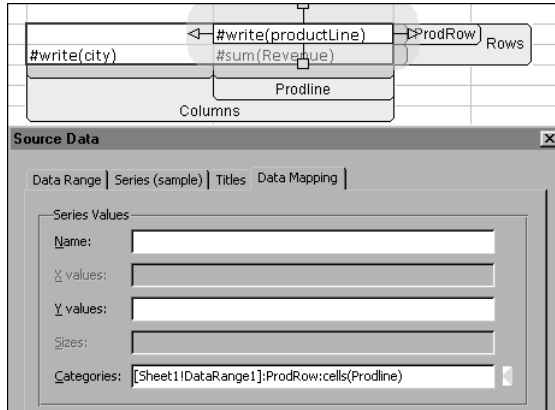


**Figure 11-3** Selecting a chart element to link with a data range

#### How to map a chart element to a cell in a data range section

- 1 In Data Mapping, under Series Values, select a field that represents a chart element. An arrow that highlights the selected field appears.
- 2 Select a cell in a data range that contains a data series value expression.  
A data range reference for the cell you selected appears in the field under Series Values. A box with handles appears around the cell in the data range.
- 3 Select the handles that surround the box, so that the arrows point along the section row.  
In the series value field, the range reference for the selected cell changes to reflect the row orientation of the data series, as shown in Figure 11-5.
- 4 On Source Data, choose Apply.

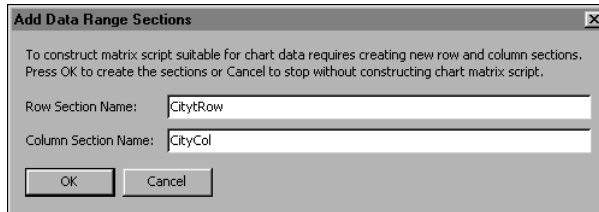
For a more complete example that demonstrates mapping a chart element to a data range, see Tutorial 3: “Creating a chart in a spreadsheet report,” in Chapter 10, “Using a chart.”



**Figure 11-4** Chart element Categories mapped to section ProdRow

## Creating a new section for data mapping

Both a row and column section must contain a cell in a data range that you map to a chart. If no, or only a row or a column section contains a data range cell that you select for mapping to a chart element, Add Data Range Sections appears, as shown in Figure 11-5.



**Figure 11-5** Creating data range sections

### How to create a section around a selected cell

On Add Data Range Sections, type text that describes a section name in either or both of the following fields. Then, choose OK.

- In Row Section Name, type text that describes a row section.
- In Column Section Name, type text that describes a column section.

## Working with Source Data

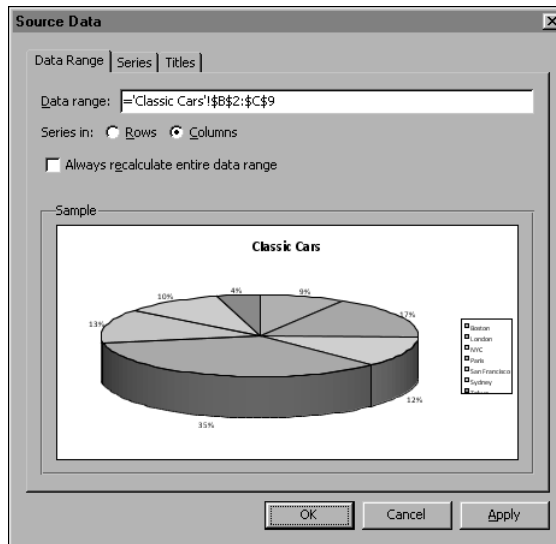
Modify the data range, series, titles and data mapping for an existing chart by selecting the same options that appear in the chart wizard when you create a new chart object. To see these options, right-click the chart area in an existing chart object, then choose Source Data from the context menu.



Use Source Data to adjust the data range references, modify the series layout, add or change titles and create data mapping expressions for the chart to which a data range links.

On Design, use Data Mapping to modify how series values map to chart elements before previewing chart appearance. After you preview a report, on View, Source Data does not display Data Mapping. Any values that you set on Data Mapping override values that you set on the Series or Titles tab.

Source Data—Sample displays an image that reflects how your last selections affect chart appearance. Figure 11-6 shows a sample image and data range options for a pie chart.



**Figure 11-6** Source Data—Data Range

Several examples in this section that show Source Data omit Sample, for brevity.

## Updating the data range for a chart

Data Range displays the cell reference for the range to which a chart links, indicates the data range layout, and specifies when to update the chart data.

## Selecting a different data range

In Data range, type or retype a range reference. For example, to change the data range that a chart displays from a range on Sheet1 to a range of the same size on Sheet 2, change the reference =Sheet1!\$B2:\$E\$7 to =Sheet2!\$B2:\$E\$7.

## Selecting series layout

In Series in, select the button that describes whether data series span rows or columns of the data range.

## Refreshing a chart

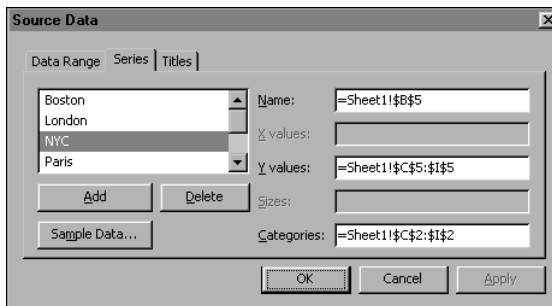
A refreshed chart displays a current view of the data range to which it is linked. To manually refresh a chart, you save the chart when you save or recalculate the worksheet that contains a linked data range. To set the option that refreshes a chart when the data range recalculates, select Always recalculate entire data range. This selection disables Data Mapping.

## Working with Series

Source Data—Series displays references for data series values linked to the selected chart object. For example, in Design, when you select a chart object that is linked to sample data, Series displays references to that sample data. In View, when you select a chart object that is linked to a data range, Series displays references to the data range. Data Range cannot represent individual series that do not reference a contiguous rectangle in a data range.

### How to add a series to an existing chart

- 1 On Series, select Add.
- 2 Optionally, in Name, type a name for the added series.
- 3 In the fields for each data series at the right, type sample data series values or enter reference expressions from a data range, as shown in Figure 11-7, then choose OK.



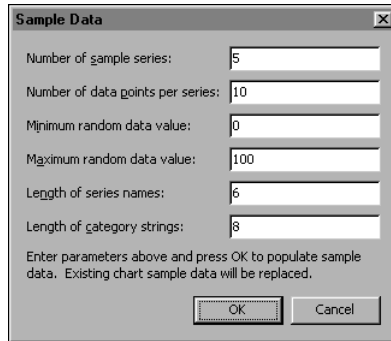
**Figure 11-7** Source Data—Series

### How to add sample data to an existing chart

- 1 Select a chart object, right-click and then choose Source Data.
- 2 On Source Data—Series, select Sample Data.

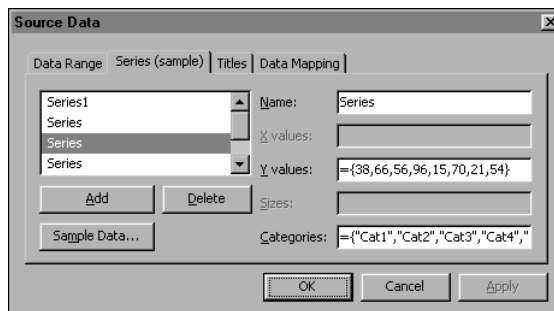
- 3 On Sample Data, edit each parameter value to display a sample image that represents how your chart object looks, then choose OK.

For example, in Number of sample series, type the number of series in an actual data range, as shown in Figure 11-8.

A dialog box titled "Sample Data" with a close button (X) in the top right corner. It contains several input fields with numerical values: "Number of sample series:" (5), "Number of data points per series:" (10), "Minimum random data value:" (0), "Maximum random data value:" (100), "Length of series names:" (6), and "Length of category strings:" (8). Below these fields is a text instruction: "Enter parameters above and press OK to populate sample data. Existing chart sample data will be replaced." At the bottom are "OK" and "Cancel" buttons.

**Figure 11-8** Entering sample data

Source Data—Series (Sample), as shown in Figure 11-9, displays an image that has the number of series that you typed.

A dialog box titled "Source Data" with tabs for "Data Range", "Series (sample)", "Titles", and "Data Mapping". The "Series (sample)" tab is active. It shows a list of series names on the left: "Series1", "Series", "Series", and "Series". The second "Series" is selected. To the right, there are fields for "Name:" (Series), "X values:" (empty), "Y values:" (={38,66,56,96,15,70,21,54}), "Sizes:" (empty), and "Categories:" (={"Cat1","Cat2","Cat3","Cat4","}). There are "Add", "Delete", and "Sample Data..." buttons on the left, and "OK", "Cancel", and "Apply" buttons at the bottom.

**Figure 11-9** Viewing sample data on Source Data—Series (Sample)

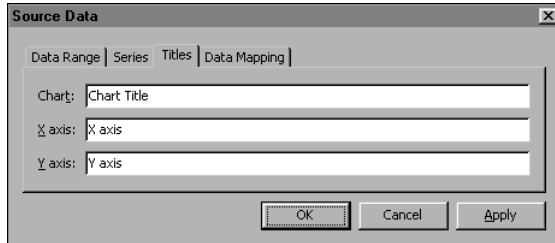
#### How to edit series values in sample data

- 1 In Series (Sample), select a series name in the left upper pane.
- 2 In Name, type text to change the selected Series name.
- 3 In data series fields, type any changes to the sample values, then choose Apply.

Source Data—Sample displays an image of applied changes.

## Working with Titles

To add text that does not vary to a chart title or an axis title, use Source Data—Titles, as shown in Figure 11-10. The chart design in the report executable file displays the same text strings that you type in chart and axis fields on Titles.



**Figure 11-10** Source Data—Titles

### How to add a text title to a chart

- 1 On Titles, in Chart, type text that describes the entire chart.
- 2 In X-axis, type text that describes the category, or x-axis.
- 3 In Y-axis, type text that describes the value, or y-axis. Then, choose OK.

---

## Refining a chart

After you create a basic chart object, you refine the chart appearance by formatting, mapping or modifying the elements of the chart.

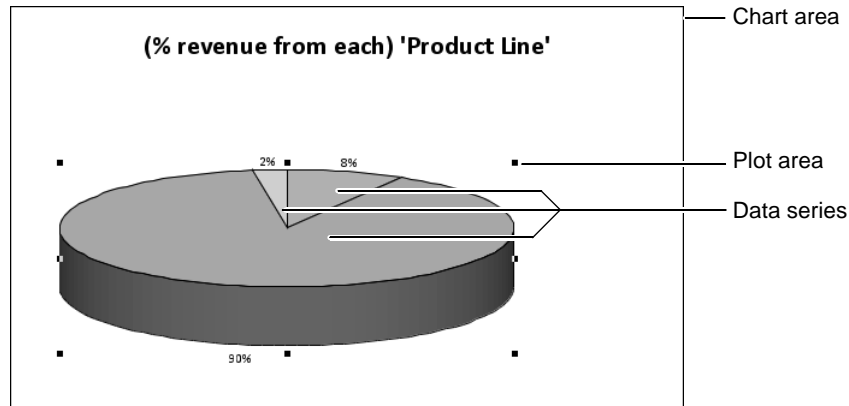
This section describes how to select areas in a chart and change the overall appearance of a chart. Many subsections explain how to modify the layout and formatting of data series, axes and data label elements.

## Working with an element in a chart area

You work separately in each chart area and with each chart element. You use a different context menu to modify each area and element of a chart. Options available in a chart area or for a chart element differ for different chart types.

### Selecting a chart area

You can select formatting options that apply to each area of a chart, shown in Figure 11-11. For example, to change the fill pattern in a pie chart, right-click the plot area, then choose Format Plot Area from the context menu.



**Figure 11-11** Selecting an area in a chart

## Selecting a chart element

You can select formatting options within each element in a chart. For example, to make changes that apply to all points in a data series, right-click a point along the data series, then choose **Format Data Series**. For a list of chart elements, see Figure 10-1 and Table 10-1 in Chapter 10, “Using a chart.”

## Showing tips

Typically, text that describes a chart area or element appears as the cursor moves over that area or element of the chart. If you do not see such descriptions, choose **Tools** ➤ **Designer Preferences**, then select **Chart**. On **Chart**, select **Show tips**.

## Formatting the chart area

When you select the chart area, you can change the chart type, apply a different formatting template, and change the patterns, line styles and font characteristics throughout the chart. In the chart area, you also control how the chart displays hidden, or non-numeric data.

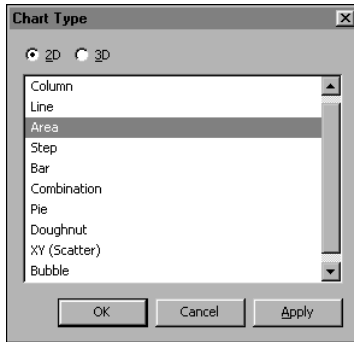
## Changing a chart type

To change the arrangement of graphic elements in an existing chart, change the chart type.

### How to change a chart type

- 1 Right-click a chart area, then choose **Chart Type**.
- 2 On **Chart Type**, as shown in Figure 11-12, select the following options, then choose **OK**:

- Number of dimensions
- A different chart type



**Figure 11-12** Selecting a different chart type

## Changing a color or pattern in the chart area

Typically, the chart area appears white. To contrast one chart area from another, change the color or pattern that fills a chart area. For example, add color or a pattern to the chart area to make the chart area distinct from the plot area.

### How to change a color or pattern in the chart area

- 1 Right-click the chart area, choose Format Chart Area, then select Patterns.
- 2 On Patterns, in Area, make the following selections. Then, choose OK.
  - To remove all colors and patterns, select None.
  - To change a color or pattern, select a color.
  - To add a pattern in the chart area, select Fill Effects. Then, make the following selections:
    - On Fill Effects, in Pattern, select an icon pattern.
    - Optionally, select a foreground color and a background color for the fill pattern.
  - To re-apply typical color and pattern settings, select Automatic.

## Changing the line that borders a chart area

Typically, the chart area appears surrounded by a one point, black line. The plot area appears surrounded by no line. To enhance the difference between the chart and plot areas, modify the border of an area. For example, to mark a boundary between the chart and plot areas, add a line that borders the plot area.

### **How to change the line that borders a chart area**

- 1 Right-click the chart area, choose Format Chart Area, then select Patterns.
- 2 On Patterns, in Border, make the following selections. Then, choose OK.
  - To remove all border lines, select None.
  - To add lines that surround the chart area, select custom, then select style, color, and weight options for the lines that border the chart area.
  - To re-apply typical border line settings, select Automatic.

### **Changing a font characteristic in the chart area**

To set the appearance of text in axis and legend labels, set font characteristics in the chart area. To keep the text size of these elements proportional to chart size when you change the size of the chart, set font scaling in the chart area. Changing font characteristics for the chart area overrides font settings that you make previously to axis and legend labels.

#### **How to change a font characteristic in the chart area**

To change the appearance of text in axis and legend labels:

- 1 Right-click the chart area, choose Format Chart Area, then select Font.
- 2 On Font, make any of the following selections. Then, choose OK.
  - In Font, select a font name.
  - In Size, select a number.
  - In Color, select a colored icon.
  - In Style, select one or more from; bold, italic, strikeout, and underline.

#### **How to resize text in axis and legend labels when you resize a chart**

- 1 Right-click the chart area, choose Format Chart Area, then select Font.
- 2 On Font, in Style, select AutoScale, then choose OK.

### **Working with hidden data**

If you link a chart to a data range that contains hidden data, then you choose whether to display the hidden data in the chart or display in the chart only data that you can see in the data range. For example, you can set a chart linked to a data range in a worksheet that contains an outline to display or not display the data not visible in the worksheet when the outline appears collapsed.

#### **How to display hidden data in a chart**

- 1 Right-click the chart area, choose Format Chart Area, then select Options.
- 2 On Options, deselect Plot visible cells only, then choose OK.

## Displaying a non-numeric data value in a chart

In a chart, BIRT Spreadsheet Designer plots 0 for a data point linked to a cell that contains text. You can choose one of three options for how a data point linked to an empty cell appears.

### How to set the display option for an empty cell

- 1 Right-click the chart area, choose Format Chart Area, then select Options.
- 2 On Options, in Empty Cells, select one of the following options. Then, choose OK.
  - Not used. No data point appears on the chart.
  - Plot as zero value. BIRT Spreadsheet Designer plots the data point at 0.
  - Interpolated. BIRT Spreadsheet Designer calculates the missing value based on values of the adjacent data points.

## Resetting a chart layout

After changing the appearance or type of a chart, you can reset formatting options for the chart object to a set of typical settings that display a basic layout. BIRT Spreadsheet Designer provides two options for resetting a chart object to a basic layout:

- To return chart elements that you have moved or resized to typical sizes and positions without changing the chart size, shape, and type, right-click the chart area, then choose Automatic Layout.
- To remove all non-typical formatting from a chart object, right-click the chart area, then choose Default Chart. Choosing Default Chart resets the chart object to a basic, column chart layout.

## Formatting an element

The appearance of each chart element affects the overall appearance of the chart image. Some chart elements, such as titles and labels, should look similar, but distinct, so that a reader recognizes their function. Other elements, such as data series, should stand out clearly for simple comparison. Finally, some elements must fit clearly in a chart. For example, names that appear as labels on the category axis must not overlap. To create a clear graphic image, change the appearance of one or more data elements.

## Showing and hiding an element

A typical chart shows a legend. A typical column chart shows axes, but a typical pie chart does not. A typical chart does not show data labels. To show or hide chart elements in the chart area, use Chart Options.



### How to show and hide an element

- 1 Right-click the chart area, then choose Chart Options.
- 2 On Chart Options, select any of the following tabs, and select Visible. Then, choose OK.
  - Axes
  - Legend
  - Data Labels  
On Data Labels, in Type, select the data label type that best describes the data.

### Changing the color or pattern in an element

Typically, an element in a chart appears black with a white or grey background. To group or contrast elements, change the color or pattern that fills each element to the same or a different color. For example, select one color for major grid lines, a lighter shade of the same color for minor grid lines and a separate color that contrasts the grid for each data series element in your chart.

#### How to change the color or pattern in an element

- 1 Right-click an element, choose Format 'elementName', then select Patterns.
- 2 On Patterns, in Area, make the following selections. Then, choose Apply.
  - To remove all colors and patterns, select None.
  - To change a color or pattern, select a color.
  - To add a pattern that fills an element, select Fill Effects:
    - On Fill Effects, in Pattern, select an icon pattern, then choose OK.
    - Optionally, select a foreground color and a background color for the fill pattern.
  - To re-apply typical color and pattern settings, select Automatic.Choose OK.

### Changing the line that borders an element

To make an element, such as an axis or a grid line, appear more distinct, change the style, color, or width of the line that represents that element. You can also change the color and width of the line that surrounds, or borders a chart element, such as a title, legend, axis label, or data label.

#### How to change the line that borders an element

- 1 Right-click an element, choose Format 'elementName', then select Patterns.

- 2 On Patterns, in Border, make the following selections. Then, choose OK.
  - To remove all lines that border an element, select None.
  - To add lines that surround an element, select custom, then select style, color and weight options for the lines that border the element.
  - To add a darker, wider border along the bottom and right edge of an element, select Shadow.
  - To re-apply typical border line settings, select Automatic.

## Changing a font characteristic for an element

To set the appearance of text in an elements such as a title or data label, set the font characteristics for that element. To keep the text size of an element proportional to chart size when you change the size of the chart, set font scaling in the chart area. Changing font characteristics for an element does not affect the font settings for the chart area.

### How to change a font characteristic in an element

- 1 Right-click an element, choose Format 'elementName', then select Font.
- 2 On Font, make any of the following selections. Then, choose OK.
  - In Font, select a font name.
  - In Size, select a number.
  - In Color, select a colored icon.
  - In Style, select one or more from; bold, italic, strikeout, and underline.

## Aligning text in an element

To more clearly describe data, adjust the position of text that describes a chart element. For example, you can change alignment of text in an axis label, an axis title, a data label, or a chart title. You can also orient text horizontally, vertically, or on an angle.

### How to change text alignment in an element

- 1 Right-click an element, choose Format 'elementName', then select Alignment.
- 2 On Alignment, make any of the following selections. Then, choose OK.
  - In Text Alignment, select a horizontal and vertical alignment option.
  - In Orientation, select an option or select Custom then specify an angle number.

Sample displays an example of the alignment and orientation options that you select.

## Changing the number format in an element

To describe numeric data more clearly, apply formats to numbers that appear in an element such as a data label or the value axis label.

### How to change the format of a number

- 1 Right-click an element, choose Format 'elementName', then select Number.
- 2 On Number, make any of the following selections. Then, choose OK.
  - In Category, select a format group name, such as Currency.
  - In Number format, select an option from the list of examples.

Sample displays how the number looks with the format option that you select.

## Working with a data series

A data series is a chart element that graphically shows a set of data points. For example, a line chart shows a data series as a line that connects two or more data points. To change the appearance of a data series, format a data series. To describe a set of data points, label a data series. To compare data series, add a series to a chart, change the order of series, stack series or plot series data points as a percentage of a category.

### Formatting a data series

Select different format options for each data series so that data points in one series appear similar to each other, yet distinct from those in another series. Topics in this section describe how to change the appearance of a data series.

## Changing shape, color and size of points in a data series

To contrast individual data series, change the patterns that mark the data points in each data series in an area, bar, column, line, step, combination, or scatter chart.

### How to change marker patterns in a data series

- 1 Right-click a data series, choose Format Data Series, then select Patterns.
- 2 In Markers, select the following options and choose Apply. Then, choose OK.
  - To remove colors and patterns, select None.
  - To change a marker shape, in Style, select a shape name.
  - To add color to a data series marker, in Foreground and Background, select a color.
  - To change the size of data series markers, in Size use the up or down arrows to increase or decrease the size of all markers.
  - To re-apply typical color and pattern settings, select Automatic.

**3** In Line, make the following selections. Then, choose OK.

- To remove a line, select None.
- To add a line that links data points along the data series, select custom, then select style, color and weight options.
- To re-apply typical line settings, select Automatic.

## **Assigning different colors to each data point in a single data series**

Typically, data points in the same series display the same color. To display in varying colors the shapes that represent points in a data series, format the data series element.

### **How to assign a different color to each data point in a data series**

- 1** Right-click the series, choose Format Data Series, then select Options.
- 2** On Options, select Vary colors, then choose OK.

## **Setting different colors for positive and negative values in a data series**

To set different, contrasting colors that represent positive and negative values in a chart, format the patterns in the data series element. For example, columns or bars can display black for positive values and red for negative values. Similarly, the foreground and background patterns that you set as fill effects reverse for positive and negative values. For more information about color and fill options, see “Changing the color or pattern in an element,” earlier in this section.

### **How to display negative chart values in a different color or pattern**

To set contrasting color and fill effects for positive and negative data series:

- 1** Right-click the data series, choose Patterns, then select Invert if negative.
- 2** Select Fill Effects, make the following selections. Then, choose OK.
  - In Foreground, select a color and optionally a fill pattern, for positive values.
  - In Background, select a color and optionally a fill pattern for negative values.
  - On Patterns, select Apply, then choose OK.

## **Plotting a data series on a second y-axis**

Two-dimensional area, bar, column, combination, line, and step chart types support plotting a data series on a second y-axis. To add a second y-axis, format the data series you want to plot on it.

### How to plot a data series on a second y-axis.

- 1 Right-click a data series, then choose Format Data Series.
- 2 On Axis, select y-axis 2. Then, choose OK.

For a combination chart, right-click a data series, choose Chart Type and then select a chart type different from the one that displays the other series. In a combination chart, the second y-axis can have a type and scale different from the first y-axis.

## Displaying y-error bars in a data series

Y Error bars show the potential error in the y-value for each data point in a series. You can add y-error bars to two-dimensional area, bar, bubble, column, line, or step chart types.

### How to add y-error bars

- 1 Right-click a data series, choose Format Data Series, then select Y-Error Bars.
- 2 On Y-Error Bars, make any of the following selections. Then, choose OK.
  - In Display, select the type of error bar to display.
  - In Show tee-top marker, determine the shape of the error bar:
    - To show the error bar as a vertical line, deselect Show tee-top marker.
    - To show the error bar as a vertical line with horizontal lines at the top and bottom of the line, select Show tee-top marker.
  - In Error amount, determine the error calculation:
    - To use a fixed value as the error amount, select Fixed value, then type the value.
    - To use a percentage of the value of the data point as the error amount, select Percentage, then type a percentage value.
    - To use the data point value's standard deviation from the mean of plotted values as the error amount, select Standard deviation, then type the number of standard deviations to use in the calculation.

## Hiding data labels for a data series

To remove text that describes points in a data series from the chart area, hide the set of data labels for one or more data series.

### How to hide a data series label.

- 1 Right-click a data series, choose Format Data Series, then select Data Labels.
- 2 On Data Labels, in Type, deselect a data label type, then choose OK.

## Changing display order of data series

Typically, a chart displays multiple data series in the same order that the series appear the data range linked to the chart.

### How to change the display order of data series

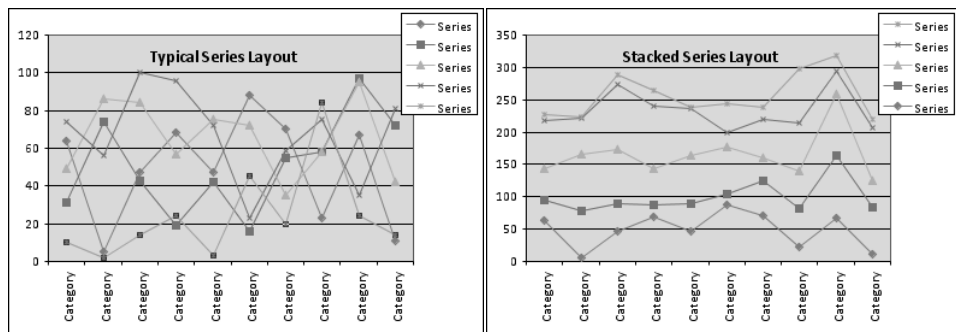
- 1 Right-click a data series, choose Format Data Series, then select Order. In Series order, the data series you selected appears highlighted.
- 2 Select Move up or Move down to move the highlighted series up or down the order.
- 3 When Sample displays the series order you want displayed in the chart, choose OK.

## Stacking series

Typically, area, bar, column, line, and step charts plot each data series along the x-axis. To show how each data point contributes to the total value of all data points in a category, you stack series. When you stack series, you change the layout so that the data points in one series appear above the data points of another series. When you stack series, you must stack all series on an axis.

BIRT Spreadsheet Designer adjusts the value axis interval of stacked series to accommodate the arrangement. If you previously adjusted the value axis interval, stacking series does not reset the interval. In this case, the settings you applied to that axis remain. If the maximum value you set is not large enough to accommodate the now-stacked data points, some data points may not appear in the chart area.

Figure 11-13 shows how stacking arranges several series along the horizontal axis, stacked vertically. The value axis adjusts to accommodate all values in all series.



**Figure 11-13** Typical series layout compared with stacked series layout

### How to stack data series

- 1 Right-click a data series, choose Format Data Series, then select Options.

- 2 On Options, select Stacked, then choose OK.

## Plotting data points as percentages

Typically, an area, bar, column, line, or step chart plots values starting at value 0 where the value axis intersects the category axis. To display each data point as a percentage of the total of all data points for each category, plot data series as percentages. Data series plotted as percentages look similar to those stacked, except that the value axis adjusts to show values from 0 to 100%. BIRT Spreadsheet Designer totals all data points in a category, then finds the percentage of that total each data point represents. The shapes that represent data series values in each category fill the plot area for that category. When you plot series as percentages of category totals, you do so for all of the series on an axis.

If you have previously adjusted the maximum or minimum value on a value axis, BIRT Spreadsheet Designer plots data points from 0 to 1 without changing your settings. If the maximum or minimum value is larger or smaller than 1, the chart data may not appear on the chart or may stretch beyond the plot area. Before plotting data points as percentages, set the value axis scale to Automatic, to avoid axis scale problems.

### How to plot data points as percentages

- 1 Right-click a data series, choose Format Data Series, then select Options.
- 2 On Options, select Percent, then choose OK.

## Formatting data series in specific chart types

Topics in this section describe how to change the appearance of a data series in the following chart types:

- Bar or column
- Bubble
- Combination
- Doughnut
- Line chart
- Pie

### Formatting a bar or column chart

To differentiate shapes that represent data series in a bar or column chart, do any of the following tasks:

- Change the space between shapes.
- Change the space between categories.

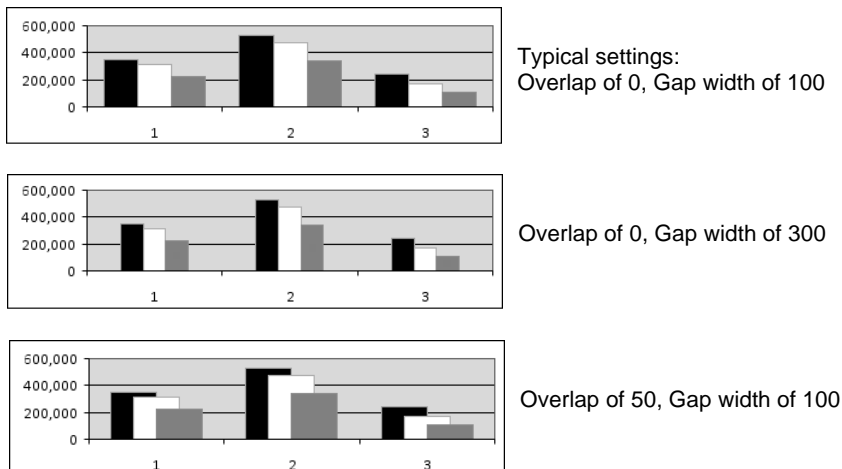
- Change the shape that represents each series.
- Add lines that connect each shape in a series.

## Setting space between groups and shapes

Shapes that represent data series on bar and column charts appear grouped by category. Each shape in a particular category represents a data point from one series. Typically, shapes from different series within a category touch, and space appears between categories.

To set the space that appears between shapes on any bar or column chart, type values in **Overlap** and **Gap width** on **Format Data Series—Options**. Figure 11-14 shows the effects of these settings.

- **Overlap** sets the amount of space between shapes in a category. The typical overlap value 0 sets no space between shapes. A positive value in **Overlap** causes shapes in a category to appear overlapped. A negative value in **Overlap** creates space between shapes in a category.  
The largest allowed overlap value is 100, which makes shapes in each category appear on top of each other. The smallest allowed overlap value is -100, which makes the width of the spaces between shapes equal to the width of the shapes.
- **Gap width** sets the amount of space between categories, or groups of shapes. Typically, gap width is 100, which sets the space between categories to 100% of the width of a shape.  
The largest allowed gap width value is 500, which sets the space between shapes to five times that of the widest shape. The smallest allowed gap width value is 0, which causes shapes in adjacent categories to touch.



**Figure 11-14** How **Overlap** and **Gap width** affect column chart appearance



### How to create space between shapes in a data series

- 1 Right-click a data series, choose Format Data Series, then select Options.
- 2 On Options, type values in the following fields. Then, choose OK.
  - In Overlap, to set the amount of space between shapes in a category, type a number between -100 and 100.
  - In Gap width, to set the amount of space between categories, type a number between 1 and 500.

### Changing the shape of a bar or column

For a bar or column chart that displays multiple data series, you may represent each series with a line instead of a bar or column. In a bar or column chart that shows depth, you can show a series as a shape such as a pyramid, cylinder, or cone.

### How to show bars as lines

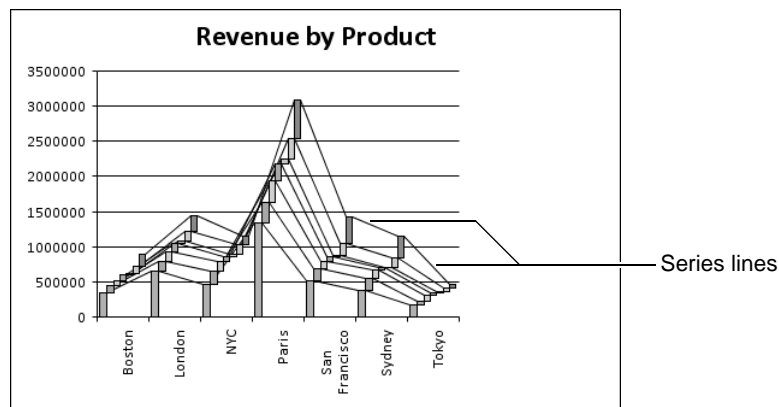
- 1 Right-click a data series, choose Format Data Series, then select Options.
- 2 On Options, select Show bar as line, then choose OK.

### How to change the shape of bars in a bar chart with depth

- 1 Right-click a data series, choose Format Data Series, then select Shape.
- 2 Select a shape, then choose OK.

### Showing series lines

Series lines connect the shapes that represent a series in a bar or column chart as shown in Figure 11-15. You can only show series lines when you stack series or when you display series data points as percentages of category totals.



**Figure 11-15** Series lines in a stacked, column chart

### How to show series lines

- 1 Right-click a data series, and choose Format Data Series. Then, select Options.
- 2 On Options, select Series lines. Then, choose OK.

### Formatting a bubble chart

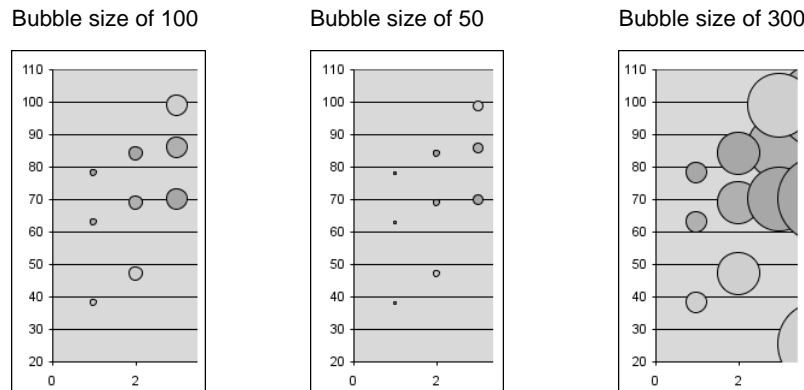
A data value determines the relative size of each bubble that appears on a bubble chart. Two other data values, the x and y values, determine the location on the chart in which the bubble shape appears. A data value that is large relative to other values in a data series plots as a bubble shape that is large relative to other bubble shapes. To clarify a bubble chart, reformat or rearrange data series that plot as groups of bubble shapes.

### Setting bubble size

When plotting a bubble chart that represents a data range, BIRT Spreadsheet Designer first uses data values to calculate a base bubble size. Then, a bubble shape that has a size proportioned to the base size appears at a point plotted from the location of data values on the x-axis and y-axis of the chart. The base bubble size has value 100.

To clarify bubble chart appearance, change Bubble size from 100 to a value between 0 and 300. Changing Bubble size changes the size of all bubble shapes on a bubble chart.

Figure 11-16 shows how changes to the value in Bubble size affect the appearance of a bubble chart.



**Figure 11-16** Changing the value in Bubble size

### How to change the bubble size

- 1 Right-click a data series in a bubble chart, choose Format Data Series, then select Options.

- 2 On Options, in Bubble size, type a value between 0 and 300, then choose OK.

#### **How to show the data value as a label for a bubble shape**

- 1 Right-click a data series in a bubble chart, choose Format Data Series, then select Data Labels.
- 2 On Data Labels, in Type, select Bubble size, then choose OK.

### **Rearranging bubbles that overlap**

If two or more data series have similar X and Y values, bubbles can overlap, and one bubble may hide another. In a chart that shows multiple data series, bubbles that display different colors belong to different data series. BIRT Spreadsheet Designer draws bubbles in the order in which they appear on the worksheet, with the data in the top worksheet row drawn first and the data in the lower rows drawn on top of the data drawn previously.

- To rearrange differently colored bubbles that overlap, change the order of the data series.
- To rearrange same colored bubbles, or bubbles that represent values in the same series, change the row order of the data range to which the chart links.

### **Adjusting axis scale on a bubble chart**

A large data value used to plot a bubble size that corresponds with a large x or y value may cause a bubble that appears at one end of an axis to extend out of the plot area. To adjust a chart that shows a bubble off the plot area, change the axis scale. For more information about formatting an axis, see “Placing and scaling an axis,” later in this chapter.

### **Formatting a combination chart**

A combination chart can display each data series using a different chart type. To format a combination chart, select each data series, then use formatting options available for the chart type selected to display that data series.

### **Formatting a doughnut chart**

In a doughnut chart, each data series looks like a ring. Multiple series form concentric rings. Categories appear as sectors along each ring. To clarify a doughnut chart, format the data series that forms each concentric ring.

### **Separating sectors of a doughnut chart**

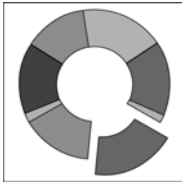
Typically, sectors in a doughnut chart touch each other. To provide space between chart categories, separate, or explode the sectors. In a doughnut chart that displays multiple series, only the outermost ring can display separated sectors.

No space displays between the inner rings of a multi-series doughnut chart when you explode the sectors of the outermost ring.

### How to separate sectors of a doughnut chart

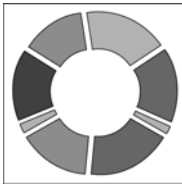
Use one of the following methods to separate one or all sectors from a ring that represents a data series:

- To separate one sector from a series ring, select a sector, drag it and drop it away from the doughnut ring, as shown in Figure 11-17.



**Figure 11-17** One category sector separated from a doughnut chart series

- To separate all sectors in the outside ring of the doughnut, right-click the ring that represents the data series, choose Format Data Series, then select Options. On Options, in Explosion percent, type a value from 0 to 100, then choose OK. Figure 11-18 shows a series with an Explosion percent value of 10.

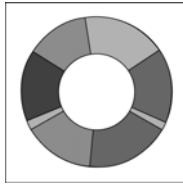


**Figure 11-18** A doughnut chart series exploded

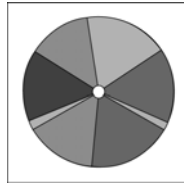
### Sizing the center of a doughnut chart

Typically, the diameter of the circle that forms the center of a doughnut chart appears half the outside diameter of the largest doughnut ring. To change the size of the doughnut center, specify a percentage of the total doughnut width as a doughnut hole size. Figure 11-19 shows the effect of changing the percentage size of the doughnut center.

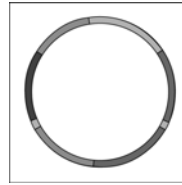
Center is 50% of  
series diameter



Center is 10% of  
series diameter



Center is 90% of  
series diameter



**Figure 11-19** Differently sized holes in a doughnut chart

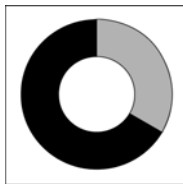
#### How to change the size of a doughnut center

- 1 Right-click the data series, choose Format Data Series, then select Options.
- 2 In Doughnut hole, type a value between 10 and 90, then choose OK.

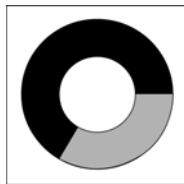
### Changing the start angle of doughnut sectors

Typically, sectors of a doughnut chart display in an order that proceeds clockwise in the same order as the data range linked to the chart. Typically, the edge of the first sector starts at 0°, or the 12 o'clock position. The edge of the first sector in each ring of a multi-series doughnut chart aligns with the start position. To change the position of sectors in a doughnut chart, change the angle relative to 0° to which the first sector(s) aligns. Figure 11-20 shows the effect of changing the start angle of the doughnut sectors.

Start angle is 0°



Start angle is 90°



**Figure 11-20** Different start angles for data series

#### How to change the start angle of doughnut sectors

- 1 Right-click the data series, choose Format Data Series, then select Options.
- 2 In Start angle, type a value between 0 and 360, then choose OK.

### Formatting a pie chart

In a pie chart, each data series appears as a full circle. Categories appear as sectors or slices of the circle. Each slice represents a portion of the whole, circular pie. The ratio of each data value to the total of data values in the series determines the size of each slice.

## Exploding sectors of a pie chart

Typically, sectors in a pie chart touch each other. To provide space between chart categories, you separate, or explode the sectors. To separate sectors, use one of the following methods:

- To separate an individual sector from a pie chart, select a sector, drag it and drop it some distance away from the other sections of the pie.
- To separate, or explode, all the sectors in the pie, right-click the chart, choose Format Data Series, then select Options. In Explosion percent, type a value from 0 to 100.

## Changing the start angle of pie sectors

Typically, sectors of a pie chart display in an order that proceeds clockwise in the same order as the data range linked to the chart. Typically, the edge of the first sector starts at 0°, or the 12 o'clock position. To change the position of sectors in a pie chart, change the angle relative to 0° to which the first sector aligns.

### How to change the start angle of pie sectors

- 1 Right-click the data series, choose Format Data Series, then select Options.
- 2 On Options, in Start angle, type a value between 0 and 360, then choose OK.

## Displaying data label lines

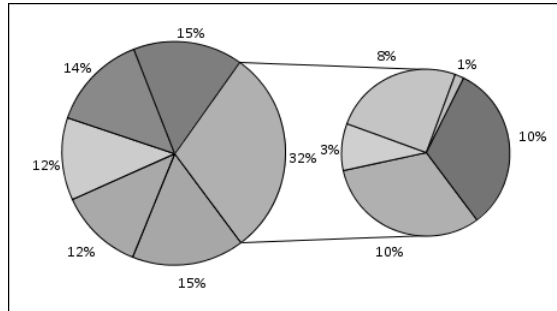
A data label line connects each sector of a pie chart to the data label that describes that sector.

### How to display data label lines

- 1 Right-click the pie chart area, choose Chart Options, then select Data Labels.
- 2 On Data Labels, select Data label lines, then choose OK.

## Formatting a data series in a pie of pie chart

A pie of pie chart shows slices that represent smaller data values as a separate pie. Figure 11-21 shows a sample image of a pie of pie chart. Format options for a pie of pie chart differ from those for a pie chart that displays a single shape.

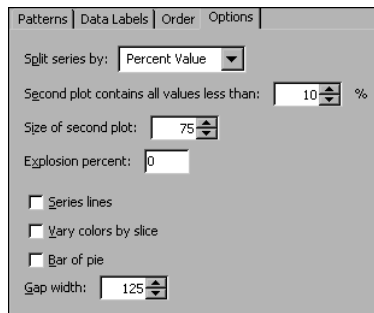


**Figure 11-21** Pie of pie chart

### How to format a data series in a pie of pie chart

Use Format—Options to control how categories appear in a pie of pie chart.

- 1 Right-click the data series, choose Format Data Series, then select Options. Figure 11-22 shows Options with the typical option, Percent value, selected in Split series by.



**Figure 11-22** Pie of pie chart options to split data series by percent value

- 2 On Option, select any of the following options. Then, choose OK.
  - In Split series by, determine how to allocate categories in the second pie by selecting one of the following options:
    - To allocate categories based on the order in which data appears in the data range linked to the chart:
      - In Split series by, select Position.
      - In Second plot contains the last, select a number of values to appear in the second pie.

The second pie appears with the number of values that you select which appear last in the data range.
    - To allocate categories based on the value of the category:

- ❑ In Split series by, select Value.
- ❑ In Second plot contains all values less than, type a value.

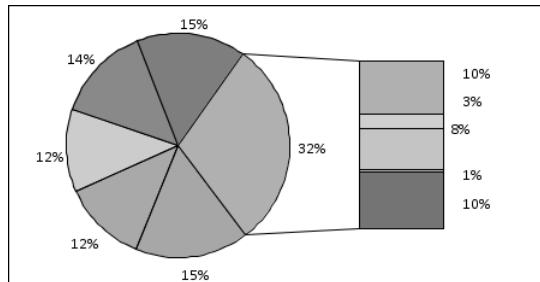
The bar or second pie displays all categories with a value that is less than the value you type.

- ❑ To allocate categories based on the value of the category compared to the total value of the main pie:

- ❑ In Split series by, select Percent Value.
- ❑ In Second plot contains all values less than, select a percent value.

The bar or second pie displays all categories that compose less than that percentage of the main pie.

- ❑ To change the relative positions of slice shapes in a pie of pie chart, select Custom. After you select Custom, use Change Position to move a slice. For more complete instructions, see “How to change the positions of slices in a pie of pie or bar of pie chart,” later in this chapter.
- In Size of second plot, select a percentage value. This value specifies the size of the second pie as a percentage of the first pie shape.
- In Explosion percent, type a value between 1 and 100. This value sets a ratio used to separate sectors in both pie shapes.
- Select Series lines to show lines that connect both pie shapes. The second pie shape appears connected to one sector in the first pie shape that represents all data that the second pie shape represents.
- Select bar of pie to show the second pie shape as stacked bars. Figure 11-23 shows a pie of pie chart with bar of pie format selected.



**Figure 11-23** Bar of pie chart

- In Gap width, select a value to adjust the space between the first and the second shape. BIRT Spreadsheet Designer uses the value you select in Gap width as a percent ratio of the size of the second shape. For example, if you type 100, the second shape appears separated from the first by a distance equal to the width of the second shape.



### **How to change the positions of slices in a pie of pie or bar of pie chart**

- 1 Right-click the data series, choose Format Data Series, then select Options.
- 2 On Options, in Split series by, select Custom, then choose OK.
- 3 Right-click a slice that represents a single category in either shape, then choose Change Position. The slice that you chose appears in the other shape. Positions of other slices in both shapes change to keep the pie shape(s) circular.

## **Adding reference lines to a chart**

Typically, each data series appears distinct from another. To relate values in a data series to an axis, or to another series, add reference lines to a chart. BIRT Spreadsheet Designer supports the following types of reference lines:

- Drop lines
- High-low lines
- Up or down bars

### **Adding drop lines**

A drop line connects the data point with the highest value in a category to the category axis. When you display drop lines for a data series, BIRT Spreadsheet Designer draws drop lines on all categories and all series on an axis. You can display drop lines for only one axis in a chart. For example, you can display drop lines on a main chart axis but not on a study chart axis. To change the appearance of a drop line, select pattern options as you would for any chart element.

#### **How to add drop lines to a chart**

- 1 Right-click a data series, choose Format Data Series, then select Options.
- 2 On Options, select Drop lines, then choose OK.

### **Adding high-low lines**

High-low lines connect the highest data point of all series in a category to the lowest data point of all series in that category. High-low lines often appear on a stock chart to show the highest and lowest price of each stock.

#### **How to add high-low lines to a line chart**

- 1 Right-click the data series, choose Format Data Series, then select Options.
- 2 On Options, select High-low lines, then choose OK.

### **Adding up or down bars**

An up or down bar connects each data point in the first series on a chart to a data point in the last series that has the same category value. For example, up or down

bars appear on a stock chart to show prices at which stocks opened and closed in a trading period. A chart displays one up or down bar on each category. Up or down bars appear only between the first and last series that appear on a chart.

#### **How to add up or down bars**

- 1 Right-click the data series, choose Format Data Series, then select Options.
- 2 On Options, select Up or down bars, then choose OK.

### **Changing the width of up or down bars**

To highlight each data value pair, change the gap width between up or down bars. Changing the gap width value for one data series changes the width of all up or down bars between data values in both first and last data series.

#### **How to change the width of up or down bars**

- 1 Right-click a data series with a up or down bars, choose Format Data Series, then select Options.
- 2 On Options, in Gap width, type a value between 0 and 500, then choose OK.

### **Working with axes**

Axis lines, grid lines, and tick marks provide spatial reference information along a chart axis. Tick mark labels provides descriptive information along a chart axis. To change the appearance of descriptive information along an axis, select that axis, then format characteristics of the axis or elements plotted along it. For example, to clarify text in tick mark labels that appear along the horizontal axis, right-click the horizontal axis, then select options on Font, Number or Alignment. Sample displays how your selections affect the appearance of the chart element.

When you modify x-axis settings, you change the settings for all x-axes on the chart or study chart. In a chart that displays more than one y-axis, you can modify one y-axis on the chart without affecting the other y-axis.

To work with lines and labels at the chart, or macro level, right-click the chart area, choose Chart Options, then select Axes.

### **Showing and hiding lines and labels**

To show or hide the lines that extend across the plot area and touch each axis, on Chart Options—Axes, select any of the following options, then choose OK:

- In Y-axis:
  - To show data labels, axis lines, and tick marks along the vertical axis, select Visible.
  - To show lines that extend horizontally across the plot area, and touch the y-axis at the value of each major unit, select Major grid.

- To show lines that extend horizontally across the plot area, and touch the y-axis at the value of each minor unit, select Minor grid.
- To hide data labels, axis lines and tick marks along the vertical axis, deselect Visible.
- In X-axis:
  - To show data labels, axis lines and tick marks along the horizontal axis, select Visible.
  - To show lines that extend vertically across the plot area, and touch the x-axis at the value of each major unit, select Major grid.
  - To hide data labels, axis lines and tick marks along the horizontal axis, deselect Visible.

## Showing or hiding lines on an axis

To add or remove reference lines along a chart axis, you select which elements along the axis you want the chart to display. Hiding an axis hides all elements along the axis, including the axis line, ticks, and axis labels.

To work with lines, tick marks and labels at the axis, or micro level, right-click an axis, choose Format Axis, then select Options.

### How to show or hide an element along a chart axis

- 1 Right-click a chart axis, choose Format Axis, then select Options.
- 2 On Options, make any of the following selections. Then, choose OK.
  - Select to show, or deselect to hide, axis lines, major and minor grid lines.
  - For each option you select as visible, select an option that positions the element relative to the axis line. For example, in Major Ticks, select Cross to display tick marks that cross the axis line.

## Showing and hiding descriptions on an axis

Tick mark labels display text that describes information along an axis. Typically, tick mark labels appear on a chart with axes. You can hide the axis labels.

### How to hide tick mark labels

- 1 Right-click an axis, choose Format Axis, then select Options.
- 2 On Options, in Tick mark labels, select None.

## Placing and scaling an axis

To adjust the placement of an axis, or the order and space between data points along an axis, modify settings on Scale. For example, a category axis can appear at

the top or bottom of a chart. Also, an axis can display specific numbers and groups of categories, such as months, quarters or years.

This section describes:

- How to change where axes appear in relation to each other
- How to rearrange data points, by reversing the order in which categories or values appear on an axis
- How to create a logarithmic value axis
- How to create a time scale category axis

## **Working with an axis intersection**

On a chart with multiple axes, the axes must intersect. Typically, the intersection occurs at the lower left corner of the plot area at a point that corresponds to the minimum value on one axis. On a value axis, the default intersection point is 0 or the value closest to 0. On a category axis, the default intersection point is the outside edge of the first category. The default intersection point on a time-scale axis is the outside edge of the earliest major division. To change the intersection on an axis, specify the value at which the other axis crosses.

### **How to adjust a y-axis intersection**

- 1 Right-click the y-axis, choose Format Axis, then select Scale.
- 2 On Scale, specify a y-axis intersection using one of the two following methods. Then, choose OK.
  - To select a y-axis value at which the category axis intersects, in Interval, deselect Intersection and type a y-axis value.
  - To set the intersection point to the maximum value on a category or value axis, select Adjacent axis intersects at maximum.

### **How to adjust an x-axis intersection**

- 1 Right-click the x-axis, choose Format Axis, then select Scale.
- 2 On Scale, specify an x-axis intersection, using one of the two following methods. Then, choose OK.
  - To set the axis intersection to the maximum x-axis value, select Y-axis crosses at maximum category.
  - To select an x-axis value at which the value axis intersects, type an x-axis value in Y-axis crosses at category number.

## **Changing the size of spaces on a category axis**

To adjust the space between labels, ticks, and grid lines on a category axis, modify settings on Scale. On a category-scale axis, adjust settings under Frequency. On a

time-scale category axis, adjust settings under Interval. For example, if a chart plots stock prices over time and uses dates as categories, you can display one tick mark for each week, one tick mark for each day, or several tick marks for each day.

#### **How to adjust frequency along a category scale axis**

- 1 Right-click a category axis, choose Format Axis, then select Scale.
- 2 Select a value for each of the following options. Then, choose OK.
  - Label frequency
  - Number of categories between ticks
  - Category number at which y-axis crosses.

#### **How to adjust intervals along a time scale category axis**

- 1 Right-click a time-scale category axis, choose Format Axis, then select Scale.
- 2 In Interval, make the following selections. Then, choose OK.
  - To apply typical settings, based on linked data values, select the item under Auto that describes each interval setting.
  - To customize time interval settings, deselect Auto, and then type or select a value for each interval parameter.

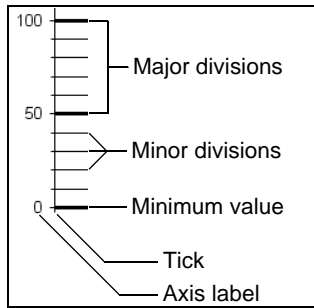
### **Changing the interval on a value axis**

The interval of a value axis requires four settings:

- Minimum value
- Maximum value
- Major unit
- Minor unit

Figure 11-24 shows a value axis with the following interval settings:

- Minimum value of 0
- Major unit of 50
- Minor unit of 10



**Figure 11-24** Chart elements that show an interval

### How to adjust the interval along a value axis

- 1 Right-click a value axis, choose Format Axis, then select Scale.
- 2 On Scale, in Interval, make the necessary selections and type appropriate values, then choose OK.

### Reversing axis order

To change the order in which values or categories appear along an axis, reverse the order of the axis. Reversing the value axis causes category labels to appear at the opposite side of a chart. Reversing axis order may affect the axis intersection.

### How to reverse order on an axis

- 1 Right-click the axis, choose Format Axis, then select Scale.
- 2 Select Reverse order, then choose OK.

### Making a value axis logarithmic

A value axis that appears logarithmic displays data points that appear along the value axis at points calculated as results of the log function. A line that connects these points appears to increase at rate which increases. Only a value axis can appear logarithmic.

### How to make a value axis logarithmic

- 1 Right-click a value axis, choose Format Axis, then select Scale.
- 2 On Scale, select Logarithmic, then choose OK.

### Creating a time-scale category axis

Only a category axis can appear as a time-scale axis. You cannot convert a value axis into a time-scale axis. When you create a time-scale axis, BIRT Spreadsheet Designer determines the axis interval settings. You can change these settings or select your own values. BIRT Spreadsheet Designer considers the following factors when setting interval values for a time-scale axis:

- **Minimum**  
The earliest date or time that appears on the axis. BIRT Spreadsheet Designer sets the minimum to a date earlier than the earliest date in the data when the base unit requires doing so. For example, weeks that appear on a time-scale axis start on Sunday. So, if a chart with a base unit of weeks contains data that starts on Tuesday, the minimum day that appears on the axis, is two days earlier than the earliest day plotted in the chart data.
- **Maximum**  
The latest date or time that appears on the axis. As with axis minimum date, BIRT Spreadsheet Designer sets the maximum date to the next date after the latest date plotted in the chart data.
- **Base unit**  
BIRT Spreadsheet Designer chooses a base unit using an internal algorithm that limits the number of major divisions, without creating too few. For example, on a chart that displays 22 days' worth of data, a base unit of weeks results in only 4 major divisions. You can specify a different base unit.
- **Major unit**  
Determines the interval and unit of major tick marks and grid lines.

#### **How to create a time scale category axis**

- 1 Right-click a chart area, choose Chart Options, then select Axes.
- 2 On Axes, in X-axis, select Time scale, then choose OK.

## **Describing elements in a data series**

A data label element describes a data series in a chart. For example, use a data label to display the actual data value that a data point represents. Data labels display the following types of information:

- Series name
- Category name
- Data point value
- Percentage of a whole series value
- Value that quantifies the size of a bubble in a bubble chart

More than one label type can describe each data point in a data series. When you use more than one type of label, specify how to arrange the label information for each data point. For example, when you display the series name and the data point value, use a comma to separate the name and value.

## Displaying data labels

Make data labels visible to show more information about a data series. A data label displays text or numbers. Format a data label like other chart elements to clearly describe a data series.

You format a data label element like you format other chart elements. Use Format Data Labels to change data label position, or change data label fill, line style, font, number format, or alignment. In a pie chart, you can make data label lines visible to connect a pie sector with a label.

### How to display data labels

- 1 Right-click a data series, choose Format Data Series, then select Data Labels.
- 2 On Data Labels, in Type, select each data label type that you want visible, then choose OK.

### How to format a data label

- 1 Right-click a data label, choose Format Data Label, then select Patterns.
- 2 Select any of the following options and modify any characteristics. Select Apply. Then, choose OK.
  - To modify colors and patterns, select Patterns.
  - To change characteristics of text, choose Font.
  - To format numbers, select Number.
  - To modify the position and orientation of data labels, select Alignment.
  - To change the position of data labels relative to data points, select Options.

## Working with a trendline

A trendline appears on the plot area of a chart as a line that illustrates a trend in the charted data. BIRT Spreadsheet Designer supports multiple trend or regression types. Base your choice of trendline type on the data that your chart represents.

### About linear trendlines

A linear trendline appears as a best-fit, straight line. Use a linear trendline to show a trend that increases or decreases at a constant rate. Use a linear trendline with simple, linear data sets. A data series in a linear data set resembles a straight line.

The Linear option creates a trendline by using the following, linear equation to calculate the least squares fit for a line:

$$y = mx + b$$



where  $m$  is the slope and  $b$  is the y-intercept.

## About logarithmic trendlines

A logarithmic trendline appears as a best-fit, curved line. Use a logarithmic trendline to show a rate of change in data that increases or decreases quickly and then levels out. A logarithmic trendline can display negative and/or positive values.

The Logarithmic option creates a trendline by using the following equation to calculate the least squares fit through points:

$$y = c \ln x + b$$

where  $c$  and  $b$  are constants, and  $\ln$  is the natural logarithm function.

## About polynomial trendlines

A polynomial trendline appears as a curved line. Used a polynomial trendline to illustrate data that fluctuates, such as value gains and losses over a large data set. Estimate the order of the polynomial based on the number of fluctuations in the data or by how many bends (hills and valleys) appear in the curve. For example, a trendline created by an Order 2 polynomial generally has only one hill or valley. Order 3 generally has one or two hills or valleys. Order 4 generally has up to three. Setting the order for a polynomial trendline too high when compared to the number of data points usually creates a trendline that oscillates wildly.

The polynomial option creates a polynomial, or curvilinear, trendline by using the following equation to calculate the least squares fit through points:

$$y = b + c_1x + c_2x^2 + c_3x^3 + \dots + c_6x^6$$

where  $b$  and  $c_1 \dots c_6$  are constants.

## About power trendlines

A power trendline appears as a curved line. Use a power trendline type to show data that increases at a specific rate. For example the acceleration of a race car at 1-second intervals. You cannot create a power trendline for data that contains zero or negative values.

The Power option creates a trendline by using the following equation to calculate the least squares fit through points:

$$y = cx^b$$

where  $c$  and  $b$  are constants.

## About exponential trendlines

An exponential trendline appears as a curved line. Use the exponential trendline type to show data values that rise or fall at an increasingly higher rate. An exponential trendline cannot depict data that contains zero or negative values.

The exponential option creates an exponential trendline by using the following equation to calculate the least squares fit through points:

$$y = ce^{bx}$$

where  $c$  and  $b$  are constants, and  $e$  is the base of the natural logarithm.

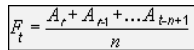
## About moving average trendlines

A moving average trendline smooths out fluctuations in a data set to show a pattern or trend more clearly. A moving average uses a specific number of data points (set by the Period option), averages them, and uses the average value as a point in the line. The number of points in a moving average trendline equals the total number of points in the series less the number you specify for the period.

For example, if you set Period to 2 for a moving average trendline, the trendline shows the average of the first two data points as the first point. The trendline shows the average of the second and third data points as the second point in the trendline, and so on.

If you add a moving average trendline to a scatter chart, BIRT Spreadsheet Designer calculates the moving average based on the order of the  $x$  values plotted in the chart. To show the trendline result you want, you might have to sort the  $x$  values before creating a moving average trendline.

The moving average option creates a trendline by using the equation shown in Figure 11-25.


$$F_t = \frac{A_t + A_{t-1} + \dots + A_{t-n+1}}{n}$$

**Figure 11-25** Moving average trendline equation

## Adding and modifying a trendline

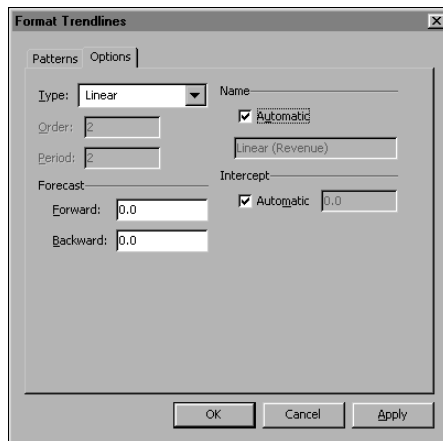
To create a trendline, right-click a data series, then choose Add Trendline.

To modify a trendline, right-click a trendline, then choose Format Trendline.

On Format Trendlines—Options, as shown in Figure 11-26, select options that describe a trend in the data range that links to the chart:

- In Type, select a type of trendline.
- For a polynomial trendline, in Order, type a number between 2 and 6. The value that you type in Order determines the order of magnitude for the polynomial that calculates the trendline.

- For a moving average trendline, in Period, type a number. The number that you type in Period determines the number of periods used to calculate an average.
- To extend the trendline forward or backward, under Forecast, in Forward or Backward, type a number. This number determines how many x-axis units to extend, or project, a forecast trendline. A category axis cannot project backward.
- In Name, BIRT Spreadsheet Designer supplies a display name for the trendline. The display name appears in the legend. To change the trendline name, under Name, deselect Automatic, then type text.
- Typically, a trendline intersects the y-axis at 0. To set the y-axis intersection for a trendline to a value other than 0, under Intercept, deselect Automatic, then type a value.



**Figure 11-26** Format Trendlines—Options



# 12

## **Designing a customizable report**

This chapter contains the following topics:

- About customizable reports
- Understanding parameter types
- Creating a parameter
- Providing a value for a parameter

---

## About customizable reports

A customizable report can display different information for different users. To create a customizable report design, you use a parameter. Parameters enable you to deliver information that a user can customize. For example, you can create a report design that requires a user to provide a state value, then generates a report that displays information for only that state. A targeted report displays information to which a user aims by providing a value.

Using parameters in a report design can help users manage reports that contain large data sets by enabling a user to display specific data. For example, using a sales report that contains nationwide data, a report user might want to view only sales in a single state, or sales over a certain amount, or sales completed in the last 30 days. When a user runs a parameterized report design, the parameter values he provides restrict or modify the data the report displays. Just one parameter can effectively limit or refine the information displayed in a report. You can use more than one parameter to provide a more specialized report.

Using parameters in a report provides the following benefits:

- **Targeted reports**  
Using parameters, a single report design can generate specialized reports, on demand, to meet the specific needs of each report user.
- **Flexible report designs**  
A report developer can design one report to suit different requirements. For example, a monthly sales report should isolate recent sales information. A report designer could create a query that requests data for January, 2006, but that query run in February no longer fetches the most recent information. If the report designer uses a parameter to filter the report information, she can deliver one report design that provides refreshed data each month.
- **Managed data in large reports**  
A spreadsheet report design can retrieve large amounts of data. For example, one report can display itemized sales orders for all customers in all cities in every country. To reduce the size of a generated report and let the user select what data to view, you can use parameters to generate only parts of the original large report.

BIRT Spreadsheet Designer offers many ways to set up parameters and request values for a report. A set of cascading parameters can request different information based on the first value a user provides. You can offer users a choice of values in a format such as a drop-down list, or require that they type an exact value. A report parameter can also accept partial values or a range of values. For example, in a report that displays order records, you can create a parameter that accepts Query By Example (QBE) syntax so that a user can supply >2100 to show order numbers over 2100.

You most commonly use a parameter to filter a data set field. You can also use security features to filter or narrow report output. For more information about using security features in spreadsheet reports, see Chapter 14, “Designing a secure report.”

---

## Understanding parameter types

You can create and use three types of parameters in your report design:

- **Ad hoc**  
An ad hoc parameter requests, but does not require, a value when a report executable file runs. For more information about using an ad hoc parameter, see “Using an ad hoc parameter,” in Chapter 4, “Connecting to a database or JDBC data source.”
- **Run time**  
A run-time parameter requests a value at run time, when you run a report executable file on a server using Actuate BIRT iServer or Actuate Information Console. The parameter filters the query in the report executable file and delivers the specified data in a report instance file. Users who open a report instance file see the data that the run-time parameters specified.
- **View time**  
A view-time parameter requests a value at view time, when a user opens a report instance file. A view-time parameter causes the query to which it is attached to run at view time, not at run time. Typically, a report instance file opens Excel to display a spreadsheet report. Each user that opens the instance file views a new copy. The report displays only data specified by the view-time parameter value.

### Using a run-time parameter

By default, a new parameter is a run-time parameter. A run-time parameter only requests input at run time. A run-time parameter is not available at view time. A user who opens a spreadsheet report from a report instance file cannot provide a value for a run-time parameter. Run-time parameters help you to reuse report designs, but they do not enable end users to specify the data to display.

### Using a view-time parameter

Use a view-time parameter when you want a report user to define what information he sees. To request input from a user at view time, when a user opens a view of a report instance file, you must create a view-time parameter.

### How to create a view-time parameter

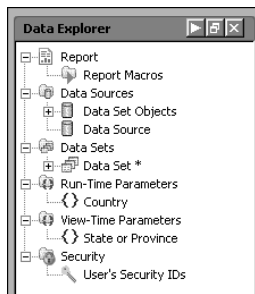
- 1 Select Report➤Create Report Parameter.
- 2 Provide required parameter attributes.  
For more information about required attributes, see “How to provide required parameter attributes,” later in this chapter.
- 3 On Report Parameter, in Display As, select Prompt for this parameter at view time.  
Choose OK.

## Using run-time and view-time parameters in a report

You can include both run-time and view-time parameters in a report design to provide customizable reports. When you use both, the view-time parameter controls report output.

For example, you can use a run-time parameter to create different report instances for each year, and add a view-time parameter to request an office location. A user who opens each yearly report provides a specific parameter that represents an office location. The report executable file generates customized, location-specific output based upon the view-time parameter that the user selects.

Data Explorer shows run-time and view-time parameters separately. For example, Figure 12-1 shows Data Explorer with two parameters.



**Figure 12-1** Data Explorer with run-time and view-time parameters

## Using a system parameter

You can also use system parameters that BIRT Spreadsheet Designer creates when you create a Data Source. You can assign a value to a system parameter in a report design. A system parameter requests a value at run time.



---

## Creating a parameter

When you add a parameter to a report, you specify the parameter name and data type, then define what type of values the parameter accepts. For example, you can enable a user to provide a QBE expression instead of a specific value.

You also determine how the user can provide a value. For example, you can set up a text box into which a user types data, or use a drop-down list from which a user selects a value. If you want to provide a list, you can type the list of values or specify a data set column from which the report executable file retrieves the values.

You can use prompt text to make parameters more comprehensible to a user. For example, you can display office names instead of office codes. You can also provide tooltip text that explains the purpose of a parameter.

To add a parameter, choose Report→Create Report Parameter. Then, use Report Parameter to select parameter attributes. Figure 12-2 shows the default attributes selected on Report Parameter.

The screenshot shows the 'Report Parameter' dialog box with the following settings:

- Properties:**
  - Name: NewParameter1
  - Prompt text: (empty)
  - Data type: String
  - Display type: Text Box
  - List of value: ☒ Static ☐ Dynamic
  - Default value: No Default Value
- Display As:**
  - Help Text: (empty)
  - List Limit: 1 values
  - ☐ Allow null value
  - ☐ Allow blank values
  - ☐ Do not echo input
  - ☐ Hidden
  - ☐ Sort alphabetically when prompting
  - ☐ Prompt for this parameter at view-time
- Create Defined Name:**
  - ☐ Name: (empty)
  - ☐ Hide In Excel

Buttons: OK, Cancel

**Figure 12-2** Examining parameter attributes

## Defining required attributes for a parameter

A parameter requires two attributes, a name and a data type. Optionally, you can add other attributes listed in Properties on Report Parameter dialog.

### Defining a parameter name

Each parameter in a report must have a unique name. BIRT Spreadsheet Designer provides default name text. You can change the default name to something more meaningful. To filter data, you refer to the parameter name in a data set or data range.

### Defining a parameter data type

To define the type of data a parameter filters, specify the parameter data type. A BIRT Spreadsheet Designer parameter can accept a static value, a QBE expression, or a formula as a parameter value. Users can provide a static value for any type of parameter, as long as that value fits the parameter type. For example, a currency parameter accepts numeric values only.

To enable users to provide formula values, a parameter must have the data type of formula. BIRT Spreadsheet Designer evaluates the formula, then uses the result value as the parameter value. For example, you can create a date-and-time parameter that accepts either of the following values:

- 06/22/06 returns data for the row with the date value June 22, 2006.
- >06/22/06 returns data for rows with date values after June 22, 2006.

You can create a formula parameter that accepts either of the following values:

- =today( ) returns data for rows with date values of today's date.
- =today( ) - 1 returns data for rows with date values of yesterday's date.

When you use formula-type parameters, instruct users how to provide values, using a tooltip or instruction in the report design.

Typically, BIRT Spreadsheet Designer treats a parameter value that begins with an equal sign as a formula. For example, if a user provides =06/20/06 as a value for a formula-type parameter, BIRT Spreadsheet Designer evaluates the value as 0.05, or 6 divided by 20 divided by 6.

To provide a value that begins with an equal sign, a user must precede the equal sign with a single quotation mark ('). For example, to provide the text =today( ) as a parameter value, a user should type:

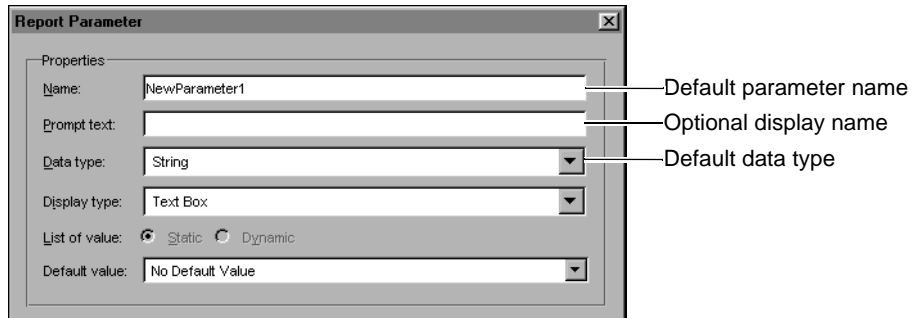
```
'=today( )
```

With this syntax, BIRT Spreadsheet Designer passes the literal text that begins with the equal sign. Without the single quotation mark, BIRT Spreadsheet Designer evaluates the formula to today's date, then uses that date as the parameter value.

### How to provide required parameter attributes

To add or edit attributes in Report Parameter, do the following steps:

- 1 In Name, type text that replaces the default parameter name or accept the default name text.
- 2 Optionally, in Prompt text, type text that appears in place of the parameter name on Requester page.
- 3 In Data Type, select a data type or accept the default data type, as shown in Figure 12-3.



**Figure 12-3** Providing attributes for a parameter

Choose OK.

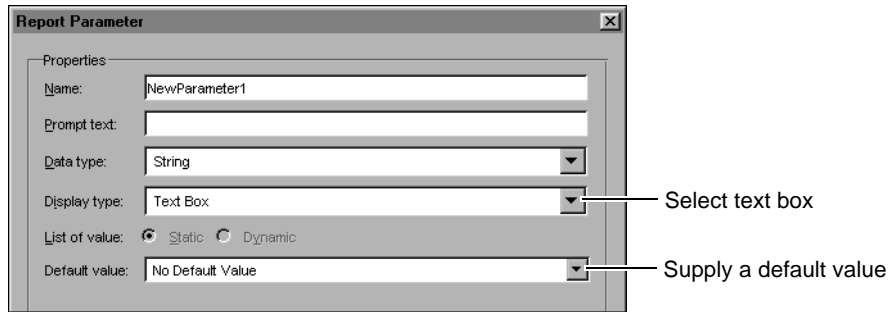
## Listing parameters on Requester Page

Requester Page presents to a report user the interface through which she enters a value for each report parameter. Typically, when a report executable file that contains parameters runs, Requester Page lists in alphabetical order the parameter names you create. A space in which the user types or selects a parameter value appears under each parameter name. Optionally, you can provide prompt text for a parameter name that displays instead of that parameter name on Requester Page.

You can use, a text box, a drop-down list, a combo box, or a list of radio buttons to request a parameter value from a user. Each display type requires different settings. After you define a set of values from which a user can choose, you define as the default selection one value from the set.

### Using a text box

To set up the simplest way to request a parameter value, use a text box. To use a text box, select Text Box in Display type, as shown in Figure 12-4. To provide a default value in the text box, type that value in Default Value.



**Figure 12-4** Text box options on Report Parameter

## Using a list

Drop-down lists, combo boxes, and sets of radio buttons each provide a different way to list items from which a user selects. A user can select only one item from a list. The following list compares and contrasts these user interface elements:

- **Drop-down list**  
A drop-down list saves space and visually distracts a user less than radio buttons because it uses only one line on Requester Page. A drop-down list requires no a default value. A user-friendly drop-down list contains fewer than 100 entries.
- **Combo list**  
A combo list looks and behaves like a drop-down list, except that a user can type a value that the list does not initially display.
- **Radio buttons**  
The radio button for each item available requires one line on Requester page. You must specify one choice as a default value. A user-friendly series of radio buttons contains fewer than 10 entries.

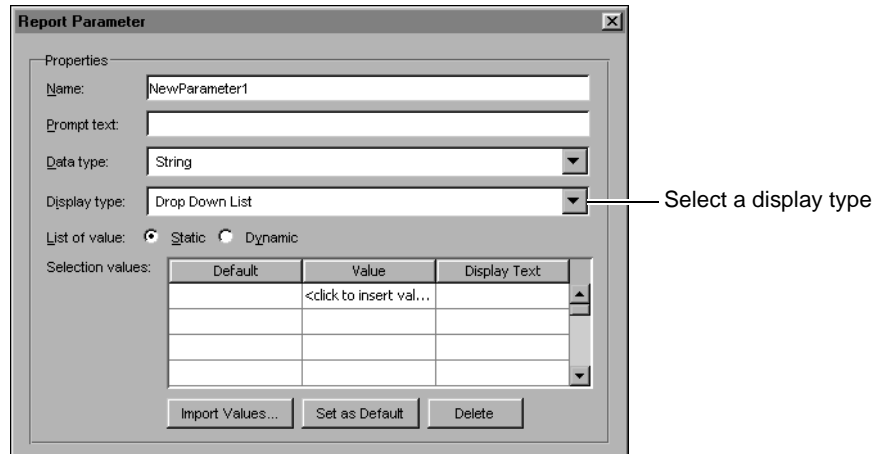
To create a list of values for a drop-down list or a series of radio buttons, you either type values, retrieve values from a data set, or link to a field in a data set. Retrieving from or linking to a data set enables a report to display current values. You can also determine whether Requester Page shows values in database order or sorted alphabetically.

### How to list on Requester Page values you provide

To provide a list of values that appear on Requester page as selections, choose the following options in Report Parameter:

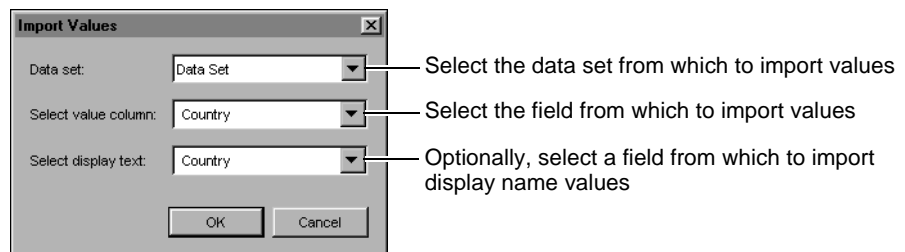
- 1 In Display Type, select Drop Down List, Combo Box, or Radio Button. The example in Figure 12-5, shows Drop Down List.

- 2 In List of Values, select Static. Report Parameter appears as shown in Figure 12-5.



**Figure 12-5** Selecting drop-down list as a display type

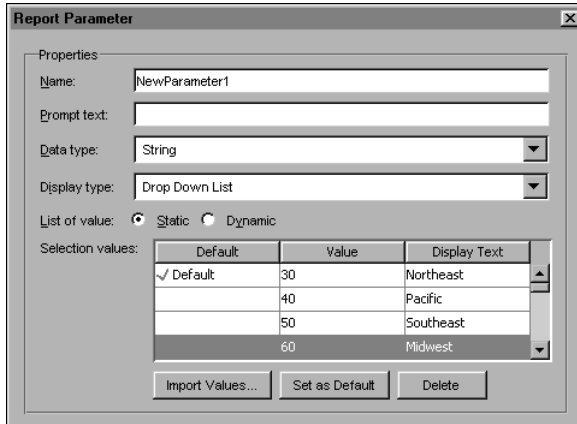
- 3 To type values, select the first cell in the Value column, then type the first value. Type a new value in each cell until you have provided all the values for the parameter.
- 4 To import values from a data set field, choose Import Values, then use the settings in Figure 12-6 to retrieve the values.



**Figure 12-6** Importing values for a parameter

- 5 To define a value as the default value, select the value, then choose Set as Default.
- 6 To use a display name for a value, type the name in Display Text.

For example, in Figure 12-7, Report Parameter shows a list of region values for a series of radio buttons. Each value uses a display name that is easier to understand than the region code that is the actual value. The first value, Northeast, is the default value on Requester Page.



**Figure 12-7** Report Parameter showing parameter values

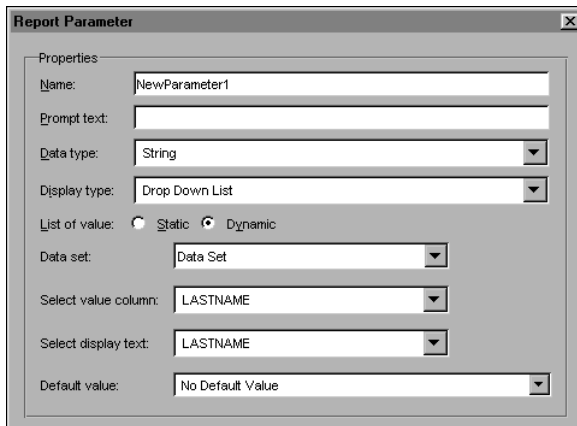
- 7 To show the list items in an alphabetically sorted list rather than in database order, in Display As, select Sort alphabetically when prompting.

When you finish defining the list of values, choose OK.

#### How to list on Requester Page values from a data set

To provide a dynamic list of values from a data set, that appear on Requester page as selections, choose the following options in Report Parameter:

- 1 In Display Type, select Drop Down List, Combo Box, or Radio Button. The example in Figure 12-8 shows Drop Down List as the display type.
- 2 In List of Values, select Dynamic. Report Parameter appears as shown in Figure 12-8.



**Figure 12-8** Using dynamic values for a drop-down list or series of radio buttons

- 3 Specify from which data set and column to retrieve the values:
    - In Data set, select a data set name.
    - In Select value column, select a data set field.
  - 4 To display a different column name as the list title, in Select display text, select the column.
  - 5 To specify a Default value, select the value in Default value.
  - 6 To show the list items in an alphabetically sorted list rather than in database order, in Display As, select Sort alphabetically when prompting.
- When you finish defining the list of values, choose OK.

### **Limiting how many parameter values Requester Page lists**

When you choose to dynamically generate the list of values that appear in a drop down or combo box list, you can limit the number of parameter values displayed for each parameter name.

#### **How to limit a dynamically generated list**

- 1 In Report Parameter, select dynamic list options.
- 2 After selecting drop down or combo box as a Display type and dynamic as List of value, type a number in List limit to the left of values.

Specify a data set for the list values, then choose OK.

### **Setting options for Requester Page users**

After defining required attributes for a parameter, you also use Report Parameter to choose options that control how users enter parameter values using Requester Page. For example, you can choose to:

- Allow null or empty values as parameter values.
- Hide characters that a user types.
- Prompt the user for parameter values using tooltip text.
- Create a defined name for a parameter.
- Hide a parameter on Requester Page.

## Allowing users to provide null or empty values

Typically, Requester Page does not accept a null or empty value for a parameter. You can specify that Requester Page accepts such values. To set Requester Page to allow a null or empty value for a parameter, on Report Parameter:

- To allow a user to provide a null value as a parameter value, select Allow null value.
- To allow a user to provide a blank, or empty value as a parameter value, select Allow blank value.

## Hiding characters that a user types

Typically, Requester Page displays the value a user types in a text box. You can hide these values. When you hide values, an asterisk appears in place of character that a user types.

To hide characters that a user types, on Report Parameter, select Do not echo input.

## Creating a tooltip

You can provide text for a tooltip that appears when a user hovers the cursor over a parameter name in Requester Page.

To provide text for a tooltip, on Report Parameter, in Help Text, type text that describes the displayed parameter name.

## Creating a defined name for a parameter

When you create a parameter, you can also create a defined name that retrieves the parameter value. You can use the default parameter name, a text string that exactly matches the parameter, or a different text string as a defined name. For example, if you want users to enter an interest rate to use when calculating mortgage totals, you can create a defined name called Rate. Use Prompt text to ask users for a number that represents the Rate parameter value, then use Rate in calculations.

### How to create a defined name for a parameter

- 1 In Report Parameter, in Create Defined Name, select Name.
- 2 Type a text string in place of the default defined name. To use the same string for the parameter and the defined name, accept the default text string.

## Hiding a parameter on Requester Page

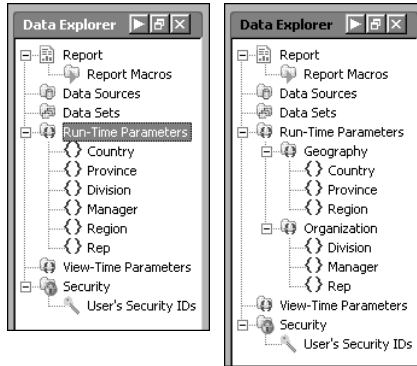
You can specify that a defined parameter not appear on Requester Page. To hide a parameter, on Report Parameter, select Hidden.



## Grouping parameters

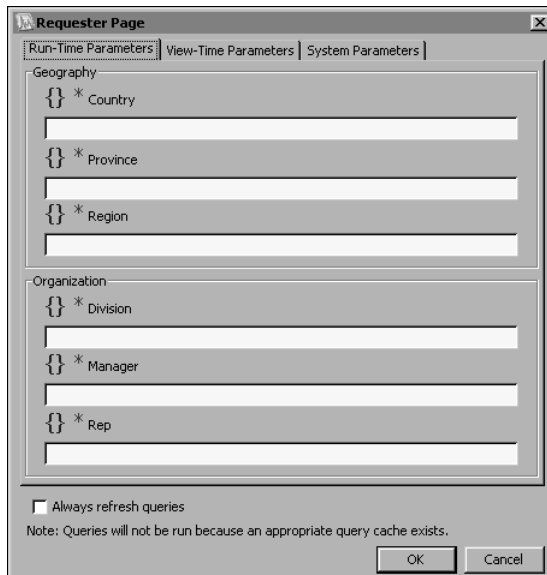
You can organize one list of many report parameters into logical groups. You create a parameter group in a report design using Data Explorer. When the report executable runs, parameter groups appear on Requester Page to help a user provide parameter values.

For example, Figure 12-9 shows two images of Data Explorer with six parameters. The image on the left displays report parameters in one list. The image on the right shows the same parameter list, grouped in two groups.



**Figure 12-9** Grouping parameters in one and two groups

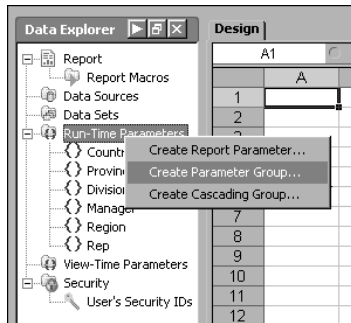
Figure 12-10 shows how two groups of parameters appear in Requester Page.



**Figure 12-10** Examining two groups of parameters on Requester Page

## Creating a parameter group

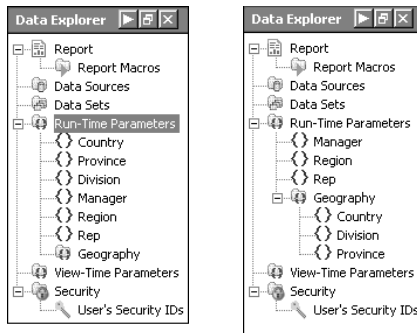
To create a parameter group, in Data Explorer, right-click Run-Time Parameters or View-Time parameters, then choose Create Parameter Group, as shown in Figure 12-11.



**Figure 12-11** Creating a parameter group

In Parameter Group, type a group name, then choose OK. Optionally, you can type a display name that replaces the group name on Requester Page.

A new group appears in the parameter list. To add an existing parameter to an existing group, drag the parameter name and drop it on the group name. For example, the image on the left in Figure 12-12 shows a new parameter group, Geography. The image on the right shows the result of dragging the Country, Region, and Province parameters and dropping them on Geography.



**Figure 12-12** Moving parameters into a group

### How to add a new parameter to an existing group

- 1 In Data Explorer, right-click a group name and choose Report Parameter.
- 2 In Parameter Group, type at least a parameter name and a data type, then choose OK.

## Creating a cascading parameter group

Cascading parameters are report parameters that have a hierarchical relationship. The top-level parameter name in a cascading parameter group should describe a set. Lower-level parameter names should describe subsets that appear as lower-level groups in the hierarchy. Table 12-1 shows an example hierarchy that contains three groups of cascading parameter names.

**Table 12-1** Example cascading parameter groups

Top-level name	Second-level name	Third-level name
Territory	Country	City
Mutual Fund Type	Fund Class	Fund
Product Type	Product	

In a cascading parameter group, each report parameter displays a set of values. A value that the report user selects for the top-level parameter determines what set of values that the next parameter displays, and so on.

Using cascading parameters can help you display fewer, more relevant parameter values to a user. For example, if a user selects North America as a territory parameter value, then the cascading parameter country displays only countries in the territory of North America. Similarly, if a user selects USA as a country parameter value, then the cascading parameter city displays only cities in the USA.

To contrast, if you create three independent parameters called territory, country, and city, then the territory parameter displays all territories, the country parameter displays all countries, and the city parameter displays all cities. The user must review three complete lists to select parameter values. Also, a user can inadvertently select an invalid combination of independent parameters, such as Japan, USA, and Paris.

### How to create a cascading parameter group

- 1 In Data Explorer, right-click Run-Time or View-Time Parameter and select Create Cascading Group.
- 2 On Parameter Group, type text in the following fields. Then, choose OK.
  - In Name, type text that identifies a parameter group.
  - Optionally, in Display Name, type text that replaces the group name on Requester Page.

---

## Providing a value for a parameter



Requester Page appears when either a report executable file or a report instance file that includes a parameter runs. To test a parameter or a parameter value during report development, select Report ➤ Run with Parameters. When you select Run with Parameters, Requester Page opens before the report executable file runs.

A report user must supply a value for any parameter listed on Requester Page that displays an asterisk. An asterisk does not display before an ad-hoc parameter. A user may optionally select, or enter a value for an ad-hoc parameter.

For example, Figure 12-13 compares how Run-Time Parameters on Requester Page prompts a report user to provide values for three run-time parameters, Amount, Start and End, and allows a null value for PortMgrID.

This example also shows parameter values that match the data type defined for the following parameters:

- Amount, with the numeric data type, accepts only numeric values.
- Start and End, with the date-and-time data type, accept only date-and-time values.

The screenshot shows a dialog box titled "Requester Page" with three tabs: "Run-Time Parameters", "View-Time Parameters", and "System Parameters". The "Run-Time Parameters" tab is active. It contains four parameter fields, each with a dropdown arrow. The first field is labeled "\* Amount" and has the value "30". The second field is labeled "\* Start" and has the value "10/1/2006". The third field is labeled "\* End" and has the value "12/1/2006". The fourth field is labeled "PortMgrID" and has the value "<null>". Below the fields is a checkbox labeled "Always refresh queries" which is checked. At the bottom, there is a note: "\* Parameter is required" and "Italicized parameters are hidden on the server." and buttons for "OK" and "Cancel".

**Figure 12-13** Requester Page—Run-Time Parameters

To provide a parameter value that begins with =, a user must type ' before typing =. For example, to create the parameter value =this value, type the following text:

'=this value

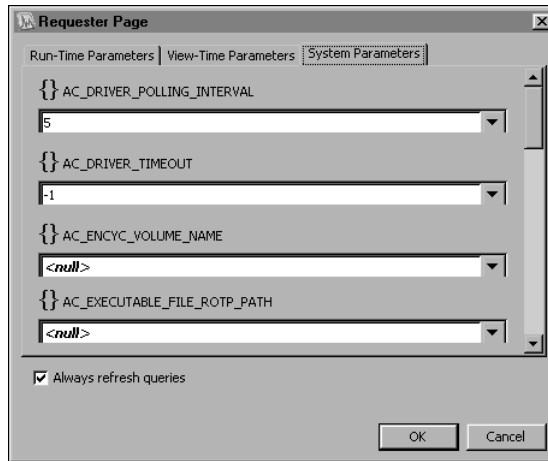
## Working with system parameters

To view and modify a system-level parameter value, you use System Parameters on Requester Page. Changes you make to system-level parameter values in a report design file do not affect BIRT iServer behavior. To change how BIRT iServer deploys a report, modify system parameters on the BIRT iServer.

### How to change the value of a system-level parameter



- 1 Choose Report➤Run with Parameters.
- 2 On Requester Page, choose System Parameters, then scroll to see a list of system parameters, as shown in Figure 12-14.



**Figure 12-14** System parameters on Requester Page

- 3 To change a parameter value, select a value from the combo-box list that appears under the parameter name, or type a new value in the combo box.
- 4 Assign values to all parameters next to which asterisks appear, then choose OK.

### Providing a report parameter value that uses locale-specific formatting

A query by example (QBE) expression can accept only US locale format. A single-value report parameter can accept non-US locale formats.



# 13

## Working with a pivot range

This chapter contains the following topics:

- About pivot ranges
- Creating a pivot range
- Working with pivot range appearance
- Working with pivot range data
- Printing a pivot range

## About pivot ranges

You use a pivot range to combine and compare data in a flexible format that is similar to an Excel pivot table. Rather than listing large amounts of data, a pivot range summarizes and arranges the data. You can easily modify the layout of a pivot range to show different sets of information. To compare data and see variance information, you can create subtotals, create custom calculations, and filter report fields.

To create a pivot range, you use the same types of data sources you use for other report ranges. You can also use a pivot range as a data source for a second pivot range.

The pivot range toolbar supports some pivot range tasks, such as refreshing the pivot range or changing the number format of a field. Figure 13-1 shows the pivot range toolbar. The pivot range toolbar appears only after you create a pivot range.



**Figure 13-1** Examining pivot range toolbar

## Examining a sample pivot range

The pivot range in Figure 13-2 displays order information for several sales representatives. The report groups the number of orders and the sales totals by quarter.

offices.officialD	(All)					
			shipByDate			
last	customName	Data	Qtr1	Qtr2	Qtr3	Grand Total
Barajas	Advanced Design Corp.	Number of orders	14		14	28
		Order totals	\$5,873,096	\$0	\$7,423,377	\$26,517,970
	Computer Engineering	Number of orders		9	4	13
		Order totals	\$0	\$2,540,202	\$2,408,400	\$10,429,073
	Design Boards Co.	Number of orders			14	14
		Order totals	\$0	\$0	\$3,288,407	\$3,288,407
	Design Systems	Number of orders			14	14
		Order totals	\$0	\$0	\$6,270,308	\$6,270,308
	Signal Engineering	Number of orders		7		7
		Order totals	\$0	\$9,891,714	\$0	\$9,891,714
Barajas	Technical Specialists Co.	Number of orders			31	31
		Order totals	\$0	\$0	\$33,958,925	\$33,958,925
Barajas	TekniSystems	Number of orders			1	1
		Order totals	\$0	\$0	\$476,850	\$476,850
Barajas Number of orders			14	16	78	108
Barajas Order totals			\$5,873,096	\$22,807,512	\$251,129,084	\$546,545,175
Castillo	Brittan Design Inc.	Number of orders	7			7
		Order totals	\$9,965,106	\$0	\$0	\$9,965,106
	Design Engineering Corp.	Number of orders		53		53
			\$0	\$50,979,872	\$0	\$50,979,872

**Figure 13-2** Examining a sample pivot range



You can modify the pivot range to show different data without designing a new report. You can make the following types of changes:

- Add or remove pivot range fields. For example, you can replace the customer name field with a customer credit rank field to show totals for customers of each rank.
- Filter pivot range data. For example, you can show data for customers who placed more than 10 orders.
- Group pivot range data. For example, you can show orders by month instead of quarter.
- Rearrange pivot range fields. For example, you can display the data fields, Number of orders and Order totals, horizontally rather than vertically.
- Recalculate pivot range totals. For example, you can change the function that displays the average number of items per order for each sales representative to one that displays the sum of revenue per order.

## Recognizing the parts of a pivot range

When you work with a pivot range, you use the following types of fields:

- Row field. A row field arranges data vertically in the pivot range. When you use multiple row fields, the row fields to the outside of the pivot range are outer row fields. The row field closest to the data is an inner row field.
- Column field. A column field arranges data horizontally in the pivot range.
- Page field. A page field can filter the entire pivot range. To filter the range, show one of the page field items. For example, you can use year as a page field to display data for all years or for selected years.
- Data field. A data field summarizes data from the data source and uses a summary function, such as sum or average.
- Total field. A subtotal displays total information for outer row or column items. A block total displays total information for all the items in an inner field. A grand total displays total information for the entire range.

You also work with these parts of a pivot range:

- Item. An item is a value in a field.
- Calculated field. A calculated field displays the value result of a formula you create. A calculated field can perform calculations using the contents of other pivot range fields.
- Calculated item. A calculated item uses a formula you create. A calculated item can perform calculations using the contents of other items in the same pivot range field.

Figure 13-3 shows how some parts of a pivot range appear in BIRT Spreadsheet Designer.

office.city	NYC				
Sum of OrderTotal	orders.category				
last	customName	1	2	3	Grand Total
Barajas	Computer Engineering	\$0	\$2,540,202	\$2,408,400	\$10,429,073
	Design Boards Co.	\$0	\$0	\$3,288,407	\$3,288,407
	Design Systems	\$0	\$0	\$6,270,308	\$6,270,308
	Technical Specialists Co.	\$3,801,654	\$1,612,524	\$3,684,499	\$33,958,925
	TekniSystems	\$0	\$0	\$476,850	\$476,850
Barajas Total		\$3,801,654	\$8,610,097	\$88,947,264	\$217,470,396
Hernandez	CompuSolutions Co.	\$0	\$0	\$49,939,929	\$49,939,929
	Computer Systems Corp.	\$0	\$0	\$3,588,980	\$3,588,980
	InfoDesign	\$0	\$0	\$1,612,524	\$1,612,524
	SigniDesign	\$0	\$0	\$87,312,904	\$87,312,904
	TekniSystems	\$0	\$0	\$12,186,894	\$12,186,894
	TeleMicroSystems	\$0	\$4,961,754	\$10,036,362	\$29,246,112
Hernandez Total		\$0	\$4,961,754	\$752,116,493	\$879,455,094
Thompson	InfoSpecialists	\$0	\$0	\$2,301,400	\$2,301,400
	Technical Specialists	\$0	\$0	\$3,432,504	\$3,432,504
Thompson Total		\$0	\$0	\$11,396,352	\$11,396,352
Tseng	Design Solutions Corp.	\$0	\$0	\$11,644,242	\$11,644,242
	InfoEngineering	\$0	\$10,079,152	\$21,673,668	\$104,609,430
	Technical Design Inc.	\$22,260,798	\$0	\$0	\$22,260,798
	Technical MicroSystems Inc.	\$0	\$0	\$1,140,963	\$1,140,963
Tseng Total		\$22,260,798	\$10,079,152	\$136,885,367	\$454,747,368
Vanauf	Exosoft Corp.	\$0	\$0	\$49,661,685	\$49,661,685
Vanauf Total		\$0	\$0	\$49,661,685	\$49,661,685
Grand Total		\$51,028,125	\$71,396,640	\$3,938,601,236	\$6,230,628,600

Figure 13-3 Basic parts of a pivot range

## Creating a pivot range

You use PivotRange Wizard to create a pivot range. First, you select the source of data to include in the pivot range. Next, you specify where to place the pivot range. After you create a pivot range, you add and arrange data fields in row, column, data, and page field areas of the pivot range.

### About sources of pivot range data

When you create a pivot range, you can use one of the following as a source of data:

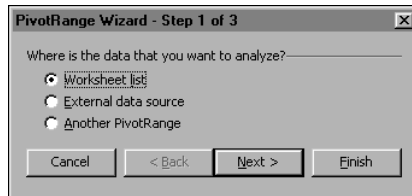
- **Worksheet range.** When you use a worksheet range as the source for pivot range data, the data and the pivot range can appear in different worksheets or workbooks. The data must appear in a contiguous list with no empty cells.
- **External data source.** You can use an ODBC or JDBC database, a text file, an SAP data source, an XML data source, an Actuate information object, or a

custom data source as the source for pivot range data. When you use an external data source, you must select an existing data set using PivotRange Wizard.

- Another pivot range. When you use an existing pivot range as the source for pivot range data, both pivot ranges link to the data source defined for the first pivot range. Updating the data in one range updates the associated data in the other range. Before you create two pivot ranges that use similar data, consider using one pivot range as a source for the other. Using a pivot range that depends on another as a data source requires less memory than creating two independent pivot ranges.

### How to create a pivot range

- 1 Select Data→PivotRange Wizard.
- 2 In PivotRange Wizard—Step 1 of 3, select a source type for the pivot range data, as shown in Figure 13-4. Then, choose Next.



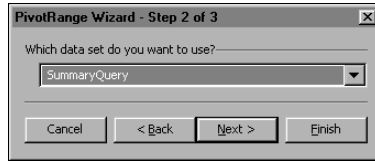
**Figure 13-4** Selecting a source type for pivot range data

- 3 In PivotRange Wizard—Step 2 of 3, perform one the following tasks, based on your selection in step 2:
  - If you selected Worksheet list, in Range, verify that the correct source data range appears in Range. If not, type a new range reference, as shown in Figure 13-5. Then, choose Next.



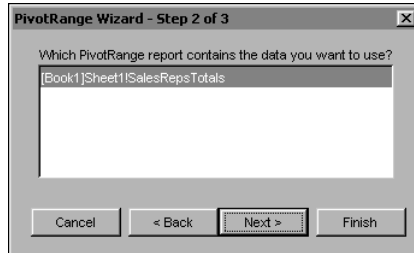
**Figure 13-5** Specifying a data range in a worksheet source

- If you selected External data source, select an available data set, as shown in Figure 13-6. Then, choose Next.



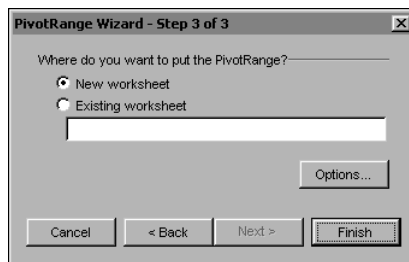
**Figure 13-6** Selecting an existing data set in an external data source

- If you selected Another pivot range, select an available pivot range, as shown in Figure 13-7, then choose Next.



**Figure 13-7** Selecting a pivot range as a data source

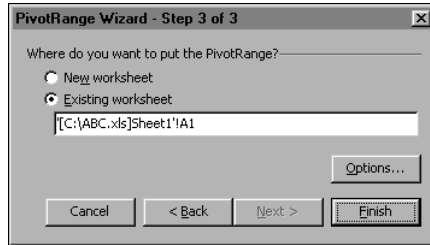
- 4 In PivotRange Wizard—Step 3 of 3, select a location for the pivot range, then choose Finish:
  - To place the pivot range on a new worksheet in the same workbook, select New worksheet, as shown in Figure 13-8, then choose Finish.



**Figure 13-8** Specifying a new worksheet location for a pivot range

- To place the pivot range on an existing worksheet, select Existing worksheet, type a valid cell reference for the starting cell of the range, then choose Finish. The existing worksheet must be open in BIRT Spreadsheet Designer. For example, to place the upper left corner of the new pivot range in cell A1 of Sheet 1 of a open workbook named ABC.xls, use the following reference, as shown in Figure 13-9:

```
'[C:\ABC.xls]Sheet1'!A1
```



**Figure 13-9** Creating a pivot range in an open, existing worksheet  
Next, you add data fields to the pivot range.

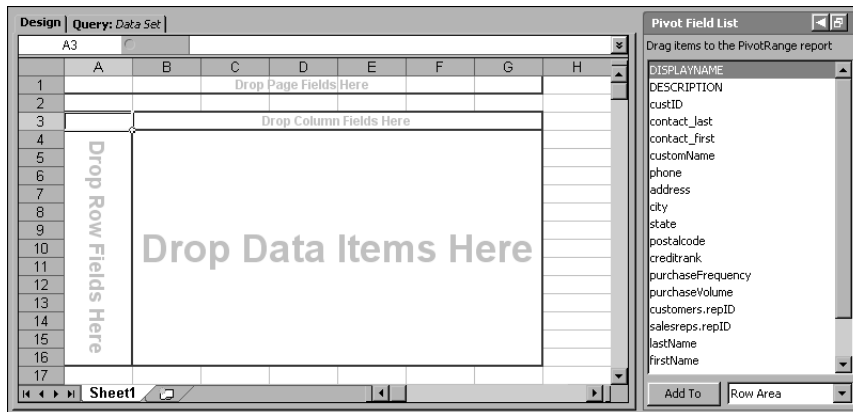
## Working with Pivot Field List

After you create a pivot range, Pivot Field List displays the fields from the selected data set that you can add to the pivot range. You add a data field to a pivot range by dragging a data field name from Pivot Field List and dropping it in the pivot range. You arrange the pivot range layout by dropping data field names in different areas of a pivot range.

If two database fields share the same field name, Pivot Field List displays the table name to identify the fields. If two worksheet fields share the same name, Pivot Field List uses a number suffix to identify the fields. For example, if a worksheet source includes two columns named Sales, the second Sales column appears as Sales2 in Pivot Field List.






You must select a cell in the pivot range to see Pivot Field List. If Pivot Field List does not display when you select a cell in a pivot range, select Pivot Range toolbar, then choose Show Field List. In Figure 13-10, Pivot Field List displays fields from the customers and items tables from the Sfddata sample database and a pivot range with cell A3 selected.



**Figure 13-10** Examining a new pivot range with Pivot Field List

## How to move Pivot Field List

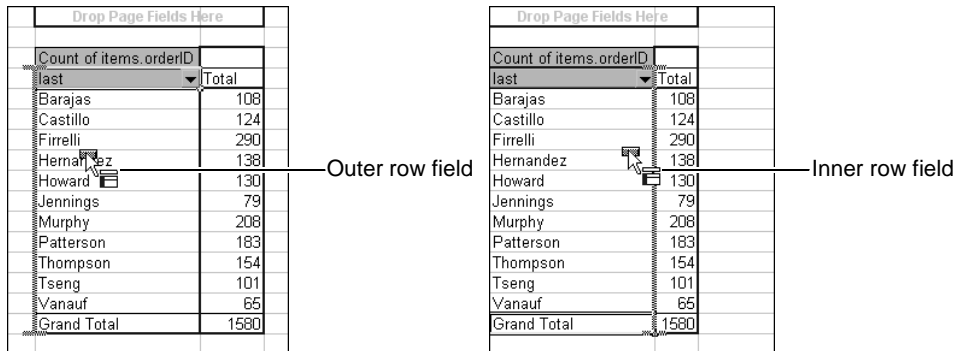
Typically, Pivot Field list appears docked and located to the right of Design tab. To move Pivot Field List, choose one of following options:

-  ■ To move the field list left or right of Design tab, choose the arrow in Pivot Field List title bar.
-  ■ To undock Pivot Field List, select maximize.
-  ■ To dock Pivot Field List next to Design tab, select minimize.

## Adding data fields to a pivot range

A pivot range displays a data field that you drop in the row area of the pivot range as a pivot range row field. A pivot range displays a data field that you drop in the column area of the pivot range as a pivot range column field. A pivot range displays a data field that you drop in the pivot range page area as a pivot range page field. You can add multiple row, column, and page fields to a pivot range.

You must add at least one row or column field and at least one data field to the pivot range. When you use multiple row or column fields, you can position a field as an outer or an inner field. A field dropped to the left of an existing row field appears as an outer row field. A field dropped to the right of an existing row field appears as an inner row field. Figure 13-11 shows how adding an outer row field differs from adding an inner row field.



Count of items.orderID	
last	Total
Barajas	108
Castillo	124
Firrelli	290
Hernandez	138
Howard	130
Jennings	79
Murphy	208
Patterson	183
Thompson	154
Tseng	101
Vanauf	65
Grand Total	1580

Outer row field

Count of items.orderID	
last	Total
Barajas	108
Castillo	124
Firrelli	290
Hernandez	138
Howard	130
Jennings	79
Murphy	208
Patterson	183
Thompson	154
Tseng	101
Vanauf	65
Grand Total	1580

Inner row field

**Figure 13-11** Adding outer and inner row fields

A pivot range displays a data field as a summary field. When you first select a data field, the field uses one of the following default functions:

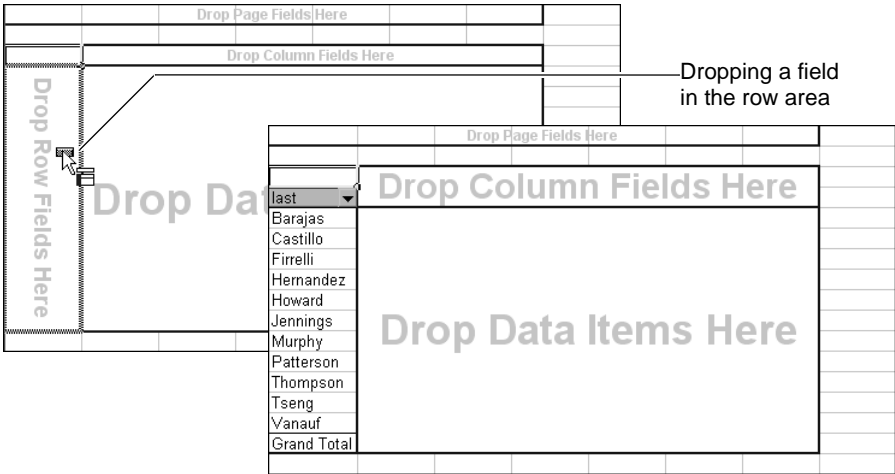
- The default function for numeric data fields is SUM.
- The default function for non-numeric fields is COUNT.

You can change the function for a data field. You can also add a calculated field or a calculated item to show additional data.

**How to add a field to a pivot range**

- 1 To add a row field to the pivot range, drag a field from the Pivot Field List, and drop it in the row field area in the pivot range.

As you drag the field, the pointer displays a shaded block. The pivot range frame highlights the current pivot range area, as shown in Figure 13-12. After you drop a field in the row area, the row area displays field data. For example, in Figure 13-12, the row area displays data for salesreps.last field.



**Figure 13-12** Adding a field to the pivot range row area

- 2 To add a data field to the pivot range, drag a field from the Pivot Field List, and drop it in the pivot range data area. After you drop a field in the data area, the data area displays summary field data. For example, in Figure 13-13, the data area displays the number of customers for each sales representative.

Drop Page Fields Here	
Count of customName	
last	Total
Barajas	108
Castillo	124
Firrelli	290
Hernandez	138
Howard	130
Jennings	79
Murphy	208
Patterson	183
Thompson	154
Tseng	101
Vanauf	65
Grand Total	1580

**Figure 13-13** Data in a pivot range data field

- 3 To add a page or column field, or use additional row or data fields, drag a field from the Pivot Field List and drop it in the appropriate area of the pivot range.

## Improving performance when adding fields to a pivot range

A pivot range typically displays a large amount of data. If each data field includes many items, adding fields to the pivot range consumes time as the pivot range refreshes each time you add a field to a row, column, page or data area. To improve performance when you create a pivot range, add row and column fields before you add data fields.



To prevent BIRT Spreadsheet Designer from refreshing the pivot range while you set it up, deselect Always Display Items on the PivotRange toolbar. When you deselect Always Display Items, BIRT Spreadsheet Designer does not populate the pivot range until you add a field to the data area of the pivot range. When you create a pivot range, Always Display Items is selected by default.

## Moving a field in a pivot range

You can rearrange the layout of a pivot range by moving a field in the pivot range. To move a field, drag the field and drop it in a new position in the row, column, page, or data area. A pivot range supports moving a field:

- Within one pivot range area that contains multiple fields
- From one pivot range area to another

To remove a field from a pivot range, drag it from the pivot range and drop it in the Pivot Field List.

## Changing the location of a pivot range

You can change the location where an existing pivot range appears.

### How to change the location where a pivot range appears

To change the location in a spreadsheet where a pivot range appears, BIRT Spreadsheet Designer supports Pivot Range Wizard and cut-and-paste functions.

- 1 Select a cell in an existing pivot range, then choose PivotRange➤PivotRange Wizard.
- 2 On PivotRange Wizard—Step 3 of 3:
  - To place the pivot range on a new worksheet, select New worksheet, or
  - To place the pivot range on an open, existing worksheet, select Existing worksheet, then type a valid cell reference for the starting cell of the range
- 3 Choose Finish.

Alternatively, select the pivot range, then use cut-and-paste to reposition it.



---

## Working with pivot range appearance

Item name, format and layout attributes affect pivot range appearance. You can change one or more of these attributes of a pivot range with no effect on the source of pivot range data. Changing pivot range appearance can more clearly focus the view of data that a pivot range provides in a spreadsheet report. For example, you can rearrange page fields or move a field from one pivot range area to another. You cannot delete the row, column, or data areas of a pivot range. You can hide pivot range data fields and items or show the details that they describe.

You can independently adjust specific aspects of pivot range layout as described throughout this section. Alternatively, you can use a formatting template to apply a set of layout and formatting attributes to a pivot range.

### Renaming a pivot range item

When you create a pivot range, BIRT Spreadsheet Designer gives the range a default name. You can rename a pivot range, a pivot range field, or a pivot range item to better identify data in a report. For example, you can rename the outer row field item "countofItems.OrderId" to "# of Items for Order". You must use a name that is different from any other field in the report and from any field in the pivot range data source.

To rename a pivot range, select a cell in the pivot range and choose PivotRange→Table Options. On PivotRange Options, in Name, type a name for the pivot range. To rename a field or item in a pivot range, select the field or item cell, then type the new name.

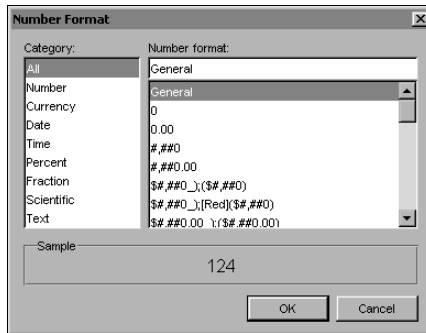
### Formatting a pivot range field

Typically, BIRT Spreadsheet Designer retains the formatting changes you make to a pivot range field. You can change this setting so the fields revert to default formatting when you refresh the pivot range or rearrange field locations. BIRT Spreadsheet Designer does not retain conditional formatting or cell borders when you refresh or rearrange a pivot range. You must reapply conditional formatting to the new field or range.

To change the format of a numeric field, change the field settings. Changing the format of a pivot range field using field settings ensures that each item in the field retains those format settings after you refresh or rearrange the pivot range.

#### How to change the number format of a field

- 1 Select an item in the field to change and choose PivotRange→Field Settings.
- 2 In PivotRange Field, choose All to see number formatting options, as shown in Figure 13-14.



**Figure 13-14** Examining all format options for a pivot range

- 3 In Number Format, select a category and number format, then choose OK.
- 4 In PivotRange Field, choose OK.

## Using tabular or outline layout

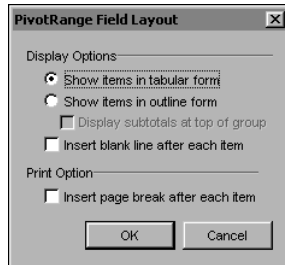
The default layout for a pivot range displays lines between each pivot range element and totals below each outer row field. The default formatting appears as a tabular format. You can change the pivot range layout to display as an outline, with fewer lines. When you display a pivot range as an outline, you can also choose to show subtotals above each outer row field. You use Field Settings to change the formatting of an outer row field from tabular to outline layout. Tabular layout displays field boundaries and looks similar to a spreadsheet. Outline layout uses space to separate inner and outer fields. Figure 13-15 shows the difference between tabular layout and outline layout.

Tabular layout						Outline layout					
Number of orders		shipByDate				Number of orders		shipByDate			
offices.city	last	Qtr1	Qtr2	Qtr3	Grand Total	offices.city	last	Qtr1	Qtr2	Qtr3	Grand Total
Boston	Castillo	16	56	52	124	Boston	Castillo	16	56	52	124
	Firrelli	21	15	42	78		Firrelli	21	15	42	78
	Murphy	146	28	28	202		Murphy	146	28	28	202
	Patterson	75	12	20	107		Patterson	75	12	20	107
	Thompson	26	59	44	129		Thompson	26	59	44	129
Boston Total		284	170	186	640	Boston Total		284	170	186	640
NYC	Barajas	14	16	78	108	NYC	Barajas	14	16	78	108
	Hernandez	89	47	2	138		Hernandez	89	47	2	138
	Thompson	3		22	25		Thompson	3		22	25
	Tseng	75	17	9	101		Tseng	75	17	9	101
	Vanauf		44	21	65		Vanauf		44	21	65
NYC Total		181	124	132	437	NYC Total		181	124	132	437
Philadelphia	Firrelli	188		24	212	Philadelphia	Firrelli	188		24	212
	Howard	114		16	130		Howard	114		16	130
	Jennings	29	26	24	79		Jennings	29	26	24	79
	Patterson	48		28	76		Patterson	48		28	76
Philadelphia Total		379	26	92	497	Philadelphia Total		379	26	92	497
Grand Total		844	320	410	1574	Grand Total		844	320	410	1574

**Figure 13-15** Tabular and outline layout of a pivot range

### How to apply a tabular or outline layout to an outer row field

- 1 Select an outer row field, then choose PivotRange→Field Settings.
- 2 In PivotRange Field, choose Layout.
- 3 In PivotRange Field Layout, under Display Options, specify the layout settings for the outer row field, as shown in Figure 13-16:



**Figure 13-16** PivotRange Field Layout

- To display lines between each element, select Show items in tabular form.
  - To display spaces between each element, select Show items in outline form.
  - To show subtotals at the top of the subtotaled group in an indented layout, select Display subtotals at top of group.
  - To display a blank line after outer field items, select Insert blank line after each item.
- 4 Under Print option, to create a separate page for each item, select Insert page break after each item.

Choose OK.

## Formatting a pivot range

To format an entire pivot range, use a format template. A format template applies a set of format attributes such as bold, shading, font size and borders to specific pivot range items. For example, the Classic template applies border lines to the page area, outer row, and total items. A new pivot range has the Classic template applied by default. Each of the format templates available in BIRT Spreadsheet Designer differs from the default, Classic template. For example, to add bold and shading attributes to column headings, use the Table 10 template.

### How to apply a format template to a pivot range.

- 1 Select a cell in the pivot range.
- 2 Choose PivotRange→Format Report.

- 3 In AutoFormat, scroll and compare examples of available templates, then select a template. To remove all formatting from a pivot range, select None. Then, choose OK.

## Preserving pivot range formatting

By default, a refreshed pivot range retains the formatting you apply using Field Settings. You can set an option that reverts a pivot range to default formatting when you refresh the pivot range.

### How to revert to default formatting when you refresh data

- 1 Select a cell in the pivot range and choose PivotRange→Table Options.
- 2 In PivotRange Options, deselect Preserve formatting, then choose OK.

## Working with pivot range layout

You can change the layout of a pivot range area by moving labels and reorganizing field items. For example, you can adjust the position of labels for row and column fields, show page fields in columns or rows, or arrange data fields horizontally or vertically. The pivot range page area layout affects how data displays in the other areas of a pivot range. You can change the layout to examine the detailed data that contributes to a data field.

### Arranging page field layout

By default, a pivot range arranges page fields vertically, in a column, as shown in Figure 13-17.

creditrnk	(All)	
status	(All)	
Count of items.orderID		
last	Total	
Barajas	108	
Castillo	124	
Firrelli	290	
Hernandez	138	
Howard	130	
Jennings	79	
Murphy	208	
Patterson	183	
Thompson	154	
Tseng	101	
Vanauf	65	
Grand Total	1580	

Page fields appear in a column

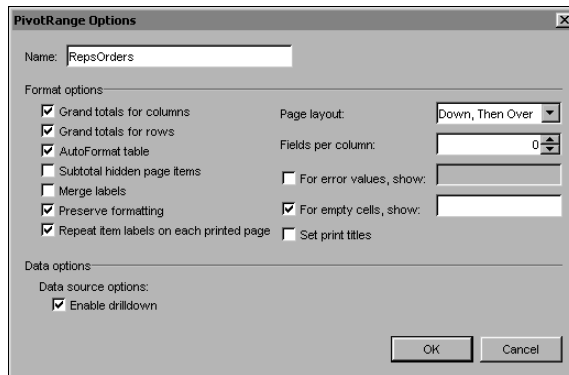
**Figure 13-17** Page fields arranged in a column

You can change the page field layout to display page fields horizontally, in rows or vertically, in columns. You can also set the number of page fields to display in each row or column.

### How to change field layout in the page area

- 1 Select a field in the pivot range, then choose PivotRange➤Table Options.
- 2 In PivotRange Options, as shown in Figure 13-18, specify page field layout:
  - To arrange page fields vertically from top to bottom:
    - In Page layout, select Down, Then Over.
    - In Fields per column, type the number of fields to display in each column.
  - To arrange page fields horizontally from left to right:
    - In Page layout, select Over, Then Down.
    - In Fields per row, type the number of fields to display in each row.

When you finish setting the pivot range options, choose OK.



**Figure 13-18** PivotRange Options

### Adding and deleting the page area in a pivot range

The page area can contain no, one, or more than one data field. To delete an empty page area from a pivot range, delete the row in which the empty page area appears. You cannot delete a page area containing a data field from a pivot range.

#### How to add a page area to a pivot range

To add a page area to a pivot range from which the page area has been removed, you must also add a field.

- 1 In Pivot Field List, select a field name to add.
- 2 Using the drop-down list at the bottom of the Pivot Field List, select Page Area. Then, choose Add To.

## Arranging data fields

When you add another field to the data area, the pivot range displays the data fields vertically, in one column in the data area, as shown in Figure 13-19.

last	Data	Total
Barajas	Count of items.orderID	108
	Sum of OrderTotal	\$546,545,175
Castillo	Count of items.orderID	124
	Sum of OrderTotal	\$756,382,814
Firrelli	Count of items.orderID	290
	Sum of OrderTotal	\$2,283,710,135
Hernandez	Count of items.orderID	138
	Sum of OrderTotal	\$1,287,314,505
Howard	Count of items.orderID	130
	Sum of OrderTotal	\$731,402,458
Jennings	Count of items.orderID	79
	Sum of OrderTotal	\$354,579,362
Murphy	Count of items.orderID	208
	Sum of OrderTotal	\$1,656,776,963
Patterson	Count of items.orderID	183
	Sum of OrderTotal	\$1,847,951,238
Thompson	Count of items.orderID	154
	Sum of OrderTotal	\$930,670,806
Tseng	Count of items.orderID	101
	Sum of OrderTotal	\$535,148,780
Total Count of items.orderID		1580
Total Sum of OrderTotal		\$107,984,553,483

Data fields in a vertical list for each row field

**Figure 13-19** Data fields arranged in one column

You can change the pivot range layout so that data fields appear horizontally, as separate columns in the data area, as shown in Figure 13-20.

	Data	
last	Count of items.orderID	Sum of OrderTotal
Barajas	108	\$546,545,175
Castillo	124	\$756,382,814
Firrelli	290	\$2,283,710,135
Hernandez	138	\$1,287,314,505
Howard	130	\$731,402,458
Jennings	79	\$354,579,362
Murphy	208	\$1,656,776,963
Patterson	183	\$1,847,951,238
Thompson	154	\$930,670,806
Tseng	101	\$535,148,780
Vanauf	65	\$49,661,685
Grand Total	1580	\$108,034,215,168

**Figure 13-20** Data fields arranged in separate columns

### How to change data field layout

To show data fields horizontally, drag a data field from a data area column and drop it in a cell at the upper right corner of the data area, as shown in Figure 13-21.

To show data fields vertically, drag a data field from a data area column and drop it in a cell at the left side of the data area, as shown in Figure 13-22.

last	Data	Total
Barajas	Count of items.orderID	108
	Sum of OrderTotal	\$546,545,175
Castillo	Count of items.orderID	124
	Sum of OrderTotal	\$756,382,814
Firrelli	Count of items.orderID	290
	Sum of OrderTotal	\$2,283,710,135
Hernandez	Count of items.orderID	138
	Sum of OrderTotal	\$1,287,314,505

To position data horizontally, move a data field to the right of the data area

**Figure 13-21** Arranging data fields horizontally, in separate columns

last	Count of items.orderID	Sum of OrderTotal
Barajas	108	\$546,545,175
Castillo	124	\$756,382,814
Firrelli	290	\$2,283,710,135
Hernandez	138	\$1,287,314,505
Howard	130	\$731,402,458
Jennings	79	\$354,579,362
Murphy	208	\$1,656,776,963
Patterson	183	\$1,847,951,238
Thompson	154	\$930,670,806
Tseng	101	\$535,148,780
Vanauf	65	\$49,661,685
Grand Total	1580	\$108,034,215,168

To position data vertically, move a data field to the left of the data area

**Figure 13-22** Arranging data fields vertically, in one column

## Arranging items in a pivot range field

By default, a pivot range displays sorted data. When you update a pivot range, new items appear at the end of the range. To rearrange the items in a pivot range field, select an item and choose one of the following options:

- To move the item to the beginning of the field, choose PivotRange→Order→Move to beginning.
- To move the item one place left, choose PivotRange→Order→Move left.
- To move the item one place right, choose PivotRange→Order→Move right.
- To move the item one place up, choose PivotRange→Order→Move up.
- To move the item one place down, choose PivotRange→Order→Move down.
- To move the item to the end of the field, choose PivotRange→Order→Move to end.

## Centering labels for row and column fields

Typically, labels for row and column fields appear left-aligned in heading cells. You can merge labels so they appear centered, vertically and horizontally, with respect to the data that they describe. To make labels in a pivot range appear centered, choose PivotRange→Table Options, then choose Merge labels.

## Hiding a pivot range field

To prevent display of a data field in a pivot range, you can hide that field.

To hide a field that appears in the row, column, page, or data area of a pivot range, right-click the field, then choose Field Settings. On PivotRange Field, choose Hide.

## Hiding a pivot range item

You can hide one or more items in a pivot range field. To hide one item in a row or column field, right-click the item, then choose Hide.

### How to hide multiple items in a pivot range field

- 1 Select the arrow at the right of the field. A list shows all available field items and whether each item is selected. For example, Figure 13-23 shows all items selected.



**Figure 13-23** Showing all items in a pivot range row or column field

- 2 For each item, make one of the following selections to set the items that display in the pivot range:
  - To show an item, select the checkbox.
  - To hide an item, deselect the checkbox.
  - To show all items, select Show All.

Choose OK.

## Hiding an item in the pivot range page area

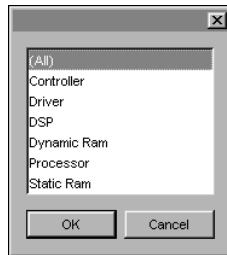
The following procedures describe how to show or hide items in the pivot range page area. You can show all items, show one item, or show multiple items.

### How to show all items or one item in a page field

- 1 Select the arrow to the right of the page field name.



BIRT Spreadsheet Designer displays a list of the page field items. For example, Figure 13-24 shows the items for an order category page field.



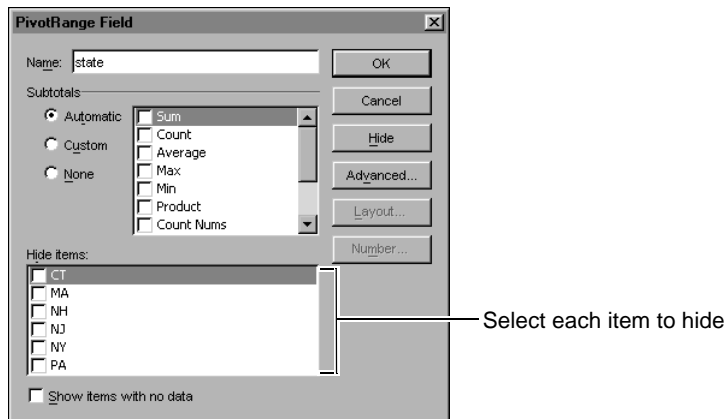
**Figure 13-24** Items in a page field

- 2 Select the items to display:
  - To display all items, select All.
  - To display one item, select the item.

Choose OK.

#### How to hide items in a page field

- 1 Right-click the field, then choose Field Settings.
- 2 On PivotRange Field, in Hide Items, select each item to hide, as shown in Figure 13-25, then choose OK.



**Figure 13-25** Hiding items in a page field

## Hiding inner field details

You can show or hide inner fields that contribute to outer fields and totals. You can also show field details to display data that contributes to a summary field.

## Consolidating a pivot range view

To consolidate a pivot range view, make it appear smaller by hiding inner field items. Hiding inner field items also hides the subtotals that contribute to the total for an outer item. For example, Figure 13-26 shows how hiding the names displayed in an inner field called last also hides the subtotals that contribute to the total calculated for the outer field item called Boston.

Count of items.orderID		
offices.city	last	Total
Boston	Castillo	124
	Firrelli	78
	Murphy	208
	Patterson	107
	Thompson	129
Boston Total		646
NYC	Barajas	108
	Hernandez	138
	Thompson	25
	Tseng	101
	Vanauf	65
NYC Total		437
Philadelphia	Firrelli	212
	Howard	130
	Jennings	79
	Patterson	76
Philadelphia Total		497
Grand Total		1580

Count of items.orderID		
offices.city	last	Total
Boston		646
NYC	Barajas	108
	Hernandez	138
	Thompson	25
	Tseng	101
	Vanauf	65
NYC Total		437
Philadelphia	Firrelli	212
	Howard	130
	Jennings	79
	Patterson	76
Philadelphia Total		497
Grand Total		1580

**Figure 13-26** Hiding inner field items

To change whether inner items display, select an outer field item, then choose one of the following options:

- To hide the inner field items associated with the outer field item, choose PivotRange→Group and Show Detail→Hide Detail.
- To show the inner field items associated with the outer field item, choose PivotRange→Group and Show Detail→Show Detail.

## Expanding a pivot range view

To expand a pivot range view, make it appear larger by showing details for inner field items. Doing this also shows subtotal values that contribute to other item totals. For example, the pivot range shown in Figure 13-27 displays details of the inner field customerName. The pivot range also displays details of one row from the inner field last. Expanding the Firelli row of last reveals not only the customer names but also the number of orders that contribute to the Firelli order total.

To show detail for an inner item, select the item and choose PivotRange→Group and Show Detail→Show Detail. On Show Detail, select from the list a field name that contains the detail you want to show, like the example shown in Figure 13-28, then choose OK.

Count of items.orderID			
offices.city	last	customName	Total
Boston	Castillo		124
	Firrelli	Advanced Solutions	6
		Computer MicroSystems Corp.	20
		InfoEngineering Corp.	5
		Signal Systems	15
		Technical Specialists Corp.	17
		TeleMicroSystems Inc.	15
	Firrelli Total		78
	Murphy		208
	Patterson		107
	Thompson		129
Boston Total			646
NYC	Barajas		108
	Hernandez		138
	Thompson		25
	Tseng		101
	Vanauf		65
NYC Total			437
Philadelphia	Firrelli	Advanced Engineering Inc.	108
		CompuBoards	61
		InfoBoards	8
		Technical Design Corp.	8
		TekniMicroSystems Co.	27
	Firrelli Total		212
	Howard		130
	Jennings		79
	Patterson		76
Philadelphia Total			497
Grand Total			1580

The pivot range displays detailed data for the Firrelli item

**Figure 13-27** Showing detail data



**Figure 13-28** Show Detail

### How to hide detail for an item

Select the item and choose PivotRange→Group and Show Detail→Hide Detail. The detail disappears, but the item name remains in the pivot range. To prevent display of the item name, drag it from the pivot range and drop it in the Pivot Field List.

## Showing detail for a data field

To see all the values that contribute to a data field in a pivot range, right-click the field, then choose Group and Show Detail → Show Detail. A new worksheet appears in the workbook. The new worksheet lists the details for the selected field.

Typically, a pivot range allows a report user to display data field details in a separate worksheet. To prevent report users from seeing the data that contributes to a data field, select the field, then choose PivotRange → Table Options. On PivotRange Options, deselect Enable drill to details, then choose OK.

---

## Working with pivot range data

You can update, or refresh, a pivot range to show current values from the selected data set. You can also select a different source for pivot range data. You can examine or modify a pivot range calculation. To change how data appears in a pivot range report, you can group, sort, and filter pivot range data. You can also use pivot range data outside the pivot range.

### Updating pivot range data

When you run a report that contains a pivot range, the pivot range refreshes so that it displays current values from the selected data set. You can also manually refresh a pivot range without running a report. When you use one pivot range as the data source for another pivot range, refreshing one pivot range refreshes both ranges.

#### How to refresh pivot range data



- 1 Select a cell in the pivot range.
- 2 Choose one of the following options:
  - On the PivotRange toolbar:
    - Select the Refresh Data icon.
    - Select PivotRange, then choose Refresh.
  - Right-click, then choose Refresh.

### Changing the source for pivot range data.

You can change the source for data displayed by an existing pivot range.

#### How to change the source for pivot range data

- 1 Select a cell in an existing pivot range, then choose PivotRange → PivotRange Wizard.

- 2 On PivotRange Wizard—Step 3 of 3, choose Back.
- 3 On PivotRange Wizard—Step 2 of 3, type a different range or select a different data set, then choose Next.
- 4 On PivotRange Wizard—Step 3 of 3, choose Finish.

## Working with empty fields and error values

Typically, pivot range cells that contain fields with no data or error values display blank. You can set a pivot range option that causes the pivot range to display text that you specify in place of empty fields or error values.

For each row, column, or page field, you can choose to hide or display items without data. For example, in a report that displays weekly order totals, you can either hide or display a row that displays a week without orders.

### How to set text that indicates empty fields or error values

- 1 Select a cell in the pivot range and choose PivotRange→Table Options.
- 2 On PivotRange Options, specify error or empty cell text:
  - To set error value text, select For error values, show, then type a text string.
  - To set empty cell text, select For empty cells, show, then type a text string.Choose OK.

### How to display row, column, or page field items with no data

- 1 Select a pivot range field and choose PivotRange→Field Settings.
- 2 On PivotRange Field, select Show items with no data.  
Choose OK.

## Working with pivot range calculations

A pivot range uses calculations in data fields, calculated fields, calculated items, and totals. You can change the expression a calculation uses or create a custom calculation. You can also add or modify subtotals and choose to include or exclude hidden items in subtotals.

To evaluate calculations, you can list all the calculations in a pivot range on a separate worksheet. You can also rearrange the solve order of the formulas or evaluate the sort order each formula follows.

### About creating a pivot range calculation

When you create a calculation in a pivot range, use the following guidelines:

- Do not include a cell reference, a range reference, a sheet name, or a range name. To refer to pivot range data, use an absolute, relative, or name reference

as described in “Referring to pivot range data from inside the pivot range,” later in this chapter.

- Do not refer to more than one cell.
- Do not use a variable function, such as NOW or RAND.
- BIRT Spreadsheet Designer interprets all names in a pivot range formula as field or item names. To use a name that begins with a special character, such as \$ or %, surround the name with single quotation marks.

## Working with a calculated field or a calculated item

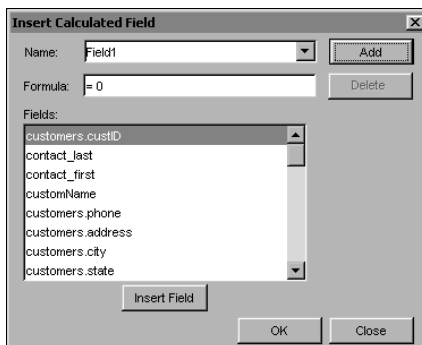
To include a field or an item based on a formula in a pivot range, you must create a calculated field or a calculated item:

- A calculated field uses fields that appear in the Pivot Range Field List in a formula that creates a new pivot range field. You add a calculated field to the data area of the pivot range. For example, you can multiply the pricequote and quantity fields to show an order total.
- A calculated item uses pivot range items in a formula that creates a new item in an existing page, row, or column field. You add a calculated item to the row, column, or page area of a pivot range. For example, you can add North and West to show North and West sales totals as one item. You cannot add a calculated item to a grouped field.

A calculated field or a calculated item can use any data in the pivot range, except subtotal or grand total data. You cannot create a calculated field or calculated item in a pivot range using Data Explorer.

### How to create a calculated field in a pivot range

- 1 Select a field in the pivot range and choose PivotRange→Formulas→Calculated Field. Insert Calculated Field appears and lists the available pivot range fields, as shown in Figure 13-29.



**Figure 13-29** Insert Calculated Field

## 2 Set up the calculated field:

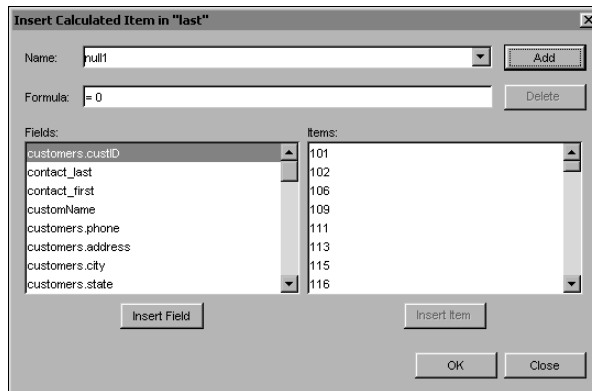
- In Name, type a name for the calculated field.
- In Formula, type the formula that calculates the value for the field.
- To add a field in a formula, select a field name, then choose Insert field.

Choose Add, then choose OK. The field appears in the data area of the pivot range.

### How to create a calculated item in a pivot range

#### 1 Select a field in the pivot range, then choose PivotRange→Formulas→Calculated Item.

Insert Calculated Item appears. Fields lists the available pivot range fields. Items lists the items associated with the selected field. For example, in Figure 13-30, Insert Calculated Item lists the items in the customers.custID field which appears highlighted.



**Figure 13-30** Insert Calculated Item

#### 2 To set up the calculated item:

- In Name, type a name for the calculated item.
- In Formula, type the formula that calculates the value for the item.
- To add a field or item in a formula, select a field or item name, then choose Insert field or Insert Item.

#### 3 Choose Add. The new calculated item appears in the selected field in the pivot range.

#### 4 When you finish creating calculated items and fields, choose OK.

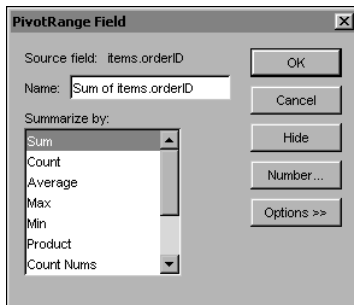
## Changing the expression a calculation uses

You can change the expression a data field, calculated field, or calculated item uses.

### How to change an expression

- 1 Select a data field, calculated field, or calculated item that contains an expression, then choose PivotRange→Field Settings.

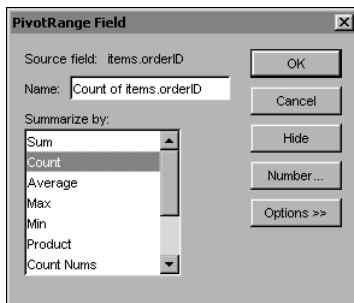
PivotRange Field—Name displays a pivot range field name for the item you selected that uses a function and a field name to describe the current field expression. For example, Figure 13-31 shows how PivotRange Field displays the sum of items.orderID.



**Figure 13-31** PivotRange Field showing a field expression

- 2 In Summarize by, select a different function.

The pivot range field name displayed in Name changes to reflect your selection. For example, in Figure 13-32, PivotRange Field shows Count of items.orderID in Name; the result of selecting Count in Summarize by.



**Figure 13-32** Changing an expression in PivotRange Field  
Choose OK to save the displayed field expression.

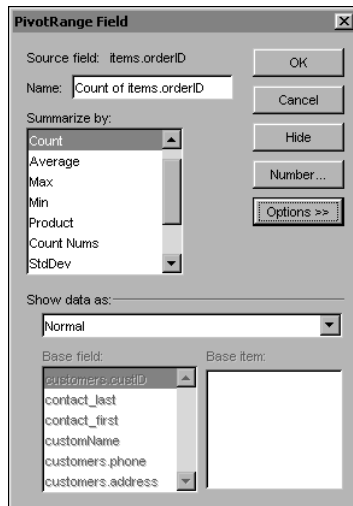


## Using variance data

You can use variance data to show how the value in a data field varies with or compares to other data fields, called base fields, in a pivot range. For example, in a report that shows orders by region, you can show each region's orders as a percentage of total orders. Other variance data calculations include showing data as a running total or showing data as a percentage of the total for a row.

### How to calculate variance data

- 1 Select a data field and choose PivotRange→Field Settings.
- 2 On PivotRange Field, choose Options to see settings similar to those shown in Figure 13-33.



**Figure 13-33** PivotRange Field with variance data calculation options

- 3 In Show data as, set up a calculation:
  - To use a standard summary calculation, select Normal.
  - To show the difference between the selected data field and another pivot range item, called a Base item, select Difference From, then select a Base field and a Base item.
  - To show the selected data field as a percentage of another pivot range item, select Percent of, then select a Base field and a Base item.
  - To show the percentage difference between the selected data field and another pivot range item, select Percent Difference From, then select a Base field and a Base item.
  - To show data as a running total, select Running Total in, then select the field in which to display the running total.

- To show data as a percentage of the total for a row, select Percent of row.
- To show data as a percentage of the total for a column, select Percent of column.
- To show data as a percentage of the total for the entire pivot range, select Percent of total.
- To show data as the result of the following calculation, select Index:
 
$$\frac{((\text{value in cell}) * (\text{Grand Total of Grand Totals}))}{((\text{Grand Row Total}) * (\text{Grand Column Total}))}$$

Choose OK to save the current calculation.

## Working with totals

Typically, a pivot range shows subtotals and grand totals. You can hide all the subtotals or grand totals in a pivot range. You can also show or hide subtotals for each field in a pivot range. When you hide a page field item in a pivot range, you can include or exclude the data for that item in the pivot range subtotals. To show totals for all the items in an inner field, use a block total.

### Working with a subtotal

Typically, a pivot range displays a subtotal below the fields which the subtotal summarizes. You can hide a subtotal or use a subtotal to summarize hidden items. You can add, change, or modify a subtotal and change the location where a subtotal displays.

#### How to hide or show a subtotal for a field

Select the field and choose PivotRange→Subtotals.

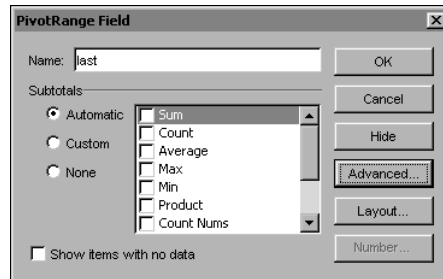
#### How to include hidden items in subtotals

- 1 Select a cell in the pivot range and choose PivotRange→Table Options.
- 2 On PivotRange Options, select Subtotal hidden page items.  
Choose OK.

#### How to modify or add subtotal attributes

- 1 To modify the attributes for a subtotal, select the field in which a subtotal appears and choose PivotRange→Field Settings.
- 2 In PivotRange Field, select one of the following subtotal options:
  - Automatic, to use the default subtotal function for a selected data field.
  - Custom, to modify the calculation for a subtotal. Then select one or more available function.
  - None, to calculate no subtotal for the field.

For example, Figure 13-34 shows the default subtotal function selected.



**Figure 13-34** Selecting the default function to calculate a subtotal  
Choose OK.

## Working with a grand total

Typically, a pivot range displays a grand total to either the right or the bottom of a pivot range.

### How to hide a grand total

- 1 Select a field in the pivot range and choose PivotRange→Table Options.
- 2 In PivotRange Options, set total options:
  - To hide grand totals for columns, deselect Grand totals for columns.
  - To hide grand totals for rows, deselect Grand totals for rows.

Choose OK.

## Working with a block total

Typically, a block total appears after the subtotal for the last outer item field and before the grand total. For example, the report shown in Figure 13-35 uses block totals to show the total and average order amounts for each order category.

Sum of OrderTotal		
last	orders.category	Total
Barajas	1	\$3,801,654
	2	\$8,610,097
Barajas Total		\$28,141,190
Hernandez	2	\$4,961,754
		\$4,961,754
Tseng	1	\$22,260,798
	2	\$10,079,152
Tseng Total		\$62,830,378
	1 Sum	\$51,028,125
	1 Average	\$51,028,125
	2 Sum	\$71,396,640
	2 Average	\$71,396,640
Grand Total		\$247,417,480

Block totals

**Figure 13-35** Using block totals to summarize data

### **How to add a block total to an inner field**

- 1 Select the inner field and choose PivotRange→Field Settings.
- 2 In PivotRange Field, in Subtotals, select Custom, then select the subtotal calculation.

Choose OK to add the block total to the pivot range.

### **Working with pivot range formulas**

To list the formulas used to calculate fields and items for a pivot range, select a field in the pivot range and choose PivotRange→Formulas→List Formulas. A new worksheet appears in the workbook. The new worksheet lists each calculated item and calculated field in the pivot range. The list order matches the current order in which the pivot range solves formulas. This list also shows each calculated item or field name and the formula used to calculate the value it displays.

### **How to modify the solve order of formulas used to calculate items in a pivot range**

- 1 Select a field in the pivot range and choose PivotRange→Formulas→Solve Order.
- 2 In Calculated Item Solve Order, select a formula, then choose one of the following options:
  - To move an item earlier in the solve order, select Move up.
  - To move an item later in the solve order, select Move down.
  - To remove a calculated item from the pivot range, select Delete.

Choose Close.

### **Grouping pivot range data**

You can group pivot range data to show different data relationships. For example you can group date, time, and number fields, by month, hour and hundreds, respectively. Alternatively, you can group those fields by week, minute and tens. You can also group selected items in a pivot range. You cannot add calculated items to a grouped field. To group items that appear in page field area of a pivot range, you must first move the field that contains those items to the row or column area of the pivot range, create the group, then move the field back to the page area.

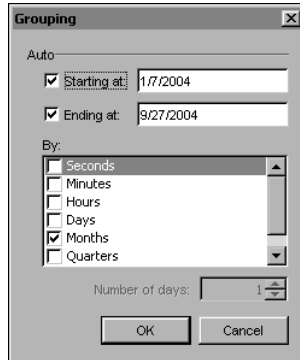
### **Grouping date, time, and number fields**

You can group date, time, and number fields in a pivot range. For example, you can group a numeric order field by the first three digits of the order number. Or, for example, you can group an order date field by the month in which orders closed.

## How to group a field of date or time items

- 1 Select an item in the field and choose PivotRange→Group and Show Detail→Group.

Grouping shows options for separating field items based on current data field values. For example, in Figure 13-36, Grouping shows the span of dates and the possible group criteria for separating items in a date field.



**Figure 13-36** Setting grouping options for a date-and-time field

- 2 Specify grouping settings:
  - In Starting at, determine the date or time at which to begin the group:
    - To have the pivot range calculate the lowest or earliest date or time using the available data, select Starting at.
    - To specify a starting date or time, deselect Starting at and type a date or time.
  - In Ending at, determine the date or time at which to end the group:
    - To have the pivot range calculate the highest or latest date or time using the available data, select Ending at.
    - To specify an end date or time, deselect Ending at and type a date or time.
  - In By, select the criteria by which the pivot range will group items. Each option you select creates another group level. For example, if you group dates by selecting Month and Quarter, then the pivot range will display dates grouped by quarter, and by month.

If you select Days, in Number of days, select the number of days you want to use as the group period. For example, to group financial data by work week, select 5.

Choose OK.

### How to group a field of numeric items

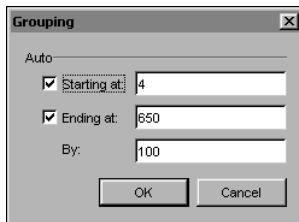
- 1 Select an item in the field to group and choose PivotRange→Group and Show Detail→Group.

Grouping appears. Grouping shows the range of items the field displays.

- 2 Specify grouping settings:

- In Starting at, determine the value at which to begin the group:
  - To have the pivot range calculate the lowest or earliest value using the available data, select Starting at.
  - To specify a starting value, deselect Starting at and type a value.
- In Ending at, determine the value at which to end the group:
  - To have the pivot range calculate the highest or latest value using the available data, select Ending at.
  - To specify an end value, deselect Ending at and type a value.
- In By, select the interval by which to size each group.

For example, the settings in Figure 13-37 group items between 4 and 650 in groups of 100.



**Figure 13-37** Setting grouping options for a numeric field

Choose OK.

### Grouping selected pivot range items

You can group selected items in a pivot range. For example, in a sales report, you can group selected sales representatives, then show subtotals for that group. In Figure 13-38, the pivot range displays subtotals for two new groups in the Salesgroups column.

#### How to group selected items

- 1 Press Ctrl-click to select each item for a group.
- 2 After selecting all group items, choose PivotRange→Group and Show Detail→Group.

The selected items appear in a new group with a default name, such as Group1, added to the pivot range. You can rename a group in the same way as you rename other pivot range fields.

			Data
offices.city	Salesgroup	last	Count of items.orderID
Boston	Group1	Firrelli	78
		Murphy	236
		Patterson	107
	Group1 Sum		579,060
	Group2	Castillo	124
		Thompson	129
	Group2 Sum		363,155
Boston Total			646

Group 1 contains items selected from Boston

**Figure 13-38** Selected-item groups

## Sorting pivot range data

When you create a pivot range report, pivot range data appears sorted in ascending order. Typically, a pivot range sorts data in the following order:

- Numbers.
- Text. The pivot range sort order is case-insensitive.
- Boolean values. False appears before true.
- Errors. Error values appear in the following order:
  - #N/A
  - #NUM!
  - #NAME?
  - #REF!
  - #VALUE!
  - #DIV/0!
  - #NULL!
- Date-and-time values
- Blank cells

You can set options to sort pivot range data in a different order. When you refresh the pivot range, it displays data sorted using the options you set.

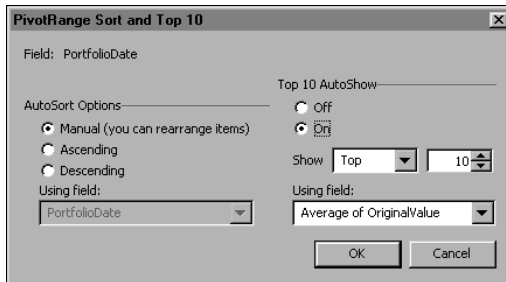
You typically sort data within items. For example, if Sales Representative is an outer row field and Customers is an inner row field, sorting the Customer field sorts the customer names in each Sales Representative item.

### How to sort pivot range items

- 1 Select an item in the field to sort and choose PivotRange→Sort and Top 10.

- 2 In PivotRange Sort and Top 10, select one of the following Autosort Options:
  - Manual, to allow a custom sort order.
  - Ascending, to sort data from lowest to highest value.
  - Descending, to sort data from highest to lowest value.

For example, Figure 13-39 shows the selected options that allow manual sorting of the PortfolioDate field.



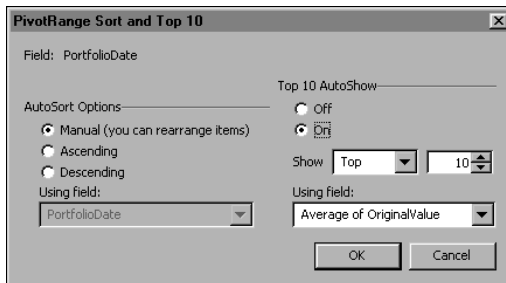
**Figure 13-39** Selecting sort conditions for a pivot range

## Filtering pivot range data

You can use a filter to display the top or bottom items in a pivot range field. For example, in a report that lists the percent gain or loss for several security values in a month, you can apply a filter to show only the five securities with the greatest gain. If security is an outer row field and type is an inner row field, you can use a filter to show the five securities of each type with the greatest gain. Data that a filter hides does not contribute to subtotals.

### How to show the top or bottom items in a row or column field

- 1 Select the field to filter, then choose PivotRange→Sort and Top 10.
- 2 In PivotRange Sort and Top 10, select the following options, as shown in Figure 13-40. Then, choose OK.



**Figure 13-40** Selecting filter options that show top 10 values



- In Top 10 AutoShow, select On.
- In Show, select Top or Bottom, then select a value.
- In Using field, select a field name from the drop-down list.

## Referring to pivot range data

When you refer to pivot range data, you cannot use a cell or range reference or a defined name as you do with other report fields. Updating or rearranging pivot range data breaks standard references. For example, if you use a cell reference to identify pivot range data in a formula, and then you rearrange the pivot range, the formula uses incorrect data. To refer to pivot range data, use one of the following methods:

- To refer to pivot range data in a cell or formula within the pivot range, use a name that identifies the location of the data within the report.
- To refer to pivot range data in a cell or formula outside the pivot range, use the `GETPIVOTDATA` function.

## Referring to pivot range data from inside the pivot range

To refer to pivot range data from a pivot range cell, you must use one of the following reference types:

- **Name reference.** A combination of the row field, column field, and item names. For example, in a pivot range that lists sales totals in several product categories, you can use Chips May to identify the total for the Chips category in the May column. You can list the row field, column field, and item names in any order. For example, Chips May and May Chips refer to the same data. If multiple items in different fields have the same name, you must use the field name when you refer to one of the items.
- **Index number reference.** A field name followed by an index number that identifies where a field or item appears in a field or the report. Enclose the index number in brackets. You can use one of the following types of index numbers:
  - An absolute index number identifies where the field or item appears in the column. For example, in a field called Category, you refer to the second item as Category [2].
  - A relative index number identifies where the field or item appears in relation to the selected cell. For example, in a field called Category, Category [-5] refers to the item five items above the selected item.

To create a reference that always refers to the same data, even when you show or hide items in the pivot range, use a name reference. An index number reference can refer to different data after you hide or show field items. For example, Figure 13-41 shows a pivot range that generates a sales report.

	A	B	C	D
1	offices.city	(Multiple Items)		
2				
3			Data	
4	last	customName	Orders	Order Totals
5	Castillo	Brittan Design Inc.	7	\$9,965,106
6		Design Engineering Corp.	53	\$50,979,872
7		Design Systems	9	\$12,773,106
8		Technical Systems Inc.	46	\$46,704,275
9		TeleBoards Co.	9	\$10,210,840
10	Castillo Total		124	\$756,382,814
11	Firrelli	Computer MicroSystems Corp.	20	\$8,572,278
12		InfoEngineering Corp.	5	\$4,631,412
13		Technical Specialists Corp.	17	\$536,724
14		TeleMicroSystems Inc.	15	\$6,907,698
15	Firrelli Total		57	\$99,127,296
16	Murphy	CompuDesign Co.	38	\$79,264,704
17		Design MicroSystems Co.	30	\$486,222
18		InfoEngineering	9	\$4,845,270
19		InfoSpecialists Inc.	70	\$47,239,424
20		TekniDesign Corp.	19	\$28,356,294
21	Murphy Total		166	\$761,895,944
22	Patterson	Advanced Solutions Inc.	11	\$3,966,864
23		Advanced Specialists Corp.	2	\$275,944
24		Design Solutions Corp.	18	\$34,902,803
25		Signal MicroSystems	2	\$2,068,360
26		Technical Systems Corp.	60	\$68,678,008
27	Patterson Total		93	\$428,479,137
28	Thompson	Design Design	30	\$2,917,348
29		Technical Boards	13	\$33,817,346
30	Thompson Total		43	\$110,447,208
31	Grand Total		483	\$9,412,958,304

**Figure 13-41** Sample pivot range showing sales data

You can identify data in the sample pivot range shown in Figure 13-41, as follows:

- Firelli Order Totals refers to the value in cell D15.
- Castillo Brittan Design Inc. Orders refers to the value in cell C5.
- Grand Total Order Totals refers to the grand total in cell D31.
- 'Last [2]' refers to the data for Firelli.
- Last customName [2] refers to the data for the second customer for each item in last.

## Referring to pivot range data from outside the pivot range

To use data from a pivot range in a formula outside the pivot range, you must use the GETPIVOTDATA function. For example, to display the grand total for a pivot range in a report on another worksheet, use GETPIVOTDATA rather than using the cell reference where the grand total appears. You cannot use GETPIVOTDATA to refer to hidden data in a pivot range or to refer to a block total. You can use the GETPIVOTDATA function in a formula. To have BIRT Spreadsheet Designer create the function for you, you can type the beginning of the formula, then select the pivot range cell in which the data appears.

GETPIVOTDATA uses pairs of field and item names to identify data. GETPIVOTDATA uses the following syntax:

```
=GETPIVOTDATA("DataField", PivotRange, Field,Item)
```

where

- DataField is either the full name of the data field in the pivot range or the name of the data source field on which the data field is based. For example, to identify a count data field based on the orderID field, you can use either "count of orderID" or "orderID".
- PivotRange is a cell or range reference in the pivot range. PivotRange enables BIRT Spreadsheet Designer to locate the pivot range.
- Field, Item uses a field name and an item name to identify the data to retrieve. Enclose field and item names in quotation marks. You can use multiple field and item name pairs.  
For example, to retrieve the data for CustomerA in the Customers field, use "Customers", "CustomerA".

For example, Figure 13-42 shows a pivot range that generates a sales report.

	A	B	C	D
1	offices.city	(Multiple Items)		
2				
3			Data	
4	last	customName	Orders	Order Totals
5	Castillo	Brittan Design Inc.	7	\$9,965,106
6		Design Engineering Corp.	53	\$50,979,872
7		Design Systems	9	\$12,773,106
8		Technical Systems Inc.	46	\$46,704,275
9		TeleBoards Co.	9	\$10,210,840
10	Castillo Total		124	\$756,382,814
11	Firrelli	Computer MicroSystems Corp.	20	\$8,572,278
12		InfoEngineering Corp.	5	\$4,631,412
13		Technical Specialists Corp.	17	\$536,724
14		TeleMicroSystems Inc.	15	\$6,907,698
15	Firrelli Total		57	\$99,127,296
16	Murphy	CompuDesign Co.	38	\$79,264,704
17		Design MicroSystems Co.	30	\$486,222
18		InfoEngineering	9	\$4,845,270
19		InfoSpecialists Inc.	70	\$47,239,424
20		TekniDesign Corp.	19	\$28,356,294
21	Murphy Total		166	\$761,895,944
22	Patterson	Advanced Solutions Inc.	11	\$3,966,864
23		Advanced Specialists Corp.	2	\$275,944
24		Design Solutions Corp.	18	\$34,902,803
25		Signal MicroSystems	2	\$2,068,360
26		Technical Systems Corp.	60	\$68,678,008
27	Patterson Total		93	\$428,479,137
28	Thompson	Design Design	30	\$2,917,348
29		Technical Boards	13	\$33,817,346
30	Thompson Total		43	\$110,447,208
31	Grand Total		483	\$9,412,058,304

**Figure 13-42** Sample pivot range sales data

You can use the following GETPIVOTDATA references, placed in a cell outside the pivot range, to identify data in the pivot range:

- =GETPIVOTDATA("Order Totals", \$A\$3,"last","Castillo") refers to the Castillo total in cell D10.
- =GETPIVOTDATA("Order Totals ", \$A\$3,"customName","Brittan Design Inc.,""last","Castillo") refers to Castillo's order total for Brittan Design, Inc. in cell D5.

- =GETPIVOTDATA("Order Totals ", \$A\$3) refers to the grand total in cell D31.

BIRT Spreadsheet Designer helps you to build a correct GETPIVOTDATA expression. To use GETPIVOTDATA, in a cell outside the pivot range, type an equal sign, then select the pivot range cell that contains the data to get. BIRT Spreadsheet Designer inserts the correct GETPIVOTDATA reference in the cell.

---

## Printing a pivot range

You print a pivot range report in the same way as you print other spreadsheet reports. When you deliver the report in Excel, you can also use the following print options:

- Insert a page break after each outer row field item.
- Use item labels as print titles in a printed pivot range.

Pivot range print options affect printing from Excel only. Pivot range print options do not affect the report in BIRT Spreadsheet Designer.

### How to insert page breaks after each outer row field item

- 1 Select an outer row field, then choose PivotRange→Field Settings.
- 2 In PivotRange Field, choose Layout.
- 3 In PivotRange Field Layout, select Insert page break after each item. Then, Choose OK.
- 4 In PivotRange Field, choose OK.

### How to set pivot range printed page titles

- 1 Select a field in the pivot range and choose PivotRange→Table Options.
- 2 Use PivotRange Options to specify printed page title settings. Then, Choose OK.
  - To print outer row field item labels as page titles at the top of each printed report page, select Repeat item labels on each printed page.
  - To use field and item labels as row and column titles on printed report pages, select Set print titles.

## Designing a secure report

This chapter contains the following topics:

- About report security
- Restricting user access
- About SmartSheet security

---

## About report security

BIRT Spreadsheet Designer supports many types of report security to help ensure the integrity of data in your report. For example, you can lock cells to prevent users from changing, moving, or deleting important data. You can also deliver a customizable report that shows a user only the data appropriate to her business role, based upon her login credential.

You can use the following security features:

- Allowing selected actions on a worksheet. To ensure that a delivered report retains the intended formatting and displays correct data, you can allow only selected action on a spreadsheet report worksheet.
- Setting password access. You can require users to provide a password before opening or saving a file or manipulating data in cells, ranges, worksheets, or graphical objects.
- Hiding a worksheet or formula. Hiding a worksheet, a formula, or formula data enables you to use data in report calculations without exposing the base data to users.
- Encrypting a workbook. Encryption makes text readable only to authorized users who have a public key that matches the encryption type and allows them to decrypt the text.
- Customizing reports using SmartSheet security. SmartSheet security technology enables you to deliver a different report for each user. A user's security ID determines the data he sees.

This chapter describes how to set up and use security features in a report.

---

## Restricting user access

You can restrict access to selected report operations and elements. For example, you allow or disallow specific report operations on each worksheet. You can also set password access for report elements. You can prevent a user from viewing a spreadsheet report worksheet and from saving any changes she makes to that worksheet until she provides a password that you set. You can also set a password that restricts access to an entire workbook.

To prevent changes to report appearance, you can lock a specific graphical object or graphical object text. To prevent changes to report content, you can also lock a cell or range of cells in a worksheet.

You can hide specific report elements. Hiding a report element prevents report users from viewing or accessing that element, without removing it from the report and without requiring a password.

## Allowing selected user actions

You can select options that allow certain actions on a worksheet. For example, you can allow or not allow changes to worksheet layout, addition or removal of rows or columns, and selection of cells or objects. Table 14-1 lists options and user actions that you can protect on each worksheet in a spreadsheet report.

**Table 14-1** Options on Protect Sheet and allowed user actions

Option	User action	Default setting
Select locked cells	Select locked cells	✓
Select unlocked cells	Select unlocked cells	✓
Format cells	Use Format Cells and the associated context menus	
Format columns	Hide, show, and resize columns	
Format rows	Hide, show, and resize rows	
Insert columns	Add columns	
Insert rows	Add rows	
Insert hyperlinks	Create hyperlinks	
Delete columns	Remove columns	
Delete rows	Remove rows	
Sort	Sort report data	
Use Autofilter	Filter a report	
Use PivotRange reports	Manipulate a pivot range	
Edit objects	Create, edit, and delete a graphical, text, or picture object	

When you allow a specific user action on a worksheet, the report user will see an active menu item for that action when that worksheet opens. When you disallow an action, the menu item for that action appears grey when that spreadsheet report worksheet opens.

For example, a report user opens a report worksheet that you protect using the default selections. When that user selects a cell on that worksheet and chooses Format, she sees no active menu option to format or insert a cell, column or row. If you select all Insert options on Protect Sheet, then that user will see the active menu option Insert, and the active submenus; column, row and hyperlink.

To control what actions users can perform on a worksheet, choose Tools>Protection>Protect Sheet. Next, select options, then select Protect worksheet and contents of locked cells just as you would when you use Excel.

## About password requirements

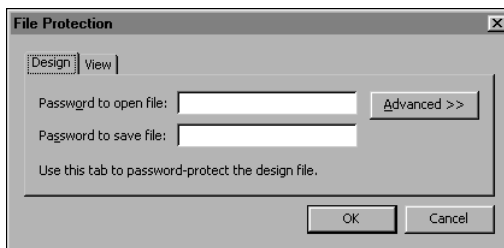
Limit passwords to 255 or fewer characters. You can use special characters, such as &, %, or \$. Although a password can contain any character, use only low-ASCII characters when you create passwords for use in multiple locales. A locale-specific character such as an ideograph may not be available to a user in a different locale. BIRT Spreadsheet Designer supports case-sensitive passwords.

## Setting a password for a workbook

Typically, a spreadsheet report workbook allows user access. You can set a password that a user must provide to open or save a spreadsheet report design. You can also set a password that a user must provide to open or save a spreadsheet report. To allow unrestricted access to a workbook, delete these passwords.

### How to set a password to open and save a workbook

- 1 To set file protection options for an open report design file, choose Tools>Protection>File Protection and Encryption.
- 2 In File Protection, select Design or View.
- 3 Type a password in one or both of the following fields shown in Figure 14-1, then choose OK:
  - Password to open file
  - Password to save file



**Figure 14-1** Setting a password to open and save a workbook file

- 4 On Verify password to open file, retype the password you set to open file, as shown in Figure 14-2 then choose OK.

If you also set a password to save the file, retype the password you set to save file, then choose OK.





**Figure 14-2** Verifying a password

BIRT Spreadsheet Designer displays the message, “Changes you make in File Protection take effect the next time you save the file.”

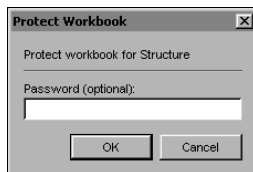
- 5 To close message window, choose OK.
- 6 To save the report with a password, save the view as a .xls file.
- 7 To save the design with a password, save the design as a .sod file.

## Setting a password for workbook structure

You can set a password that protects workbook structure. A user must provide this password to add, hide, move, delete, or rename individual worksheets in a workbook.

### How to set a password that allows changes to workbook structure

- 1 In a workbook you want to protect, choose Tools→Protection→Protect Workbook.
- 2 In Protect Workbook, in Password (optional) as shown in Figure 14-3, type characters, then choose OK.



**Figure 14-3** Setting a password in Protect Workbook

- 3 In Protect Workbook, in Verify password, retype the password characters you typed in step 2, then choose OK.

## Setting a password for a worksheet

Typically, a spreadsheet report worksheet allows user access. You can set a password for a worksheet and lock all, or lock specific cells on a worksheet. A user who attempts to type data in a locked cell sees an error message.

Before she can access a locked cell or worksheet, a user must type the password you set for the worksheet.

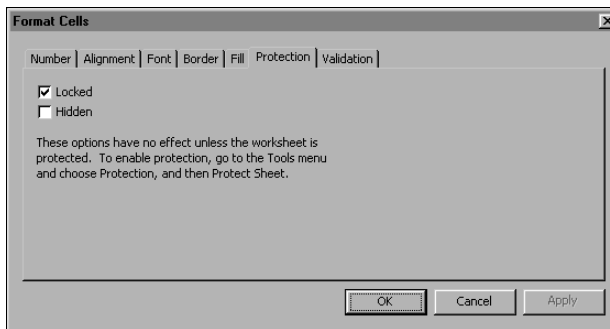
To set a password for a worksheet, choose Tools→Protection→Protect Sheet, type characters in Password, then select protect sheet and contents of locked cells just as you would when you use Excel.

## Unlocking selected cells in a worksheet

When you set a password for a worksheet, you typically lock all cells in the worksheet. Use the following procedure to unlock a cell before you protect the worksheet.

### How to unlock a specific cell

- 1 Select a cell that you want unprotected, then choose Format→Cells.
- 2 In Format Cells, select Protection to see the options shown in Figure 14-4.



**Figure 14-4** Format→Cells→Protection showing Locked for a cell

- 3 Deselect Locked, then choose OK.

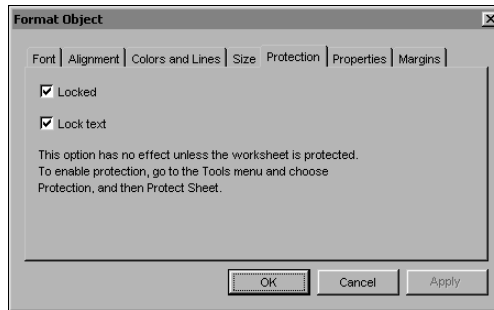
After unlocking selected cells, set a password for the worksheet to lock the remaining cells, as described earlier in this chapter.

## Locking a graphical object

Unlike worksheet cells, any graphical objects on a worksheet remain unlocked when you set a password for that worksheet. You can lock a graphical object or graphical object text.

### How to lock a graphical object on a worksheet

- 1 Right-click the object, choose Format Object, then choose Protection to see the protection options shown in Figure 14-5.
- 2 Make either or both of the following selections. Then, choose OK.
  - Select Locked to lock the image.
  - Select Locked text to lock the text in the image.

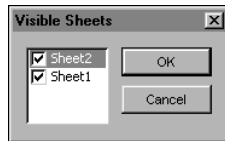


**Figure 14-5** Locking a graphical object

After setting object protection, set the worksheet password as described earlier in this chapter.

## Hiding a worksheet

To restrict access to a worksheet, you hide the worksheet. You cannot hide all the worksheets in a workbook. At least one worksheet must remain visible. To hide a worksheet, choose **Format**→**Sheet**→**Visibility**. On **Visible Sheets**, checkmarks appear next to visible worksheets, as shown in Figure 14-6. To hide a worksheet, deselect the worksheet name, then choose **OK**.



**Figure 14-6** Two visible worksheets

## Hiding formula results

When an active cell contains a formula, that cell typically displays the result returned by the formula. At the same time, the formula bar displays the formula text. To hide the result that a formula returns, you can format a worksheet to display formula text in each cell that contains a formula, as shown in Figure 14-7.

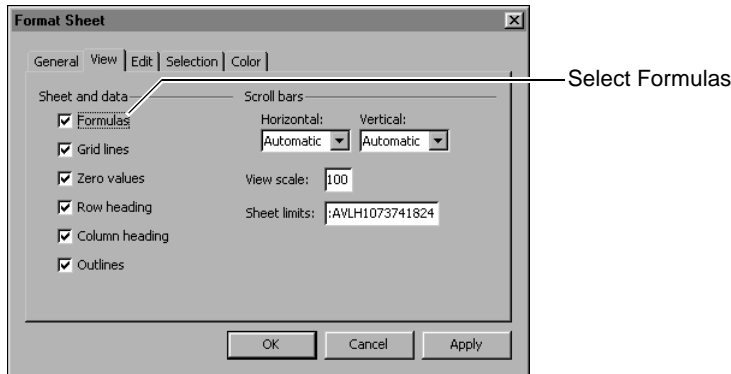
B1		=SUM(Income!A1:A12)		
	A	B	C	
1	Total Income	=SUM(Income!A1:A12)		Formula text
2	Total Expenses	=SUM(Expenses!A2:A14)		
3				
4	Net	=B1-B2		
5				

**Figure 14-7** Displaying formula text in a cell

### How to show formula text in place of a formula result

- 1 In an unprotected worksheet, choose **Format**→**Sheet**→**Properties**.

- 2 In View, select Formulas as shown in Figure 14-8. Then, choose OK.



**Figure 14-8** Showing formula text

## Hiding formula text

To prevent display and change of formula text, you can format a cell to prevent formula text in that cell from appearing in the formula bar. When a user selects a cell that you have formatted as hidden, the formula bar displays nothing.

To hide formula text, select a cell or range and choose Format→Cells, then select Hidden, just as you would when you use Excel.

After you select hidden as a cell format, you must also select locked and then protect the worksheet to prevent change of formula text in each cell. Formula text remains visible until you protect the worksheet.

## Using file encryption

When you use file encryption, you add an algorithm and ciphers to the file. When you save an encrypted spreadsheet report as an Excel file, the algorithm applies ciphers that make text in the file unreadable. Users who provide a password that you set activate ciphers that decrypt the report text. To implement file encryption, you must first select an encryption level, then set a password that enables the encryption/decryption process. You can encrypt a spreadsheet report design and the resulting report view.

### How to set file encryption

- 1 In a spreadsheet report file you want to encrypt, select Tools→Protection→File Protection and Encryption.
- 2 On File Protection, choose Advanced to see the advanced protection options in Figure 14-9.



**Figure 14-9** Advanced file protection options

- 3 Select Design or View.
- 4 Select one of the following options. Then, choose OK.
  - In most cases, select Office 97/2000 and BIRT Spreadsheet.
  - Select Weak encryption only if the workbook is designed for use in a country, such as France, that does not permit strong encryption.

After you select an encryption type, you must set a password to open the file. For more information about setting a password for a file, see “Setting a password for a workbook,” earlier in this chapter.

- 5 In Password to open file, type characters.
- 6 In Password to save file, type characters, then choose OK.
- 7 To verify the passwords required to open and save the file, in Verify Password retype the password characters you typed in steps 5 and 6.
- 8 To save the report as an encrypted file, save the view as a .xls file.
- 9 To save the design as an encrypted file, save the design as a .sod file.

---

## About SmartSheet security

SmartSheet security is a BIRT Spreadsheet Designer feature that uses a grant expression to create a customized report view. When a reader opens a report that uses SmartSheet security, a grant expression evaluates the login credential that the user provides. The report displays to that user only information appropriate for that user’s role. For example, you can use SmartSheet security in a report design that displays to a sales manager all the data that represents the sales results of her direct reports. A sales representative who opens the report at the same time sees only information that represents his sales results. SmartSheet security is available only to users who purchase the Actuate SmartSheet Security option for use with BIRT iServer.

SmartSheet security works by comparing the report user's security IDs (SIDs) to the access control list (ACL) for all or part of a report. An ACL lists privileges that allow access to a report element. To create the ACL for that element, you build a grant expression that evaluates to a security ID. If any of a report user's security IDs match any security ID in the ACL for that element, the report user gains access to the data in that element.

A report that uses SmartSheet security evaluates a user login credential and uses that result to filter the report. A report that uses SmartSheet security makes no request for a parameter value to filter the report. SmartSheet security minimizes handling of user input values and controls access to report information based upon matching a user credential to a defined role in a spreadsheet report.

## About security IDs

A security ID provides information to Actuate BIRT iServer about a report user's access privileges to report elements that use SmartSheet security. A security ID can be based on any of the following identifiers:

- The user's name
- The user's Encyclopedia volume security roles
- A user name the Report Server Security Extension (RSSE) provides  
For more information about Report Server Security Extension, see *Using BIRT iServer Integration Technology*.
- A virtual security ID  
A virtual security ID can provide greater access restrictions to a section. For example, you can concatenate Encyclopedia volume security roles to create a virtual security ID.

## Using security IDs in a data range

To return the current user's security ID, use the security report script function. Use the following syntax:

```
security()
```

You can use the security report script function to display information about the user in a data range cell. If the user has more than one security ID, such as Manager, Corporate, and France, security returns all the ID values.

You can also use security to set up conditions in a data range. Use the following syntax:

```
security(ID_value)
```

where ID\_value is a user ID.

For example, the following report script function displays different data if the user is a CEO or Manager than if she does not have CEO or Manager as a security ID:

```
#if (security({"CEO","Manager"}), formula("SUM(" &  
    cells(prodLine.detail) & ")", sum(revenue) )
```

If the user is a CEO or Manager, she can see which detail data contributed to a total. If the user does not have either security ID, she can only see the totals, not the detail data.

For more information about setting up security, including a user's security IDs, see *Managing an Encyclopedia Volume*.

## About creating grant expressions

A grant expression includes text strings and data set columns. Typically, you use report script functions to create a grant expression. Do not precede the report script function text with a number sign. You can add text before or after a data set field, as you do with other report script function expressions. You can associate a grant expression with a data set, a worksheet, a data range, or a data range section. When the report runs, the grant expression evaluates to a string.

For example, you could define the following expression as the grant expression GrantExp, then attach it to a data range section:

```
[offices.city] & "Office Manager"
```

For each value of [offices.city] in the database, GrantExp returns a security ID that corresponds to a security role in the Encyclopedia volume that contains the report. For example, for Boston, GrantExp returns

```
Boston Office Manager
```

When the value of [offices.city] changes in a data range section, the ACL of the section changes. If the Boston Office Manager opens an Excel view of the report, he will see only the sections for which Boston is the offices.city value.

The following list describes more ways to create a grant expression:

- Use a field name as a grant expression to return a value or list of values. For example, use the following grant expression to return Country values:

```
[Country]
```

You can specify the data set from which to return values. For example, use the following grant expression to return values from Country in the Dev data set:

```
Dev: [Country]
```

You cannot, however, apply a grant expression to a data set if the grant expression refers to data from a different data set.

- Use report script functions to change the values a field returns in a grant expression. For example, you can use the upper report script function to change the State field values to upper case:

```
upper([State])
```

- Use the if report script function or the value of a parameter to set up a conditional expression. For example, the following report script function returns the value of State if the US parameter is true and Province if the US parameter is false:

```
if(:US=True, [field 1], [field 2])
```

- Apply aggregate report script functions, such as stdev or sum, to the value of a field in a grant expression. For example, the following report script function expression returns the maximum value from the Price field:

```
max([Price])
```

- Use a Java class to create a grant expression. For example, the following grant expression uses MyClass.Method to return an array of grant expression strings:

```
MyClass.MyMethod(repID)
```

- Combine terms in a grant expression. Use commas to separate terms. For example, the following grant expression evaluates to CEO, a comma, and a second term that uses Dept and the value of DeptID:

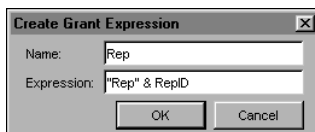
```
"CEO", "Dept" & [DeptID]
```

The report script function evaluates to expressions such as CEO, Dept201. A user with either the CEO security ID or the Dept201 security ID can access data that this grant expression protects.

### How to create a grant expression

- 1 In Data Explorer, right-click Security and choose Create Grant Expression.
- 2 In Create Grant Expression, type the following characters. Then, choose OK.
  - In Name, type characters that name the grant expression.
  - In Expression, type a grant expression string.

For example, Figure 14-10 shows the characters you type to create a grant expression called Rep that concatenates the word Rep with values from the RepID field.



**Figure 14-10** Creating a grant expression



The grant expression name appears under Security in Data Explorer.

## Adding a grant expression to a report element

After you create a grant expression, you add it to a report element such as a data set, data range section, or worksheet. Adding a grant expression to a report element ensures that only authorized viewers see data in that report element.

### How to add a grant expression to a report element

- 1 Select a report element.

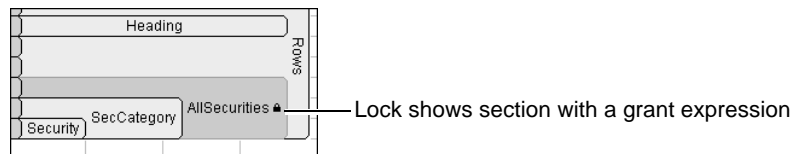
For example, to apply a grant expression to a:

- data set, in Data Explorer, right-click the data set name
- data range section, right-click the section tab
- worksheet, right-click the worksheet tab

- 2 Select Set Security from the context menu.

- 3 Select the grant expression name you want applied to the report element.

After you add a grant expression to a report element, a lock image appears next to the element name in the design. For example, Figure 14-11 shows a data range section, AllSecurities, with a lock.



**Figure 14-11** Data range section with a grant expression

To remove a grant expression from a report element, right-click the report element, select Set Security, then select None.

## Testing a grant expression

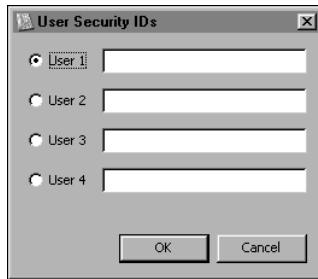
You should test a new grant expression before you publish a report. To test, you provide sample security IDs, then run the report to see which data the report displays. The previewed report should display only the sections or data in which the result generated by the grant expression matches the security ID you provide.

To set up sample security IDs, in Data Explorer, right-click Users's Security IDs and choose Edit, as shown in Figure 14-12.



**Figure 14-12** Opening User's Security IDs

Then provide sample values, using User Security IDs, as shown in Figure 14-13. You can provide up to four security IDs for testing.



**Figure 14-13** Providing sample security IDs

For example, if you created a grant expression called Rep that shows the text Rep followed by an employee ID, provide a user ID in the same format, such as the following text:

Rep101

To continue testing, select one of the sample user IDs, then choose OK. When you run the report, BIRT Spreadsheet Designer uses the selected user ID to generate the correct report information.

## Using SmartSheet security to deliver secure reports

To apply SmartSheet security features to a report, you first create a grant expression. Next, you associate the grant expression with a report element such as a data set, worksheet, data range, or data range section. Then, you run the report on the report server to create a report instance file. The BIRT iServer administrator must apply secure read privilege to the report instance file.

### Using SmartSheet security on a data set

When you apply a grant expression to a data set, only users whose security ID matches a security ID in the data set ACL see that data. For example, Figure 14-14 shows a report that was run without a grant expression that restricts the data set. The report displays all clients and all portfolio values to all users.

Figure 14-15 shows the same report run with a grant expression that restricts the data set to show only clients from the user's country. A user from Spain who runs the report sees only clients from Spain.

Another grant expression prevents some employees from seeing sensitive financial data. For example, a temporary employee can see client names but not portfolio values, as shown in the report in Figure 14-16.

	A	B	C	D	E
1	<b>Client List</b>				
2	Generated: 30-Aug-06				
3	<b>Client</b>	<b>Country</b>	<b>Mgr</b>	<b>Portfolio Value</b>	
4	Aaron	USA	4	\$16,220,915	
5	Abner	USA	2	\$29,803,945	
6	Anderson	USA	1	\$369,730	
7	Anderson	USA	2	\$474,950	
8	Anderson	USA	4	\$458,900	
9	Arendt	Germany	1	\$250,360	
10	Ashby	USA	2	\$448,440	
11	Ashley	USA	1	\$194,165	

**Figure 14-14** Sample report without a grant expression

	A	B	C	D	E
1	<b>Client List</b>				
2	Generated: 30-Aug-06				
3	<b>Client</b>	<b>Country</b>	<b>Mgr</b>	<b>Portfolio Value</b>	
4	Avila	Spain	1	\$314,305	
5	Barajas	Spain	1	\$155,310	
6	Benitez	Spain	1	\$99,020	
7	Benitez	Spain	3	\$371,590	
8	Cabanillas	Spain	3	\$538,920	
9	Cabanillas	Spain	4	\$399,440	
10	Camacho	Spain	1	\$355,400	
11	Campos	Spain	4	\$637,860	

**Figure 14-15** Sample report with a country grant expression

	A	B	C	D	E
1	<b>Client List</b>				
2	Generated: 30-Aug-06				
3	<b>Client</b>	<b>Country</b>	<b>Mgr</b>		
4	Aaron	USA	4		
5	Abner	USA	2		
6	Anderson	USA	1		
7	Anderson	USA	2		
8	Anderson	USA	4		
9	Arendt	Germany	1		
10	Ashby	USA	2		
11	Ashley	USA	1		

**Figure 14-16** Sample report with a employee-level grant expression

## Using SmartSheet security on a worksheet

By applying SmartSheet security to a worksheet in the report design, you can display different report worksheets to each user, based on the user's security role. Using an appropriate grant expression for each worksheet allows you to provide an appropriate spreadsheet view from one data set to each user. You can also show more or less of a report design to a user, based on the user's security role.

The following example shows how you can limit display of report worksheets to users whose security IDs contain one city. At the same time, you can provide a view of all worksheets to a user whose security ID contains a high level role. Figure 14-17 shows a report design with the grant expression city,"CEO" applied to the byCity worksheet.

	A	B	C	D	E	F
1						
2						
3		#city				
4						
5						
6		#customerName				
7		#orderNumber	#productName	#quantity	#Revenue	
8						
9						
10						
11						
12						

Summary byCity

**Figure 14-17** Grant expression on the byCity Sheet

Figure 14-18 shows the view generated for a user whose security ID contains the city Tokyo, and CEO, a job title field. The report includes a summary page and worksheets for all cities in the data set.

	A	B	C	D	E	F	G	H
1								
2		Summary						
3								
4								
5		City	orders	revenue				
6		Boston	276	892,539				
7		London	456	1,436,951				
8		NYC	353	1,157,590				
9		Paris	959	3,083,762				
10		San Francisco	445	1,429,064				
11		Sydney	370	1,147,176				
12		Tokyo	137	457,110				

Summary Boston London NYC Paris San Francisco Sydney Tokyo

**Figure 14-18** Report view for User with CEO security ID

To contrast the preceding example, Figure 14-19 shows the view generated for a user whose security ID contains only the city Tokyo. This user can see only the summary page and the Tokyo page.

	A	B	C	D	E	F
1						
2						
3		Tokyo				
4						
5						
6		Cruz & Sons Co.				
7			10108 1968 Ford Mustang	33	5,458	
8			10108 1968 Dodge Charger	45	4,334	
9			10108 1970 Plymouth Hemi Cuda	39	2,957	
10			10108 1969 Dodge Charger	36	3,856	
11			10108 1948 Porsche 356-A Roadster	38	2,575	
12			10108 1969 Dodge Super Bee	26	1,902	

Summary Tokyo

**Figure 14-19** Report view for User with Tokyo security ID

The preceding examples show how applying an appropriate grant expression to a report element provides secure access to that element. It also shows a simple report layout that provides useful information to broad range of users; one that

can be called scalable. SmartSheet security allows a report developer to provide a secure, scalable report design that requires low overhead to develop and deploy.



## Using a hyperlink

This chapter contains the following topics:

- About hyperlinks
- Working with a fixed hyperlink
- Working with a variable hyperlink
- Using hyperlinks in a BIRT iServer environment

---

## About hyperlinks

In a spreadsheet report, you can create a hyperlink that points from a cell to a worksheet range, web page, file, or e-mail address. You can also create a hyperlink that opens a report created using another Actuate application.

When you create a hyperlink, you specify a location, or target, to which the hyperlink points. You can create the following two kinds of hyperlinks:

- A fixed hyperlink, which includes a static or fixed target value, for example, a cell location or file name. A fixed hyperlink does not change when the data in the report changes. A fixed hyperlink points to the same target in a published report as in the report design.
- A variable hyperlink, which includes a dynamic or variable target value, for example, a dynamic field or parameter. A variable hyperlink changes when report data changes. When you create a variable hyperlink expression for a field or a group in a report design file, the published report displays a hyperlink for each item in the field or the group. When the report data changes, a variable hyperlink changes accordingly.

## Understanding hyperlink targets and target types

BIRT Spreadsheet Designer supports the following hyperlink target types:

- E-mail address. An e-mail address that you use as a hyperlink target must use the following syntax: <name>"@"<address>".<xyz>, where xyz specifies a domain, such as com, net, org, or gov.

You can also create a more specific hyperlink that opens the local e-mail client, creates a new message and populates the message subject line. To create this type of target, use the `mailto:` function and the `?subject` element. For example, to create an e-mail message about Hyperlinking addressed to `jdoe` at Actuate, use the following target element:

```
mailto:jdoe@actuate.com?subject=Hyperlinking
```

- File. A file name that you use as a hyperlink target must be associated with an executable file on the computer that hosts the target. You can create a hyperlink that points to the following kinds of files:
  - A spreadsheet workbook created using either BIRT Spreadsheet Designer or Microsoft Excel
  - A report executable file created using an Actuate application
  - A PDF file
  - An image file
  - An XML file



- A formatted or unformatted text file

You can type an absolute path, such as C:\My Documents\calendar.xls, or a relative path from the working directory, such as \calendar.xls. You can also point a hyperlink to an existing bookmark in a Microsoft Word or PDF file. For example, to point a hyperlink to the bookmark named Files in Introduction.doc, use the following target:

```
\Books\Getting Started\Introduction.doc#Files
```

The following example shows the syntax for a hyperlink to a report that is served by Actuate Information Console running on the local system:

```
http://localhost:8700/portal/servlet/CorpSales.sox
?userid=administrator&password=mySecret?volume=myVol
&_folder=/&_wait=True&salesYear=2008
```

The preceding example launches Actuate Information Console, which executes the CorpSales report immediately after assigning the value 2008 to the report salesYear parameter.

- Range. A range that you use as a hyperlink target must include a valid reference to a cell or defined name on any worksheet in a workbook. For example, to create a hyperlink that points to cell A1 on an Excel or BIRT Spreadsheet Designer worksheet Sheet2, use the following target:

```
Sheet2!A1
```

To point a hyperlink to a defined name in a workbook, use the defined name as the bookmark. For example, to create a hyperlink that points to the defined name Receipts in an Excel workbook named Checkbook.xls, use the following target:

```
\Checkbook.xls#Receipts
```

- Web page URL. A URL that you use as a hyperlink target to a HTML, JSP, ASP, or another type of web page must use valid syntax for <domainsecurity>.<sitename>.<domain>/<pagename>.<pagetype>. To specify a bookmark in the web page, type # and the name of the bookmark after the URL. For example, to create a hyperlink that points to the Credits bookmark on a public web page named Index.html, use the following target:
- ```
www.website.com/Index.html#Credits
```
- Variable. You can use a field name or a parameter name as a hyperlink target. For more information about the syntax you use to create a variable hyperlink, see “Working with a variable hyperlink,” later in this chapter.

## Working with hyperlinks

You create a hyperlink in a worksheet cell. You can create a hyperlinked cell either inside or outside a data range. You can create a hyperlink in an empty cell, or in a cell that contains text, a value or formula, or an image. When you create a

hyperlink in an empty cell, the cell displays the hyperlink text. A cell that contains information before you create a hyperlink in it displays that same information after you create the hyperlink. Text in a hyperlinked cell appears blue and underlined. You can edit and format a hyperlinked cell without affecting the hyperlink.

## **Working with a hyperlinked cell**

If you move the cursor over a hyperlinked cell and click, you activate the hyperlink. To select a hyperlinked cell without activating the hyperlink, select a cell near the hyperlinked cell and then use arrow keys to navigate to the hyperlinked cell. Alternatively, move the cursor over a hyperlinked cell and right-click to open the cell context menu. When you select a hyperlinked cell, the formula bar displays the cell content. You can change content displayed in a hyperlinked cell using the formula bar.

## **Working with hyperlink text**

You can select hyperlink text within a hyperlinked cell without activating the hyperlink. To select hyperlink text, move the cursor over hyperlink text, then right-click. Hyperlink commands available on the cell context menu are Edit Hyperlink, Open Hyperlink and Remove Hyperlink.

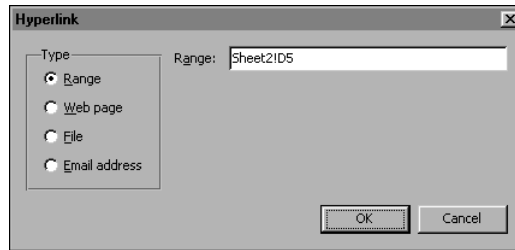
---

# **Working with a fixed hyperlink**

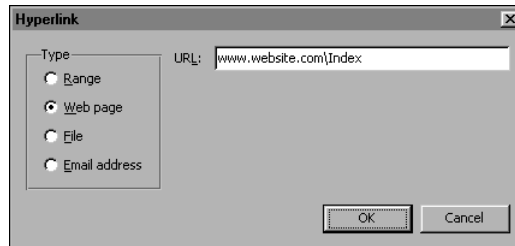
Creating a fixed hyperlink is similar to creating a hyperlink using other spreadsheet applications such as Excel. BIRT Spreadsheet Designer provides a user interface that supports creating all the available hyperlink types. You can also edit or delete a fixed hyperlink.

### **How to create a fixed hyperlink**

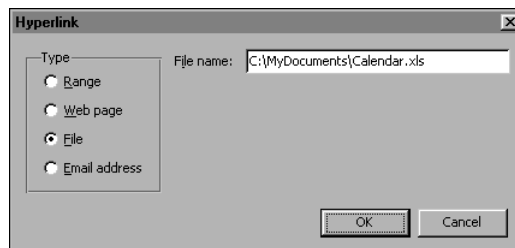
- 1 Select a cell, then choose Insert→Hyperlink. Alternatively, right-click a cell, then choose Hyperlink from the context menu.
- 2 In Hyperlink, in Type, select the hyperlink target type to which the hyperlink will point and then type the specific string that defines the target:
  - To point to a range, in Type, select Range. In Range, type an absolute or relative reference to a cell or range in the current workbook, as shown in Figure 15-1.
  - To point to a web page, in Type, choose Web page. In URL, type the web page URL, as shown in Figure 15-2.
  - To point to a file, in Type, choose File. In File name, type a path and filename, as shown in Figure 15-3.



**Figure 15-1** Pointing to a range

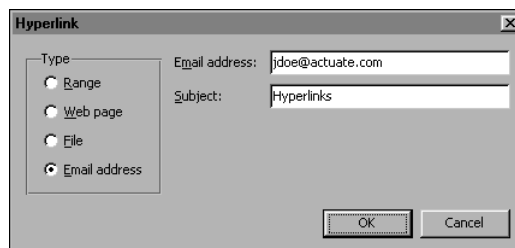


**Figure 15-2** Pointing to a web page



**Figure 15-3** Pointing to a file

- To point to an e-mail address, in Type, choose E-mail address. In E-mail address, type a valid e-mail address, as shown in Figure 15-4. Choose OK.

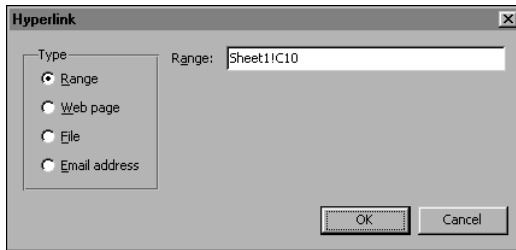


**Figure 15-4** Linking to an e-mail address

#### How to edit a fixed hyperlink target

- 1 Right-click the hyperlink and choose Edit Hyperlink from the context menu.

- 2 In Hyperlink, in Type, select Range, Web page, File, or E-mail address, as shown in Figure 15-5.



**Figure 15-5** Editing a fixed hyperlink to a range

- 3 To change the target, select the displayed Range, URL, File name, or E-mail address field, type the change, then choose OK.

#### How to remove a fixed hyperlink

Right-click the hyperlink, then choose Remove Hyperlink using the context menu.

---

## Working with a variable hyperlink

To create a variable hyperlink in a data range, you use the `#hyperlink` report script function to create a hyperlink expression that uses a data field or expression. `#hyperlink` supports the same hyperlink types as a static hyperlink. You must specify the hyperlink type as well as the target. `#hyperlink` only creates the link. You add static text or another report script function to the expression to display values in the result report cells. The displayed values vary as the values in each data field. For example:

```
#hyperlink([repEMail],1) write([repLast]&" "&[repFirst])
```

where

- `repEMail` is the name of the field that contains sales representative e-mail addresses.
- `1` denotes an email hyperlink.
- `repFirst` and `repLast` are names of the fields that contain the first and last names of the sales representatives.

This example creates a hyperlink using an absolute URL for each representative's e-mail address. Cells in the report display the representative's last and first names separated by a comma and formatted as hyperlink text.

---

## Using hyperlinks in a BIRT iServer environment

You can use a hyperlink in a report design to point to other Encyclopedia volume locations or reports. Spreadsheet reports deployed on BIRT iServer support fixed and variable hyperlinks. You can also create a hyperlink that points to different targets based on the value of one or more parameters when the report runs.

### Linking to a report in an Encyclopedia volume

You can create a hyperlink that points to a published report in an Encyclopedia volume. You can also create a hyperlink that generates and opens a new report. To create a hyperlink that opens another report, you must know the web server and port number to use, and the name and location of the report executable file. You supply the location of the report as the link parameter in a #hyperlink report script function. Use the following syntax for the link parameter:

```
http://<webserver:port>/<context>/servlet/  
    <filename>?volume=<encyclopedia>&__wait=True  
    [&<reportparameter>=<value>]
```

where

- <webserver:port> is the name and port number of the web server on which Actuate Information Console runs.
- <context> specifies the application in which to run the report. iportal is the default context for Actuate Information Console.
- servlet/<filename> specifies a report file name. To link to a published spreadsheet report executable, provide a complete file name that includes the .sox file-name extension.
- <encyclopedia> defines which Encyclopedia volume contains a report file.
- Optionally, <reportparameter> identifies a report parameter to pass to the report executable. If you specify a parameter in <reportparameter>, use <value> to specify the parameter value.
- Optionally, <value> specifies the value of a parameter that you pass to the report.

For example, the following formula creates a hyperlink to connect to the Ames order in the BuyerTotals report that a BIRT iServer generates in the Buyers Encyclopedia volume:

```
#hyperlink("http://Sales:8700/iportal/servlet/BuyerTotals.sox  
    ?volume=Buyers&__wait=True&Order_name=Ames",1,"Click link")
```

## Creating a hyperlink that varies at run time

To create a hyperlink that changes when a report runs, use a system-level parameter name instead of specific text in the link parameter in a formula that uses the #hyperlink report script function. When the report runs, BIRT iServer replaces the parameter name with the current system-level parameter value. You can use any system-level parameter name in a hyperlink. Report developers typically find the following two system-level parameters useful:

- AC\_EXECUTABLE\_ENCYCLOPEDIA\_PATH specifies the Encyclopedia volume path of the report executable file.
- AC\_LAST\_REPORT\_RUN\_TIME specifies the last time the report ran.

For example, the following hyperlink generates and opens the CustomerTotals report in a specific Encyclopedia volume path:

```
#hyperlink("http://Sales:8700/iportal/servlet  
/CustomerTotals.sox?volume=AC_EXECUTABLE_ENCYCLOPEDIA_PATH  
&__wait=True&order_name="&[name],1,"Click link")
```

When the report runs, BIRT iServer generates the hyperlink and inserts the path of the Encyclopedia volume where AC\_EXECUTABLE\_ENCYCLOPEDIA\_PATH appears. If the parameter value changes, the hyperlink changes accordingly.

# 16

## **Using a spreadsheet report on a production system**

This chapter contains the following topics:

- Publishing a spreadsheet report to a BIRT iServer System
- Configuring a data source for a production environment

---

## Publishing a spreadsheet report to a BIRT iServer System

When you publish a spreadsheet report, you store a copy of the report executable file on a server to make the report available to other users. To publish an Actuate report, you use Actuate BIRT iServer or BIRT Deployment Kit. Publishing to BIRT Deployment Kit requires copying files to the file system that BIRT Deployment Kit accesses. A report user locates and views a report using Information Console.

When you publish a report to BIRT iServer, you use BIRT Spreadsheet Designer's publishing feature to upload the report executable file to BIRT iServer and store it in a repository called an Encyclopedia volume. To publish a spreadsheet report, you must have a user login for the Encyclopedia volume and a BIRT iServer profile. This chapter explains how to create a report executable file, create an profile, and publish the report executable file to an Encyclopedia volume on BIRT iServer.

After you publish a report file, you can use BIRT iServer to manage and deliver the report. For example, you can control user access to a report, run a report, schedule a job, distribute a report over the web, archive an Encyclopedia volume item, and manage version control. For more information about BIRT iServer and Encyclopedia volumes, see *Using Information Console*.

Actuate Information Console supports accessing and working with information using a web browser. A report that you publish to an Encyclopedia volume is visible to a user logged in to Information Console, based on the user's access privileges. For information about distributing a report using Information Console, see *Using Information Console*.

### Preparing to publish a report

Before you publish a report, you must save the report and create a report executable file. Typically, a report developer previews report output from the report design file. Previewing a report design automatically saves the design file and creates a report executable file. You can also create a report executable file without previewing.

Select one of the following options to create a report executable file:



- Select Report➤Run to create an executable file, then preview the report.



- Select Report➤Run with Parameters to supply parameter values, create an executable file, then preview the report.
- Select Report➤Build to build an executable file without previewing.



## Creating a BIRT iServer profile

To publish an executable file to an Encyclopedia volume, you must have at least one BIRT iServer profile. The profile indicates the BIRT iServer, Encyclopedia volume, and destination folder to which you can publish a file. When the file you publish has the same name as an existing file in the Encyclopedia volume, the profile specifies whether to replace the existing file or create a new version of it. The profile also indicates whether to copy privileges from the latest version of a file to a new file that has the same name. Each profile can publish to a different destination and can use different publication settings.

Table 16-1 describes the information you need to set up a BIRT iServer profile.

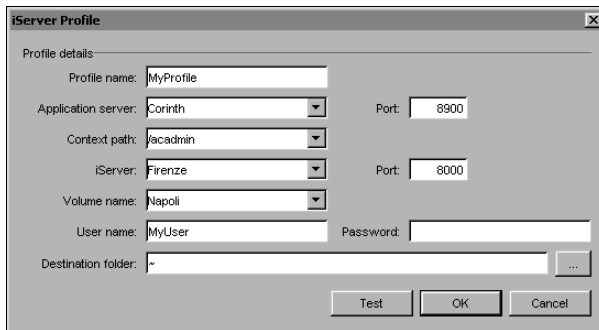
**Table 16-1** Actuate BIRT iServer profile settings

| Profile item                | Description                                                                                                                                                                                                                                                   |
|-----------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Profile name                | Descriptive name for the BIRT iServer profile.                                                                                                                                                                                                                |
| Application server and Port | Machine name and port number for the application server. The application server is the machine that hosts the BIRT iServer executable file. The default port is 8900. Actuate HTTP service listens on this port.                                              |
| Context path                | Part of the URL that specifies the volume administration path. Use /acadmin for BIRT iServer.                                                                                                                                                                 |
| BIRT iServer and Port       | Machine name and port number that BIRT Spreadsheet Designer uses to contact BIRT iServer. The default BIRT iServer port number is 8000.                                                                                                                       |
| Volume name                 | Name of the Encyclopedia volume to which you want to publish files using this profile.                                                                                                                                                                        |
| User name                   | User name with which to log into the Encyclopedia volume.                                                                                                                                                                                                     |
| Password                    | Password with which to log into the Encyclopedia volume.                                                                                                                                                                                                      |
| Destination folder          | Folder into which you want to upload the file, such as: <ul style="list-style-type: none"><li>■ ~ for the current user's home folder</li><li>■ /Sales/Forecasts for the Forecasts folder in the Sales folder in the Encyclopedia volume root folder</li></ul> |

### How to create and test a BIRT iServer profile

- 1 With a spreadsheet report open in BIRT Spreadsheet Designer, choose File➤Publish Report.
- 2 On Publish and Preview Options, in BIRT iServer profile information, choose Add.

- 3 In BIRT iServer Profile, use the descriptions in Table 16-1 as a guideline to provide profile details, as shown in Figure 16-1. In User name, type the name you use to login to the Encyclopedia volume in place of MyUser.

The image shows a dialog box titled "iServer Profile". It contains several fields for configuring a BIRT iServer profile. The fields are: Profile name (text box with "MyProfile"), Application server (dropdown menu with "Corinth"), Port (text box with "8900"), Context path (dropdown menu with "/acadmin"), iServer (dropdown menu with "Firenze"), Port (text box with "8000"), Volume name (dropdown menu with "Napoli"), User name (text box with "MyUser"), Password (text box), and Destination folder (text box with "\*"). At the bottom, there are three buttons: "Test", "OK", and "Cancel".

**Figure 16-1** BIRT iServer Profile

- 4 Choose Test.

BIRT Spreadsheet Designer tests the BIRT iServer connection using the values you provide. If the connection succeeds, a confirmation message appears. If the connection fails, an error message appears. Choosing Test does not verify the presence of the destination folder.

- 5 Choose OK to close the confirmation message.
- 6 Choose OK to close BIRT iServer Profile.

Publish and Preview Options appears.

## Publishing a report

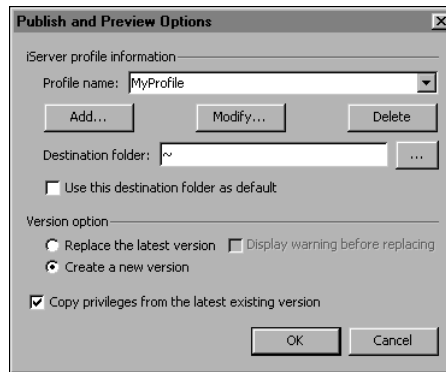
When you publish a report executable file, BIRT Spreadsheet Designer:

- Generates an executable file from the report design
- Logs in to the Encyclopedia volume that your BIRT iServer profile uses
- Uploads the executable file to the destination folder your profile specifies
- Performs other tasks your publication settings require, such as opening your destination folder, displaying the output, or copying privileges from an existing version of the file

### How to publish an executable file to an Encyclopedia volume

- 1 With a spreadsheet report open in BIRT Spreadsheet Designer, choose File➤Publish Report.
- 2 In Publish and Preview Options:
  - 1 In BIRT iServer profile information:

- ❑ Select a profile name from the drop-down list.
  - ❑ Select a destination folder.
- 2 In Version options, select Create a new version, as shown in Figure 16-2.



**Figure 16-2** Publishing an executable file to an Encyclopedia volume

- 3 Choose OK.

The report executable file uploads to the BIRT iServer. To view the report, you must run the report using BIRT iServer or Information Console. For more information about BIRT iServer and Encyclopedia volumes, see *Managing an Encyclopedia Volume*.

## Managing file versions

When a file you publish has the same name as an existing file in the destination folder in the Encyclopedia volume, you can use Version options to replace the existing file or create a new version of it. If you replace the existing file, you can display a warning message before overwriting the file.

## Copying file properties

When a file you publish has the same name as an existing file in the destination folder in the Encyclopedia volume, you can copy the permissions settings of the existing file to the file you publish. The permissions settings you copy come from the latest version of the existing file. You can copy permissions whether you replace the existing file or create a new version of it.

## Naming a report

Typically, a spreadsheet report that BIRT iServer generates from a report executable file has the name of the report design. For example, if you publish and run the report executable file EuropeSales.so, then the spreadsheet report file that BIRT iServer generates has the name EuropeSales.

Alternatively, you can use a system parameter in a report design to define a new name for a spreadsheet report. BIRT iServer assigns or dynamically generates the new name for the spreadsheet report file when the report executable file runs. A report name you define in a report design file using a system parameter overrides a report name value or expression set on BIRT iServer.

To add the name for the spreadsheet report to the report design, create a defined name for the system parameter, `AC_REPORT_OUTPUT_PATH`, then set the report name on Requester page:

- To assign a static name to a spreadsheet report file, type a slash, then the new name characters. For example, including the following value in the `AC_REPORT_OUTPUT_PATH` system parameter produces a report named NewReport:

```
/NewReport
```

- To generate a new name for a spreadsheet report file dynamically, use spreadsheet functions to create a name expression. Precede a formula with an equal sign and enclose text values in quotation marks. For example, to use a date in the name, you can use the today function. Including the following expression in the `AC_REPORT_OUTPUT_PATH` system parameter produces a report name that begins with today's date:

```
=CONCATENATE("/",TEXT(TODAY(),"m-d-yy"), "_Sales.xls")
```

For example, if the report runs on June 30, 2008, the report name is:

```
6-30-08_Sales.xls
```

When you assign a static name value or define a name value using a function expression, use the following guidelines:

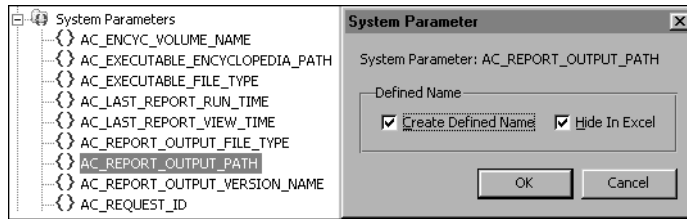
- The first character in a name must be a slash.
- Do not use number formats that use slash separators. BIRT iServer interprets a slash as a directory separator and creates new directories to match. For example, the following name produces a file called 05.xls in the /6/30 directory:

```
/6/30/05
```

After you set the report name on Requester page, you publish the report executable file to the Encyclopedia volume.

#### **How to name a spreadsheet report using a system parameter**

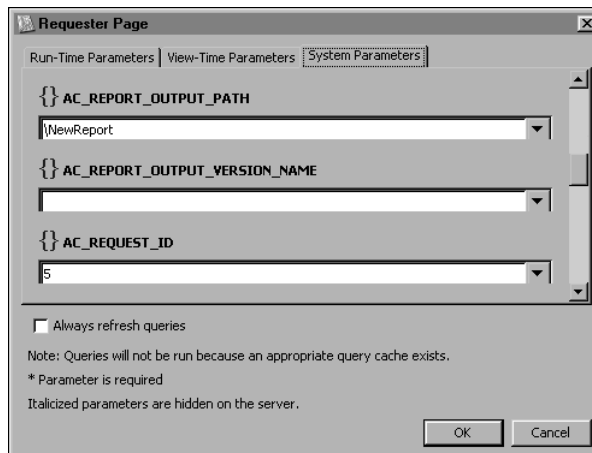
- 1 Before publishing a spreadsheet report, in Data Explorer, select System Parameters, then double-click `AC_REPORT_OUTPUT_PATH`.
- 2 On System Parameter, select Create Defined Name, as shown in Figure 16-3, then choose OK.



**Figure 16-3** Creating a defined name for a system parameter



- 3 Choose Report→Run with Parameters.
- 4 In Requester Page, choose System Parameters to see all system parameters.
- 5 Scroll to AC\_REPORT\_OUTPUT\_PATH, select Default Value, then type a name value or expression, such as /NewReport shown in Figure 16-4.



**Figure 16-4** Typing a system parameter name in Requester Page

- 6 Choose OK.

When you publish, then run a report that includes a defined name parameter, you generate a report that uses the defined name as its name. For example, the report output generated from the preceding example has the name NewReport. .

## Configuring a data source for a production environment

In BIRT Spreadsheet Designer, you create a data source to provide the connection information for a database or other data source. When you run a report using Actuate BIRT iServer System, you can use a different data source connection than the connection that you provided in the report. You can change the connection information for one report or update the connection information for every report

that you run on a BIRT iServer machine. To set run-time connection information, you use a connection configuration file.

## Using a connection configuration file

A connection configuration file is an XML file contains data source connection settings to use when Actuate BIRT iServer system runs a report. The settings in a connection configuration file can override the settings in a report. For example, you can develop a report using one data source, such as a test database. Then, you create a connection configuration file to run the report with a different data source, such as the equivalent production database. To change or add connection information for a spreadsheet report, you create or modify an existing connection configuration file with the connection information.

You can use the same connection configuration file for every spreadsheet report that you run on an Actuate BIRT iServer. BIRT reports, e.reports, and spreadsheet reports share the same connection configuration file. Each entry in the file is specific to a particular product.

Actuate BIRT iServer expects the file to be in UTF8 encoding, which supports a wide variety of characters. You can also use a file with only ASCII characters.

There is no default location for the connection configuration file. To use a connection configuration file, you create the file and then specify its name and location using the ConnConfigFile parameter in Actuate Management Console. Set the ConnConfigFile parameter to the absolute path and name of your connection configuration file. If you have a cluster of BIRT iServers, each BIRT iServer in the cluster must have access to the file. The path can be a local absolute path on each machine and must be specified for each BIRT iServer in the server configuration. If you use a single copy of the file for a cluster, put the file in a shared location and then specify the path to that shared location for all BIRT iServers in the cluster.

When you run a report, Actuate BIRT iServer uses the ConnConfigFile parameter to identify where to find a configuration file. Actuate BIRT iServer uses the connection information in the file to connect to a data source. If no configuration file exists, or if the file does not correctly specify a data source, then Actuate BIRT iServer uses the connection information from the data source definition in the spreadsheet report.

## Setting up a connection configuration file

To create a connection configuration file, create an XML file and provide connection values for one or more data sources. Each entry in a connection configuration file must include the name of the data source and one or more of the available properties for a connection to that data source. For example, you could use the user name and password properties to change the account that accesses a particular data source. You can include entries for data sources in BIRT

Spreadsheet Designer reports in the same file as entries for Actuate Basic and BIRT data sources.

To override the connection information in a BIRT Spreadsheet Designer data source, create a `ConnectOptions` element with one or more property sub-elements, using the following syntax:

```
<Config>
  <Runtime>
    <ConnectOptions Type="EssDataSource.connection_name">
      <Property PropName="datasource"> data_source_name
    </Property>
      <Property PropName="property_name"> prop_value </Property>
    </ConnectOptions>
  </Runtime>
</Config>
```

where

- `Connection_name` is an identifier used to identify a set of connection properties for a data source. The connection name must be unique within the set of connection name identifiers for a particular product. However, you can use the same connection name for two different products. For example, you can use `Production_db` as a connection name for a BIRT Spreadsheet Designer connection and for a BIRT connection. The connection name does not need to match the name of the data source in the report.
- `Data_source_name` is the name of the data source that the report uses. The data source name must match the name of the data source in the report.
- `Property_name` is the case-insensitive name of the property to set.
- `Prop_value` is the property's value.

After you create the connection configuration file, specify its name and location using the `ConnConfigFile` parameter in Management Console.

Table 16-2 describes available connection configuration properties for an Actuate Data Integration Service data source.

**Table 16-2** Actuate Data Integration service connection configuration file properties

Property name	Description
Datasource	The name of the BIRT Spreadsheet Designer Actuate Data Integration Service Connection data source.
ServerUri	URI of Actuate BIRT iServer for the information object.

(continues)

**Table 16-2** Actuate Data Integration service connection configuration file properties (continued)

Property name	Description
Volume	Name of the Encyclopedia volume in which the information object appears.
Username	User name for Actuate BIRT iServer.
Password	Password for Actuate BIRT iServer.

Table 16-3 describes available connection configuration properties for a flat file data source.

**Table 16-3** Connection configuration file properties for a flat file data source

Property name	Description
Datasource	The name of the BIRT Spreadsheet Designer Flat File data source.
File	Full path or URL to the file.
Filetype	File type. Must be TEXT. Required.
Columnpositions	Comma-separated list of column positions in ascending order. If this property is set, and the filetype=TEXT, BIRT Spreadsheet Engine or BIRT Spreadsheet Designer assumes that the file's data is fixed-width text.
Characterdelimiter	Delimiter. The characters that are used to delimit the file can be a single character or a string of characters and can contain special characters, such as back slash (\). If this property is set, the filetype=TEXT, and the columnpositions property is not set, BIRT Spreadsheet Engine or BIRT Spreadsheet Designer assumes that the file's data is delimited.
Textqualifier	The character that is used to qualify text when processing delimited text. Must be a single character. This property is ignored if the data source is a delimited text file. The default value is nothing.
Treatconsecutivedelimitersasone	Must be true or false. This property is ignored if this data source is not a delimited text file. The default value is false.



**Table 16-3** Connection configuration file properties for a flat file data source

Property name	Description
Codepage	Any one of the following text encodings: ANSI UTF8 UTF16BE UTF16LE
Columnnames.delimiters	Character or list of characters to be recognize as the delimiters of the columnnames property. If this property is not set, it uses its default value, ";". This property is ignored unless this data source is a fixed-width text file.
Columnnames	Delimited list of names to assign to columns. This property is ignored unless this data source is a fixed-width or delimited text file.
Columntypes	Comma-delimited list of number formatting to assign to columns. For example, G,G,N,S,T is a list of number formats for five columns. This property is ignored unless this data source is a fixed-width or delimited text file. Valid characters include: <ul style="list-style-type: none"><li>■ G is the general format.</li><li>■ S skips the column entirely, including data.</li><li>■ T is the text format.</li><li>■ N is the number format.</li></ul>
Startrow	0-based row number from which to start processing. Only valid if filetype=TEXT. If this property is not set, it uses its default value, 0.
Refreshindex	This data source's index number in the refresh order. Any number from 0 to 128, or -1 to set no specific refresh index. The default value is -1. Lower number values indicate higher placement in the refresh order.

Table 16-4 describes available connection configuration properties for a JDBC database data source.

**Table 16-4** Connection configuration file properties for a JDBC data source

Property name	Description
Datasource	The name of the BIRT Spreadsheet Designer JDBC Source data source.
Driver	Fully qualified JDBC driver class's name. Required unless the jndi.name property is set.
Database	Database URL. Required unless the jndi.name property is set.
User	Database user name to use when establishing a connection. The default value is an empty string.
Password	Database password to use when establishing a connection. The default value is an empty string.
Jndi.name	Fully qualified JNDI name of a valid javax.sql.DataSource object in the JNDI tree. Required unless both driver and database are set. This property takes precedence over the .driver and .database properties when all three are set.
Refreshindex	This data source's index number in refresh order. Any number from 0 to 128, or -1 to set no specific refresh index. The default value is -1. Lower numbers indicate higher placement in the refresh order.
Intrxmlopt.*	An intermediate XML schema option. Replace the asterisk with a valid intermediate XML schema option's name. The value of the property must be a valid value for the given option. For more information, including valid options and values, see the setIntermediateXMLOption method in the API.

**Table 16-4** Connection configuration file properties for a JDBC data source

Property name	Description
Connectionproperty.*	<p>Prefix for properties that are extracted, put into a new java.util.Properties object, and passed to the setAdvancedConnectionProperties method.</p> <p>Example:</p> <pre>&lt;Property PropName =     "connectionproperty.username"&gt;     jim &lt;/Property&gt;</pre> <p>In this case, the property name that is extracted to the new java.util.Properties object is username, and the value of the property, jim, is set as the value of the new username property in the new Properties object.</p> <p>These properties are ignored if the jndi.name property is set. If any of these properties are set, the username and password properties are ignored.</p>

## Examining a sample connection configuration file

The following example shows how the settings in a connection configuration file can direct Actuate BIRT iServer to use a production server when it runs a report. Both the development database and the production database use the Oracle thin driver. The connection configuration file changes the SalesConnection data source from the development database, devserver, to the production database, prodserver:

```
<Config>  
  <Runtime>  
    <ConnectOptions Type="EssDataSource.SalesConn">  
      <Property PropName="datasource"> SalesConnection  
        </Property>  
      <Property PropName="database">  
        jdbc:oracle:thin:@prodserver:1521:sales </Property>  
    </ConnectOptions>  
  </Runtime>  
</Config>
```



# Personalizing BIRT Spreadsheet Designer

This chapter contains the following topics:

- Setting BIRT Spreadsheet Designer preferences
- Setting workbook properties
- Setting sheet properties
- Improving spreadsheet report performance

---

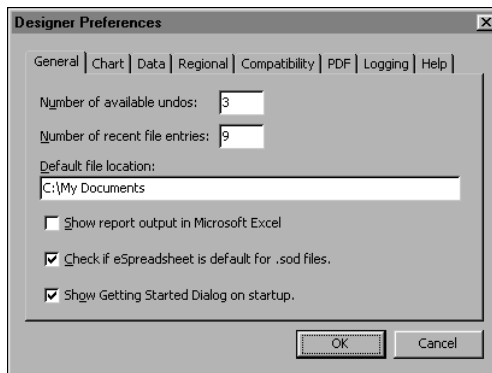
# Setting BIRT Spreadsheet Designer preferences

You can customize the appearance and functionality of BIRT Spreadsheet Designer to meet your preferences. For example, if you use many worksheets and workbooks, you can increase the number of recently opened files that appear on the File menu. You can also improve application performance by changing how BIRT Spreadsheet Designer recalculates formulas. This chapter describes how you change typical Designer preferences, workbook, and worksheet properties to personalize BIRT Spreadsheet Designer.

## Changing general designer preferences

Use Designer Preferences to change how you work with spreadsheet report files. Figure 17-1 shows typical file handling preference settings on Designer Preferences—General.

This page also supports changing the number of actions a report developer can undo using Edit→Undo. BIRT Spreadsheet Designer saves the state of the report design for the number of undo actions specified. A complex report design using a large amount of data requires more memory than a simple design. For this reason, reduce the number of undo actions if you experience out-of-memory errors during report design. A higher number of undo actions increases the flexibility of the report design process.



**Figure 17-1** Typical BIRT Spreadsheet Designer file handling preferences

### How to change file handling preferences

- 1 Choose Tools→Designer Preferences, then select General.
- 2 In General, set your preferences for the following fields. Then, choose OK.
  - In Number of available undos, type a value between 0 and 100. The default value is 3.

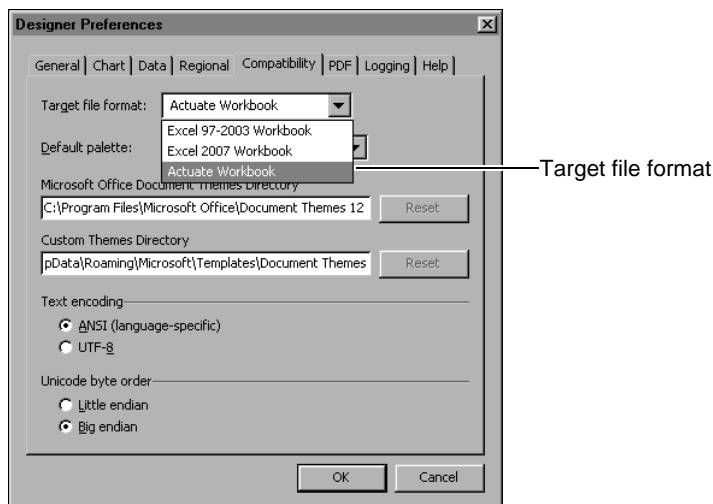
- In Number of recent file entries, to change how many recently opened files appear on the File menu, type a number between 0 and 9. The default value is 4.
- In Default file location, type a directory path to set the location in which e.Sreadsheet Designer saves a spreadsheet report design file. A report developer can change this location before saving a file.
- To open Excel and display a spreadsheet report file using the Excel application, select Show report output in Microsoft Excel.
- To open a spreadsheet design file using another application, deselect Check if BIRT Spreadsheet is default for .sod files.
- To prevent the Getting started dialog from appearing each time BIRT Spreadsheet Designer opens, deselect Show Getting Started Dialog on startup.

## Viewing a report in BIRT Spreadsheet Designer

Typically, BIRT Spreadsheet Designer displays a spreadsheet report as an Excel 97 workbook. To view a report having more than 1,048,576 rows, you must view the report as an Actuate workbook. Viewing a report as an Actuate workbook is not an available option for reports that are stored on an Encyclopedia volume.

### How to view a report as an Actuate workbook

- 1 Choose Tools→Designer Preferences, then select Compatibility.
- 2 In Compatibility, in Target file format, select Actuate workbook, as shown in Figure 17-2, then choose OK.

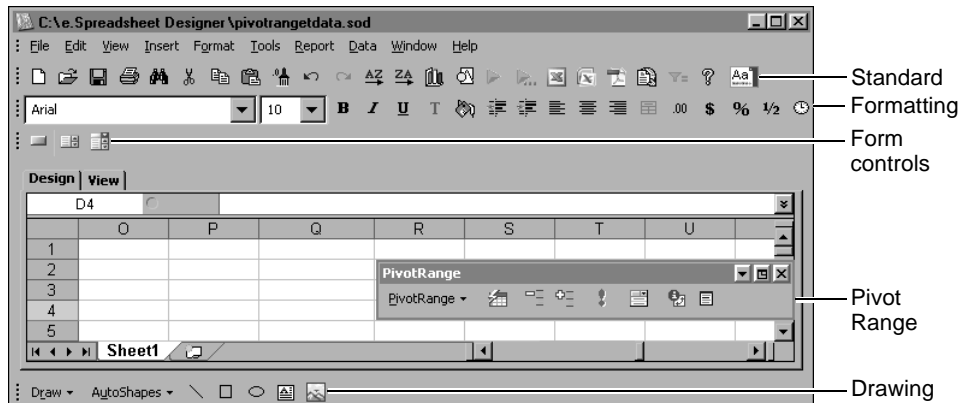


**Figure 17-2** Setting how BIRT Spreadsheet Designer shows report output

To view a report as an Excel workbook when the BIRT Spreadsheet Designer target file format is Actuate workbook, choose Report and then select Open with Excel 97-2003 or Open with Excel 2007.

## Moving a toolbar

The BIRT Spreadsheet Designer interface provides five toolbars. Figure 17-3 shows each BIRT Spreadsheet Designer toolbar docked in a typical location. The PivotRange toolbar typically appears undocked at the upper right, but overlays the worksheet in this example, to compact the view.



**Figure 17-3** BIRT Spreadsheet Designer toolbars in typical locations

### How to move a toolbar

For a docked toolbar, move the cursor over the three dots at the left side. For an undocked toolbar, move the cursor over the grey bar at the top. When the cursor appears as crossed lines, drag the toolbar and drop it in a different location.

Alternatively, right-click, then choose one of the following options from the context menu:

- To dock a toolbar above the workbook window, choose Dock on Top Side.
- To dock a toolbar below the workbook window, choose Dock on Bottom Side.
- To undock a toolbar, choose Undock.
- To move a docked toolbar up or down in a stack of toolbars, choose Move Up or Move Down.
- To hide a toolbar, choose Close. To show a hidden toolbar, use View→Toolbars.

When BIRT Spreadsheet Designer opens next, each toolbar appears positioned as it was when BIRT Spreadsheet Designer closed.

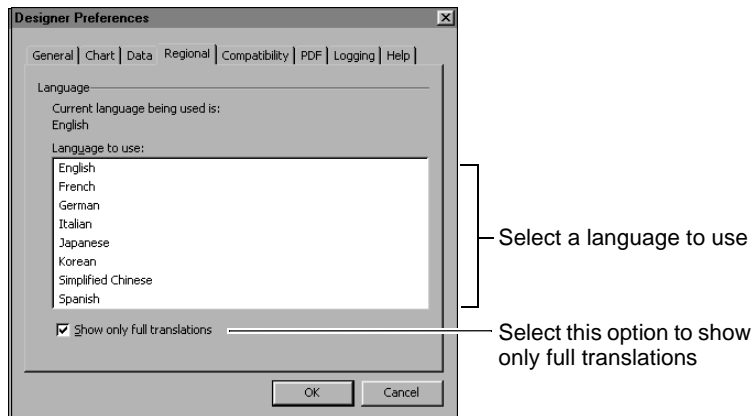


## Changing language preferences

BIRT Spreadsheet Designer features typically appear in English. Use Preferences—Regional to change the language in which text on menus and dialog boxes appears. For more information about setting regional preferences, see Chapter 22, “Using BIRT Spreadsheet Designer with multiple locales.”

### How to change language settings

- 1 Choose Tools→Designer Preferences, then select Regional.
- 2 In Regional, select language and translation options, as shown in Figure 17-4, then choose OK.



**Figure 17-4** Setting regional preferences

## Changing compatibility preferences

The settings you select on Compatibility determine:

- The target file format of a new spreadsheet report workbook
- The default color palette for a new spreadsheet report workbook
- A file location for the Microsoft Office document themes files
- A file location for custom themes files
- An encoding format to use when no code page exists for a workbook

### Selecting a target file format

The target file format determines how a new spreadsheet report workbook appears on View when you preview report output. BIRT Spreadsheet Designer uses one of three target file formats. An Actuate workbook can contain unlimited rows and columns. No workbook features appear hidden when you select Actuate workbook as the target file format.

Selecting the target file format Excel 2007 Workbook limits a workbook as follows:

- Hides rows with row number greater than 1048576
- Hides columns with column number greater than XFD
- Locks validation rules
- Prevents user access to row and column headings

Selecting the target file format Excel 97-2003 Workbook introduces all of the above limitations and the following:

- Hides rows with row number greater than 65536
- Hides columns with column number greater than IV
- Prevents selecting theme-based colors
- Limits custom color selection

The default target file format for a new spreadsheet report workbook is Excel 97-2003 Workbook.

## Selecting the default palette

The default palette determines the set of colors available as fill, border and font formatting styles for cells in a new spreadsheet workbook. Typically, a BIRT Spreadsheet Designer workbook includes the Excel 97 color scheme as the default color palette.

Typically, BIRT Spreadsheet Designer uses the color palette files in a directory that contains Microsoft Office Themes files. You can also define a directory location for customized themes files.

## Selecting encoding options

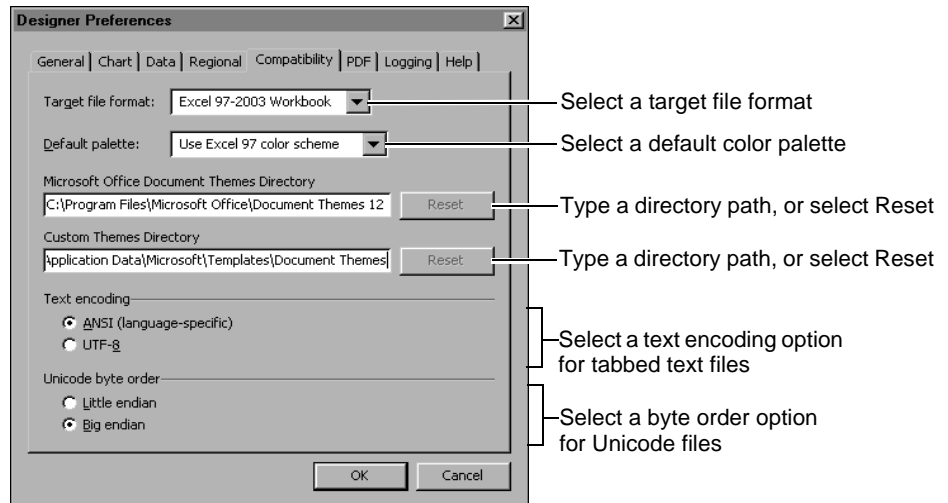
The code page specifies the encoding method an application uses to store text in a file. If the contents of a workbook file do not indicate the code page, BIRT Spreadsheet Designer applies a selected encoding method that depends upon the file type.

BIRT Spreadsheet Designer supports two encoding methods, tabbed text and Unicode. When you select tabbed text, BIRT Spreadsheet Designer saves the file as a tab-delimited text file. When you select Unicode, BIRT Spreadsheet Designer saves the file using a character coding system that supports using the file for multiple locales.

For a tabbed text file, BIRT Spreadsheet Designer supports ANSI, the default, and UTF-8, a Unicode translation format, as the character standards. BIRT Spreadsheet Designer supports both Big-Endian and Little-Endian byte order for UTF-16 files.

### How to set compatibility preferences

- 1 Choose Tools→Designer Preferences, then select Compatibility.
- 2 In Compatibility, select options, as shown in Figure 17-5, then choose OK.



**Figure 17-5** Setting file compatibility preferences

### How to set the target file format for an existing workbook

- 1 Choose File→Save Design As.
- 2 In Save, in Files of type, select a spreadsheet workbook file type.
- 3 Choose Save.
- 4 Choose Tools→Designer Preferences, then select Compatibility.
- 5 In Target file format, select a target file format that is equally or less restrictive than the file type in which you saved the workbook in step 2.
- 6 Choose File→Open.
- 7 In Open, select the file name of the workbook you saved in step 2.

## Changing data preferences

BIRT Spreadsheet Designer data preferences affect how you create data sources, data sets and joins. Specifically, data preferences determine whether to:

- Create joins between tables that contain common columns.
- Include sample data sets.
- Show unavailable drivers for data sources.

Joins between tables that contain common columns are called automatic joins. Use Preferences—Data to change data preferences settings. For more information about data sources, data sets and joins see Part 2, “Accessing data.”

### How to change data preferences

- 1 Choose Tools→Designer Preferences, then select Data.
- 2 On Data, select data options, as shown in Figure 17-6, then choose OK.

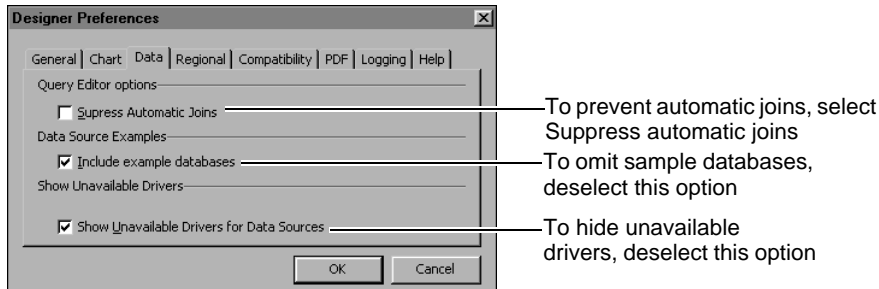


Figure 17-6 Setting Data preferences

## Changing error logging preferences

BIRT Spreadsheet Designer typically logs file error messages at the INFO level, and console error messages at the WARNING level. Typically, developers of callback classes use these messages to check their code. The log file captures messages from a Logger object. The console displays messages from a Logger object and the System.out and System.err streams. Use Preferences—Logging to change the error level that triggers display and capture of log messages. Console messages appear in the Output window in BIRT Spreadsheet Designer. To see these messages, choose View→Output. The default location for the log file for messages is:

C:\Documents and Settings\<username>\.f1j\logs

### How to change error logging levels

- 1 Choose Tools→Designer Preferences, then select Logging.
- 2 In Logging, select level options, as shown in Figure 17-7, then choose OK.

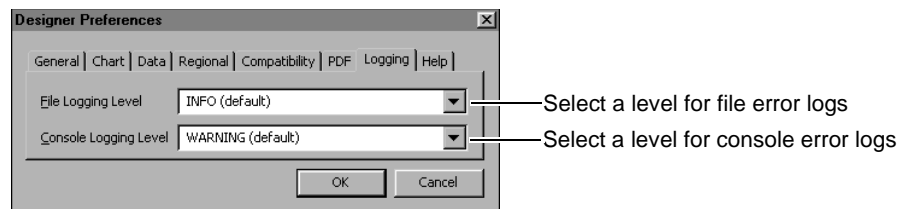


Figure 17-7 Setting error log level preferences

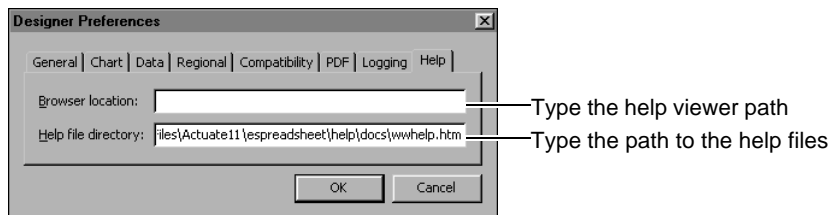
## Setting the location for online help files

BIRT Spreadsheet Designer online help files typically install in the same directory path that the application files install. When you choose Help→Contents, BIRT Spreadsheet Designer opens the files from this location. If you have installed the help files in a different folder, use Preferences—Help to instruct BIRT Spreadsheet Designer to open online help from that location. Typically, BIRT Spreadsheet Designer opens the online help using the system's default browser. You can use Preferences—Help to define a different browser.

### How to set directory paths for online help files

- 1 Choose Tools→Designer Preferences, then select Help.
- 2 In Help, type a directory path in each of the following fields, as shown in Figure 17-8, then choose OK:
  - In Browser location, type the location of the help viewer executable file.
  - In Help file directory, type the location where help content files install.

If you type no path in either of these fields, BIRT Spreadsheet Designer uses default locations to locate and display help content files.



**Figure 17-8** Setting directory paths for online help files

## Showing tips on charts

When you activate tips for a chart, then move the cursor over an element in a chart, a text box appears. Text in the box describes the chart element. For example, if you hover the cursor over a bar shown along the value-axis of a bar chart, a text box appears. Text in the box shows the value represented by the bar. When you move the cursor over a different chart element, a box appears that shows different text. BIRT Spreadsheet Designer charts typically appear with no tips.

### How to show tips on a chart

- 1 Choose Tools→Designer Preferences, then select Chart.
- 2 In Chart, select Show tips, then choose OK.

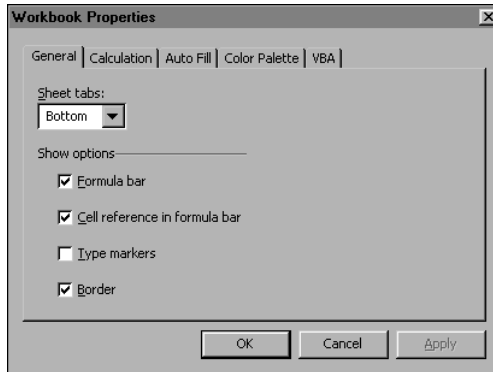
---

## Setting workbook properties

Use workbook properties to change the location and color of workbook elements such as Sheet tabs, formula bars and cell borders. You also use workbook properties to set preferences that control how BIRT Spreadsheet Designer calculates formulas and displays colors and values in worksheet cells.

### Controlling workbook element display

Use Workbook Properties—General to change the location of, or hide, the worksheet tabs in a workbook. For an Actuate workbook, you can select to show, or hide the formula bar, the cell reference in the formula bar, or the workbook border. You can also select to display markers that show the data type in each cell. Figure 17-9 shows Workbook Properties—General for an Actuate workbook target file type. For an Excel Target file type, only Sheet tabs appears.



**Figure 17-9** Examining typical workbook properties

#### How to change display settings for an Actuate workbook

- 1 Choose Tools→Workbook Properties, then select General.
- 2 In General, in Sheet tabs, select a location from the drop-down list.
- 3 In Show options, select any of the following options, then choose OK:
  - To hide the formula bar, deselect Formula bar.
  - To prevent the cell reference from appearing in the formula bar when you select a cell, deselect Cell reference in formula bar.
  - To hide the workbook border, deselect border.

## Using type markers

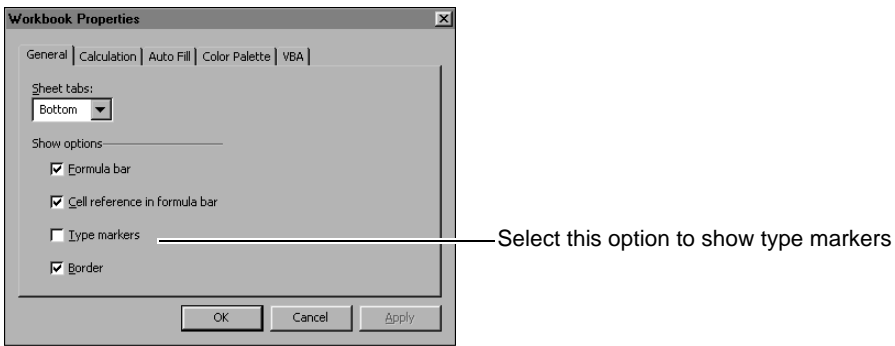
A type marker outlines a cell using colored border to identify the type of data that the cell contains. Table 17-1 lists type markers available in BIRT Spreadsheet Designer. Typically, BIRT Spreadsheet Designer does not display type markers.

**Table 17-1**      Type markers

Color	Indicates
Blue	Empty, formatted cells
Green	Values
Red	Formulas and functions

### How to display type markers

- 1 Choose Tools→Workbook Properties, then select General.
- 2 In General, select Type markers, as shown in Figure 17-10, then choose OK.



**Figure 17-10**      Selecting type markers

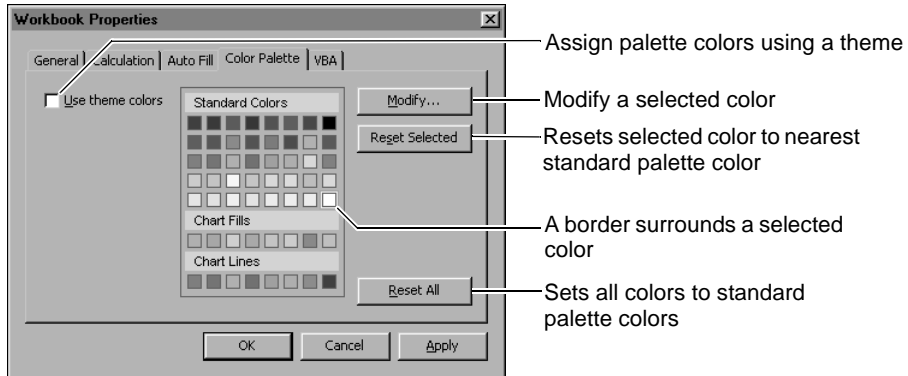
## Modifying the color palette

BIRT Spreadsheet Designer provides a default color palette that includes 56 pre-set colors. Use Workbook Properties—Color Palette set the color palette of a workbook to use theme colors. Alternatively, you can modify one or more of the color values in the color palette to your preference.

### How to change the color palette

- 1 Choose Tools→Workbook Properties.
- 2 In Workbook Properties, select Color Palette to see the settings in Figure 17-11.
- 3 To set the palette colors using a predefined theme, select Use theme colors.
- 4 To change a color, deselect Use theme colors. Under Standard Colors, select a color icon, then choose one of the following:

- To customize the selected color, choose Modify.
- To reset the selected color to the nearest standard color, select Reset Selected.
- To reset all colors to the nearest standard colors, select Reset All.



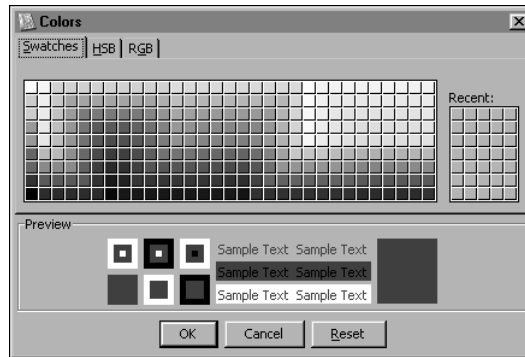
**Figure 17-11** Modifying the color palette for a workbook

### How to modify a selected color

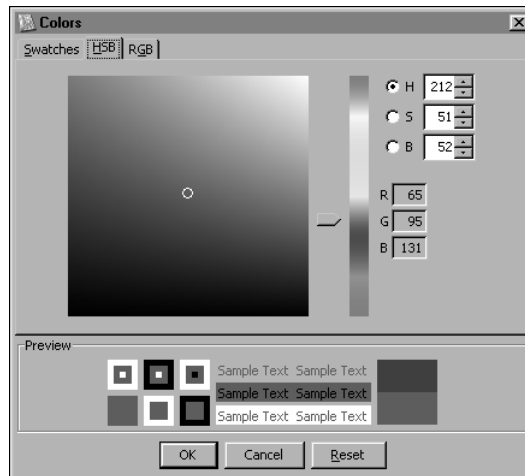
When you select a palette color and choose Modify, Colors provides three options to adjust values that comprise the selected color; Swatches, HSB, and RGB, as described in the following procedure:

- 1 In Color Palette, select a color icon, then choose Modify.
- 2 In Colors, select any of the following options and adjust color values. Then, choose OK.
  - Select Swatches. In Swatches, select a colored swatch, as shown in Figure 17-12, then choose OK.
  - Select HSB. In HSB, select values for hue, saturation and brightness, as shown in Figure 17-13, then choose OK.
  - Select RGB. In RGB, perform any of the following tasks that change red, green and blue color component values as shown in Figure 17-14. Then, choose OK.
    - ❑ Move the slider bar located between Red, Green, Blue and the corresponding color value, to modify the value of each color component.
    - ❑ Type a color component value in a selector box.
    - ❑ Change each color component value, using the up or down arrow selector.

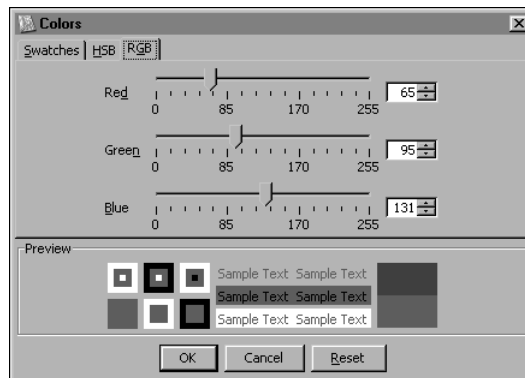




**Figure 17-12** Selecting a color swatch



**Figure 17-13** Setting hue, saturation and brightness values



**Figure 17-14** Setting Red, Green and Blue values using slider bars

## About calculation preferences

As you change a spreadsheet report design, BIRT Spreadsheet Designer adjusts cell and formula references to point to current worksheet cells. When your system processor idles, BIRT Spreadsheet Designer recalculates formulas without user input. Automatically recalculating formulas when you change large, complex, or multiple worksheets can slow system processing. In these situations, turn off automatic recalculation.

You can apply iteration to solve a formula that contains a circular reference. A circular reference occurs when a formula refers, either directly or indirectly, to the cell that contains the formula. Iteration solves a formula by repeatedly calculating the formula until meeting a specified condition. BIRT Spreadsheet Designer supports iteration through settings you specify on Workbook Properties—Calculation.

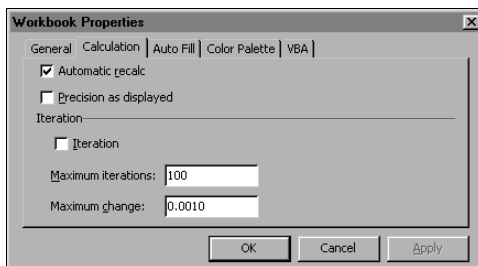
Calculation precision determines how closely the BIRT Spreadsheet engine rounds numbers used in formulas and functions. You specify a number of decimal places in each cell number format. You then set precision to use the currently displayed number format as rounding limits. Using fewer numbers after each decimal point minimizes the amount of numeric data the engine stores when it calculates each formula. Precision applies to all or no worksheets in a BIRT Spreadsheet Designer workbook. If you set precision on a workbook that already contains values and formulas, BIRT Spreadsheet Designer rounds the constant and calculated values to the number of decimal places displayed in formatted cells. You cannot undo calculations that use rounding.

## Setting calculation preferences

Use Workbook Properties—Calculation to set the following BIRT Spreadsheet Designer calculation preferences for Automatic recalculation, Iteration, and Precision.

### How to set calculation preferences

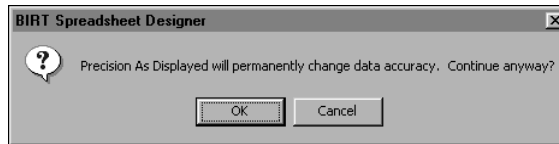
- 1 Choose Tools>Workbook Properties.
- 2 In Workbook Properties, select Calculation to see the settings in Figure 17-15.



**Figure 17-15** Setting calculation preferences

- 3 Select any of the following calculation preferences. Then, choose OK.
- To stop automatic recalculation, deselect Automatic recalc.
  - To specify circular reference iteration settings:
    - Select Iteration.
    - In Maximum iterations, type the maximum number of iterations to execute.
    - In Maximum change, type the maximum change between iterations. A smaller number increases accuracy.
  - To set precision to current cell formats, select Precision as displayed.

If you selected Precision as displayed, the message shown in Figure 17-16 appears.



**Figure 17-16** Precision message

To close the message box, choose OK.

## Recalculating worksheet formulas

You manually recalculate formulas on a worksheet by pressing specific key combinations. Typically, BIRT Spreadsheet Designer recalculates only formulas that refer to changed cells. This method is called minimal recalculation. Minimal recalculation provides results faster than full recalculation. To recalculate changed cells on a worksheet using minimal recalculation, press F9. To recalculate every formula on a worksheet, press Ctrl+Shift+F9.

---

## Setting sheet properties

Use sheet properties to control the appearance of each worksheet in your spreadsheet report design and view. You also use sheet properties to select a set of rules that each worksheet uses to evaluate formulas. You can control user access to elements and actions for each worksheet.

## Selecting evaluation rules

Typically, BIRT Spreadsheet Designer evaluates formulas in natural order, the same way Excel does. Alternatively, you can select Lotus style formula evaluation

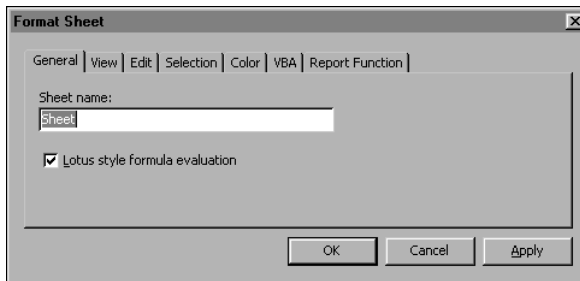
to evaluate formulas on each worksheet. You can select either set of evaluation rules for each worksheet in a workbook.

Lotus 1-2-3 evaluates formulas in a slightly different way than Excel does. Lotus 1-2-3 rules always interpret the logical value TRUE as 1, FALSE as 0. Lotus 1-2-3 also evaluates text as 0 (zero) in cells to which a formula or function argument refers.

### How to use Lotus style formula evaluation

To change the set of rules used to evaluate a spreadsheet report worksheet:

- 1 Choose Format→Sheet, then select Properties.
- 2 In Format Sheet, select General, then select Lotus style formula evaluation, as shown in Figure 17-17, then choose OK.



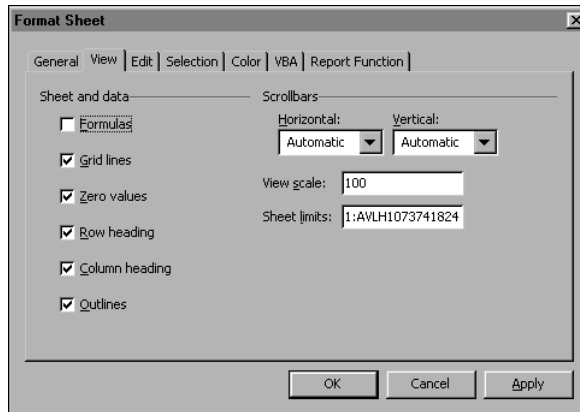
**Figure 17-17** Setting Lotus 1-2-3 evaluation rules for a worksheet

## Controlling worksheet appearance

Use Format Sheet—View to control the appearance of elements on a BIRT Spreadsheet Designer worksheet. For example, you select whether each cell shows a formula or a result. You can also select whether the value or result in a cell that equals zero shows as a zero value or as a blank. You can magnify or minimize the worksheet scale and range. You determine whether to display or hide specific worksheet elements such as horizontal and vertical scroll bars, grid lines, row and column headings, and outlines.

### How to control worksheet appearance

- 1 Choose Format→Sheet, then select Properties.
- 2 In Format Sheet, select View to see the options shown in Figure 17-18.



**Figure 17-18** Setting worksheet appearance preferences

- 3 In View, select any of the following options, then choose OK:
  - In Sheet and Data:
    - To display formulas instead of computed values in worksheet cells, select Formulas.
    - To show worksheet cells without borders, deselect Grid lines.
    - To show 0 instead of a blank in a cell where you type 0 or where a calculation result equals zero, select Zero values.
    - To show the number at the left of each row, select Row heading.
    - To show the letter at the top of each column, select Column heading.
    - To show outline levels on the worksheet, select Outlines.
  - In Scroll bars, select On to show scroll bars, Off to hide them, or Automatic to display them only for an active workbook.
  - In View scale, to magnify or shrink the worksheet view, type a value between 10 and 400. This value corresponds to a value you select using View→Zoom.
  - In Sheet limits, define the worksheet range by setting limits using cell references. You can set a worksheet to span a range from a single row and column to more than a billion rows and up to 32,768 columns. The largest worksheet you can use includes cells A1:AVLH1073741824. The largest worksheet you can use in Excel, however, includes cells A1:IV65536. For more information about worksheet size and Excel compatibility, see “Working with worksheet differences” in Appendix B, “Comparing Excel and BIRT Spreadsheet Designer.”

---

## Improving spreadsheet report performance

To improve BIRT Spreadsheet Designer performance, you can modify the BIRT Spreadsheet Designer configuration file, limit the number of actions a user can undo, and adjust your workstation memory settings. To improve performance of your spreadsheet reports, optimize query design, limit query scope and use defined report script functions. A spreadsheet report typically performs better when it runs on BIRT iServer than when it runs in BIRT Spreadsheet Designer, because each engine use a different method to cache data.

### Modifying the BIRT Spreadsheet Designer configuration file

A file called eSpreadsheet.voptions contains BIRT Spreadsheet Designer configuration information. You can modify eSpreadsheet.voptions to fit your needs. For example, to decrease application load time, you can add your JDK path to eSpreadsheet.voptions. If you use many large files simultaneously, you can increase the BIRT Spreadsheet Designer maximum memory allocation, specified in eSpreadsheet.voptions.

BIRT Spreadsheet Designer installs eSpreadsheet.voptions in the root of your BIRT Spreadsheet Designer installation directory. For example, if you accept the default installation location, eSpreadsheet.voptions installs in C:\Program Files\Actuate11\espreadsheet. You can use any text editor to open and modify eSpreadsheet.voptions.

eSpreadsheet.voptions includes the default path to the Java executable file using the following text:

```
javaexe=C:\Program Files\Actuate11\espreadsheet\jre\bin\javaw.exe
```

Before you modify eSpreadsheet.voptions, you must verify that the path to the Java executable file is correct for your system setup. If the Java executable path is incorrect, changes you make to eSpreadsheet.voptions do not affect BIRT Spreadsheet Designer behavior. For example, if you install BIRT Spreadsheet Designer to C:\espreadsheet, you must change the eSpreadsheet.voptions text that begins javaexe= to the following text:

```
javaexe=C:\espreadsheet\jre\bin\javaw.exe
```

Always save a backup copy of eSpreadsheet.voptions before making any changes to the file.

#### How to modify the BIRT Spreadsheet Designer configuration file

- 1 In a text editor, open eSpreadsheet.voptions.
- 2 Verify that the path to the Java executable is correct for your system setup.
- 3 Make the necessary changes.

- 4 Save eSpreadsheet.voptions as a text file.
- 5 Close BIRT Spreadsheet Designer, if necessary.
- 6 To start BIRT Spreadsheet Designer and run the update to the eSpreadsheet.voptions file, select Start→Programs→Actuate 11→Actuate BIRT Spreadsheet Designer.

## Improving BIRT Spreadsheet Designer performance

This section provides general tips and references for improving BIRT Spreadsheet Designer performance.

### Allocating memory

BIRT Spreadsheet Designer allocates memory to save previous user actions. To improve performance, minimize the number of saved user actions, as described in “Changing general designer preferences,” earlier in this chapter.

BIRT Spreadsheet Designer buffers result set data in memory. To improve performance, edit eSpreadsheet.voptions to set the JVM limit for BIRT Spreadsheet Designer high enough to process your optimized report design. For more information about eSpreadsheet.voptions, see “Modifying the BIRT Spreadsheet Designer configuration file,” earlier in this chapter.

### Optimizing a query

To optimize results that a SQL query returns:

- Use a WHERE clause that limits the number of rows returned from a data source.
- Select only table columns that your report requires. Use “Select\*” only when absolutely necessary.
- Simplify complicated queries that perform multiple joins, if possible, into multiple simple requests.
- Consider a query of a query. BIRT Spreadsheet Designer supports querying a query.

For more information about query optimization, see “Creating a SQL query using BIRT Spreadsheet Designer,” in Chapter 4, “Connecting to a database or JDBC data source.”

### Using report script functions

BIRT Spreadsheet Designer supports report script functions that allow you to join and create relationships between data sets. For more information about BIRT Spreadsheet Designer report script functions, see Chapter 18, “Report script reference guide.”





# Part Four

---

## Extending BIRT Spreadsheet Designer



# Report script reference guide

This chapter contains the following topics:

- About BIRT Spreadsheet Designer report script
- BIRT Spreadsheet Designer report script function overview
- BIRT Spreadsheet Designer report script function reference

---

## About BIRT Spreadsheet Designer report script

This chapter provides a reference guide to BIRT Spreadsheet Designer report script functions, the functions and expressions you use to develop a data range. When you type a report script function, use the following guidelines:

- Report script is available only in a data range. You cannot use report script functions in a cell outside a data range or in a pivot range.
- Any part of a data range can accept multiple report script functions. Multiple functions are interpreted from left to right. For example, consider the following expression that applies to a row or column section:

```
group( [Product] ) if( sum([sales])<1000, delete( ) )
```

The `group( )` function is interpreted first. The `if( )` function is interpreted second.

- When typing report script in a cell, precede the report script function or series of report script functions with a number sign (#), as shown in the following example:

```
#average( [quantity] * [itemPrice] )
```

In any other part of a data range, do not precede the report script with a number sign.

- Surround text expressions or values in double or single quotation marks. You must also use quotation marks to surround report script functions that evaluate to text expressions or formulas. To display double quotation marks in the text, use single quotation marks to surround the text expression. Alternatively, in a complex expression that uses both double and single quotation marks, escape the quotation mark by typing it twice, as shown in the following example:

```
#write( "Ernesto's son said, ""Hola!"" " )
```

- To concatenate text values, use an ampersand (&), as shown in the following example:

```
#write( [lastName] & ", " & [firstName] )
```

- Type data set field names as they appear in the data source. If a data set field name does not begin with an alphabetic character or if the name includes characters other than alphanumeric characters, underscores, or periods, you must surround the name in square brackets, [ ]. To make report script functions more clear to report developers, you can enclose all field names in square brackets. In the following example, the data set has a field, count. Using square brackets improves the readability of the expression:

```
#count( distinct( [count] ) )
```

- To access a data set that is not the default one for a data range, precede a report script function by the data set name and a colon (:). Similar to a field name, if the data set name includes characters other than alphanumeric characters, underscores, or periods, you must surround the data set name in square brackets, [ ], as shown in the following example:

```
[Data Set 1]:select( [productLine] = "Planes" )
```

- Surround lists of values in curly brackets, { }, as shown in the following example:

```
#sum( orderValue ) if( "Manager" in security( ) )
```

## Placing report script functions in a data range

Each report script function is available in one or more areas of the data range. When you type an expression in the data range, BIRT Spreadsheet Designer checks that each function in the expression is applicable to the area where you put the expression. If a function is not available in that area, BIRT Spreadsheet Designer displays an error message. The following list describes the types of report script functions:

- **General report script functions**  
These are report script functions that you can use in any area of the data range. You use general report script functions to perform functions such as testing for null values, converting text values to upper case, or creating a computed field.  
  
General report script functions also include report script functions that you use to create aggregated totals or comparisons in a row or column section or the body of a data range. An aggregate report script function is different from a spreadsheet formula. You use an aggregate report script function to summarize data fields. To create a formula that summarizes data range cells or ranges, you must use text and report script functions to create the formula.
- **Data area report script functions**  
You use these report script functions to control the data in a cell. For example, you can use a data area report script function to format a value in a cell.
- **Row section and column section report script functions**  
You use these functions to control the data in a row or a column section. For example, you use a row or column report script function to identify the data a section uses.
- **Sheet area report script functions**  
These are report script functions that you use to control the data on a sheet. For example, you use a sheet function to determine how the report splits data into separate worksheets.

- Chart report script functions

Charts can use a subset of the functions that are available in the data area of a data range. For example, you can display the distinct count of a set of values or the maximum value in a chart.

## Using mathematical operators with report script functions

You can use common mathematical operators in an expression. Table 18-1 lists the available operators.

**Table 18-1** Report script function operators

Operator	Definition
=	Equals.
<	Less than.
>	Greater than.
<=	Less than or equal to.
>=	Greater than or equal to.
<>	Not equal to.
in	Use to test an array of possible values. Surround the array in curly brackets, {}. Some functions return an array. Use the in operator to test these return values.
and	Use to define multiple conditional expressions. For the complete expression to evaluate to true, both expressions must evaluate to true.
or	Use to define multiple conditional expressions. For the complete expression to evaluate to true, at least one expression must evaluate to true.
not	Use to define a conditional expression that must evaluate to false.
xor	Use to define multiple conditional expressions. For the complete expression to evaluate to true, only one expression can evaluate to true.
like	Use to match a wildcard expression. This operator is not case sensitive. For example, [state] like "N*" matches any of NB, Nb, nB, nb, NY, Ny, nY, and ny.
+	Add.
-	Subtract.
*	Multiply.
/	Divide.

## Using a list

Some report script functions, such as `color()` and `security()`, accept a list as a parameter value or return a list. To define a list of static values, you surround the values in curly brackets, `{ }`, as shown in the following example:

```
#{"AK", "NY", "CA"}
```

To create a dynamic list from a data set field, use the `distinct()` function, as shown in the following example:

```
#distinct([stateCode])
```

## Using operators with lists

You can use the addition (+) and subtraction (-) operators with lists. When you add one list to another, the contents of the list on the right are appended to the list on the left. When you subtract one list from another, any element in the list on the right is removed from the list on the left. For example, the following expression:

```
#{1,1,2,3} + {1,3}
```

displays

```
1,1,2,3,1,3
```

The following expression:

```
#{1,1,2,3} - {1,3}
```

displays:

```
2
```

## Sorting a list

To sort the values in a list, you use the unary plus (+) and minus (-) operators, as shown in the following examples. The plus operator sorts the values in ascending order. The minus operator sorts the values in descending order:

```
#+{"AK", "NY", "CA"}
```

returns:

```
AK, CA, NY
```

```
#-distinct([creditRating])
```

returns the values in the data set field, `creditRating`, in descending order.

## Using wildcards with report script functions

You can use the comparison operator, like, to filter data rows that match wildcard values. typically, you use the like operator in conjunction with one or more of the following three wildcard symbols:

- Asterisk (\*). Use \* as a placeholder for zero or more characters.
- Question mark (?). Use ? as a placeholder for one character.
- Tilde (~). Use ~ to search for an asterisk, a question mark, or a tilde symbol.

For example, the following expression selects all records where Customer includes the text United:

```
select([Customer] like "*United*")
```

The following expression tests to see if Customer begins with United. If the value begins with United, BIRT Spreadsheet Designer returns Yes. If the value does not begin with United, BIRT Spreadsheet Designer returns No:

```
#if([Customer] like "United*", "Yes", "No")
```

The following expression selects data for which Customer matches United\* and two more characters:

```
select([Customer] like "United~*??")
```

## Using multiple report script functions in a cell or section

You can add one or more report script functions to a cell or section. BIRT Spreadsheet Designer executes a series of report script function expressions in the same order in which it reads them, from left to right and from top to bottom. You can place report script functions on the same line or on different lines in a cell. To start a new line, use Alt+Enter. You can use a report script function more than once in the same cell or section.

## Accessing a non-default data set

You can use values from a non-default data set as well as values from the default data set in report script. To access a non-default data set, precede a function or data set field name by the data set name and a colon (:). To process the current set of rows in the non-default data set, precede a function by the data set name, as shown in the following example:

```
[Second Data Set]:group([productName] )  
select([MODEL]=groupVal( ) )
```

To process a value from the first row in the non-default data set, precede a field name by the data set name. Typically, you use this syntax only if the value of the field is the same for the entire set of rows or if the data set has only one row. The



following code is equivalent to the preceding example because the field, `productName` is the same for all rows in Second Data Set in the current group:

```
[Second Data Set]:group( [productName] )
select( [MODEL]=[Second Data Set]:[productName] )
```

The following examples are not equivalent:

```
select( [inStock] > [Second Data Set]:average( [inStock] ) )
select( [inStock] > average( [Second Data Set]:[inStock] ) )
```

The first line compares the value of the `inStock` field in the default data set to the average of the `inStock` field in all current rows in Second Data Set. The second line compares the value of the `inStock` field in the default data set to the average of the `inStock` field in the first row in Second Data Set. This line probably does not produce a meaningful result. Consider a set of rows in Second Data Set such that the average of `inStock` for Second Data Set is 200 and the value of `inStock` in its first row is 0. In this case, the first line selects rows from the default data set having a value of `inStock` greater than 200, and the second line selects all rows from the default data set that have a positive value for `inStock`.

### Adding a comment

A complex spreadsheet report can use hundreds of report script functions. You can add a comment to a cell to add confidential information to a report design, leave instructions for other report developers, or remind yourself of the purpose of a report script function expression. To add a comment, type two slashes (`//`), then type the comment text. BIRT Spreadsheet Designer ignores any text that follows the slashes. To add a report script function after the comment text in a cell, you must start a new line.

---

## BIRT Spreadsheet Designer report script function overview

Table 18-2 lists all the available report script functions and the areas of a data range in which you can use each function. In the table, an X indicates that a function is available in the corresponding area.

**Table 18-2** Report script functions

Report script function	Data area	Row or column area	Sheet area	Charts
average	X	X	X	X

*(continues)*

**Table 18-2** Report script functions (continued)

Report script function	Data area	Row or column area	Sheet area	Charts
cells	X			X
chart			X	
col	X			
color	X	X	X	
count	X	X	X	X
date	X	X	X	X
day	X	X	X	X
delete		X	X	
detail		X	X	
distinct	X	X	X	X
error	X	X	X	X
eval	X	X	X	X
exclude		X	X	
extract	X	X	X	X
first	X	X	X	X
fontcolor	X	X	X	
format	X			
formula	X			
getEntry	X	X	X	X
group		X	X	
groupNum	X	X	X	X
groupVal	X	X	X	X
hour	X	X	X	X
hyperlink	X			
if	X	X	X	X
include		X	X	
isnull	X	X	X	X
last	X	X	X	X
len	X	X	X	X
lock	X	X	X	

**Table 18-2** Report script functions (continued)

Report script function	Data area	Row or column area	Sheet area	Charts
lower	X	X	X	X
max	X	X	X	X
merge	X	X		
min	X	X	X	X
minute	X	X	X	X
mod	X	X	X	X
month	X	X	X	X
name	X	X	X	
now	X	X	X	X
nth	X	X	X	X
pageBreak		X		
product	X	X	X	X
quarter	X	X	X	X
rollup		X	X	
round	X	X	X	X
row	X			
rownum	X	X	X	X
second	X	X	X	X
sectioncount	X	X		
security	X	X	X	X
select		X	X	
setchart	X	X	X	X
setentry	X	X	X	X
sheet	X	X	X	
size		X		
stdev	X	X	X	X
stdevp	X	X	X	X
substitute	X	X	X	X
sum	X	X	X	X

*(continues)*

**Table 18-2** Report script functions (continued)

Report script function	Data area	Row or column area	Sheet area	Charts
time	X	X	X	X
today	X	X	X	X
tonumber	X	X	X	X
toText	X	X	X	X
trim	X	X	X	X
trunc	X	X	X	X
upper	X	X	X	X
values	X	X	X	X
var	X	X	X	X
varp	X	X	X	X
write	X			
year	X	X	X	X

## BIRT Spreadsheet Designer report script function reference

This section provides an alphabetical list of all report script functions. Each entry includes a general description of the function, its syntax, a description of its parameters, and states where in a data range this function is available.

### average

**Syntax** average( numeric\_expression )

**Description** Computes the average value in a data set field or expression. Average( ) ignores any null values in the field.

**Parameter** **numeric\_expression**  
The data set field or expression to average.

**Available in** All areas of a data range.

**Example** The following example computes the average value in the data set field, sales:  
average( [sales] )

## cells

<b>Syntax</b>	<code>cells( section )</code>
<b>Description</b>	<p>Returns a spreadsheet reference to the range of cells in a row or a column section. If <code>section_name</code> specifies a row section, the range is the set of cells for that section in the current column. If <code>section_name</code> specifies a column section, the range is the set of cells for that section in the current row.</p> <p>If <code>section_name</code> specifies a section that is immediately inside the current section, the returned range is contiguous, such as <code>A22:N22</code> or <code>F76:F97</code>. If section name specifies a section that is nested more deeply inside the current section, can return a disjoint range, such as <code>H7:H17,H19:H29,H31:H41</code>.</p> <p>Typically, you use the returned range to process the values of those cells in an Excel formula or as the argument to another report script function.</p>
<b>Parameters</b>	<p><b>section_name</b> The name of a section.</p>
<b>Available in</b>	The data area and charts.
<b>Example</b>	<p>In the following example, the <code>cells( )</code> function returns the range reference for the <code>sales.city.state.line1</code> row or column section. The <code>formula( )</code> function creates an Excel formula of the form, <code>SUM(&lt;range&gt;)</code>:</p> <pre>#formula("SUM((" &amp; cells( sales.city.state.line1 ) &amp; "))")</pre>

## chart

<b>Syntax</b>	<code>chart( chart_name, chart_data, series_in_rows )</code>
<b>Description</b>	Identifies the source data of a chart that uses data range data and appears in the completed worksheet. When you run the report, <code>chart( )</code> links the chart design to the updated data in the data range. To link data to a chart that appears on a sheet other than the data range worksheet, use <code>setchart( )</code> .
<b>Parameters</b>	<p><b>chart_name</b> The name of the chart to update. The chart name cannot be the default chart name.</p> <p><b>chart_data</b> The defined name that identifies the data to use in the chart.</p> <p><b>series_in_rows</b> Indicates whether BIRT Spreadsheet Designer defines rows or columns as series. Values include <code>True</code> or <code>False</code>.</p>
<b>Available in</b>	The sheet area.
<b>Example</b>	<p>The following example links the chart named <code>QuantityChart</code> to the data that the <code>Quantity</code> defined name identifies. The chart uses columns as series:</p> <pre>chart( "QuantityChart","Quantity", False )</pre>

## col

**Syntax** col( )

**Description** Returns the identifier of the worksheet column in which col( ) is executed.

**Available in** The data area.

**Example** The following example identifies the worksheet column in which the report script function is executed. For example, if you type this report script function appears in a cell that becomes cell B1 in the data range, it returns B:

```
#col( )
```

## color

**Syntax** color( color\_expression )

**Description** Sets the cell pattern foreground color of the specified area. This color is the background color of the cells.

**Parameter** **color\_expression**

An expression that evaluates to a color. You can specify the color value in the following two ways:

- A string containing an RGB hexadecimal value. If the output document is an Excel 97 file, this color is mapped to the closest color on the Excel color palette.
- A list of three integer values between 0 and 255 that specify the red, green, and blue components of the color. If the output document is an Excel 97 file, this color is mapped to the closest color on the Excel color palette.

**Available in** All areas of a data range except charts.

**Example 1** The following example sets the color to magenta:

```
#color( "FF00FF" )
```

**Example 2** The following example sets the color to green:

```
#color( { 0, 255, 0 } )
```

## count

**Syntax** count( [expression] )

**Description** Counts the number of values returned for the specified data set field or expression. If you do not specify a value for the parameter, count( ) counts the number of query rows that contribute to the group section, row, or column. Count( ) does not include null or empty fields.

**Parameter** **expression**

The optional data set field or expression to count.

**Available in** All areas of a data range.

**Example 1** The following example counts the number of values in the sales field:

```
count( [sales] )
```

**Example 2** The following example counts the number of values in the state field that match "NY":

```
count( [state]="NY" )
```

**Example 3** The following example counts the number of values in the state field that begin with "A" with sales greater than one million:

```
count( [state]like"A*" and [sales] > 1000000 )
```

## date

**Syntax** date( year, month, day, hr, min, sec, ms )

**Description** Returns a number that represents a date. You can use this function to perform arithmetic on date values.

**Parameters** **year**  
A one-, two-, three-, or four-digit year value to convert.

**month**  
A month value to convert.

**day**  
A day value to convert.

**hr**  
An hour value to convert.

**min**  
A minute value to convert.

**sec**  
A second value to convert.

**ms**  
A microsecond value to convert.

**Available in** All areas of a data range.

**Example 1** The following example returns December 10, 2001, as 37235:

```
#date( 2001, 12, 10 )
```

**Example 2** The following example returns the value for the date that is 45 days earlier than the current date, formatted as a date like 2007-04-14:

```
#date( year( ), month( ), day( )-45 ) format( "yyyy-mm-dd" )
```

## day

**Syntax** day( [date] )

**Description** Displays a date value as the day of the month. If there is no parameter, day( ) returns the day of the month for today's date.

**Parameter** **date**  
An optional date field to use when calculating the day values.

**Available in** All areas of a data range.

**Example** The following example displays the day part of the data set field, purchaseDate:  
`#day( [purchaseDate] )`

## delete

**Syntax** delete( )

**Description** Deletes a section or a sheet. Use delete( ) with if( ) to remove items that the group( ) report script function creates. When called from a section, the section and its child sections do not appear in the report. When called from a sheet, the sheet and its child sheets do not appear in the report.

**Available in** Row sections, column sections, and sheets.

**Example** The following example groups data by the values of the data set field, Product, then uses if( ) and delete( ) to remove the group if the sum of the sales field is less than 1000:

```
group( [Product] ) if( sum([sales])<1000, delete( ) )
```

## detail

**Syntax** detail( sort\_key )

**Description** Returns all data rows in a section. You can provide a sort key by which to order the returned data rows. To show data rows in the order in which the data set returns them, do not specify a sorting key. To sort in descending order, type a minus sign before the sort field.

Typically, a sort key is an expression based on data set fields. This type of sort key produces a dynamic set of detail sections. You can also use detail( ) to create a static set of detail sections by using literal values for sort\_key in the following ways:

- Use a list of values to create a set of detail sections having known sort key values.
- Use an integer to create that number of detail sections. In this case, the sort key values are the integers one to the value of sort\_key.



**Parameter** **sort\_key**  
An expression or list of expressions or static values to use as a sort key.  
Alternatively, use a number to create a pre-defined number of detail sections.

**Available in** Row sections, column sections, and sheets.

**Example 1** The following example returns detail data:

```
detail( )
```

**Example 2** The following example returns detail data and sorts rows in descending order on the data set field, country:

```
detail( -[country] )
```

**Example 3** The following example creates four detail sections:

```
detail( 4 )
```

## distinct

**Syntax** `distinct( expression )`

**Description** Returns only unique values for a data set field or expression.

**Parameter** **expression**  
The data set field or expression from which to return unique values.

**Available in** All areas of a data range.

**Example 1** The following example counts the unique values from the Region field:

```
#count( distinct( [Region] ) )
```

**Example 2** The following example selects rows with a DeptID data set field that matches one of the unique values from the DeptID field in the Europe data set:

```
select( DeptID in Europe:distinct( [DeptID] ) )
```

## error

**Syntax** `error( expression )`

**Description** Displays an error message. Use with `if( )` to determine when to show the message.

**Parameter** **expression**  
An error message expression. The message can consist of report script functions or text expressions.

**Available in** All areas of a data range.

**Example** The following example displays a message when a select statement returns no data:

```
select( [Status]="A" ) if( ( count( )=0,  
    error("There are no rows with Status = 0" ) )
```

## eval

**Syntax** eval( expression )

**Description** Processes a text string as a report script function or as input for a report script function at report generation time.

**Parameter** **expression**  
A report script function or a text expression.

**Available in** All areas of a data range.

**Example** The following example evaluates the parmCustID parameter value as a comma-separated list as part of the select statement:

```
select( [custID] ) in eval( "{" & :parmCustID & "}" ) )
```

## exclude

**Syntax** exclude( condition )

**Description** Uses the specified condition to filter returned data.

**Parameter** **condition**  
The criteria to use to filter the data. The row, column, or section does not include data that matches the criteria.

**Available in** Row sections, column sections, and sheets.

**Example** The following example returns data for which age is not less than 18:

```
exclude( [age] < 18 )
```

## extract

**Syntax** extract( source\_text, extract\_pattern )

extract( source\_text, length )

extract( source\_text, start, end )

**Description** Displays a portion of a text string. Use extract() with text fields or expressions only.

**Parameters** **source\_text**  
The text or text-type field from which to extract a section.

**extract\_pattern**

An expression that indicates how to select a section. Use the at sign, @, to identify the part of the source\_text to return. Use wildcards to identify the part of the source\_text to ignore.

**length**

A number that indicates how many characters to extract. If length is a positive number, extract() returns the specified number of characters, starting from the

first character. If length is a negative number, `extract()` returns the specified number of characters, starting from the last character.

**start**

A number that indicates at what character to begin the selection to extract. If start is a negative number, the selection begins at the end of the expression and goes backwards toward the beginning.

**end**

A number that indicates at what character to end the selection to extract. If end is a negative number, the selection begins at the end of the expression and goes backwards toward the beginning.

**Available in** All areas of a data range.

**Example 1** The following example returns Text:

```
#extract( "Example-Text", "*-@" )
```

**Example 2** The following example returns 20:

```
#extract( "05/20/2005", "*/@/*" )
```

**Example 3** The following example returns the domain name from each value in an e-mail address field:

```
#extract( [Email], "*~@@" )
```

**Example 4** The following example returns the text Exa:

```
#extract( "Example", 3 )
```

**Example 5** The following example returns the text ple:

```
#extract( "Example", -3 )
```

**Example 6** The following example returns the text xampl:

```
#extract( "Example", 2, 6 )
```

**Example 7** The following example returns the text ple:

```
#extract( "Example", -3, -1 )
```

## **first**

**Syntax** `first( expression )`

**Description** Returns the first non-null value in a set of records or a list.

**Parameter** **expression**

The data set field or expression from which to return the first value.

**Available in** All areas of a data range.

**Example 1** The following example displays the first non-null value for the data set field, shippedDate:

```
#first( [shippedDate] )
```

**Example 2** The following example displays the first value for the data set field, shippedDate, when the data set field, state, is neither null nor "NY":

```
#first( if ( [state]<>"NY", [shippedDate] ) )
```

**Example 3** The following example displays the first value returned by groupVal( ), which is the group key of the row section:

```
#first( groupVal( ) )
```

## fontcolor

**Syntax** fontcolor( color\_expression )

**Description** Sets the font color of the specified area.

**Parameter** **color\_expression**

An expression that evaluates to a color. You can specify the color value in the following two ways:

- A string containing an RGB hexadecimal value. If the output document is an Excel 97 file, this color is mapped to the closest color on the Excel color palette.
- A list of three integer values between 0 and 255 that specify the red, green, and blue components of the color. If the output document is an Excel 97 file, this color is mapped to the closest color on the Excel color palette.

**Available in** All areas of a data range except charts.

**Example 1** The following example sets the color to magenta:

```
#fontcolor( "FF00FF" )
```

**Example 2** The following example sets the color to green:

```
#fontcolor( { 0, 255, 0 } )
```

## format

**Syntax** format( value\_format )

**Description** Applies a value format to a cell.

**Parameter** **value\_format**

A value format string.

**Available in** The data area.

**Example** The following example displays the value of cost and formats the value as a number:

```
#[cost] format( "$#,##0_"; [Red] ($#,##0) " )
```

## formula

- Syntax** formula( expression )
- Description** Evaluates the expression, which contains a Microsoft Excel formula, using the API locale. This locale is largely equivalent to the US English locale.
- Parameter** **expression**  
The Excel formula to evaluate. This expression can contain data set fields.
- Available in** All areas of a data range.
- Example** The following example displays the sum of the values in the defined name, Country, with the value from the data set field, countryName, appended:  
`#formula( "SUM(Country" & [countryName] & ")" )`

## getEntry

- Syntax** getEntry( [sheetname], row, column )  
getEntry( cells )
- Description** Returns the computed value of a cell or range of cells. You use numeric or string expressions to specify the cell or range of cells in the spreadsheet report.  
  
If the arguments to getEntry( ) evaluate to a range of cells, getEntry( ) gets values from each output cell in a row from left to right and proceeds from top to bottom. GetEntry( ) then displays those values as a comma-separated list using ascending numeric output cell order. For example, a list generated from the range of cells A1:B4 displays in the following order:  
  
`(A1value, A2value, B1value,...B4value)`
- Parameters** **sheetname**  
An optional worksheet name. GetEntry( ) requires a sheetname parameter value only if you intend it to return values from a different worksheet. If you specify no sheetname parameter value, GetEntry( ) returns values from the worksheet in which you place the function.
- row**  
The row number of the cell in the spreadsheet report. Row numbers begin at zero. For example, to indicate row number 3, use 2.
- column**  
The column number of the cell in the spreadsheet report. Column numbers begin at zero. For example, to indicate column B, use 1.
- cells**  
An absolute or relative cell reference in a spreadsheet report.
- Available in** All areas of a data range.

**Example 1** The following example displays a single value from the first row and second column of the Sheet1 worksheet:

```
getEntry("Sheet1",0,1)
```

**Example 2** The following example displays a single value from cell C4 of the current worksheet:

```
getEntry(3,2)
```

**Example 3** The following example displays a single value from cell A1 of the Sheet1 worksheet:

```
getEntry("Sheet1!$A1")
```

**Example 4** The following example lists in ascending row and column order all values from the range that spans cells A1 to B4 on Sheet1:

```
getEntry("Sheet1!A1:B4")
```

## group

**Syntax** group( group\_key, sort\_key )

**Description** Uses a group key to break the data returned in a data range row or column into subsets known as group sections. Each group section appears in a different row or column. BIRT Spreadsheet Designer sorts groups in ascending order using the group key as the sort key. To use a different sort key, use the sort\_key parameter. To sort in descending order, type a minus sign before sort\_key.

Typically, a group key is an expression based on data set fields. This type of key produces a dynamic set of group sections. You can use group( ) to create a static set of group sections by using literal values for group\_key in the following ways:

- Use a list of values to create a set of group sections having known group keys.
- Use an integer to create that number of group sections. In this case, the group keys are the integers one to the value of group\_key.

**Parameters** **group\_key**  
An expression or list of expressions or static values to use as a group key. Alternatively, use a number to create a pre-defined number of group sections.

**sort\_key**  
An expression or list of expressions to use as a sort key.

**Available in** Row sections, column sections, and sheets.

**Example 1** The following example creates a group for each BusinessUnit value and sorts the rows in each group by SalesRep:

```
group( [BusinessUnit], [SalesRep] )
```

**Example 2** The following example creates a group for each BusinessUnit with a value greater than 2:

```
select( [BusinessUnit]>2 ) group( [BusinessUnit] )
```

**Example 3** The following example creates Region groups within the current BusinessUnit:

```
group( { [BusinessUnit], [Region] } )
```

**Example 4** The following example creates three groups with group keys, NorthEast, Central, and SouthEast, in that order and selects the data rows applicable to each group:

```
group( { "NorthEast", "SouthEast", "Central" } )  
select( [Region] = groupVal( ) )
```

**Example 5** The following example creates four groups and selects data rows for each quarter to display in the appropriate group. If there are no data rows for a quarter, the data area can display text such as "No orders for this quarter":

```
group( 4 ) select( quarter( [orderDate] ) = groupNum( ) )
```

## groupNum

**Syntax** groupNum( )

**Description** Returns the number of a group within a grouped section or sheet. If the data range contains nested groups, groupNum( ) returns the number of the innermost grouped section for its context. If no enclosing section in the data range uses either group( ) or detail( ), groupNum( ) returns 0. Similarly, if you use groupNum( ) on a sheet that does not use group( ), it returns 0. If you use this function in a cell in the data area, groupNum( ) returns a list of two group numbers. The first number in the list is the row group number. The second number is the column group number.

**Available in** All areas of a data range except in calculated fields.

**Example** The following example creates a section for data row and colors alternate rows in light gray:

```
detail( ) if( mod( groupNum( ), 2 ) = 0, color( "DDDDDD" ) )
```

## groupVal

**Syntax** groupVal( )

**Description** Returns the key value of a group within a grouped section or sheet. If the data range contains nested groups, groupVal( ) returns the key value of the innermost grouped section for its context. If no enclosing section in the data range uses group( ), groupVal( ) returns null. Similarly, if you use groupVal( ) on a sheet that does not use group( ), it returns null. If you use this function in a cell in the data area, groupVal( ) returns a list of two group key values. The first value in the list is the row group value. The second value is the column group value.

**Available in** All areas of a data range except in calculated fields.

**Example** The following example creates a section for each value of the data set field, country, then deletes the section with a group key value of USA:

```
group( country ) if( groupVal( )="USA", delete( ) )
```

## hour

**Syntax** hour( [date] )

**Description** Displays the hour part of a date-and-time value. If there is no parameter, hour( ) returns the hour value for the current time.

**Parameter** **date**  
An optional date field to use when calculating the hour value.

**Available in** All areas of a data range.

**Example** The following example displays the hour part of the data set field, startLog:  

```
#hour( [startLog] )
```

## hyperlink

**Syntax** hyperlink( link, target\_type[, tooltip] )

**Description** Creates an embedded hyperlink in a cell.

**Parameters** **link**  
A string that specifies a hyperlink target.

**target\_type**  
An integer that specifies the type of target to which the hyperlink points, as described in the following list:

- 0, a range in the same spreadsheet report as the hyperlink
- 1, an absolute URL
- 2, a relative URL
- 3, an absolute file path
- 4, a relative file path
- 5, a network file path

**tooltip**  
An optional string that displays information when the user hovers the mouse pointer over the cell.

**Available in** The data area.

**Example 1** The following example displays the value of the data set field, customerName, and links to the first cell in a sheet that displays the details for the customer:

```
#write( [customerName] ) hyperlink( [customerID] & "!"$A$1", 0 )
```



**Example 2** The following example displays a static string, links to an absolute URL, and displays a tooltip:

```
#hyperlink( "http://www.actuate.com/products/index.asp", 1,  
            "Actuate's products" )
```

## if

- Syntax** if( Boolean\_expression, true\_condition[, false\_condition] )
- Description** Sets a cell or range value depending on the value of a Boolean expression.
- Parameters** **Boolean\_expression**  
A Boolean expression to evaluate.
- true\_condition**  
An expression to evaluate if the Boolean expression evaluates to true.
- false\_condition**  
An optional expression to evaluate if the Boolean expression evaluates to false.
- Available in** All areas of a data range.
- Example** The following example displays High if cost is greater than 500 and Low if cost is not greater than 500:
- ```
#if( [cost] > 500, "High", "Low" )
```

## include

- Syntax** include( condition )
- Description** Uses the specified condition to add data rows to a result set. Typically, include( ) is used in conjunction with select( ).
- Parameter** **condition**  
The criteria to use to filter the data. The row, column, or section includes only data that matches the criteria.
- Available in** Row sections, column sections, and sheets.
- Example** The following example selects data for which age is 65 or 75, creates a group section for each age value, and populates the group sections with all data for which age is less than 50:
- ```
select( [age] in {65,75} ) group( [age] ) include( [age] < 50 )
```

## isnull

- Syntax** isnull( expression[, value\_if\_null] )
- Description** Determines if a value in a field or range is null. If value\_is\_null is not specified, isnull( ) returns TRUE or FALSE. If value\_if\_null is specified, isnull( ) returns the value of expression or the value of value\_if\_null.

<b>Parameters</b>	<p><b>expression</b> The data set field or expression to test for null values.</p> <p><b>value_if_null</b> The value to return if expression is null.</p>
<b>Available in</b>	All areas of a data range.
<b>Example 1</b>	<p>The following example selects rows in which the data set field, sales, is not null:</p> <pre>select( not isnull( [sales] ) )</pre>
<b>Example 2</b>	<p>The following example returns the data set field, invoiceDate, if its value is not null. If invoiceDate is null, the function returns the value of the shipByDate data set field:</p> <pre>#isnull( [invoiceDate], [shipByDate] )</pre> <p>This example is equivalent to the following expression that uses isnull()'s single-parameter syntax:</p> <pre>#if( isnull( [invoiceDate] ), [shipByDate], [invoiceDate] )</pre>
<b>Example 3</b>	<p>The following example returns the data set field, orderNumber, if its value is not null. If orderNumber is null, the function returns No Orders:</p> <pre>#isnull( [orderNumber], "No Orders" )</pre>

## last

<b>Syntax</b>	last( expression )
<b>Description</b>	Returns the last non-null value in a set of records.
<b>Parameter</b>	<p><b>expression</b> The data set field or expression from which to return the last value.</p>
<b>Available in</b>	All areas of a data range.
<b>Example</b>	<p>The following example displays the last non-null value for the data set field, shippedDate:</p> <pre>#last( [shippedDate] )</pre>

## len

<b>Syntax</b>	len( value )
<b>Description</b>	Returns the number of characters in a string. For numeric or other non-string data, len() converts the value to a string equivalent and returns the number of characters.
<b>Parameter</b>	<p><b>value</b> The field or expression for which to calculate the number of characters.</p>
<b>Available in</b>	All areas of a data range.

**Example 1** The following example returns the value, 5:

```
#len( "Hello" )
```

**Example 2** The following example returns the value, 6:

```
#len( 123.45 )
```

## lock

**Syntax** lock( Boolean\_expression )

**Description** Locks a cell, range of cells, row or column section, or sheet. To enable cell locking, you must activate sheet protection. You can use this function in conjunction with the security( ) function to lock cells according to a user's Encyclopedia volume access control list (ACL). When Boolean\_expression evaluates to true, the report user cannot modify the cells.

**Parameter** **Boolean\_expression**  
The field or Boolean expression that defines whether to lock the cells.

**Available in** All areas of a data range except charts.

**Example 1** The following example locks the cells when the data set field, country, is Japan:

```
lock( [Country] = "Japan" )
```

**Example 2** The following example locks the cells when the user's ACL does not include the role, Manager:

```
lock( not security( "Manager" ) )
```

## lower

**Syntax** lower( text\_expression )

**Description** Converts a text value to lowercase.

**Parameter** **text\_expression**  
The field or text expression to convert to lowercase.

**Available in** All areas of a data range.

**Example** The following example selects data rows whose ID values, when converted to lowercase, match the string, abc123:

```
select( lower( [ID] )="abc123" )
```

## max

**Syntax** max( expression )

**Description** Computes the maximum value of a data set field or expression. max( ) ignores null values in the data set.

**Parameter** **expression**  
The data set field or expression for which to calculate the maximum value.

**Available in** All areas of a data range.

**Example** The following example calculates the maximum value in the data set field, sales:

```
max( [sales] )
```

## merge

**Syntax** merge( [key\_value] )

**Description** When used in the data area, applies the specified key value to all cells the data cell or range expands to include, then merges cells with the same key value. When used in row or column sections, merge( ) tests the key value on the data rows for all cells in the row, column, or section, then merges adjacent cells that have the same key value.

**Parameter** **key\_value**  
An optional identifier to apply to the cells to merge. The key value can be a text, numeric, or Boolean expression. Use this parameter if you need to merge adjacent cell ranges separately.

**Available in** The data area, row sections, and column sections.

**Example** The following example merges all adjacent cells with equal Region values:

```
merge( [Region] )
```

## min

**Syntax** min( expression )

**Description** Computes the minimum value of a data set field or expression. min( ) ignores null values in the data set.

**Parameter** **expression**  
The data set field or expression for which to calculate the minimum value.

**Available in** All areas of a data range.

**Example** The following example calculates the minimum value in the data set field, sales:

```
min( [sales] )
```

## minute

**Syntax** minute( [date] )

**Description** Returns the minute part of a date-and-time value. If there is no parameter, minute( ) returns the minute value for the current time.

**Parameter** **date**  
An optional date field to use when calculating the minute values.

**Available in** All areas of a data range.

**Example** The following example displays the minute part of the data set field, startLog:  
`#minute( [startLog] )`

## mod

**Syntax** `mod( dividend, divisor )`

**Description** Returns the remainder of a division. If either of the arguments is null, mod( ) returns null.

**Parameters** **dividend**  
The data set field or numeric expression to divide by divisor.

**divisor**  
The data set field or numeric expression to divide into dividend.

**Available in** All areas of a data range.

**Example 1** The following example returns 0:  
`#mod( 9, 3 )`

**Example 2** The following example returns 1:  
`#mod( 9, 4 )`

## month

**Syntax** `month( [date] )`

**Description** Displays a date values as the month part of the date. The month values are 1 through 12. If there is no parameter, month( ) returns the month of today's date.

**Parameter** **date**  
An optional data set field or expression of date data type to use when calculating the month values.

**Available in** All areas of a data range.

**Example** The following example displays the month part of the data set field, startLog:  
`#month( [startLog] )`

## name

**Syntax** `name( name )`

**Description** Creates a defined name to describe a row or column section, or a cell or a range of cells. When you create a defined name in a cell, the defined name refers to the range of cells the section or cell expands to include. A defined name you create for a row is row relative and column absolute. A defined name that you create for

a column is column relative and row absolute. You cannot use a defined name more than once. Each defined name must have a unique name.

To create a defined name on a range of cells for use as a data source for a chart, the range must be contiguous and rectangular. For example, you can set up a defined name on cells D7, E7, and F7, but not on D7 and F7 only.

**Parameter**    **name**  
The defined name to use to describe the row or column.

**Available in**    All areas of a data range except charts.

**Example 1**    The following example creates a defined name, `productData`, on a cell that displays the data set field, `productCode`:

```
#write( [productCode] ) name( "productData" )
```

**Example 2**    The following example creates a sheet-level defined name by prefixing the sheet name to the value `"!productData"`:

```
name( sheet( ) & "!productData" )
```

## now

**Syntax**    `now( )`

**Description**    Returns the current date-and-time.

**Available in**    All areas of a data range.

**Example**    The following example returns a string that contains the current year:

```
"Current year is " & year( now( ) )
```

## nth

**Syntax**    `nth( expression, index )`

**Description**    Returns from a set of records the non-null value that occupies a place marked by the value of `index`. `nth` ignores null values.

**Parameters**    **expression**  
The data set field or expression from which to return a value.

**index**  
A number that marks the place held by a data set element. A positive value marks data set elements from first to last. A negative value marks data set elements from last to first.

**Available in**    All areas of a data range.

**Example 1**    The following example returns 5:

```
#nth( {5,7,8,12,71} , 1 )
```

This example is equivalent to:

```
#first( {5,7,8,12,71} )
```

**Example 2** The following example returns 7:

```
#nth( {8,7,5,12,71} , 2 )
```

**Example 3** The following example returns 71:

```
#nth( {12,8,5,7,71} , -1 )
```

This example is equivalent to:

```
#last( {12,8,5,7,71} )
```

**Example 4** The following example returns the second, non-null customerNumber value:

```
#nth( [customerNumber], 2 )
```

**Example 5** The following example returns the next-to-last, non-null customerNumber value:

```
#nth( [customerNumber], -2 )
```

**Example 6** The following example returns the second-smallest customerNumber value:

```
#nth( + distinct [customerNumber], 2 )
```

## pageBreak

**Syntax** pageBreak( break\_before, break\_between, break\_after )

**Description** Creates a page break in the report. Each page break creates a new page when you print the spreadsheet report.

**Parameters** **break\_before**  
A Boolean expression that specifies whether to create a page break before the first instance of the section.

**break\_between**  
A Boolean expression that specifies whether to create a page break between instances of the section.

**break\_after**  
A Boolean expression that specifies whether to create a page break after the last instance of the section.

**Available in** Row sections and column sections.

**Example** The following example creates a page break between every instance of the section:

```
pageBreak( false, true, false )
```

## product

**Syntax** product( numeric\_expression )

**Description** Computes the product from the values that a data range cell expands to include.

**Parameter** **numeric\_expression**  
The data set field or numeric expression for which to compute a product.

**Available in** All areas of a data range.

**Example** The following example computes the product of the values in the sales field:  

```
product ( [sales] )
```

## quarter

**Syntax** `quarter( date )`

**Description** Displays a date value as the number of the quarter in the year. The resulting quarter values are 1 through 4. Quarter 1 begins on January 1.

**Parameter** **date**  
The date field to use when calculating the quarter values.

**Available in** All areas of a data range.

**Example** The following example displays the quarter part of the data set field, purchaseDate:  

```
#quarter( [purchaseDate] )
```

## rollup

**Syntax** `rollup( match_in_child_expression, match_in_parent_expression )`

**Description** Searches through one or more levels of a hierarchy. Typically, you use `rollup( )` with data in a parent-child relationship. BIRT Spreadsheet Designer iterates through the hierarchy that the match-in-parent expression defines to find all rows and descendant rows that meet the match-in-child expression value.

**Parameters** **match\_in\_child\_expression**  
Defines the value to use when searching the hierarchy for child-level data rows. BIRT Spreadsheet matches the value of `match_in_parent_expression` in a parent-level row to the `match_in_child_expression` value in a child-level row.

**match\_in\_parent\_expression**  
Defines the value to use for parent-level data rows.

**Available in** Row sections, column sections, and sheets.

**Example** The following example retrieves data for all rows in the Eastern region and all the descendants of each Eastern region row:  

```
select ( [Region]="Eastern" ) rollup( [ParentRegion], [Region] )
```



## round

- Syntax** `round( number[, decimal_places] )`
- Description** Rounds a number to the specified number of decimal places. Numbers higher than 5 are rounded up. The default number of decimal places is 0. If either of the arguments is null, `round( )` returns null.
- Parameters** **number**  
The value to round.
- decimal\_places**  
The optional number of decimal places to which to round the number. To round to a whole number power of ten, use a negative value for `decimal_places`. For example, to round to the nearest ten, use -1.
- Available in** All areas of a data range.
- Example 1** The following example rounds the Totals field values to three decimal places:  
`round( [Totals], 3 )`
- Example 2** The following example rounds the Totals field values to the nearest hundred:  
`round( [Totals], -2 )`

## row

- Syntax** `row( )`
- Description** Returns the identifier of the worksheet row in which it is executed.
- Available in** The data area.
- Example** The following example identifies the worksheet row in which the report script function is executed. For example, if you type this text in a cell that becomes B15 in the data range, the report script function returns 15:  
`row( )`

## rownum

- Syntax** `rownum( )`
- Description** Returns the query row number of the first row in the result set. Use `rownum( )` with `select( )` or `group( )` to create single-row result sets.
- Available in** All areas of a data range.
- Example** The following example creates a new group for each returned row:  
`group( rownum( ) )`

## second

**Syntax** second( [date] )

**Description** Displays the seconds part of a date-and-time value. If there is no parameter, second( ) returns the seconds value for the current time.

**Parameter** **date**  
An optional date field to use for calculating the second values.

**Available in** All areas of a data range.

**Example** The following example displays the seconds part of the data set field, startLog:  
`#second( [startLog] )`

## security

**Syntax** security( [security\_expression] )

**Description** Returns TRUE if the security expression matches a value in the user's Encyclopedia volume access control list (ACL). If there is no parameter, security( ) returns the user's list of security IDs.

**Parameter** **security\_expression**  
An optional string expression to test against the user's ACL. This expression can be a list of values.

**Available in** All areas of a data range.

**Example 1** The following example tests the user's ACL against the value in the data set field, salesRegion:

`#security( [salesRegion] )`

**Example 2** The following example tests for either of the values, MANAGER and NORTHEAST, in the user's ACL:

`#security( { "MANAGER", "NORTHEAST" } )`

**Example 3** The following example returns all the security IDs in the user's ACL:

`#security( )`

## sectioncount

**Syntax** sectioncount( section\_name )

**Description** Returns the number of instances the group report script function produces for the named section.

**Parameter** **section\_name**  
The name of the section for which to count section instances.

**Available in** The data area, row sections, and column sections.

**Example** The following example returns the number of unique sections in Section1:

```
sectioncount( section1 )
```

## select

**Syntax** select( condition )

**Description** Specifies criteria to use to return data rows.

**Parameter** **condition**  
A comparison expression to use to return data rows.

**Available in** Row sections, column sections, and sheets.

**Example 1** The following example selects data rows where age is equal to 5 or 8, then groups the results by month in date of birth:

```
select( [age] = 5 or [age] = 8 )  
group( month( [dateOfBirth] ) )
```

**Example 2** The following example selects data rows where age is 5 or 7 and sex is male and rows where age is 4 or 6 and sex is female:

```
#select( ( [age] in { 5, 7 } ) and [sex] = "male" ) or ( [age] in  
{ 4, 6 } and [sex] = "female" ) )
```

## setchart

**Syntax** setchart( sheet\_name, chart\_name, chart\_data, series\_in\_rows )

**Description** Identifies the source data of a chart that uses data range data and appears in a worksheet other than the data range worksheet. When you run the report, setchart( ) links the chart design to the updated data in the data range. To link data to a chart that appears on the data range, use chart( ).

**Parameters** **sheet\_name**  
The name of the sheet on which the chart appears.

**chart\_name**  
The name of the chart to update. The chart name cannot be the default chart name.

**chart\_data**  
The defined name that identifies the data to use in the chart.

**series\_in\_rows**  
Indicates whether BIRT Spreadsheet Designer defines rows or columns as series. A value of true indicates that rows are series, and false indicates that columns are series.

**Example** The following example links the chart named QuantityChart on the Summary worksheet to the data that the defined name, Quantity, identifies. The chart uses rows as series:

```
setchart( "Summary", "QuantityChart","Quantity", true )
```

**Syntax** setchart( chart\_sheet\_name, chart\_data, series\_in\_rows )

**Description** Identifies the source data of a chart that uses data range data and appears in its own worksheet. When you run the report, setchart( ) links the chart design to the updated data in the data range. To link data to a chart that appears on the data range, use chart( ).

**Parameters** **chart\_sheet\_name**  
The name of the sheet on which the chart appears.

**chart\_data**  
The defined name that identifies the data to use in the chart.

**series\_in\_rows**  
Indicates whether BIRT Spreadsheet Designer defines rows or columns as series. A value of true indicates that rows are series, and false indicates that columns are series.

**Example** The following example links the chart on the ChartSheet1 worksheet to the data that the defined name, Quantity, identifies. The chart uses columns as series:

```
setchart( "ChartSheet1","Quantity", false )
```

## setentry

**Syntax** setentry( sheet\_name, row, column, text )

**Description** Places the supplied text in a specified cell.

**Parameters** **sheet\_name**  
The worksheet on which to place the supplied text.

**row**  
The row number of the cell in which to place the supplied text. Row numbers begin at zero. For example, to indicate row number 3, use 2.

**column**  
The column number of the cell in which to place the supplied text. Column numbers begin at zero. For example, to indicate column B, use 1.

**text**  
The text string to place in the cell. If supplying a formula, preface the formula with an equal sign.

**Available in** All areas of a data range.

**Example 1** The following example places the text Totals in cell A10 on the Summary worksheet:

```
setEntry( "Summary", 9, 0, "Totals" )
```

**Example 2** The following example places the formula =Location in cell C20 on the Sheet1 worksheet:

```
setEntry( "Sheet1", 19, 2, "=Location" )
```

## sheet

**Syntax** sheet( [name[, codename]] )

**Description** Names the data range worksheet or returns the name of the current sheet. To use sheet( ) to name a worksheet, use a name parameter. To use sheet( ) to return the name of the current worksheet, do not include a value for the name parameter. When you use group( ) to display data on more than one sheet, you can use sheet( ) after the group( ) report script function to rename all sheets. A worksheet name cannot be more than 31 characters long.

The codename parameter enables you to set the VBA codename of a worksheet.

**Parameters** **name**

The name for the worksheet.

**codename**

The optional code name for the worksheet.

**Available in** All areas of a data range except charts.

**Example 1** The following example names the sheet MySheet:

```
#sheet( "MySheet" )
```

**Example 2** The following example creates a new sheet for each state and names each sheet with the associated state name:

```
group( [state] ) sheet( [state] )
```

**Example 3** The following example creates a new sheet for each city in each state and names each sheet with the concatenated state name and city name:

```
group( { [state], [city] } ) sheet( [city] & ", "& [state] )
```

**Example 4** The following example returns the name of the current worksheet:

```
#sheet( )
```

## size

**Syntax** size( [size\_value[, max\_size]] )

- Description** Sets the row height or column width to the specified value. If you supply both arguments, the first argument is the minimum size and the second argument is the maximum size.
- Parameters** **size\_value**  
An optional value to set the width of the column or the height of the row, in twips. To size the height to the tallest row contents or the width to the widest column contents, use a value of zero, or do not provide a size value.
- max\_size**  
An optional value to set the maximum width of the column or the maximum height of the row, in twips. If you provide a value for this parameter, size\_value is the minimum size of the row or column. The row or column uses the smallest size that can contain its contents, within the range size\_value to max\_size. A value of 0 indicates that there is no maximum size. The row or column expands to fit its largest contents.
- Available in** Row sections and column sections.
- Example 1** The following example sets the column width or row height to the size of the largest cell contents:  
`#size( )`
- Example 2** The following example sets the column width or row height to 144 twips, which is one-tenth of an inch:  
`#size( 144 )`
- Example 3** The following example sets the minimum column width or row height to 1440 twips, which is one inch, and the maximum to two inches:  
`#size( 1440, 2880 )`
- Example 4** The following example sets the maximum column width or row height to 1440 twips, which is one inch. If the row or column values fit in a smaller height or width, the size is reduced. If the row or column contains no values, it is not visible in the report:  
`#size( 0, 1440 )`
- Example 5** The following example sets the minimum column width or row height to 144 twips, which is one-tenth of an inch. If the values in the row or column require more space, the size expands to show the largest value:  
`#size( 144, 0 )`

## stdev

- Syntax** stdev( numeric\_expression )
- Description** Computes the standard deviation of a data set field or expression.
- Available in** All areas of a data range.

**Parameter** **numeric\_expression**

The data set field or numeric expression for which to compute the standard deviation.

**Example** The following example computes the standard deviation of the totals field:

```
stdev( [totals] )
```

## stdevp

**Syntax** stdevp( numeric\_expression )

**Description** Computes the standard deviation from the values that a cell expands to include in the data range, based on the entire population.

**Parameter** **numeric\_expression**

The data set field or numeric expression for which to compute the standard deviation.

**Available in** All areas of a data range.

**Example** The following example computes the standard deviation of the totals field:

```
stdevp( [totals] )
```

## substitute

**Syntax** substitute( text\_expression, search\_text, replace\_with[, occurrence] )

**Description** Replaces one or more substrings in the text expression with another substring.

**Parameters** **text\_expression**

The value in which to substitute the search text.

**search\_text**

The substring to find in the text expression.

**replace\_with**

The value to substitute for the search text.

**occurrence**

An optional numeric argument that specifies which occurrence of search text to substitute with the replacement text. The default value of zero indicates that substitute() changes all occurrences of the search text. A negative value indicates that substitute() searches from the end of the text expression string rather than the start.

**Available in** All areas of a data range.

**Example 1** The following example changes the word, banana, into binini:

```
#substitute("banana", "a", "i")
```

**Example 2** The following example changes the word, binini, into bikini:

```
#substitute("binini", "n", "k", -2)
```

## sum

**Syntax** sum( numeric\_expression )

**Description** Computes the sum of a data set field or expression.

**Parameter** **numeric\_expression**  
The data set field or numeric expression for which to compute a sum.

**Available in** All areas of a data range.

**Example 1** The following example computes the sum of the values in the sales field:

```
sum( [sales] )
```

**Example 2** The following example computes the sum of the values in the sales field for completed records and the values in the deferredAmount field for other records:

```
#sum( if ([status]="Complete", [sales], [deferredAmount] ) )
```

## time

**Syntax** time( hour, minute, second )

**Description** Returns a number that represents a time value.

**Parameters** **hour**  
An hour value. Hour must be a number between 0 and 23.

**minute**  
A minute value. Minute must be a number between 0 and 59.

**second**  
A second value. Second must be a number between 0 and 59.

**Available in** All areas of a data range.

**Example** The following example converts 10:10 to a number value:

```
time( 10, 10, 0 )
```

## today

**Syntax** today( date\_string[, language[, country]] )

**Description** Converts a string to a date value. Returns null if you do not provide a value for date\_string.

**Parameters** **date\_string**  
The data set field or string expression to convert to a date. The string must contain a date value in one of the standard spreadsheet date formats.



**language**

Optional. A two-character language code that specifies the language to use when converting date\_string to a date. The default value is the language of the current locale.

**country**

Optional. A two-character country code that specifies the country to use when converting date\_string to a date. The default value is the country of the current locale.

**Available in** All areas of a data range.

**Example 1** The following example turns a static string to a date:

```
todate( "2007-10-04" )
```

**Example 2** The following example converts order date string values to dates for the English language in the country, United Kingdom:

```
todate( [orderDateString], "en", "gb" )
```

**tonumber**

**Syntax** tonumber( string[, language[, country]] )

**Description** Converts a value to a number. tonumber( ) returns null if you do not provide a value for string.

**Parameters** **string**  
The value to convert.

**language**

Optional. A two-character language code that specifies the language to use when converting the number. The default value is the language of the current locale.

**country**

Optional. A two-character country code that specifies the country to use when converting the number. The default value is the country of the current locale.

**Available in** All areas of a data range.

**Example 1** The following example converts customer ID values to numbers:

```
tonumber( [customerID] )
```

**Example 2** The following example converts exchange rate values to numbers for the French language in the country, France:

```
tonumber( [exchangeRate], "fr", "fr" )
```

**toText**

**Syntax** toText( expression[, format] )

**Description** Converts a date or numeric expression to a single text value or a list of formatted text values. `toText()` returns null if you do not provide any arguments. If you specify only expression, `toText()` returns a single text value. If expression evaluates to a single value and you specify format, `toText()` returns a single formatted text value. If expression evaluates to multiple values and you specify format, `toText()` returns a list of formatted text values.

**Parameters** **expression**  
The value to convert.

**format**  
An optional format to apply to the value.

**Available in** All areas of a data range.

**Example 1** The following example converts price values to text:

```
toText( [price] )
```

**Example 2** The following example converts the data set field, `purchaseDate`, to a formatted string of the format, June 8, 2007:

```
toText( [purchaseDate], "mmm d, yyyy" )
```

**Example 3** The following example converts customer IDs to a list of formatted strings:

```
toText( distinct( [customerID], "000000" ) )
```

## trim

**Syntax** `trim( text_expression )`

**Description** Removes leading and trailing whitespace from the text expression.

**Parameter** **text\_expression**  
The value to trim.

**Available in** All areas of a data range.

**Example** The following example removes leading or trailing whitespace from the `CustomerName` values:

```
trim( [CustomerName] )
```

## trunc

**Syntax** `trunc( numeric_expression[, decimal_places] )`

**Description** Truncates a decimal value to the specified number of decimal places. The default number of decimal places is 0. If either argument is null, `trunc()` returns null.

**Parameters** **numeric\_expression**  
The value to truncate.

### **decimal\_places**

The number of decimal places to which to truncate the numeric expression. To truncate to a whole number power of ten, use a negative value for decimal\_places. For example, to truncate to the nearest ten, use -1.

**Available in** All areas of a data range.

**Example 1** The following example returns the value, 45:

```
trunc( 45.9 )
```

**Example 2** The following example returns the value, 45.2:

```
trunc( 45.237, 1 )
```

**Example 3** The following example returns the value, 40.0:

```
trunc( 45.237, -1 )
```

### **upper**

**Syntax** upper( text\_expression )

**Description** Converts a text values to uppercase.

**Parameter** **text\_expression**  
The field or text expression to convert to uppercase.

**Available in** All areas of a data range.

**Example** The following example selects ID values and converts them to uppercase:

```
select( upper( [ID] )="ABC123" )
```

### **values**

**Syntax** values( expression )

**Description** Retrieves all the values for the specified data set field or expression in all rows in the result set.

**Parameter** **expression**  
The data set field or expression from which to retrieve values.

**Available in** All areas of a data range.

**Example 1** The following example returns the ID values from the data source data1 that match the ID values in the data source data2:

```
data1:select( [id] in data2:values([id] )
```

**Example 2** The following example displays all the values in the amount field:

```
#values( [amount] )
```

## var

**Syntax** var( numeric\_expression )

**Description** Computes the variance for a data set field or expression.

**Parameter** **numeric\_expression**  
The data set field or numeric expression for which to compute the variance.

**Available in** All areas of a data range.

**Example** The following example computes the variance for sales:

```
var( [sales] )
```

## varp

**Syntax** varp( numeric\_expression )

**Description** Computes the variance from the values that a cell expands to include in the data range, based on the entire population.

**Parameter** **numeric\_expression**  
The data set field or numeric expression for which to compute the variance.

**Available in** All areas of a data range.

**Example** The following example computes the variance of the sales field:

```
varp( [sales] )
```

## write

**Syntax** write( text\_expression )

**Description** Evaluates a text expression and displays the result in a cell.

**Parameter** **text\_expression**  
A text expression.

**Available in** The data area.

**Example 1** The following example displays the value of the data set field, customerName:

```
#write( [customerName] )
```

**Example 2** The following example concatenates static strings and data set field values into a single cell:

```
#write( "Sales contact: " & [lastName] & ", " & [firstName] )
```

## year

**Syntax** year( [date] )

<b>Description</b>	Displays the year part of a date value. If there is no parameter, year( ) returns the year for today's date.
<b>Parameter</b>	<b>date</b> An optional date field to use when calculating the year values.
<b>Available in</b>	All areas of a data range.
<b>Example</b>	The following example displays the year part of the data set field, purchaseDate: <code>#year( [purchaseDate] )</code>



# Working with callback classes

This chapter contains the following topics:

- About callback classes
- Writing a callback class
- Deploying a callback class
- Testing and debugging a callback class
- Writing a multiple-class callback

---

## About callback classes

BIRT Spreadsheet Designer is a powerful and flexible tool for generating spreadsheet reports. Java programmers can expand the power and flexibility of BIRT Spreadsheet Designer by writing custom Java classes, called callback classes. Actuate BIRT Spreadsheet Engine and API users also use BIRT Spreadsheet Designer, but the reports they design do not use callback classes. Callback classes can access a subset of the application programming interface (API) that Actuate BIRT Spreadsheet Engine and API applications can access. For more information about developing applications for Actuate BIRT Spreadsheet Engine and API, see *Using BIRT Spreadsheet Engine and API*.

A callback class is a Java class that you write and attach to a BIRT Spreadsheet workbook. All callback classes must implement the ReportCallback interface, which specifies two methods that you include in your callback class. Those two methods are the start() method and the end() method. The start() method runs before data populates your workbook. The end() method runs afterwards.

You attach your custom callback class to a workbook using the BIRT Spreadsheet Designer interface, as explained in “Deploying a callback class,” later in this chapter. When you are ready to publish a report, you also deploy the class to Actuate BIRT iServer System, as explained in “Deploying a callback class to Actuate BIRT iServer System,” later in this chapter. A workbook can only have one callback class but that class can initiate many operations. A callback class can call most of the thousands of public methods in classes in the BIRT Spreadsheet API. It can also call methods internal to the class and public methods of other classes that you provide.

Most of the packages of classes in the BIRT Spreadsheet API are available to callback writers, but there are also some packages reserved for use by Actuate BIRT Spreadsheet Engine and API customers. For more information about the BIRT Spreadsheet API and its components, see the BIRT Spreadsheet API Javadoc. By accessing the classes and methods of the BIRT Spreadsheet API library, you can affect almost every aspect of a report and the report generation process. The BIRT Spreadsheet API includes functionality that allows you to manage and manipulate your workbook in many ways, such as:

- Affect the presentation, visibility, and organization of data.
- Add, delete, and modify any of the following:
  - Cells
  - Columns
  - Rows
  - Ranges
  - Worksheets



- Graphical objects
- Charts
- Read and set cell values and formulas.
- Create or alter the SQL statements that determine what data populates a workbook.
- Break large worksheets into smaller sheets.
- Write data to a custom data stream.
- Save a worksheet as an HTML or XML file.
- Perform security checks based on user names and passwords.
- Do conditional operations based on any of the following:
  - Cell values
  - Report parameter values
  - Defined names
  - Information retrieved from a file or database

---

## Writing a callback class

The signature of the required `start()` method of a callback class is:

```
public void start(BookModel bk, Object obj)
```

Similarly, the signature of the required `end()` method is:

```
public void end(BookModel bk, Object obj)
```

The first argument to both methods is a `BookModel` object, which occurs in the `com.flj.ss` package. The second argument is reserved for future use. The `BookModel` object provides access to the workbook and all associated objects, including sheets, queries, and data.

## Using the BookModel object

You can use the following two primary resources to explore the full range of functionality that a callback class can access through the `BookModel` object:

- The BIRT Spreadsheet API Javadoc  
The BIRT Spreadsheet API Javadoc documents every package, class, interface, method, and field in the BIRT Spreadsheet API. You can find the Javadoc in the javadoc directory. The default location is:

```
C:\Program Files\Actuate11\espreadsheet\javadoc
```

- Examples of callbacks

Chapter 20, “Examining samples of callback classes,” contains a set of examples of callback classes, illustrating many techniques that you can use in your callback class.

## Understanding a simple example

The following code is an example of a simple callback class that does nothing more than place the word Start in cell A1 and the word Finish in cell A2. The callback class modifies the contents of cells A1 and A2 with calls to the `setText()` method of the `BookModel` object in both its `start()` method and its `end()` method.

```
import com.flj.ss.*;
import com.flj.ss.report.*;
import com.flj.util.*;

public class MyCallback implements ReportCallback {
    public void start(BookModel bk, Object o) {
        try {
            bk.getLock();
            bk.setText(0,0,"Start");
        } catch (FlException ex) {
            // Exception handling code goes here
        }
        finally{
            bk.releaseLock();
        }
    }
    public void end(BookModel bk, Object o) {
        try {
            bk.getLock();
            bk.setText(0,1,"Finish");
        } catch (FlException ex) {
            // Exception handling code goes here
        }
        finally{
            bk.releaseLock();
        }
    }
}
```

The following sections explain different aspects of this example.

### Using import statements

The three import statements at the beginning of the file are not necessary, but without them you must provide fully qualified class names for things like `BookModel`, `ReportCallback`, and `FlException`.

For example, compare this statement containing a fully qualified class name:

```
public class MyCallback implements  
    com.flj.ss.Report.ReportCallback {
```

with the same statement containing a simple class name:

```
public class MyCallback implements ReportCallback {
```

## Using try-catch blocks

All the code that affects the workbook in this example is bracketed in try-catch blocks. This is necessary because almost all of the methods in the BIRT Spreadsheet API can throw exceptions. How you handle the exceptions is up to you. The Messages example, described in “Writing a multiple-class callback,” later in this chapter, provides one method of dealing with exceptions. Some very simple callback classes have no statements that throw exceptions. In this case you must remove the try-catch block to prevent compile errors.

## Using getLock( ) and releaseLock( )

You should always place a call to getLock( ) at the beginning of each callback method, usually as the first statement in the try block. You should also include a call to releaseLock( ) in the finally clause of the matching catch block. This assures that the lock is always released, even when the program throws an exception. These calls relate to multithreading and assure that your code is thread-safe, a requirement for running on Actuate BIRT iServer System.

If your callback class has no statements that throw exceptions be sure to move the releaseLock( ) statement from the finally clause to the end of the main line of code.

## Compiling the Java file

You can use your favorite Java Development Environment (JDE) to compile your callback class. Alternatively, you can use javac on the command line in a command window. In either case, you must include essd11.jar in your class path.

You can temporarily add the JAR file to your class path by specifying it in the command line of the compile statement, as shown in the following example:

```
javac -classpath .;C:\Actuate11\spreadsheet\essd11.jar  
    MyCallback.java
```

An easier alternative is to permanently add the JAR file to your class path by adding it to your CLASSPATH environment variable, in which case your compile statement is:

```
javac MyCallback.java
```

Both of these javac statements assume that the current directory contains a Java source file named MyCallback.java. The result of a successful compile is a file

named `MyCallback.class`, which `javac` creates in the current directory. This file is the CLASS file that you attach to your workbook, as described in “Deploying a callback class,” later in this chapter.

There are many options that you can set on the command line to affect the compilation and directories used. For full information about those options, see the documentation that comes with Java.

---

## Deploying a callback class

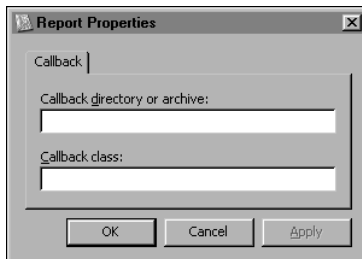
After you compile a Java file into a class file, you must deploy the class file. You deploy the callback class to BIRT Spreadsheet Designer for testing and debugging. When the callback functions as desired and you are ready to publish the report to Actuate BIRT iServer System, you must also deploy the callback to BIRT iServer System.

### Deploying a callback class to BIRT Spreadsheet Designer

Deploying a callback class to BIRT Spreadsheet Designer requires that you copy the class file to a special directory and that you associate the class with a report design, as in the following instructions.

#### How to deploy a callback class to BIRT Spreadsheet Designer

- 1 Open BIRT Spreadsheet Designer.
- 2 Load the design for which your callback class is intended.
- 3 Copy the class file to the `\extensions` directory. By default, this directory is:  
`C:\Program Files\Actuate11\espreadsheet\extensions`
- 4 With the design open, choose **Report**→**Report Properties** to see the options in Figure 19-1.



**Figure 19-1** Report callback options

- 5 On Report Properties—Callback, identify the callback class:

- In Callback directory or archive, specify the location of the callback class:
    - To use a callback class in the \extensions folder, do not type a location.
    - To use a callback class in a directory other than the \extensions directory, enter the location.
  - In Callback class, type the class name without the .class extension.
- Choose OK.

## Deploying a callback class to Actuate BIRT iServer System

If you deploy a spreadsheet report that uses a callback class to an Encyclopedia volume, you also need to deploy the callback .class file in the following directory:

```
...\Actuate11\iServer\reportengines\engines\ess\lib
```

---

## Testing and debugging a callback class

After you deploy the class file to BIRT Spreadsheet Designer, you can test its functionality and verify that it works as expected. You do this by running the report. You run the report by choosing Report➤Run or choosing one of the two Run buttons on the tool bar.

If you run the report more than once, you might not get the same results every time you run it. This could happen if your code modifies the workbook in some way that affects subsequent executions of the code. All changes to the workbook and worksheets, including formatting changes, and addition and deletion of worksheets, persist after you run the report.

## Logging messages from a callback class

The most common methods Java programmers use to display error and debugging messages are `System.err.println` and `System.out.println`. If you use these methods in BIRT Spreadsheet Designer, the messages appear in the output window, below the report. There is no output window when the report runs on Actuate BIRT iServer System, so only use these methods for debugging a report in BIRT Spreadsheet Designer. To log error and debugging messages on BIRT iServer System, use a `java.util.logging.Logger` object. This technique logs messages in a log file and is available both in BIRT Spreadsheet Designer and on BIRT iServer System.

The Messages example, described in “Writing a multiple-class callback,” later in this chapter, shows an alternative technique for logging messages. When you use the Messages class, it records messages when you run the report on BIRT iServer System or in BIRT Spreadsheet Designer.

## How to use a Logger object to provide debug messages

Because the two callback methods can run in different JVMs, you must instantiate the static Logger object outside the callback methods.

- 1 Declare and instantiate the logger object as a static class variable, as shown in the following code, where myCallback is the name of the callback class:

```
static private Logger logger = Logger.getLogger(  
    myCallback.class.getName() );
```

- 2 Log messages at the desired level, as shown in the following code:

```
logger.info("\nWritten first cell\n");
```

## Using a Logger object in BIRT Spreadsheet Designer

To set the logging level at which BIRT Spreadsheet Designer displays log messages in the output window or log file, use Tools→Designer Preferences. A Logger object writes messages to the output window and to the BIRT Spreadsheet Designer log file when you view the report in BIRT Spreadsheet Designer. Values in the file, flj.logging.properties, specify the location and name of the log file, using the format supported by java.util.logging.FileHandler.pattern. This properties file is located in the .flj subdirectory of the user's designated home folder. Typically, this folder is C:\Documents and Settings\<user name>. The default location of the log file is .flj\logs in the user's home folder. The default name of the log file is a unique number followed by .log.

## Using a Logger object on BIRT iServer System

To set the logging level at which BIRT iServer System records messages, use BIRT iServer Configuration Console. Messages that the Logger object creates appear in the Java server or Java factory log files when the report runs on BIRT iServer System, as shown in the following list:

- When running or running and saving a report, the Java server logs all callback messages.
- When scheduling a report, the Java factory logs messages that the start() method produces.
- When viewing a spreadsheet report (.soi), the Java server logs messages that the end() method produces.

BIRT iServer System's log files are located in its installation folder's log subdirectory. Java server logs have names that begin with jsrvr10. Java factory logs have names that begin with jfctsrvr10.

## Modifying and retesting a callback class

If your first pass at testing a callback class requires you to make additions or adjustments to your code, complete the following tasks in this order:

- Modify your source code.
- Recompile.
- Copy the new class file to the \extensions directory.
- Close BIRT Spreadsheet Designer.
- Open BIRT Spreadsheet Designer.
- Load the workbook.
- Run the report.

Close and restart BIRT Spreadsheet Designer every time you update your class file. BIRT Spreadsheet Designer caches class files and always uses the version in its cache. If you save your workbook at any time after you attach the callback class, you do not need to re-attach the callback class. Closing and reloading the workbook does not cause the program to find an updated class file. You must close and restart BIRT Spreadsheet Designer.

## Testing in a Java development environment

If you prefer to test and debug a callback class in a JDE, complete the following tasks:

- Add the following JAR files as libraries in your project:  
`essd11.jar`  
`derby.jar`
- Set the debugger to run the `main()` method for BIRT Spreadsheet Designer, which is in the following class:  
`com.flj.swing.designer.Designer`

## Testing a callback on Actuate BIRT iServer System

Some reports are fully testable only when you run them on BIRT iServer System. For example, if a callback class executes code that runs conditionally, based on the name of the host system at run time, the conditional code never runs when you run the report on BIRT Spreadsheet Designer.

You can test a callback on Actuate BIRT iServer System by first publishing your report, as described in “Deploying a callback class to Actuate BIRT iServer System,” earlier in this chapter, then running it using Actuate Information Console, as described in *Using Information Console*.

You can debug a callback class by viewing and analyzing the report you create on BIRT iServer System. To see debugging output, use a `java.util.logging.Logger` object and the BIRT iServer System log files, as described in “Logging messages from a callback class,” earlier in this chapter.

You can also debug the callback class using remote debugging in a Java Development Environment (JDE). You must configure BIRT iServer System and set up the Java processes that run, view, and perform scheduled generation of spreadsheet reports, as described in the following sections.

## Configuring BIRT iServer System for remote Java debugging

To debug a callback class on the BIRT iServer System machine remotely from your local machine, set the start arguments for the Default Java Async and Default Java View resource groups to support communication with the JDE. Add an argument of the following form to the start arguments:

```
-Xdebug -Xrunjdp:transport=dt_socket,server=y,suspend=n,  
address=<port number>
```

Use a different port number for each resource group. Set up your JDE to communicate with the debug ports that you specified for the BIRT iServer System’s Java resource groups.

For information about setting up remote debugging in your JDE, see your JDE documentation. To pass control to the debugger, set a break point in the callback method that you need to debug. When you run or schedule a report on BIRT iServer System, the debugger takes control when the Java process reaches that line of code. You can step through the code or make changes as supported by your JDE.

### How to set Java resource group start properties for debugging on BIRT iServer

The following procedure shows how to set up debugging on Java resource groups on Actuate BIRT iServer.

- 1 Open BIRT iServer Configuration Console and log in.
- 2 On the Configuration Console menu bar, choose Show Advanced View.
- 3 To set debugging arguments for the Default Java View resource group, perform the following steps:
  - 1 On the side menu, choose System Resource Groups. On Resource Groups, select Default Java View.
  - 2 On Resource Groups > Default Java View : Properties, select and copy the text in Start Arguments.
  - 3 On the side menu, choose Servers. On Servers, select the name of the server on which you plan to run your spreadsheet report.
  - 4 On Servers > Server Name : Properties, select Resource Groups.



- 5 Paste the text that you copied in Step 2 into Start Arguments for Default Java View.
- 6 Type the additional text similar to the following line into the same Start Arguments field. Change 4001 to the port number on which your JDE communicates with the Java View process:
 

```
-Xdebug -Xrunjdpw:transport=dt_socket,server=y,suspend=n,  
address=4001
```
- 7 In Max factories, type 1. If Min factories is greater than 1, in Min factories, type 1.
- 8 Choose OK.
- 4 To set debugging arguments for the Default Java Async resource group, perform Step 3 for Default Java Async. Use a different port number in the Default Java Async startup arguments from the one you used for Default Java View.
- 5 To restart the Java resource group services, restart the BIRT iServer by performing the following steps:
  - 1 On the side menu, choose System.
  - 2 On System : Status, choose Stop.
  - 3 On the message, Are you sure you want to perform as system-wide shutdown, choose OK. Wait until the System : Status page shows that the system is offline.
  - 4 Choose Start system.

## Testing the BIRT iServer System Java processes

After changing the Java resource groups' start arguments to support debugging, as described in the previous section, you must test that they still produce reports. You run a Java-based report that is known to succeed, using each process. Testing the processes ensures that they are started. You cannot debug a spreadsheet callback remotely until these processes are started.

When the BIRT iServer System starts, it starts the Java server for running and viewing spreadsheet reports. By default, the BIRT iServer System does not start the Java factory that generates a scheduled spreadsheet report until a scheduled Java report runs. To debug a scheduled spreadsheet report, you must start this process manually, by scheduling a Java-based report to run immediately, as described later in this section.

After testing the Java processes successfully, you can debug your callback remotely in a JDE. Run your spreadsheet report that uses a callback class using the steps described in "How to test the Java server," and "How to start and test the Java factory," later in this chapter.

### **How to test the Java server**

This procedure describes starting the Java server by using BIRT iServer Management Console to run a Java-based report. You can also run a report using Information Console.

- 1 Open BIRT iServer Management Console and log in.
- 2 Navigate to a folder that contains a BIRT or spreadsheet report executable, for example, /Public/BIRT and Business Reports Examples.
- 3 Move the mouse pointer over the arrow to the left of a BIRT report design or a spreadsheet report executable that you know runs successfully, then choose Run. Do not use the report that you need to debug.
- 4 On Parameters, type or choose values for any required parameters. Then choose OK.
- 5 If the job did not succeed, move the mouse over the arrow to the left of the job and choose Properties. Review the properties for error messages. If you supplied incorrect start arguments to the Default Java Async resource group, as described in “How to set Java resource group start properties for debugging on BIRT iServer,” earlier in this chapter, it can fail to start. Review the messages in the most recent jfctsrvr log file. Correct the issues and restart the server if necessary. Repeat this procedure until the scheduled job succeeds.

### **How to start and test the Java factory**

This procedure describes starting the Java factory by using BIRT iServer Management Console to schedule a Java-based report. You can also schedule a report using Information Console.

- 1 Open BIRT iServer Management Console and log in.
- 2 Navigate to a folder that contains a BIRT or spreadsheet report executable, for example, /Public/BIRT and Business Reports Examples.
- 3 Select a BIRT report design or a spreadsheet report executable that you know runs successfully. Do not choose the report that you need to debug.
- 4 On Schedule, select Parameters. Type or choose values for any required parameters. Then choose OK.
- 5 On the side menu, choose Jobs, then select Completed. Refresh this page until your job appears.
- 6 If the job did not succeed, move the mouse over the arrow to the left of the job and choose Properties. Review the properties for error messages. If you supplied incorrect start arguments to the Default Java Async resource group, as described in “How to set Java resource group start properties for debugging on BIRT iServer,” earlier in this chapter, it can fail to start. Review the messages in the most recent jfctsrvr log file. Correct the issues and restart the server if necessary. Repeat this procedure until the scheduled job succeeds.

---

## Writing a multiple-class callback

You can only attach one callback class to a workbook, but you can use any number of your own Java classes in combination with your callback class. To use functionality in other Java classes than those in the BIRT Spreadsheet API, complete the following tasks:

- Select a package name and assign your additional classes to that package.
- Create a directory under the \extensions directory, giving it the name of that package.
- Copy the class files for your additional classes in that directory.
- Add an import statement to your callback class that specifies your package name.

### Examining a multiple-class callback example

The following example, `MyCallback2`, is a slight modification to the simple example that appears in “Understanding a simple example,” earlier in this chapter. The changes enable the example program to display debugging and error messages on worksheets that it adds to the workbook.

`Messages.java` is the helper class for this example. `MyCallback2.java` calls a method in the `Messages` class that displays a message in a named worksheet. There are also methods in `Messages` to disable all messages, a method to re-enable all messages, and methods to enable and disable messages for specific named worksheets. You can use `Messages` with any callback class that you write.

The source code for `MyCallback2` is:

```
import com.flj.ss.*;
import com.flj.ss.report.*;
import com.flj.util.*;
/***** enable access to Messages class *****/
import messages.*;
public class MyCallback2 implements ReportCallback {
    /***** define names of new worksheets for displaying
        debugging and error messages *****/
    private static String errorMsgs = "Error Messages";
    private static String debugMsgs = "Debugging Messages";
    public void start(BookModel bk, Object o) {
        try {
            bk.getLock();
            bk.setText(0,0,"Start");
        }
    }
}
```

```

/***** Display a debugging message *****/
    Messages.display(bk, "Got to the start method",
        debugMsgs);
    } catch (FlException ex) {
/***** Display an error message *****/
        Messages.display(bk, ex + " in start method " +
            ex.getMessage(), errorMsgs);
    }
    finally{ bk.releaseLock(); }
}

public void end(BookModel bk, Object o) {
    try {
        bk.getLock();
        bk.setText(0,1,"Finish");
/***** Display a debugging message *****/
        Messages.display(bk, "Got to the end method",
            debugMsgs);
    } catch (FlException ex) {
/***** Display an error message *****/
        Messages.display(bk, ex + " in start method " +
            ex.getMessage(), errorMsgs);
    }
    finally{ bk.releaseLock(); }
}
}

```

Because all methods of Messages are static, MyCallback2 never needs to instantiate a Messages object. All calls to Message methods are similar in form to the following statement:

```
Messages.display(jb, printArea, debugMessages);
```

Methods you call in your own additional classes might not be static. In those cases you must instantiate an object of the class and call methods on that object.

## Specifying a package for additional classes

Any time that a callback class calls methods in classes that you provide, include the import statement for the packages that contain additional classes. The following statement in MyCallback2 tells the compiler which package contains the Messages class:

```
import messages.*;
```

Likewise, additional classes must belong to a Java package. The Messages class has the following package statement, identifying the package to which it belongs:

```
package messages;
```

## Deploying additional classes

There are two ways to deploy additional classes. The first method requires you to create a jar file and the second method requires you to create one or more new directories.

### Deploying additional classes using a jar file

To deploy additional classes using a jar file, you create a Java jar file containing the callback class and any classes your callback class uses. All the classes must be in the same jar file. The additional classes must be part of a package, as explained in “Specifying a package for additional classes,” earlier in this chapter.

Once you have a jar file with all the necessary classes, to deploy the classes to BIRT Spreadsheet Designer, place a copy of the jar file in the \extensions directory that is located at:

```
\Program Files\Actuate11\espreadsheet\extensions
```

To deploy the jar file to Actuate BIRT iServer System, place a copy of the jar file in the BIRT Spreadsheet library directory:

```
\Program Files\Actuate11\iServer\reportengines\engines\ess\lib
```

### Deploying additional classes by creating new directories

If you do not create a jar file with which to deploy the additional classes that your callback class uses, you must create directories under the \extension directories.

For BIRT Spreadsheet Designer to locate the package that contains your additional classes, you must complete the following tasks:

- Create a directory under the \extensions directory on your BIRT Spreadsheet Designer machine. The default location is:

```
C:\Program Files\Actuate11\espreadsheet\extensions
```

- Give the directory the same name as the package.
- Place copies of all the class files for that package in the new directory.

For Actuate BIRT iServer System to locate your package, you must complete the following tasks:

- Create a directory under the BIRT Spreadsheet library directory on the BIRT iServer System machine. The default location is:

```
C:\Program Files\Actuate11\iServer\reportengines\engines\ess  
  \lib
```

- Give the directory the same name as the package.
- Place copies of all the class files for that package in the new directory.

You can have more than one package, and you can access classes in more than one package, but each package must have its own directory on both machines.

## Locating the example source files

The source files for all examples are in the \callback directory on your BIRT Spreadsheet Designer machine. The default location for this directory is:

C:\Program Files\Actuate11\espreadsheet\examples\callback

The source code for the Messages.java file is:

```
// The Messages class

package messages;
import com.flj.ss.*;
import com.flj.util.*;
import java.util.*;

public class Messages{

    private static Vector disabledList;
    private static boolean disabledListExists = false;

    public static void enable(String sheetName){
        if (!disabledListExists)
            return;
        for(int i = 0; i < disabledList.size(); i++){
            if (sheetName.equals((String)disabledList.elementAt(i)))
                disabledList.removeElementAt(i);
            break;
        }
    }

    // Disable messages for a specified sheet
    public static void disable(String sheetName){
        if (!disabledListExists){
            disabledList = new Vector();
            disabledListExists = true;
        }
        for(int i = 0; i < disabledList.size(); i++){
            if (sheetName.equals(
                (String)disabledList.elementAt(i)))
                return;
        }
        disabledList.addElement(sheetName);
    }

    private static boolean disabled = false;
}
```

```

// Disable all message generation
public static void disable(){
    disabled = true;
}
// Enable all message generation
// (except any in the explicit disable list)
public static void enable(){
    disabled = false;
}
// Display the specified message in the specified sheet,
// providing the messages were not disabled
public static void display (BookModel jb, String msg,
    String messageSheetName){

    // Check to see if we have disabled all messages
    if(disabled) return;

    // Check to see if this sheet should not get messages
    if(disabledListExists){
        for(int i = 0; i < disabledList.size(); i++){
            if (messageSheetName.equals(
                (String)disabledList.elementAt(i)))
                return;
        }
    }
    // Check to see if this sheet should not get messages
    int msgSheet = 0;

    try{

        // Save which was the active sheet coming in
        String origSheetName = jb.getSheetName(
            jb.getSheet());

        // Loop through all the sheets looking for one with
        // the right name
        boolean foundMsgSheet = false;
        for (int i=0;i<jb.getNumSheets();i++) {
            if(!jb.getSheetName(i).equals(messageSheetName)){
                continue;
            }
            else{
                foundMsgSheet = true;
                msgSheet = i;
                break;
            }
        }
    } // end for

```

```

        if(!foundMsgSheet){
            // Doesn't exist, need to create it
            jb.setSheet(0);
            jb.editInsertSheets();
            jb.setSheetName(0, messageSheetName);
            msgSheet=0;
        }
        // Set sheet to the message sheet
        jb.setSheet(msgSheet);

        // Set the new message in the next available row
        int nextRow = jb.getLastRow() + 1;
        jb.setText(nextRow, 0, msg);
        jb.setColWidthAuto(0,0,nextRow,0,true);

        // Reset active sheet
        for (int i=0;i<jb.getNumSheets();i++) {
            if(!jb.getSheetName(i).equals(origSheetName)){
                continue;
            }
            else{
                jb.setSheet(i);
                break;
            }
        } // End for
    } // End try
    catch (FlException ex){
        try{
            jb.setSheet(0);
            jb.setText(1,1,
                "Exception creating Callback message sheet:" +
                ex.getMessage());
        }catch (FlException ex2){}
    }
} // End of display method
} // End of Message class

```



## Examining samples of callback classes

This chapter contains the following topics:

- About the callback examples
- Using the BIRT Spreadsheet API
- Using the BIRT Spreadsheet API Javadoc
- Understanding the examples

---

## About the callback examples

The examples in this chapter are complete, functioning callback classes. Each example illustrates one or more techniques that you can use in your own callback classes. The examples show how to access the BIRT Spreadsheet API through the BookModel object. By understanding these examples, you learn various ways to control and modify a BIRT Spreadsheet report.

The examples in this chapter illustrate:

- Retrieving and using a parameter or other defined name
- Retrieving and modifying a query
- Reading and writing a data file
- Modifying cell formatting
- Locking a cell or range
- Creating a password-protected workbook
- Changing a style sheet
- Setting print ranges for all sheets in a workbook
- Changing an image in the report
- Modifying a report based on data values
- Creating a chart in the report
- Creating a pivot range in a BIRT Spreadsheet report

While these examples do not show you every possible thing you can do with a callback, they do give you a good overview of the BIRT Spreadsheet API and how you can use it to expand the power of BIRT Spreadsheet Designer and improve your reports.

---

## Using the BIRT Spreadsheet API

Callback classes use the BIRT Spreadsheet API for all operations associated with the workbook. You gain to access the BIRT Spreadsheet API through the BookModel object, which is the first argument to the start( ) method and the end( ) method of all callback classes.

The BookModel object represents the workbook to which you attach the callback class. It provides the gateway to all the other components you can access. The following list identifies many of the objects that the BookModel object can access:

- Worksheets

- Tabs
- Cells and the data they hold
- Rows
- Columns
- Report ranges
- Data sources
- Data queries
- Formatting information
- Parameters
- Defined names
- Selections
- Print ranges
- Formulas
- Print information

---

## Using the BIRT Spreadsheet API Javadoc

To see the full range of objects that you can access with the `BookModel` object, look at the methods of the `BookModel` class, which you can find in the BIRT Spreadsheet API Javadoc. The default location for the BIRT Spreadsheet API Javadoc is:

```
C:\Program Files\Actuate11\espreadsheet\Javadoc
```

You typically retrieve objects from a `BookModel` object using methods that have names that begin with `get`, such as `getSheet()` and `getRow()`. You set or change properties of the `BookModel` object using methods whose names begin with `set`, such as `setAllowTabs()` and `setCellFormat()`. You can test various states of the `BookModel` object using methods whose names begin with `is`, such as `isAutoRecalc()` and `isBookProtected()`. You change the appearance of a workbook using methods whose names begin with `setShow`, such as `setShowGridLines()` and `setShowColHeadings()`.

The naming conventions for methods of the `BookModel` class also apply to the objects that you get from the `BookModel` class. For example, the `Sheet` class has the `getDefaultColWidth()` method, and the `setName()` method, and the `isPasswordProtected()` method.

Every class in the BIRT Spreadsheet API Javadoc has a list of all its public methods. Each entry in the list is a hyperlink to a full description of the method.

Javadoc also displays all parameters and return values as hyperlinks, providing rapid access to the Javadoc for those classes.

The best way to understand the structure and functionality of the BIRT Spreadsheet API is to scan the methods, constructors, and fields of the API classes. Learning what each class offers improves your ability to use callback classes.

---

## Understanding the examples

The following examples demonstrate some of the useful things that you can do with callback classes. A brief explanation precedes each example.

The comments in the source code provide more detailed information about programming techniques and how to access the BIRT Spreadsheet API for the desired effect. Code comments are more numerous and more detailed in the earlier examples. Comments in later examples are limited to techniques or concepts that are not covered in earlier examples.

### Using and testing the examples

The source files for all examples in this chapter are available on Actuate's BIRT Exchange web site. The location of the examples is:

<http://www.birt-exchange.com/modules/wfdownloads/singlefile.php?cid=4&lid=254>

The following sections also contain the full source code of every example. You can copy and paste directly from the source files into your own callback classes. Most examples have an associated spreadsheet design (.sod) file with which you can test the callback class. These files have the same base name as the associated callback class. The SOD files are ready to use after you compile and deploy the associated callback class, as described in Chapter 19, "Working with callback classes." Many of the examples use the Messages class, which you must compile and deploy by following the steps in "Writing a multiple-class callback," in Chapter 19, "Working with callback classes."

### Understanding the SetPrintAreas example

The SetPrintAreas example finds all the worksheets in a workbook and sets the print range for each sheet to include cell A1 through the last cell containing data in that sheet. This example illustrates the following techniques:

- Setting and releasing the lock on a workbook
- Handling BIRT Spreadsheet exceptions
- Finding the last cell in a sheet

- Setting the print areas of sheets in a workbook
- Displaying debugging messages by using the Messages class

You can test this example using SetPrintAreas.sod.

```

/*****
 * The SetPrintAreas class
 *****/

/*****
 * The com.flj.ss.report package contains the
 * ReportCallback interface.
 *****/
import com.flj.ss.report.*;

/*****
 * The com.flj.ss package contains most of the classes and
 * interfaces in the BIRT Spreadsheet API.
 *****/
import com.flj.ss.*;

/*****
 * The com.flj.util package contains the fljException class.
 *****/
import com.flj.util.*;

/*****
 * The messages package contains the Messages class. To compile
 * SetPrintAreas, you must either have a messages directory
 * directly under the current directory at compile time or
 * include the path to the messages directory in your class
 * path. The messages directory must contain Messages.class.
 *****/
import messages.*;

/*****
 * Every callback class must implement ReportCallback interface,
 * which specifies the two methods, start( ) and end( ). The
 * class name, SetPrintAreas in this case, must be the same as
 * the file name, SetPrintAreas.java.
 *****/
public class SetPrintAreas implements ReportCallback {

/*****
 * Define the names of the worksheets in which the program
 * displays debugging and error messages through the display( )
 * method of the Messages class.
 *****/

```

```

*****/
    private String debugMsgs = "Debugging messages";
    private String errMsgs = "Error Messages";

/*****
 * This is one of the two methods specified by the
 * ReportCallback interface. It is called before data populates
 * the workbook. In this case it does nothing, so has an empty
 * code block.
 *****/
    public void start(BookModel bk, Object o) {}

/*****
 * This is the other method specified by the ReportCallback
 * interface. It is called after data populates the workbook,
 * but before the book is displayed. Since SetPrintAreas needs
 * data to determine the print range, the operative code for
 * this callback must be in this method rather than the start
 * method. The first argument to this method is bk, the
 * BookModel object that represents the workbook. This object is
 * the gateway object through which the program gains access to
 * all the other objects and functionality of the API.
 * The second argument to this method is reserved for future
 * use.
 *****/
    public void end(BookModel bk, Object o) {

/*****
 * All the code of most callback methods needs to be within
 * try-catch blocks to catch exceptions that most of the
 * BIRT Spreadsheet API methods throw. See the catch block at the
 * end of the method.
 *****/
    try {

/*****
 * It is important to get a lock on the workbook before doing
 * any operation on it. This is a multi-threading issue and is
 * required especially if the report is destined for publication
 * on Actuate BIRT iServer. It is equally important to release the
 * lock in the finally clause of the try-catch block.
 *****/
        bk.getLock();

/*****
 * Iterate through all the sheets in the book. Find the number
 * of sheets in the book with the getNumSheets( ) method of the
 * bk object.
 *****/

```

```

        for (int i=0;i<bk.getNumSheets();i++) {
/*****
* To get a Sheet object, perform the following steps:
*   1. Get a Book object from the BookModel object
*   2. Get a Sheet object from the Book object.
*       The BookModel interface does not contain a method to get
*       a sheet by index, but Book does.
*****/
            Sheet sht = bk.getBook().getSheet(i);

/*****
* Get the last row and column for the sheet and use them to
* build a formatted string, such as $I$13.
*****/
            int lastCol = sht.getLastCol();
            int lastRow = sht.getLastRow();
            String lastColRow = bk.formatRCNr(lastRow,
                lastCol, true);

/*****
* Build the string that contains a partial print range,
* including the sheet name in single quotes, followed by an
* exclamation point, followed by the row-column range. This
* string should have the form, 'Sheet1'!$B$2:$I$13.
*****/
            String printArea = "'" + bk.getSheetName(i) + "!" +
                "$B$2:" + lastColRow;

/*****
* Set the print area for the book. Note that for each sheet you
* have to set the active sheet before setting the print area.
* This is because setPrintArea only operates on the active
* sheet.
*****/
            bk.setSheet(i);
            bk.setPrintArea(printArea);
        }

/*****
* After all print ranges are set, this loop generates a message
* for every sheet. Use the Messages class to display messages.
* The Messages.display( ) method inserts messages into a
* worksheet having the name of the third argument. If no sheets
* exist by this name, Messages.display( ) creates one. It
* inserts each new message below the last one in the sheet.
*****/

```

```

        for(int i = 0; i < bk.getNumSheets(); i++){
            bk.setSheet(i);
            Messages.display(bk,"Sheet "+ i + ", Print range = "
                +bk.getPrintArea(),debugMsgs);
        }

/*****
* Catch the FlException, the superclass of all the exceptions
* thrown by methods in the BIRT Spreadsheet API.
*****/
    } catch (FlException ex) {

/*****
* Handle exceptions in the catch clause block. Use the Messages
* display( ) method to display errors. This is identical to the
* debugging message display except that it uses a different
* worksheet, defined by the errMsgs string.
*****/
        Messages.display(bk, ex + " in start method ",errMsgs);
    }

/*****
* Put the call to release the lock in the finally block and not
* at the end of the main line of the code. This is to insure
* that releaseLock( ) executes even if an exception occurs. If
* releaseLock( ) never executes, the callback could hang in
* Actuate BIRT iServer.
*****/
        finally{
            bk.releaseLock();
        }
    } // end of end method
} // end of class

```

## Understanding the ModifyQuery example

The ModifyQuery example changes a query based on the value of a defined name. This example illustrates the following techniques:

- Getting the query object from the BookModel object
- Getting a defined name
- Modifying and resetting a query

You can test this example using modifyQuery.sod.

```

/*****
* The ModifyQuery class
*****/

```



```

/*****
 * The com.flj.data package contains the DataSourceCollection
 * and DataQueryCollection classes.
 *****/
import com.flj.data.*;

/*****
 * The following import is necessary for the DataQuery and
 * DatabaseQuery classes.
 *****/
import com.flj.data.query.*;

/*****
 * The com.flj.data.source package contains the Source class.
 *****/
import com.flj.data.source.*;

import com.flj.ss.report.*;
import com.flj.ss.*;
import messages.*;

public class ModifyQuery implements ReportCallback {
    String debugMsgs = "Debug Messages";
    String errMsgs = "Error Messages";

    public void end(BookModel bk, Object obj) {}

/*****
 * Use the start( ) method for code that modifies the query that
 * populates the workbook.
 *****/
    public void start(BookModel bk, Object obj) {
        try {
            bk.getLock();

/*****
 * To get a SQL statement for a query, perform the following
 * steps:
 * 1. Get the data source collection from the BookModel
 *    object.
 * 2. Get an array of Source objects representing the data
 *    sources from the data source collection. If there
 *    exists a data cache, the first data source is the second
 *    element in the array. If there is no cache, the first
 *    data source element is the zeroth element.
 * 3. Get a collection of Data Queries from one of the Source
 *    objects (the zeroth one in this case).
 *****/

```

```

*      4. Get an array of data queries from the
*          DataQueryCollection object.
*      5. Cast one of the data queries (the zeroth one in this
*          case) as a DatabaseQuery object.
*      6. Get the SQL statement from the DataBaseQuery object.
*****/
    DataSourceCollection dsc = bk.getDataSourceCollection();
    Source[] src = dsc.get();
    int position;
    if(dsc.hasDataSetCache())
        position=1;
    else position = 0;
    DataQueryCollection dqc =
        src[position].getDataQueryCollection();
    DataQuery dq[] = dqc.get();
    DatabaseQuery dbq = (DatabaseQuery) dq[0];
    String sql = dbq.getQuery();

/*****
* Get the value of a defined name
*****/
    String userName = bk.getDefinedName("UserName");

/*****
* Test userName for a specific value, and conditionally add a
* new WHERE clause to the SQL statement.
*****/
    if (userName.equalsIgnoreCase("Pierre")) {

/*****
* Find the WHERE clause in the SQL statement. Assume that the
* WHERE clause comes at the end of the statement, if at all,
* and truncate the SQL statement in preparation for appending a
* new WHERE clause.
*****/
        int lastIndex = sql.toLowerCase().lastIndexOf(
            "where");
        if(lastIndex != -1){
            sql = sql.substring(0, lastIndex);
        }
        sql = sql + " WHERE state = 'MA'";
    }

/*****
* Make the new SQL statement active by setting the SQL
* statement as the query in the DataBaseQuery object.
*****/

```

```

        dbq.setQuery(sql,false);
    } catch (Exception ex) {
        System.out.println("Exception in callback: "+ex);
    }
    finally{
        bk.releaseLock();
    }
}
}
}

```

## Understanding the DynamicImage example

The DynamicImage example illustrates how to add an image to a worksheet by opening an input stream for a URL.

This example illustrates the following techniques:

- Getting a defined name from the workbook
- Placing an image on a worksheet

You can test this example using DynamicImage.sod.

```

/*****
 * The DynamicImage callback class
 *****/
import java.sql.*;
import java.net.*;
import com.flj.ss.report.*;
import com.flj.ss.*;
import com.flj.util.*;
import messages.*;

public class DynamicImage implements ReportCallback {
    String debugMsgs = "Debug Messages";
    String errMsgs = "Error Messages";

    public void end(BookModel bk, Object o) {
    }

    public void start(BookModel bk, Object o) {
        try {
            bk.getLock();

/*****
 * Get the value of a defined name and build a path based on its
 * value. Assume that the workbook contains a defined name of
 * "state" and that there is a .gif file on the hard drive with
 * the same base name as the defined name value.
 *****/

```

```

*****/
    String stateCode = bk.getDefinedName("state");
    String imagePath =
        "C:\\Program Files\\Actuate11\\espreadsheet\\examples"
        + "\\callback\\DynamicImage\\" + stateCode + ".gif";

/*****
 * To add an image to a worksheet, use the following steps:
 *
 * 1. Create a file name from the state code.
 * 2. Create a URL from the file name and a path.
 * 3. Open an input stream to the URL.
 * 4. Get a DrawingModel object associated from the
 *    BookModel object.
 * 5. Use the DrawingModel object to create a ShapeAnchor.
 * 6. Add the picture to the DrawingModel object using
 *    the ShapeAnchor and the input stream.
 *****/
    URL picURL = new URL("file:\\" + imagePath);
    java.io.InputStream inputStream = picURL.openStream();
    DrawingModel dm = bk.getDrawing();
    double left = 1.0;
    double top = 1.0;
    double right = 5.5;
    double bottom = 6.0;
    ShapeAnchor sa = dm.createShapeAnchor(left, top, right,
        bottom);
    dm.addPicture(inputStream, sa);

/*****
 * Close the input stream and re-open it before calling
 * addPicture( ) a second time.
 *****/
    inputStream.close();
    inputStream = picURL.openStream();

/*****
 * Change the ShapeAnchor and add the same image to the
 * worksheet in a different location and with different
 * dimensions. The proportions of the displayed image vary with
 * the width and height of the cells that form its coordinates.
 *****/
    sa = dm.createShapeAnchor(left + 6, top, right + 9.5,
        bottom + 10);
    dm.addPicture(inputStream, sa);
} catch (Exception ex) {

```

```

        Messages.display(bk, ex + " in start method",
            errMsgs);
    }
    finally{
        bk.releaseLock();
    }
}
}
}

```

## Understanding the CellLocking example

The CellLocking example illustrates how to lock specific cells and ranges of cells on a worksheet. This example illustrates the following techniques:

- Unlocking all cells on a worksheet
- Locking a range of cells on a worksheet
- Setting the protection features of a worksheet
- Setting a password for locking and unlocking a worksheet

You can test this example using CellLocking.sod.

```

/*****
* The CellLocking callback class
*****/
import com.flj.ss.report.*;
import com.flj.ss.*;
import com.flj.util.*;
import messages.*;

public class CellLocking implements ReportCallback {
    String debugMsgs = "Debug Messages";
    String errMsgs = "Error Messages";

    public void start(BookModel bk, Object o){}

    public void end(BookModel bk, Object o) {
        try {
            bk.getLock();
        }
    }
}

/*****
* To set protection properties of cells, perform the following
* steps:
* 1. Get a CellFormat object from the BookModel object.
* 2. Get a ProtectionFormat object from the CellFormat
*    object.
* 3. Set the locked property of the ProtectionFormat object.
* 4. Set the selection to the range of cells to lock or
*    unlock.
*****/

```

```

*      5. Call the setCellFormat( ) method of the BookModel
*      object.
*      6. Set the type of protection by calling the
*      setSheetProtection( ) object of the BookModel object
*      with the desired set of flags.
*****/
    CellFormat cf = bk.getCellFormat();
    ProtectionFormat pf = cf.protection();

/*****
* Set the locked flag to false, signifying unlocked.
*****/
    pf.setLocked(false);

/*****
* Set the selection range from cell A1 to the end of all the
* data, locking all the cells in the sheet.
*****/
    bk.setSelection(0,0,bk.getMaxRow(),bk.getMaxCol());
    bk.setCellFormat(cf);

/*****
* Set the locked flag to true, signifying locked.
*****/
    pf.setLocked(true);

/*****
* Unlock the cells in the range A1:D4. Specify the cell range
* in the setCellFormat( ) method, as compared to setting the
* selection before calling setCellFormat( ).
*****/
    bk.setCellFormat(cf,0,0,3,3);
    int activeSheet = bk.getSheet();

/*****
* Set the first argument of setSheetProtection( ) to a sheet
* index. Set the second argument to true to enable protection
* and to false to disable protection. The third parameter is an
* optional password, which if not null, must be a string that
* the user must match. Use the fourth argument to specify what
* kind of protection to set. Set this value to the sum of the
* types of operations to allow.
* The possible types of protection are
*
* kAllowEditObjects
* kAllowFormatCells
* kAllowFormatColumns
* kAllowFormatRows
* kAllowInsertColumns

```

```

* kAllowInsertRows
* kAllowInsertHyperlinks
* kAllowDeleteColumns
* kAllowDeleteRows
* kAllowSelectLocked
* kAllowSort
* kAllowUseAutoFilter
* kAllowUsePivotRanges
* kAllowSelectUnlocked
* kAllowNone
*****/
        bk.setSheetProtection(activeSheet, true, null,
            Constants.kAllowSelectUnlocked);
    }
    catch (FlException ex)
    { Messages.display(bk, ex + " in end() method",
        errMsgs); }
    finally{
        bk.releaseLock();
    }
}
}

```

## Understanding the CellFormatting example

The CellFormatting example illustrates several of the ways you can affect the formatting of cell data using a callback class. This example illustrates the following techniques:

- Setting alignment
- Setting a background color and pattern
- Setting a border
- Setting font characteristics

You can test this class using CellFormatting.sod.

```

/*****
* The CellFormatting callback class
*****/
import com.flj.ss.report.*;
import com.flj.ss.*;
/*****
* The com.flj.paint package contains AlignFormat, FillFormat,
* and FontFormat interfaces.
*****/
import com.flj.paint.*;
import com.flj.util.*;

```

```

import com.f1j.ss.FontFormat;
import com.f1j.ss.FillFormat;
import messages.*;

public class CellFormatting implements ReportCallback {
    String debugMsgs = "Debug Messages";
    String errMsgs = "Error Messages";

    public void start(BookModel bk, Object o){}

    public void end(BookModel bk, Object o) {
        try {
            bk.getLock();

/*****
* Cell Alignment
*
* To align the contents of cells, perform the following steps:
* 1. Get the CellFormat object from the BookModel object.
* 2. Get the AlignFormat object from the CellFormat object.
* 3. Call the appropriate method on the AlignFormat object,
*    passing it the appropriate static AlignFormat field.
* 4. Call setCellFormat() on the current selection.
*
* See the Javadoc for com.f1j.paint.AlignFormat for all
*    the available methods and static fields.
*****/
            CellFormat cf = bk.getCellFormat();
            com.f1j.paint.AlignFormat af = cf.align();
            af.setHorizontalAlignment(
                com.f1j.paint.AlignFormat.eHorizontalCenter);
            int firstRow = 0;
            int firstCol = 0;
            int lastRow = 3;
            int lastCol = 3;

/*****
* Set the formatting for a range of cells, specifying the
*    CellFormat object
* and the coordinates of the cell range.
*****/
            bk.setCellFormat(cf, firstRow, firstCol, lastRow,
                lastCol);

/*****
* Set the selection before getting the CellFormat object when
* you want to retain the formatting properties of the cells in
* that selection. Don't reuse a CellFormat object unless you
* want to apply the properties set in that object.
*****/

```



```

        bk.setSelection(4,0,6,3);
        cf = bk.getCellFormat();

/*****
 * Setting colors and background patterns
 *
 * Use the FillFormat object to set fill properties. This
 * process is identical to setting cell alignment, except that
 * you use the FillFormat object and set its properties instead
 * of the AlignFormat object.
 *
 * See the Javadoc for com.f1j.paint.FillFormat for all
 * the relevant methods and static fields.
 *****/
        FillFormat fillf = cf.fill();
        fillf.setPattern(FillFormat.ePatternTrellis);
        bk.setCellFormat(cf);
/*****
 * Setting borders
 *
 * Use the BorderFormat object to set border properties. This
 * process is identical to setting cell alignment, except that
 * you use the BorderFormat object and set its properties
 * instead of the AlignFormat object.
 * See the Javadoc for com.f1j.paint.BorderFormat for all
 * the methods and static fields you can specify.
 *****/
        bk.setSelection(7,0,11,3);
        cf = bk.getCellFormat();
        BorderFormat bf = cf.border();
        bf.setStyleAndColor(
            BorderFormat.eBorderBottom,
            BorderFormat.eBorderDouble,
            java.awt.Color.magenta.getRGB() );
        bf.setStyleAndColor(
            BorderFormat.eBorderTop,
            BorderFormat.eBorderDouble,
            java.awt.Color.magenta.getRGB() );
        bf.setStyleAndColor(
            BorderFormat.eBorderRight,
            BorderFormat.eBorderDouble,
            java.awt.Color.magenta.getRGB() );
        bf.setStyleAndColor(
            BorderFormat.eBorderLeft,
            BorderFormat.eBorderDouble,
            java.awt.Color.magenta.getRGB() );
        bk.setCellFormat(cf);

```

```

/*****
* Setting font characteristics
*
* Use the FontFormat object to set font properties. This
* process is identical to setting cell alignment, except that
* you use the FontFormat object and set its properties instead
* of the AlignFormat object.
* See the Javadoc for com.f1j.paint.FontFormat for all
* the methods and static fields.
*****/
        FontFormat ff= cf.font();
        bk.setActiveCell(0, 0);
        ff.setBold(true);
        ff.setItalic(true);
        ff.setUnderline(FontFormat.eUnderlineSingle);
        bk.setCellFormat(cf);
/*****
* Setting the default font
*
* Set the default font by calling the setDefaultFont() method
* on the BookModel object, specifying the name of the font and
* the font size. If you specify a TrueType font, text scales
* proportionally. Specify the font size in twips, which are 20
* times point size.
*****/
        bk.setDefaultFont("Times New Roman", 20 * 18);
/*****
* Showing formulas compared to showing results
*
* Use setShowFormulas( ) of the BookModel object to display
* formulas in the cells as compared to showing the values. The
* current selection has no effect on this operation. It is a
* global operation. Pass setShowFormulas( ) a value of true to
* show formulas and a value of false to show values.
*****/
        bk.setShowFormulas(true);
        bk.setShowFormulas(false);
    }
    catch (FlException ex)
    { Messages.display(bk, ex + " in end() method",
        errMsgs); }
    finally{
        bk.releaseLock();
    }
}
}

```

## Understanding the WordFormatting example

The WordFormatting example illustrates how to set font characteristics for individual words within a text string in a single cell. This example illustrates the following techniques:

- Setting the active cell
- Selecting individual words in a cell that contains text
- Applying formatting options to individual words

You can test the WordFormatting example using WordFormatting.sod.

```
/******
 * The WordFormatting callback class
 *****/
import com.flj.ss.report.*;
import com.flj.ss.*;
import com.flj.ss.FontFormat;
import com.flj.paint.*;
import com.flj.util.*;
import messages.*;

public class WordFormatting implements ReportCallback {
    String debugMsgs = "Debug Messages";
    String errMsgs = "Error Messages";

    public void start(BookModel bk, Object o){
    }
    public void end(BookModel bk, Object o) {
        try {

/******
 * To apply distinct formatting attributes to individual words
 * in a string, perform the following steps:
 *
 * 1. Set the active cell using the setActiveCell() method.
 * 2. Select the words you want to format using the
 *    setTextSelection() method.
 * 3. Get a CellFormat object from the BookModel object.
 * 4. Get a FontFormat object from the CellFormat object.
 * 5. Apply the attribute with a set method of the FontFormat
 *    object.
 * 6. Set the cell format using the setCellFormat() method of
 *    the book object.
 *
 * The following code loops through the text in cell A1 and
 * bolds the first word, italicizes the second word, underlines
```

```

* the third word and sets the font size to 14 point for the
* fourth word.
*****/
    bk.setActiveCell(0,0);
    int start = 0;
    int stop = 0;
    String text = bk.getText(0,0);
    CellFormat cf = bk.getCellFormat();
    FontFormat ff = cf.font();

    for(int i=0; i < 4; i++){
        stop = text.indexOf(" ", start);
        stop = (stop==-1)?text.length() -1:stop;
        bk.setTextSelection(start,stop);
        cf = bk.getCellFormat();
        ff = cf.font();
        switch(i){
            case 0: ff.setBold(true); break;
            case 1: ff.setItalic(true); break;
            case 2: ff.setUnderline(
                FontFormat.eUnderlineSingle); break;
            case 3: ff.setSizePoints(14); break;
            default:
        } // end switch
        bk.setCellFormat(cf);
        start = stop + 1;
    } // end for
} // end try
catch (FlException ex)
{ Messages.display(
    bk, ex + " in end() method", errMsgs); }
finally{ bk.releaseLock(); }
}

```

## Understanding the CustomNumberFormatting example

The CustomNumberFormatting example illustrates how to set the formatting for numeric values that appear in a range of cells on a worksheet. This example illustrates the following techniques:

- Creating a fractional format
- Creating a currency format
- Highlighting negative values using color and parentheses

- Creating date and time formats
- Creating a Scientific Notation format

You can test this example using CustomNumberFormatting.sod.

```

/*****
 * The CustomNumberFormatting callback class
 *****/
import com.flj.ss.report.*;
import com.flj.ss.*;
import com.flj.paint.*;
import com.flj.util.*;
import messages.*;

public class CustomNumberFormatting implements ReportCallback {
    String debugMsgs = "Debug Messages";
    String errMsgs = "Error Messages";

    public void end(BookModel bk, Object o){}

    public void start(BookModel bk, Object o) {
        try {
            bk.getLock();
        }
    }
}

/*****
 * Custom Number Formatting
 *
 * To assign a custom format for a number, perform the following
 * steps:
 * 1. Define a mask to define the format.
 * 2. Call setCustomFormat( ) to apply the mask to the
 *    NumberFormat object.
 * 3. Get a CellFormat object from the BookModel object.
 * 4. Get the NumberFormat object from the CellFormat object.
 *
 * The format mask contains four sections, separated by
 * semicolons. The first section defines the format for positive
 * numbers, the second for negative numbers, the third for zero
 * values, and the fourth is a string that displays when the
 * cell entry is a non-numeric value.
 *
 * The following are examples of custom number format masks:
 * Fractional:          "# ??/??;-??/??"
 * Percentage:          "0%;-0%;0%;;"\Err:31\"
 * Red for negative:    "[Black]###;[Red]###;#;\Err:2\"
 * Parentheses for negative: "#,###;(,###;#);#"
 * Currency:           "$#,###,###.##;-$,###,###.##;$0;"INVALID ENTRY"

```

```

* You can find the complete list of formatting symbols in
* the Javadoc for com.flj.util.NumberFormat.setCustomFormat().
*****/
    bk.setActiveCell(0,0);
    CellFormat cf = bk.getCellFormat();
/*****
* It is not possible to use an import statement to avoid using
* a fully-qualified name for NumberFormat because there is a
* class named NumberFormat in com.flj.ss and an interface in
* com.flj.util, so the compiler cannot disambiguate them.
*****/
    com.flj.util.NumberFormat nf = cf.number();
    String fractionalMask = "# ??/??;-# ??/??";
    nf.setCustomFormat(fractionalMask);
    bk.setCellFormat(cf);

    bk.setActiveCell(1,0);
    cf = bk.getCellFormat();
    nf = cf.number();
    String negativeMask = "#,###.00;[Red](#,###.00);0";
    nf.setCustomFormat(negativeMask);
    bk.setCellFormat(cf);

/*****
* Date and time formatting
*
* Dates and time values are a special case of number
* formatting. To set a date format, use one of, or a variation
* of, one of the following strings for the format mask:
*
* "mm/dd/yyyy"      Month, day, and year displayed as integers
* "mm/dd/yy"        Month, day, and year displayed as integers
* "dd/mm/yy"        Day, month, and year displayed as integers
* "yy/mm"           Year and month displayed as integers
* "dddd/mm/yy"      Day of the week displayed in text
* "ddd/mm/yyyy"     Day of the week displayed as abbreviaton
* "dd/mmmm/yy"      Month spelled out
* "dd/mmm/yy"       3-letter abbreviation for month
*
* To set a time format, use one of the following strings for
* the format mask:
*
* hh:mm:ss          Leading zeros on all values
* h:m:s             No leading zeros
* hh:mm:ssAM/PM     12 hour time
* [h]               Total number of hours
* [m]               Total number of minutes
* [s]               Total number of seconds

```

```

* s.0                      Seconds, including fractional part, with
*                          lead zeros
*
* Find the complete list of date and time formatting symbols
* in the Javadoc for
* com.flj.util.NumberFormat.setCustomFormat().
*****/
    bk.setActiveCell(2,0);
    cf = bk.getCellFormat();
    nf = cf.number();
    String dateMask = "dddd, mmmm dd, yyyy";
    nf.setCustomFormat(dateMask);
    bk.setCellFormat(cf);

    bk.setActiveCell(3,0);
    cf = bk.getCellFormat();
    nf = cf.number();
    String timeMask = "h:mm:ssAM/PM";
    nf.setCustomFormat(timeMask);
    bk.setCellFormat(cf);

    bk.setActiveCell(4,0);
    cf = bk.getCellFormat();
    nf = cf.number();
    nf.setCustomFormat(dateMask + ", " + timeMask );
    bk.setCellFormat(cf);

/*****
* Scientific Notation formatting
*
* Format numbers with scientific notation by specifying
* any of the following three format masks:
*
* 0.00E-00                Displays like 3.14E02
* 0.00000E+00            Displays like 3.14159E+02
* 0.000e-                Displays like 3.14E02
* ##0e+0                Displays like 314E+0
*****/
    bk.setActiveCell(5,0);
    cf = bk.getCellFormat();
    nf = cf.number();
    String scientificMask = "0.00000e+00";
    nf.setCustomFormat(scientificMask);
    bk.setCellFormat(cf);
}
catch (FlException ex)
{ Messages.display(bk, ex + " in end() method",
  errMsgs); }

```

```

        finally{
            bk.releaseLock();
        }
    }
}

```

## Understanding the FileWriting example

The FileWriting example illustrates how to use a callback class to create a variety of files that contain the data in a workbook. This example illustrates the following techniques:

- Setting password protection on an output file
- Setting the code page for an output file
- Writing an Excel output file
- Writing a tabbed-text output file

You can test this example using FileWriting.sod.

```

/*****
 * The FileWriting callack class
 *****/
import com.flj.ss.report.*;
import com.flj.ss.*;
import com.flj.data.*;
import com.flj.util.*;
import java.io.*;
import messages.*;

public class FileWriting implements ReportCallback {
    String debugMsgs = "Debug Messages";
    String errMsgs = "Error Messages";

    public void start(BookModel bk, Object o){
    }

    public void end(BookModel bk, Object o) {
        try{
            bk.getLock();

/*****
 * To write a file containing the contents of your workbook,
 * perform the following steps:
 *     1. Get a Book object from the BookModel object.
 *     2. Build or code a path to the file you want to write.
 *     3. Create a Book object from the BookModel object
 *     4. Create a Document object from the Book object
 *****/

```



```

*      5. Get a DocumentSaveOptions object from the Document
*      object
*      6. Pass false to the setAutoExecute() method of the
*      DocumentSaveOptions object to prevent the callback
*      from re-running endlessly
*      7. Call the setSaveOptions() method on the Document object
*      8. Call fileSaveAs( ) on the Document object, passing the
*      type of file as a constant of the Document class
*****/
String dir =
    "c:\\Program Files\\Actuate11\\espreadsheet"
    + "\\examples\\callback\\FileWriting";
String fileSeparator =
    System.getProperty("file.separator");
Book book = bk.getBook();
Document doc = book.getDocument();

/*****
* Set the autoExecute option to false so this callback doesn't
* run again on the new document. If you need the new
* document to auto-execute, you can set a boolean class
* variable to test whether this callback has run before and
* return immediately if it has, thus avoiding an endless loop
* of document creation.
*****/

DocumentSaveOptions docSaveOptions =
    doc.getDocumentSaveOptions();
docSaveOptions.setAutoExecute(false);
doc.setDocumentSaveOptions(docSaveOptions);

/*****
* Create an output file in Excel 97 format
*****/
String fileName = dir + fileSeparator +
    "fileWriting.xls";
java.io.File outfile = new java.io.File(fileName);
doc.fileSaveAs(outfile,
    DocumentType.EXCEL_97_WORKBOOK, null);

/*****
* Create an output file in tab-delimited text format
*****/
fileName = dir + fileSeparator + "tabDelimitedText.txt";
outfile = new java.io.File(fileName);
doc.fileSaveAs(outfile, DocumentType.TABBED_TEXT, null);

```

```

/*****
 * Create a password protected Actuate 11 workbook file.
 *
 * The maximum length of a password is 15 characters. Password
 * protection is only supported in the BIRT Spreadsheet and Excel
 * file formats.
 *
 * Password-protected files are encrypted. The types of
 * encryption are explained in the Javadoc for the Document
 * class. You can set two kinds of passwords, one for opening
 * the file and another for saving it. The password protection
 * is set in the DocumentSaveOptions object. The process for
 * using the DocumentSaveOptions object is
 * 1. Get a DocumentSaveOptions object from the Document
 *    object
 * 2. Call the setOpenPassword() or setModifyPassword( )
 *    method on the DocumentSaveOptions object
 * 3. Call the setSaveOptions() method on the Document object
 *****/
    fileName = dir + fileSeparator +
        "Actuate11Workbook.sod";
    outfile = new java.io.File(fileName);
    docSaveOptions.setOpenPassword("KlaatuBarataNik");
    doc.setDocumentSaveOptions(docSaveOptions);
    doc.fileSaveAs(outfile,
        DocumentType.ACTUATE_10_WORKBOOK, null);
} catch(Exception e){
    Messages.display(bk, e + " write failure", errMsgs);
}
bk.releaseLock();
}
}

```

## Understanding the DrawingObjects example

The DrawingObjects example illustrates how to use a callback class to draw objects on a worksheet. This example illustrates the following techniques:

- Adding a graphic object to a DrawingModel object
- Drawing graphic shapes on a worksheet

You can test this example using DrawingObjects.sod.

```

/*****
 * The DrawingObjects callback class
 *****/
import com.flj.ss.report.*;
import com.flj.ss.*;

```

```

import com.flj.util.*;
import messages.*;

public class DrawingObjects implements ReportCallback {
    String debugMsgs = "Debug Messages";
    String errMsgs = "Error Messages";

    public void end(BookModel bk, Object o) {}

    public void start(BookModel bk, Object o) {
        try {
            bk.getLock();

/*****
*   To draw four graphical objects on a worksheet, perform the
*   following steps:
*       1. Get a DrawingModel object from the BookModel object.
*       2. Define the the location of where to draw the object
*           on the sheet.
*       3. Get a ShapeAnchor from the DrawingModel.
*       4. Add a graphic object to the DrawingModel object,
*           specifying the kind of object.
*
*       The following arguments specify possible shapes:
*           eShapeLine
*           eShapeOval
*           eShapeRectangle
*           eShapeTextBox
*****/
            DrawingModel dm = bk.getDrawing();
            double left = 1.0;
            double top = 1.0;
            double right = 4.4;
            double bottom = 10.5;
            ShapeAnchor sa = dm.createShapeAnchor(left, top, right,
                bottom);
            dm.addShape(dm.eShapeOval, sa);

            left = 4.8;
            top = 1.8;
            right = 7.6;
            bottom = 5.7;
            sa = dm.createShapeAnchor(left, top, right, bottom);
            dm.addShape(dm.eShapeRectangle, sa).setText(
                "text for a rectangle?");
            left = 5.0;
            top = 2.0;

```

```

        right = 7.4;
        bottom = 5.5;
        sa = dm.createShapeAnchor(left, top, right, bottom);
        dm.addShape(dm.eShapeLine, sa);

        left = .5;
        top = 11.3;
        right = 4.0;
        bottom = 15.0;
        sa = dm.createShapeAnchor(left, top, right, bottom);
        dm.addShape(dm.eShapeTextBox, sa).setText(
            "My very own text box");
    } catch (Exception ex) {
        Messages.display(bk, ex + " in start() method",
            errMsgs);
    }
    finally{
        bk.releaseLock();
    }
}
}
}

```

## Understanding the DrawingCharts example

The DrawingCharts example illustrates how to use a callback class to add charts to a workbook. This example illustrates the following techniques:

- Setting the type of a worksheet
- Drawing a chart on a worksheet
- Setting the link range for a chart
- Setting the axis titles for a chart
- Setting the name of a series for a chart
- Setting the title of a chart
- Setting a trendline for a chart
- Setting the style of a chart

You can test this example using DrawingCharts.sod.

```

/*****
 * The DrawingCharts callback class
 *****/
import com.flj.ss.report.*;
import com.flj.ss.*;

```

```

import com.flj.util.*;
import messages.*;

public class DrawingCharts implements ReportCallback {
    String debugMsgs = "Debug Messages";
    String errMsgs = "Error Messages";

    public void end(BookModel bk, Object o) {}

    public void start(BookModel bk, Object o) {
        try {
            bk.getLock();

/*****
*   To create a chart sheet, perform the following steps:
*
*       1. Insert a new sheet into the workbook.
*       2. Get a Book object from the BookModel object.
*       3. Get the new sheet object from the Book object.
*       4. Make the new sheet the active sheet.
*       5. Set the type of the new sheet to Chart type.
*       6. Get a DrawingModel object from the BookModel object.
*       7. Define the coordinates for a ShapeAnchor object.
*       8. Use the DrawingModel to create a ShapeAnchor.
*       9. Create a ChartGraphic object with the DrawingModel
*           object.
*       10. Get a ChartModel object from the FcChart object.
*****/
            bk.insertSheets(0,1);
            Book book = bk.getBook();
            Sheet sht = book.getSheet(0);
            bk.setSheet(0);
            sht.setSheetType(Constants.eSheetTypeChart);
            DrawingModel dm = bk.getDrawing();
            int left = 0;
            int top = 19;
            int right = 6;
            int bottom = 29;
            ShapeAnchor sa = dm.createShapeAnchor(left, top, right,
                bottom);
            ChartGraphic chart =
                (ChartGraphic)dm.addChart(sa).getGraphic();
            com.flj.chart.ChartModel cm = chart.getChartModel();

/*****
*   To define the characteristics of the chart, perform the
*   following steps:
*

```

```

*      1. Set the chart type.
*      Available chart types are listed in the interface
*      com.flj.chart.Constants
*      2. Set the link range.
*      3. Set the axis titles.
*      4. Set the names of the series.
*      5. Set the chart title.
*      6. Set the chart style.
*
* See the Javadoc for ChartGraphic and ChartModel for the
* complete list of settable chart features.
*****/
    cm.setChartType(com.flj.chart.Constants.eBar);
    chart.setLinkRange("Sheet1!$a$1:$e$5",false);
    chart.setAxisTitle(com.flj.chart.Constants.eXAxis, 0,
        "X-axis data");
    chart.setAxisTitle(com.flj.chart.Constants.eYAxis, 0,
        "Y-axis data");
    chart.setSeriesName(0, "My Series number 1");
    chart.setSeriesName(1, "My Series number 2");
    chart.setSeriesName(2, "My Series number 3");
    chart.setSeriesName(3, "My Series number 4");
    chart.setTitle("My Chart");
    cm.set3Dimensional(true);
} catch (Exception ex) {
    Messages.display(bk, ex + " in start() method",
        errMsgs);
}
    finally{
        bk.releaseLock();
    }
}
}
}

```

## Understanding the CreatePivotRange example

The CreatePivotRange example illustrates how to create a pivot range in a BIRT Spreadsheet report. This example illustrates the following techniques:

- Creating a JDBC connection
- Creating a query on a connection
- Creating a pivot range model and applying a pivot range definition
- Creating a pivot range and its area objects
- Creating pivot row, column, and data fields
- Setting a pivot range summarizing method

- Setting the formatting for the summary fields
- Setting column grouping by date

You can test this example using CreatePivotRange.sod.

```

/*****
 * The CreatePivotRange callback class
 *****/
import com.flj.data.*;
import com.flj.data.query.*;
import com.flj.data.source.*;
import com.flj.ss.report.*;
import com.flj.ss.*;
import com.flj.ss.CellFormat;
import com.flj.util.*;

/*****
 * The com.flj.ss.pivot package contains all the classes and
 * interfaces necessary to create a pivot range.
 *****/
import com.flj.ss.pivot.*;
import java.util.*;
import messages.*;

public class CreatePivotRange implements ReportCallback {

    String debugMsgs = "Debug Messages";
    String errMsgs = "Error Messages";

    public void start(BookModel bk, Object obj) {
    }

    public void end(BookModel bk, Object obj) {
        try {
            bk.getLock();
        }
    }

/*****
 * To create a JDBC connection and query, use the following
 * steps:
 *
 * 1. Get the collection of data sources from the BookModel
 *    object.
 * 2. Create a JDBC data source object.
 * 3. Set the data source driver to
 *    sun.jdbc.odbc.JdbcOdbcDriver.
 * 4. Set the database using the factory() method.
 * 5. Get the collection of queries from the data source.
 * 6. Create a query, from the collection, using the factory()
 *    method.
 *****/

```

```

* 7. Set the data handler type to kJDBCResultSet.
* 8. Set the query string.
*****/
    DataSourceCollection sourceCollection =
        bk.getDataSourceCollection();
    com.flj.data.source.JDBC jdbc =
        (com.flj.data.source.JDBC)
        sourceCollection.factory("jdbc",
            com.flj.data.DataSourceCollection.kJDBC);
    jdbc.set("sun.jdbc.odbc.JdbcOdbcDriver",
        "jdbc:odbc:esd_securities", null, null, null);

    DataQueryCollection queryCollection =
        jdbc.getDataQueryCollection();
    JdbcQuery query = (com.flj.data.query.JdbcQuery)
        queryCollection.factory("query");
    query.setDataHandlerType(
        com.flj.data.handler.Handler.kJDBCResultSet);
    query.setQuery("SELECT " +
        "Client.Last, Client.Country, Employee.Last, " +
        "PortfolioHistory.PortfolioDate, " +
        "PortfolioHistory.CurrentValue, " +
        "PortfolioHistory.OriginalValue, " +
        "Security.Name FROM PortfolioHistory, " +
        "Security, Client, Employee WHERE " +
        "Client.ClientID=PortfolioHistory.ClientID AND " +
        "PortfolioHistory.SecurityID=Security.SecurityID " +
        "AND Client.PortfolioMgrID=Employee.EmployeeID " +
        "AND ((PortfolioHistory.PortfolioDate is not null))",
        false);

/*****
* To create a pivot range model and apply a pivot range
* definition, use the following steps:
* 1. Get the PivotRangeModel object from the BookModel
*    object.
* 2. Get the PivotRangeDef object from the PivotRangeModel
*    object.
* 3. Get the DataSourceInfo object from the PivotRangeDef
*    object.
* 4. Set the query on the DataSourceInfo object.
* 5. Set the location in the workbook on the PivotRangeDef
*    object.
* 6. Apply the definition to the PivotRangeModel object.
*****/
    PivotRangeModel model = bk.getPivotRangeModel();
    PivotRangeDef def = model.getPivotRangeDef();

```



```

        DataSourceInfo sourceInfo = def.getDataSourceInfo();
        sourceInfo.setDataQuery(query);
        def.setLocation(bk.getBook().getSheet(0), 0, 0);
        model.applyPivotRangeDef(def);

/*****
* To get the pivot range and its areas, use the following
* steps:
* 1. Get the PivotRange object from the PivotRangeModel
*    object.
* 2. Get the rowArea object from the PivotRange object.
* 3. Get the columnArea object from the PivotRange object.
* 4. Get the dataArea object from the PivotRange object.
*****/
        PivotRange pivotRange = model.getActivePivotRange();
        Area rowArea = pivotRange.getArea(PivotRange.EArea.row);
        Area columnArea =
            pivotRange.getArea(PivotRange.EArea.column);
        Area dataArea =
            pivotRange.getArea(PivotRange.EArea.data);

/*****
* To create the row, column, and data fields, use the following
* steps:
* 1. Get each field from the pivot range object by name
* 2. Add each field to the appropriate area
*****/
        Field rowField = pivotRange.getField("Employee.last");
        rowArea.addField(rowField);
        Field dataField = pivotRange.getField("CurrentValue");
        dataArea.addField(dataField);
        Field columnField =
            pivotRange.getField("PortfolioDate");
        columnArea.addField(columnField);

/*****
* To set the summarize-by type, use the following steps:
*
* 1. Call getDataField() on the PivotRange object to get a
*    special data field (the field that you used to create the
*    data field will not work!).
* 2. Get a special summary Field from the special data field by
*    calling getSummary(index). If there is only one summary
*    field, the index value is 0.
* 3. Get a SummaryFieldSettings object from the special summary
*    field.
* 4. Call setFunction() on the SummaryFieldSettings object.

```

```

* 5. Set the Summary function name by calling setName on the
*   SummaryFieldSettings object.
*****/
    Field dataFld = pivotRange.getDataField();
    Field summary= dataFld.getSummary(0);
    SummaryFieldSettings sfs =
        summary.getSummaryFieldSettings();
    sfs.setFunction(
        BaseFieldSettings.EFunctionType.eAverage);
    sfs.setName(dataField.getSourceName() + " Average");

/*****
* To set the formatting for the summary data, use the following
* steps:
*   1. Get a CellFormat object from the BookModel object.
*   2. Get a NumberFormat object from the CellFormat object.
*   3. Create a custom formatting mask.
*   4. Pass the custom formatting mask to the NumberFormat
*       object.
*   5. Call setNumberFormat on the the SummaryFieldSettings
*       object.
*   6. Call the setSummaryFieldSettings method on the summary
*       field object.
*****/
    CellFormat cf = bk.getCellFormat();
    com.flj.util.NumberFormat nf = cf.number();
    String mask = "$#,##_);[Red](#,###.00);0";
    nf.setCustomFormat(mask);
    sfs.setNumberFormat(nf);
    summary.setSummaryFieldSettings(sfs);

/*****
* To set column grouping by date, use the following steps:
*   1. Get a FieldGroupDef object by calling
*       createFieldGroupDef() on the column field.
*   2. Call setDateTypeEnabled on the FieldGroupDef object,
*       specifying the type of grouping. Valid values are:
*       FieldGroupDef.EDateType.eYears
*       FieldGroupDef.EDateType.eQuarters
*       FieldGroupDef.EDateType.eMonths
*       FieldGroupDef.EDateType.eDays
*       FieldGroupDef.EDateType.eHours
*       FieldGroupDef.EDateType.eMinutes
*       FieldGroupDef.EDateType.eSeconds
*   3. To set the beginning and ending values for the
*       grouping, use the following steps:

```

```

*      a. Call setWantsDefaultFinalValue(false).
*      b. Call setWantsDefaultInitialValue(false).
*      c. Call setInitialValue(date string).
*      d. Call setFinalValue(date string).
* 4. Call applyFieldGroupDef(FieldGroupDef) on the column
*    field.
*
* Note! If there are any null values, or non-date values in
* the data for the field on which grouping is based,
* createFieldGroupDef() returns a null value! The same is
* true if the field is numeric, not date type and there are
* non-numeric values in the data.
*****/
    FieldGroupDef fg = columnField.createFieldGroupDef();
    if(fg !=null){
        fg.setDateTypeEnabled(
            FieldGroupDef.EDateType.eQuarters ,true);
        fg.setWantsDefaultFinalValue(false);
        fg.setWantsDefaultInitialValue(false) ;
        fg.setInitialValue("01/01/2002");
        fg.setFinalValue("12/31/2002");
        columnField.applyFieldGroupDef(fg);
    }
} catch (Exception ex) {
    Messages.display(bk, ex.getMessage(), errMsgs);
}
finally{
    bk.releaseLock();
}

/*****
* An array of summarize-by methods. This is used by
* getFunctionType() to select the value of the function type.
*****/
    private String functionNames[] = {
        "Sum", "Count", "Average", "Max", "Min", "Product",
        "CountNums", "StdDev", "StdDevp", "Var", "Varp"};

/*****
* A utility method to return the right function type object for
* a given summarize-by type. The summarize-by type is a string
* that must match one of the entries in functionNames[]. If it
* matches no names, this method returns the function type
* object for the Count function.
*****/

```

```

private BaseFieldSettings.EFunctionType getFunctionType(String
    fName) {
    for(int i = 0; i < functionNames.length; i++){
        if(fName.equals(functionNames[i]))
            return BaseFieldSettings.s_functions[i];
    }
    return BaseFieldSettings.s_functions[1];
}
}

```

## Understanding the CreatingADataRange example

The CreatingADataRange example illustrates how to create a data range in a BIRT Spreadsheet report. This example illustrates the following techniques:

- Creating a data range model and applying a data range definition
- Creating a data range and its area objects
- Adding row sections to a data range
- Setting the grouping for a data range
- Adding script commands to a data range

The following example assumes that the Sfddata database is installed on the system on which the example runs:

```

/*****
 * The CreatingADataRange class
 *****/
import com.flj.data.*;
import com.flj.data.query.*;
import com.flj.data.source.*;
import com.flj.ss.*;
import com.flj.util.*;
import com.flj.ss.report.*;

/*****
 * The com.flj.ss.datarange package contains all the classes and
 * interfaces necessary to create a data range.
 *****/
import com.flj.ss.datarange.*;
import com.flj.ss.report.functions.*;
import com.flj.data.handler.*;
import java.util.*;

public class CreatingADataRange implements ReportCallback {
    static DataSourceCollection sources;
    static JDBC source;
}

```

```

static DataQueryCollection queries;
static DataQuery query;
static DatabaseQuery dbQuery;
static com.flj.ss.datarange.DataRangeModel model;
static com.flj.ss.datarange.DataRangeDef definition;
static com.flj.ss.datarange.DataRange datarange;
static com.flj.ss.datarange.Section rowRoot;
static com.flj.ss.datarange.Section firstLeaf;
static com.flj.ss.datarange.Section section1;
static com.flj.ss.datarange.Section secondLeaf;
static com.flj.ss.datarange.Section section2;
static BookModel book;

    public void end (BookModel bk, Object obj) { }

/*****
*   The start method calls four local methods to do the
*   following tasks:
*       1) Add a data source to the workbook
*       2) Create an empty data range
*       3) Add row sections to the data range
*       4) Add cell commands to the data range
*
*****/

    public void start (BookModel bk, Object obj) {
        book = bk;
        try {
            addDataSource( );
            createEmptyDataRange( );
            addRowSections( );
            addCellCommands( );
        } catch (FlException ex) {
            System.out.println("Exception = " + ex);
        }
    }

/*****
*   The addDataSource method does the following tasks:
*       1) Get the data source collection object from the book
*           model object
*       2) Create a JDBC data source and name it connection1
*       3) Attach the data source to a specific data source and
*           driver
*       4) Get the data query collection object
*       5) Create a new query and name it query1
*       6) Set the handler type to JDBC result set

```

```

*      7) Cast the query to DatabaseQuery which has a
*      setQuery( ) method
*****/
private static void addDataSource( ) throws FlException {
    try{
        sources = book.getBook().getDataSourceCollection();
        source = (JDBC)sources.factory("connection1",
        DataSourceCollection.kJDBC);
        source.set("sun.jdbc.odbc.JdbcOdbcDriver",
        "jdbc:odbc:sfdata, null, null, null);

        queries = source.getDataQueryCollection();
        query = queries.factory("query1");
        query.setDataHandlerType(Handler.kJDBCResultSet);

        dbQuery = (DatabaseQuery)query;
        dbQuery.setQuery("SELECT * FROM customers", false);
    } catch (FlException ex) {
        System.out.println("Exception = " + ex);
    }
}

/*****
* The createEmptyDataRange method does the following tasks:
* 1) Get a DataRangeModel object from the book object
* 2) Get the DataRangeDefinition object from the model
* 3) Apply the definition object to the data range model,
* getting a DataRange object in the process
*****/
private static void createEmptyDataRange( ) throws
    FlException {
    try{
        model = book.getDataRangeModel();
        definition = model.getDataRangeDef();
        datarange = model.applyDataRangeDef(definition);
    } catch (FlException ex) {
        System.out.println("Exception = " + ex);
    }
}

/*****
* AddRowSections builds the rows of data range,
* performing the following tasks:
* 1) Get the row root section from the DataRange object
* 2) Get the first leaf section object from the row root
* object
* 3) Create a section in first row by creating a parent
* of the first leaf section

```

```

*      4) Get the second leaf section of the row root
*      5) Add a section named "Section2" in row 2
*      6) Name the section in row1
*      7) Give section2 some data commands
*****/
private static void addRowSections( ) throws FlException {
    try{
        rowRoot = datarange.getRowRootSection();
        firstLeaf = rowRoot.getChild(0);
        section1 = firstLeaf.createParent("FirstLeafParent");
        secondLeaf = rowRoot.getChild(1);
        section2 = secondLeaf.createParent("Section2");
        section1.setName("Section1");
        section2.setCommands("group(custid)");
    } catch (FlException ex) {
        System.out.println("Exception = " + ex);
    }
}

/*****
* AddCellCommands adds the commands in the
* scripting language
*****/
private static void addCellCommands( ) throws FlException {
    try{
        book.setEntry(0, 0, "CustID");
        book.setEntry(1, 0, "#custid");
        book.setEntry(0, 1, "Last");
        book.setEntry(1, 1, "#contact_last");
        book.setEntry(0, 2, "First");
        book.setEntry(1, 2, "#contact_first");
    } catch (FlException ex) {
        System.out.println("Exception = " + ex);
    }
}
}

```





## Working with VBA

This chapter contains the following topics:

- Including VBA functionality in a spreadsheet report
- Creating and using a template file
- Using a form control object
- Reviewing a VBA template example
- Using workbook and worksheet code names

---

## Including VBA functionality in a spreadsheet report

You can include VBA functionality in a spreadsheet report you publish using Actuate BIRT iServer System. To use VBA functionality, you must develop the VBA code in an Excel file, then use that file as a template for the spreadsheet report. You can use VBA to add custom features such as macros, custom calculations, auto shapes, or cell comments. You can also use VBA to add interactive features. For example, you can use VBA to enable a user to write data back to a database.

When you develop a VBA template with a spreadsheet report, follow these guidelines:

- To enable users to trigger the code in a completed report, you can use custom toolbars, menu items, or VB forms.
- BIRT Spreadsheet Designer supports the following form control object types for use in Excel 97-2003 (.xls) workbooks:
  - Button
  - Drop down list
  - Combo box
- You cannot call VBA code from a cell formula.

The following sections describe how to set up and use a VBA template with a spreadsheet report.

---

## Creating and using a template file

To use an Excel file as a template for a spreadsheet report, you must first create the report file and the Excel template. As you develop the VBA code, you can test and refine the code by opening the design in Excel. To distribute the report, you publish it using Actuate BIRT iServer System.

### Creating and naming a template file

To associate a VBA template with a spreadsheet report, create the BIRT Spreadsheet Designer file, then create an Excel file that includes VBA code. Save the Excel file and the BIRT Spreadsheet Designer file in the same directory. When you name the Excel file, use the following format:

```
filename.sod.xls
```

where filename is the name of the BIRT Spreadsheet Designer file.

For example, if the BIRT Spreadsheet Designer file is named MyVBAReport.sod, you must name the Excel template file MyVBAReport.sod.xls.

## Testing VBA code with a spreadsheet report

You can test and refine the VBA code as you develop it, using actual report data and formatting. You must first run the report to create a spreadsheet report. Then, to merge the spreadsheet report file with the Excel template, you save the SOI file as an Excel XLS file, then open the file in Excel. You use the same file name as the as the template, without the .sod section of the file name.

For example, to test a report design named VBA.sod with an Excel template file named VBA.sod.xls, save report preview as VBA.xls, then open VBA.xls in Excel.

### How to test VBA code with a spreadsheet report



- 1 In BIRT Spreadsheet Designer, run the report.
- 2 When the report preview appears, choose File→Save As, then save the file as an Excel file using the following format:

filename.xls

where filename.sod.xls is the name of the Excel template file.

- 3 Close the XLS file, then open it in Excel.

The file includes both the BIRT Spreadsheet Designer data and formatting and the Excel VBA features.

- 4 Make your VBA changes, then save the template file.

## Publishing a report with a VBA template



When you use Publish Report to upload the spreadsheet report to Actuate BIRT iServer System, ensure that the template file is in the same directory as the BIRT Spreadsheet Designer file and that the template name matches the name of the SOD file. BIRT Spreadsheet Designer includes the template file when it creates and uploads the spreadsheet report executable file. When users open the published and distributed report, the VBA functionality appears in the report.

## Testing VBA code with Excel 2007

To test VBA code using Excel 2007, set the BIRT Spreadsheet Designer target file format to Excel 2007. Follow the same procedure described previously for testing VBA code with earlier Excel versions, but select the XLSM file format when you save the report preview.

For more information about compatibility with Excel 2007, see Chapter 17, “Personalizing BIRT Spreadsheet Designer.”

---

## Using a form control object

To run a macro in a spreadsheet report, you can add a button, drop-down list, or combo box to the spreadsheet report design (.sod) file. Before adding a form control object to a spreadsheet report, set the BIRT Spreadsheet Designer target file format to Excel 97-2003 Workbook. Then, select View→Toolbars and choose Forms. Next, open the Excel workbook in which you have at least one macro.

To add a form control to a spreadsheet design file, in e.Spreadsheet Designer, select an option from the Forms toolbar. While the cursor appears as a cross, press the left mouse button and draw a rectangle to locate and size the control. When you release the mouse button, the form control appears on Design.

To modify the form control, right-click the object, then select one of the following options from the context menu:

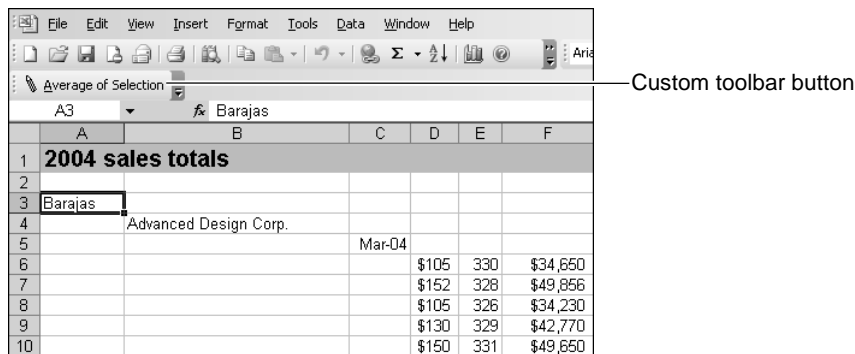
- Select Assign Macro to bind the form control object to a defined macro name.
- Select Format Object to modify object properties in the same way as in Microsoft Excel.

After modifying the form control object, run the report and save the report file as an Excel 97-2003 Workbook (\*.xls) file. Then view the file in Excel.

---

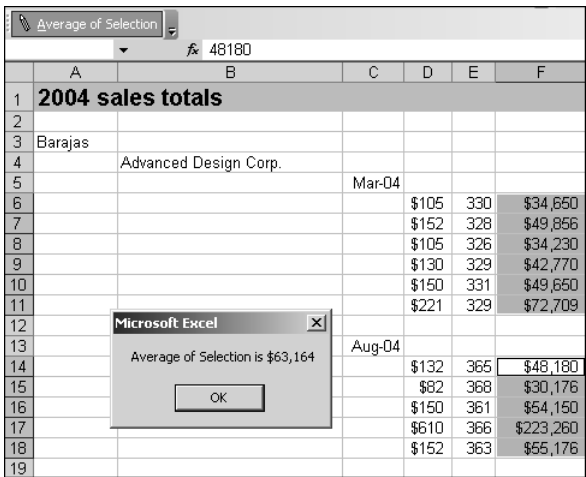
## Reviewing a VBA template example

This section describes how to set up a VBA template to use with a spreadsheet report. In the following procedures, you learn how to create the Excel template file, test the VBA template using spreadsheet data, publish the report with the template, and open the published report. When you complete these steps, the published report looks like the one in Figure 21-1. The button in the upper left corner launches a macro that calculates the average of the selected cells.



**Figure 21-1** Spreadsheet report with custom button

When you select a range of numeric cells, then choose Average of selection, the report displays a window that shows the average of the selected cells, similar to the one in Figure 21-2.



**Figure 21-2** Spreadsheet report with custom calculation

To create the button, you create a custom toolbar feature using Customize—Toolbars in Excel. To implement the average function and display the results, you associate the following macro code with the button:

```
Sub AverageOfSelection()  
    MsgBox "Average of Selection is " + _  
        WorksheetFunction.Text( _  
            WorksheetFunction.Average(ActiveWindow.RangeSelection), _  
            ActiveWindow.ActiveCell.NumberFormatLocal)  
End Sub
```

For more information about creating and using VBA features in Excel, see your Microsoft Excel documentation.

The procedures in this section assume you have created a BIRT Spreadsheet Designer report with which to use the new button. The file is named AverageVBA.sod and is stored in \VBA.

### How to create a VBA template

- 1 In Excel, open a blank workbook.
- 2 Save the file as to the same directory as the BIRT Spreadsheet Designer file with which to use the template. Use a file name with the following syntax:

Filename.sod.xls

where Filename is the name of the BIRT Spreadsheet Designer file.

For this example, save the file to a directory called VBA and use the following file name:

AverageVBA.sod.xls

- 3 Open Customize—Toolbars and create a custom toolbar button.
- 4 Create a macro that implements the average function, then associate that macro with the toolbar button you created.
- 5 Save and close the file.

#### How to test the VBA feature with spreadsheet data



- 1 In BIRT Spreadsheet Designer, open the file with which to use the custom button, then run the report.
- 2 Save the report as an Excel file in the same directory, using the following syntax for the file name:

Filename.xls

where Filename.sod.xls is the name of the Excel template file.

For this example, save the file to the VBA directory using the following file name:

AverageVBA.xls

- 3 If a message appears warning you that the file format does not support all features, choose Yes.
- 4 Close the file in BIRT Spreadsheet Designer.
- 5 In Excel, open the new file, AverageVBA.xls. The file appears, displaying the data from your report and the custom Excel button. For this example, the report looks like the one in Figure 21-3.

	A	B	C	D	E	F
1	<b>2004 sales totals</b>					
2						
3	Barajas					
4	Advanced Design Corp.					
5			Mar-04			
6				\$105	330	\$34,650
7				\$152	328	\$49,856
8				\$105	326	\$34,230
9				\$130	329	\$42,770
10				\$150	331	\$49,650
11				\$221	329	\$72,709

**Figure 21-3** Testing a report with VBA features

- 6 Test the VBA feature, then close the file.

### How to publish a file that uses a VBA template

- 1 In BIRT Spreadsheet Designer, open the file to publish. For this example, open AverageVBA.sod.
- 2 Then publish the file using Actuate BIRT iServer System. When BIRT iServer System uploads the file, it also uploads the VBA features from the Excel template. When a user opens the published file, it displays the VBA features you added.

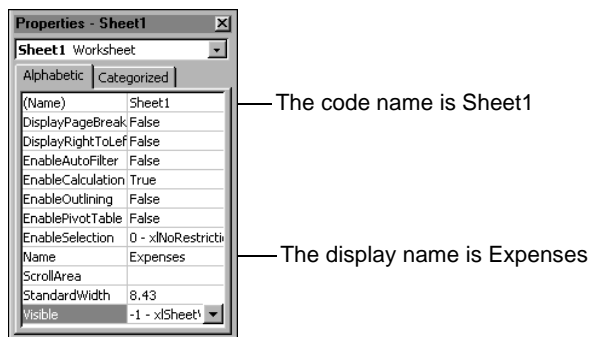
---

## Using workbook and worksheet code names

In VBA code, a workbook or a worksheet often uses a different name from the name the user sees in a file name or on a worksheet tab:

- The name the user sees as the workbook file name or the worksheet name is called the workbook or worksheet display name. The display name appears as Name in Microsoft Visual Basic—Properties.
- The identifier the code uses to specify a workbook or a worksheet is called the workbook or worksheet code name. You use the code name to bind VBA code to a workbook or worksheet. The code name appears as (Name) in Microsoft Visual Basic—Properties.

For example, Figure 21-4 shows Microsoft Visual Basic—Properties for a worksheet. The name the user sees is Expenses. The name the code uses is Sheet1.



**Figure 21-4** Microsoft Visual Basic—Properties

When you first create a BIRT Spreadsheet Designer or Excel workbook or worksheet, the default code name of the workbook or worksheet matches the name, the identifier the user sees. BIRT Spreadsheet Designer and Excel update worksheet code names differently:

- In a file you create and edit only in BIRT Spreadsheet Designer, when you change a worksheet name, BIRT Spreadsheet Designer updates the code name

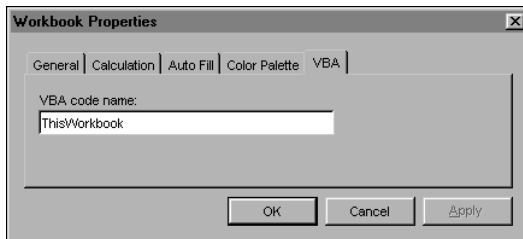
by default. The code name changes to match the new name. You can use Format Sheet—VBA to change this behavior.

- In a file you create or edit in Excel, when you change a worksheet name, Excel does not update the code name. To change the code name, you must use Microsoft Visual Basic—Properties. If you open an Excel file in BIRT Spreadsheet Designer, the Excel default setting persists. To have the code name update automatically in BIRT Spreadsheet Designer, you must use Format Sheet—VBA to change the behavior.

You can use BIRT Spreadsheet Designer to specify the code name of a workbook or worksheet.

### How to change a workbook code name

- 1 Choose Tools→Workbook Properties.
- 2 On Workbook Properties—General, choose VBA to see the setting in Figure 21-5.

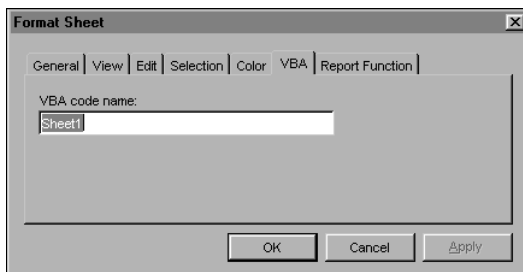


**Figure 21-5** VBA options

- 3 In VBA code name, type the new code name. Choose OK.

### How to change a worksheet code name

- 1 Choose Format→Sheet→Properties.
- 2 On Format Sheet—General, choose VBA to see the options in Figure 21-6.



**Figure 21-6** Code name options

- 3 Type a new name in VBA code name, then choose OK.



# Part Five

---

**Working in multiple locales using BIRT  
Spreadsheet technology**



# Using BIRT Spreadsheet Designer with multiple locales

This chapter contains the following topics:

- Deploying BIRT Spreadsheet Designer in a translated language
- Deploying BIRT Spreadsheet Designer in a supported language
- Using regional settings with BIRT Spreadsheet Designer
- Using international symbols in XML
- Using non-English versions of BIRT Spreadsheet Designer
- Understanding BIRT Spreadsheet Designer limitations

---

# Deploying BIRT Spreadsheet Designer in a translated language

By default, BIRT Spreadsheet Designer deploys with several translated languages:

- Chinese (Simplified)
- English
- French
- German
- Italian
- Japanese
- Korean
- Spanish

Each translated language includes a completely translated graphical user interface (GUI). Number format syntaxes, formula names, and other internal items match Excel’s translations. For more information about using BIRT Spreadsheet Designer in a translated language, see Appendix A, “BIRT Spreadsheet Designer internationalization information.”

During installation, BIRT Spreadsheet Designer writes a JAR file for each supported language in the local directory within the BIRT Spreadsheet Designer installation directory. Files are named based on a language designation as shown in the following table.

## About BIRT Spreadsheet Designer translated languages

Table 22-1 lists the languages BIRT Spreadsheet Designer can use and the translation files necessary to use each language.

**Table 22-1** BIRT Spreadsheet Designer supported languages

Language	File
Chinese (Simplified)	ess11_zhs.jar
English	default
French	ess11_fr.jar
German	ess11_de.jar
Italian	ess11_it.jar

**Table 22-1** BIRT Spreadsheet Designer supported languages

Language	File
Japanese	ess11_ja.jar
Korean	ess11_ko.jar
Spanish	ess11_es.jar

#### How to use a translated version of BIRT Spreadsheet Designer

- 1 Choose Tools➤Preferences.
- 2 Select the Regional tab.
- 3 Select Show Only Full Translations.
- 4 Select one of the translated languages.
- 5 Choose OK.

The new language version takes effect on the next instantiation.

To deploy BIRT Spreadsheet Designer in one of the translated languages, developers must:

- Include the JAR file that corresponds to the deployment language with their application.
- Put the JAR file on the client computer's class path.

You must have the appropriate licensing agreement to deploy in a translated language.

## Using code to override locale settings

By default, when you use the BIRT Spreadsheet API to create a JBook, it instantiates using the client machine's locale settings. Similarly, BIRT Spreadsheet Designer uses the client machine's locale settings. The following code example demonstrates how to override the default.

Developers who want to control the language and locale in which BIRT Spreadsheet Designer is deployed can create a Group object to override the default settings. The Group object determines the language in which the user interface appears and the manner in which numbers are formatted. The constructors for the Group class support specifying a user interface (UI) language and a locale.

This code example overrides the default settings for a newly created JBook by creating a Group object, then passing that Group object to the JBook class constructor.

**Example 1** This code sets the UI language to German, and uses the user's default locale for formatting:

```
com.flj.util.Group group = new
    com.flj.util.Group(java.util.Locale.GERMAN);
com.flj.swing.JBook jbook = new com.flj.swing.JBook(null, group);
```

**Example 2** This code sets the UI language to French and the locale to Canada:

```
com.flj.util.Group group
    = new com.flj.util.Group(java.util.Locale.FRENCH,
        java.util.Locale.CANADA);
com.flj.swing.JBook jbook = new com.flj.swing.JBook(null, group);
```

**Example 3** This code sets the UI language to Spanish and the locale to United States:

```
com.flj.util.Group group = new
    com.flj.util.Group(java.util.Locale.SPANISH,
        java.util.Locale.US);
com.flj.swing.JBook jbook = new com.flj.swing.JBook(null, group);
```

---

## Deploying BIRT Spreadsheet Designer in a supported language

Developers can deploy BIRT Spreadsheet Designer's supported language versions within an application written in their home language by including the appropriate JAR file in the class path on the machine where the application is installed. Supported versions do not include a translated GUI, but do include translations of cell errors, number format syntax, formula syntax, and other non-GUI items that are compatible with the Excel translations. Error messages are in English. Cell error messages match the corresponding Excel translation.

Supported language files, stored in the BIRTSpreadsheetDesigner/local directory, are named based on the language designations (with some modifications) from the ISO 639-2 standard. Table 22-2 lists supported languages and their corresponding JAR files.

**Table 22-2** Supported languages and JAR files

Language	File	Language	File
Chinese (Traditional)	ess11_zht.jar	Polish	ess11_pl.jar
Croatian	ess11_hr.jar	Portuguese (Brazil)	ess11_ptb.jar
Czech	ess11_cs.jar	Portuguese (Portugal)	ess11_pti.jar
Danish	ess11_da.jar	Romanian	ess11_ro.jar
Dutch	ess11_nl.jar	Russian	ess11_ru.jar

**Table 22-2** Supported languages and JAR files

Language	File	Language	File
Finnish	ess11_fi.jar	Slovak	ess11_sk.jar
Greek	ess11_el.jar	Slovenian	ess11_sl.jar
Hungarian	ess11_hu.jar	Swedish	ess11_sv.jar
Norwegian	ess11_no.jar	Turkish	ess11_tr.jar

To deploy BIRT Spreadsheet Designer in a supported language, developers must:

- Include the JAR file that corresponds to the supported language with their application.
- Put the JAR file on the client computer's class path.

You must have the appropriate licensing agreement to deploy in a supported language.

---

## Using regional settings with BIRT Spreadsheet Designer

As it loads into memory, BIRT Spreadsheet Designer picks up regional settings from the system it is loading on. If regional settings are changed after BIRT Spreadsheet Designer is loaded, they will not take effect in the spreadsheet.

The Java Virtual Machine (JVM) that BIRT Spreadsheet Designer runs under also limits the regional settings. The Java 2 VM recognizes the operating system's country and language settings and adjusts date, time, number, and currency formatting accordingly. Current versions of Sun's JVM, however, ignore the customization of these settings that is allowed in some operating systems such as Microsoft Windows.

---

## Using international symbols in XML

BIRT Spreadsheet Designer specifies value formats in XML using English formats and English special characters. When BIRT Spreadsheet Designer processes the XML, the XML parser may interpret some characters intended as international symbols as English characters instead, which can produce unexpected output. When this occurs, use English formats and symbols to indicate the current locale's corresponding format and number symbols.

The following examples illustrate how to use the Euro symbol in a German locale.

## Viewing localized currency

When viewed in the German locale, the correct example below displays 1.234,56 € in the worksheet cell.

Correct:

#,##0.00 [\$€]

Incorrect:

#.##0,00 €

## Viewing localized dates

When viewed in the German locale, the correct examples below display a date in a two-digit, date/month/year format in the worksheet cell.

Correct, using localized date separator:

dd/mm/yy

BIRT Spreadsheet Designer replaces the slash symbol (/) with local data separator.

Correct, using period as date separator:

dd\ .mm\ .yy

Incorrect:

TT.MM.JJ

---

## Using non-English versions of BIRT Spreadsheet Designer

When you run BIRT Spreadsheet Designer for the first time, the language BIRT Spreadsheet Designer uses is based on your system locale. Changing the system locale does not change the language. To change the language, use the Regional tab of the Tools➤Preferences dialog.

---

## Understanding BIRT Spreadsheet Designer limitations

BIRT Spreadsheet Designer has the following limitations:

- Multilingual data is not always displayed correctly in dialogs.  
On Actuate BIRT iServer System, however, Unicode data is extracted correctly from the database. To display the data correctly in Excel, design the report



with the appropriate fonts for the language. The fonts must be installed on the report user's computer.

- The input method editor (IME) appears in a pop-up window.  
In BIRT Spreadsheet Designer, if you press F2 or click the Edit bar before typing, the IME appears in place. If, however, you start typing without pressing F2 first, the IME appears in a pop-up window.
- The worksheet function PHONETIC is not supported.  
The Japanese language function PHONETIC is not supported.
- The worksheet function BAHTTEXT is not supported.  
The Thai language function BAHTTEXT is not supported.
- Number format symbols are not fully supported:
  - DBNum*n*  
For Chinese, Japanese, and Korean, DBNum*n*, where *n* specifies a number shape, is not supported.
  - [\$-nnccllll]  
[\$-nnccllll] is not supported in BIRT Spreadsheet Designer. It is supported in Excel, however. You can use this format style and save the file in Excel format. Additional format symbols for Arabic and Thai are not supported.
    - nn specifies the number shape.
    - cc specifies the calendar.
    - llll specifies the parsing of the number format.





# **BIRT Spreadsheet Designer internationalization information**

This appendix contains tables listing the following information:

- About internationalization information
- Using the Japanese translation

## About internationalization information

This appendix contains Table A-1, Table A-2, and Table A-3, which list information pertinent to BIRT Spreadsheet Designer's internationalization features.

### Format symbols

**Table A-1** International format symbols

Symbol Description	English	French	German	Italian	Spanish	Japanese
Escape Char	\	\	\	\	\	!
Year	y	a	J	a	a	y
Month	m	m	M	m	m	m
Day	d	j	T	g	d	d
Hour	h	h	h	h	h	h
Minute	m	m	m	m	m	m
Second	s	s	s	s	s	s
General	General	Standard	Standard	Standard	Estándar	G/General
Black	Black	Noir	Schwarz	Nero	Negro	黒
Blue	Blue	Bleu	Blau	Blu	Azul	青
Cyan	Cyan	Cyan	Zyan	Celeste	Agua-marina	水
Green	Green	Vert	Grün	Verde	verde	緑
Magenta	Magenta	Magenta	Magenta	Fucsia	Fucsia	紫
Red	Red	Rouge	Rot	Rosso	Rojo	赤
White	White	Blanc	Weiss	Bianco	Blanco	白
Yellow	Yellow	Jaune	Gelb	Giallo	Amarillo	黄色
Color	Color	Couleur	Farbe	Colore	Color	色

## Values and errors

**Table A-2** International values and errors

English/ Japanese	French	German	Italian	Spanish
TRUE	VRAI	WAHR	VERO	VERDADERO
FALSE	FAUX	FALSCH	FALSO	FALSO
#NULL!	#NUL!	#NULL!	#NULO!	#¡NULO!
#DIV/0!	#DIV/0!	#DIV/0!	#DIV/0!	#¡DIV/0!
#VALUE!	#VALEUR!	#WERT!	#VALORE!	#¡VALOR!
#REF!	#REF!	#BEZUG!	#RIF!	#¡REF!
#NAME?	#NOM?	#NAME?	#NOME?	#¿NOMBRE?
#NUM!	#NOMBRE!	#ZAHL!	#NUM!	#¡NUM!
#N/A	#N/A	#NV	#N/D	#N/A

## Function names

**Table A-3** International function names

English/ Japanese	French	German	Italian	Spanish
ABS	ABS	ABS	ASS	ABS
ACCRINT	INTERET.ACC	AUFGELZINS	INT.MATURATO. PER	INT.ACUM
ACCRINTM	INTERET.ACC. MAT	AUFGELZINSF	INT.MATURATO. SCAD	INT.ACUM.V
ACOS	ACOS	ARCCOS	ARCCOS	ACOS
ACOSH	ACOSH	ARCCOSHYP	ARCCOSH	ACOSH
ADDRESS	ADRESSE	ADRESSE	INDIRIZZO	DIRECCION
AMORDEGRC	AMORDEGRC	AMORDEGRK	AMMORT.DEGR	AMORTIZ PROGRE
AMORLINC	AMORLINC	AMOR LINEARK	AMMORT.PER	AMORTIZLIN
AND	ET	UND	E	Y
AREAS	ZONES	BEREICHE	AREE	AREAS
ASC	ASC	ASC	ASC	ASC

(continues)

**Table A-3** International function names (continued)

English/ Japanese	French	German	Italian	Spanish
ASIN	ASIN	ARCSIN	ARCSEN	ASENO
ASINH	ASINH	ARCSINHYP	ARCSENH	ASENOH
ATAN	ATAN	ARCTAN	ARCTAN	ATAN
ATAN2	ATAN2	ARCTAN2	ARCTAN.2	ATAN2
ATANH	ATANH	ARCTANHYP	ARCTANH	ATANH
AVEDEV	ECART.MOYEN	MITTELABW	MEDIA.DEV	DESVPROM
AVERAGE	MOYENNE	MITTELWERT	MEDIA	PROMEDIO
AVERAGEA	AVERAGEA	MITTELWERTA	MEDIA.VALORI	PROMEDIOA
BESSELI	BESSELI	BESSELI	BESSEL.I	BESSELI
BESSELJ	BESSELJ	BESSELJ	BESSEL.J	BESSELJ
BESSELK	BESSELK	BESSELK	BESSEL.K	BESSELK
BESSELY	BESSELY	BESSELY	BESSEL.Y	BESSELY
BETADIST	LOI.BETA	BETAVERT	DISTRIB.BETA	DISTR.BETA
BETAINV	BETA.INVERSE	BETAINV	INV.BETA	DISTR.BETA. INV
BIN2DEC	BINDEC	BININDEZ	BINARIO. DECIMALE	BIN.A.DEC
BIN2HEX	BINHEX	BININHEX	BINARIO.HEX	BIN.A.HEX
BIN2OCT	BINOCT	BININOKT	BINARIO.OCT	BIN.A.OCT
BINOMDIST	LOI. BINOMIALE	BINOMVERT	DISTRIB.BINOM	DISTR.BINOM
CALL	FONCTION. APPELANTE	AUFRUFEN	RICHIAMA	LLAMAR
CEILING	PLAFOND	OBERGRENZE	ARROTOND. ECESSO	MULTIPLO. SUPERIOR
CELL	CELLULE	ZELLE	CELLA	CELDA
CHAR	CAR	ZEICHEN	CODICE. CARATT	CARACTER
CHIDIST	LOI.KHIDEUX	CHIVERT	DISTRIB.CHI	DISTR.CHI
CHIINV	KHIDEUX. INVERSE	CHIINV	INV.CHI	PRUEBA.CHI. INV
CHITEST	TEST.KHIDEUX	CHITEST	TEST.CHI	PRUEBA.CHI
CHOOSE	CHOISIR	WAHL	SCEGLI	ELEGIR

**Table A-3** International function names (continued)

English/ Japanese	French	German	Italian	Spanish
CLEAN	EPURAGE	SÄUBERN	LIBERA	LIMPIAR
CODE	CODE	CODE	CODICE	CODIGO
COLUMN	COLONNE	SPALTE	RIF.COLONNA	COLUMNA
COLUMNS	COLONNES	SPALTEN	COLONNE	COLUMNAS
COMBIN	COMBIN	KOMBINATIONEN	COMBINAZIONE	COMBINAT
COMPLEX	COMPLEXE	KOMPLEXE	COMPLESSO	COMPLEJO
CONCATENATE	CONCATENER	VERKETTEN	CONCATENA	CONCATENAR
CONFIDENCE	INTERVALLE. CONFIANCE	KONFIDENZ	CONFIDENZA	INTERVALO. CONFIANZA
CONVERT	CONVERT	UMWANDELN	CONVERTI	COMNVERTIR
CORREL	COEFFICIENT. CORRELATION	KORREL	CORRELAZIONE	COEF.DE. CORREL
COS	COS	COS	COS	COS
COSH	COSH	COSHYP	COSH	COSH
COUNT	NB	ANZAHL	CONTA. NUMERI	CONTAR
COUNTA	NBVAL	ANZAHL2	CONTA. VALORI	CONTARA
COUNT BLANK	NB.VIDE	ANZAHL LEEREZELLEN	CONTA.VUOTE	CONTAR. BLANCO
COUNTIF	NB.SI	ZÄHLEN WENN	CONTA.SE	CONTAR.SI
COUPDAYBS	NB.JOURS. COUPON. PREC	ZINSTERM TAGVA	GIORNI.CED. INIZ.LIQ	CUPON.DIAS. L1
COUPDAYS	NB.JOURS. COUPONS	ZINSTERM TAGE	GIORNI.CED	CUPON.DIAS
COUPDAYS NC	NB.JOURS. COUPON.SUIV	ZINSTERM TAGNZ	GIORNI.CED. NUOVA	CUPON.DIAS. L2
COUPNCD	DATE. COUPON.SUIV	ZINSTERMNZ	DATA.CED. SUCC	CUPON.FECHA. L2

(continues)

**Table A-3** International function names (continued)

English/ Japanese	French	German	Italian	Spanish
COUPNUM	NB.COUPONS	ZINSTERM ZAHL	NUM.CED	CUPON.NUM
COUPPCD	DATE. COUPON.PREC	ZINSTERMVZ	DATA.CED. PREC	CUPON.FECHA. L1
COVAR	COVARIANCE	KOVAR	COVARIANZA	COVAR
CRITBINOM	CRITERE.LOI. BINOMIALE	KRITBINOM	CRIT.BINOM	BINOM.CRIT
CUMIPMT	CUMUL.INTER	ZUMZINSZ	INT.CUMUL	PAGO.INT. ENTRE
CUMPRINC	CUMUL. PRINCPER	ZUMKAPITAL	CAP.CUM	PAGO.PRINC. ENTRE
DATE	DATE	DATUM	DATA	FECHA
DATEDIF	DATEDIF	DATEDIF	DATA.DIFF	SIFECHA
DATEVALUE	DATEVAL	DATWERT	DATA.VALORE	FECHA NUMERO
DAVERAGE	BDMOYENNE	DBMITTEL WERT	DB.MEDIA	BDPROMEDIO
DAY	JOUR	TAG	GIORNO	DIA
DAYS360	JOURS360	TAGE360	GIORNO360	DIAS360
DB	DB	GDA2	AMMORT.FISSO	DB
DBCS	DBCS	DBCS	DBCS	DBCS
DCOUNT	BDNB	DBANZAHL	DB.CONTA. NUMERI	BDCONTAR
DCOUNTA	BDNBVAL	DBANZAHL2	DB.CONTA. VALORI	BDCONTARA
DDB	DDB	GDA	AMMORT	DDB
DEC2BIN	DECBIN	DEZINBIN	DECIMALE. BINARIO	DEC.A.BIN
DEC2HEX	DECHEX	DEZINHEX	DECIMALE.HEX	DEC.A.HEX
DEC2OCT	DECOCT	DEZINOKT	DECIMALE.OCT	DEC.A.OCT
DEGREES	DEGRES	GRAD	GRADI	GRADOS
DELTA	DELTA	DELTA	DELTA	DELTA



**Table A-3** International function names (continued)

English/ Japanese	French	German	Italian	Spanish
DEVSQ	SOMME. CARRES. ECARTS	SUMQUAD ABW	DEV.Q	DESVIA2
DGET	BDLIRE	DBAUSZUG	DB.VALORI	BDEXTRAER
DISC	TAUX. ESCOMPTE	DISAGIO	TASSO.SCONTO	TASA.DESC
DMAX	BDMAX	DBMAX	DB.MAX	BDMAX
DMIN	BDMIN	DBMIN	DB.MIN	BDMIN
DOLLAR	FRANC	DM	VALUTA	MONEDA
DOLLARDE	PRIX.DEC	NOTIERUNG DEZ	VALUTA.DEC	MONEDA.DEC
DOLLARFR	PRIX.FRAC	NOTIERUNG BRU	VALUTA.FRAZ	MONEDA. FRAC
DPRODUCT	BDPRODUIT	DBPRODUKT	DB.PRODOTTO	BDPRODUCTO
DSTDEV	BDECARTYPE	DBSTDABW	DB.DEV.ST	BDDVEST
DSTDEVP	BDECARTYPEP	DBSTDABWN	DB.DEV.ST.POP	BDDVESTP
DSUM	BDSOMME	DBSUMME	DB.SOMMA	BDSUMA
DURATION	DUREE	DURATION	DURATA	DURACION
DVAR	BDVAR	DBVARIANZ	DB.VAR	BDVAR
DVARP	BDVARP	DBVARIANZE N	DB.VAR.POP	BDVARP
EDATE	MOIS. DECALER	EDATUM	DATA.MESE	FECHA.MES
EFFECT	TAUX. EFFECTIF	EFFEKTIV	EFFETTIVO	INT.EFFECTIVO
EOMONTH	FIN.MOIS	MONATSENDE	FINE.MESE	FIN.MES
ERF	ERF	GAUSSFEHLER	FUNZ.ERROR	FUN.ERROR. COMPL
ERFC	ERFC	GAUSS FKOMPL	FUNZ.ERROR. COMP	FUN.ERROR. COMPL
ERROR.TYPE	TYPE.ERREUR	FEHLER.TYP	ERRORE.TIPO	TIPO.DE. ERROR

(continues)

**Table A-3** International function names (continued)

English/ Japanese	French	German	Italian	Spanish
EURO CONVERT	EURO CONVERT	EURO CONVERT	EURO CONVERT	EURO CONVERT
EVEN	PAIR	GERADE	PARI	REDONDEA. PAR
EXACT	EXACT	IDENTISCH	IDENTICO	IGUAL
EXP	EXP	EXP	EXP	EXP
EXPONDIST	LOI.EXPONEN TIELLE	EXPONVERT	DISTRIB.EXP	DISTR.EXP
FACT	FACT	FAKULTÄT	FATTORIALE	FACT
FACTDOUBLE	FACTDOUBLE	ZWEI FAKULTÄT	FATT.DOPPIO	FACT.DOUBLE
FALSE	FAUX	FALSCH	FALSO	FALSO
FDIST	LOI.F	FVERT	DISTRIB.F	DISTR.F
FIND	TROUVE	FINDEN	TROVA	ENCONTRAR
FINDB	TROUVERB	FINDENB	TROVA.B	ENCONTRARB
FINV	INVERSE.LOI.F	FINV	INV.F	DISTR.F.INV
FISHER	FISHER	FISHER	FISHER	FISHER
FISHERINV	FISHER. INVERSE	FISHERINV	INV.FISHER	PRUEBA. FISHER.INV
FIXED	CTXT	FEST	FISSO	DECIMAL
FLOOR	PLANCHER	UNTER GRENZE	ARROTONDA. DIFETTO	MULTIPLO. INFERIOR
FORECAST	PREVISION	SCHÄTZER	PREVISIONE	PRONOSTICO
FREQUENCY	FREQUENCE	HÄUFIGKEIT	FREQUENZA	FRECUENCIA
FTEST	TEST.F	FTEST	TEST.F	PRUEBA.F
FV	VC	ZW	VAL.FUT	VF
FVSCHEDULE	VC. PAIEMENTS	ZW2	VAL.FUT. CAPITALE	VF.PLAN
GAMMADIST	LOI.GAMMA	GAMMAVERT	DISTRIB. GAMMA	DISTR.GAMMA
GAMMAINV	LOI.GAMMA. INVERSE	GAMMAINV	INV.GAMMA	DISTR.GAMMA. INV
GAMMALN	LNGAMMA	GAMMALN	LN.GAMMA	GAMMA.LN

**Table A-3** International function names (continued)

English/ Japanese	French	German	Italian	Spanish
GCD	PGCD	GGT	MCD	M.C.D
GEOMEAN	MOYENNE. GEO METRIQUE	GEOMITTEL	MEDIA. GEOMETRICA	MEDIA.GEOM
GESTEP	SUP.SEUIL	GGANZZAHL	SOGLIA	MAYOR.O. IGUAL
GETPIVOT DATA	LIREDONNEES TABCROIS DYNAMIQUE	PIVOTDATEN ZUORDNEN	INFO.DATI.TAB. PIVOT	IMPORTAR DATOS DINAMICOS
GROWTH	CROISSANCE	VARIATION	CRESCITA	CRECIMIENTO
HARMEAN	MOYENNE. HARMONIQUE	HARMITTEL	MEDIA. ARMONICA	MEDIA.ARMO
HEX2BIN	HEXBIN	HEXINBIN	HEX.BINARIO	HEX.A.BIN
HEX2DEC	HEXDEC	HEXINDEZ	HEX.DECIMALE	HEX.A.DEC
HEX2OCT	HEXOCT	HEXINOKT	HEX.OCT	KHEX.A.OCT
HLOOKUP	RECHERCHEH	WVERWEIS	CERCA.ORIZZ	BUSCARH
HOURL	HEURE	STUNDE	ORA	HORA
HYPERLINK	LIEN_ HYPERTEXTE	HYPERLINK	COLLEG. IPERTESTUALE	HIPER VINCULO
HYPGEOM DIST	LOI.HYPER GEO METRIQUE	HYP GEOMVERT	DISTRIB. IPERGEOM	DISTR. HIPERGEOM
IF	SI	WENN	SE	SI
IMABS	COMPLEXE. MODULE	IMABS	COMP.MODULO	IM.ABS
IMAGINARY	COMPLEXE. IMAGINAIRE	IMAGIN ÄRTEIL	COMP. IMMAGINARIO	IMAGINARIO
IM ARGUMENT	COMPLEXE. ARGUMENT	IMARGUMENT	COMP. ARGOMENTO	IM.ANGULO
IM CONJUGATE	COMPLEXE. CONJUGUE	IM KONJUGIERTE	COMP. CONIUGATO	IM. CONJUGADA
IMCOS	COMPLEXE. COS	IMCOS	COMP.COS	IM.COS

(continues)

**Table A-3** International function names (continued)

English/ Japanese	French	German	Italian	Spanish
IMDIV	COMPLEXE. DIV	IMDIV	COMP.DIV	IM.DIV
IMEXP	COMPLEXE. EXP	IMEXP	COMP.EXP	IM.EXP
IMLN	COMPLEXE.LN	IMLN	COMP.LN	IM.LN
IMLOG10	COMPLEXE. LOG10	IMLOG10	COMP.LOG10	IM.LOG10
IMLOG2	COMPLEXE. LOG2	IMLOG2	COMP.LOG2	IM.LOG2
IMPOWER	COMPLEXE. PUISSANCE	IMAPOTENZ	COMP.POTENZA	IM.POT
IMPRODUCT	COMPLEXE. PRODUIT	IMPRODUKT	COMP. PRODOTTO	IM.PRODUCT
IMREAL	COMPLEXE. REEL	IMREALTEIL	COMP.PARTE. REALE	IM.REAL
IMSIN	COMPLEXE. SIN	IMSIN	COMP.SEN	IM.SENO
IMSQRT	COMPLEXE. RACINE	IMWURZEL	COMP.RADQ	IM.RAIZ2
IMSUB	COMPLEXE. DIFFERENCE	IMSUB	COMP.DIFF	IM.SUSTR
IMSUM	COMPLEXE. SOMME	IMSUMME	COMP.SOMMA	IM.SUM
INDEX	INDEX	INDEX	INDICE	INDICE
INDIRECT	INDIRECT	INDIREKT	INDIRETTO	INDIRECTO
INFO	INFO	INFO	AMBIENTE.INFO	INFO
INT	ENT	GANZZAHL	INT	ENTERO
INTERCEPT	ORDONNEE. ORIGINE	ACHSEN ABSCHNITT	INTERCETTA	INTERSECCION .EJE
INTRATE	TAUX.INTERET	IMZINSSATZ	TASSO.INT	TASA.INT
IPMT	INTPER	ZINSZ	INTERESSI	PAGOINT
IRR	TRI	IKV	TIR.COST	TIR
ISBLANK	ESTVIDE	ISTLEER	VAL.VUOTO	ESBLANCO
ISERR	ESTERR	ISTFEHL	VAL.ERR	ESERR

**Table A-3** International function names (continued)

English/ Japanese	French	German	Italian	Spanish
ISERROR	ESTERREUR	ISTFEHLER	VAL.ERRORE	ESERROR
ISEVEN	EST.PAIR	ISTGERADE	VAL.PARI	ES.PAR
ISLOGICAL	ESTLOGIQUE	ISTLOG	VAL.LOGICO	ESLOGICO
ISNA	ESTNA	ISTNV	VAL.NON.DISP	ESNOD
ISNONTEXT	ESTNONTEXTE	ISTKTEXT	VAL.NON.TESTO	ESNOTEXTO
ISNUMBER	ESTNUM	ISTZAHL	VAL.NUMERO	ESNUMERO
ISODD	EST.IMPAIR	ISTUNGERADE	VAL.DISPARI	ES.IMPAR
ISREF	ESTREF	ISTBEZUG	VAL.RIF	ESREF
ISTEXT	ESTTEXTE	ISTTEXT	VAL.TESTO	ESTEXTO
KURT	KURTOSIS	KURT	CURTOSI	CURTOSIS
LARGE	GRANDE. VALEUR	KGRÖSSTE	GRANDE	K.ESIMO. MAYOR
LCM	PPCM	KGV	MCM	M.C.M
LEFT	GAUCHE	LINKS	SINISTRA	IZQUIERDA
LEFTB	GAUCHEB	LINKSB	SINISTRAB	IZQUIERDAB
LEN	NBCAR	LÄNGE	LUNGHEZZA	LARGO
LENB	LENB	LÄNGEB	LUNGB	LARGOB
LINEST	DROITEREG	RGP	REGR.LIN	ESTIMACION. LINEAL
LN	LN	LN	LN	LN
LOG	LOG	LOG	LOG	LOG
LOG10	LOG10	LOG10	LOG10	LOG10
LOGEST	LOGREG	RKP	REGR.LOG	ESTIMACION. LOGARITMICA
LOGINV	LOI.LOGNOR MALE.INVERS E	LOGINV	INV.LOGNORM	DISTR.LOG.INV
LOGNORM DIST	LOI.LOG NORMALE	LOGNORM VERT	DISTRIB.LOG NORM	DISTR.LOG. NORM
LOOKUP	RECHERCHE	VERWEIS	CERCA	BUSCAR
LOWER	MINUSCULE	KLEIN	MINUSC	MINUSC

*(continues)*

**Table A-3** International function names (continued)

English/ Japanese	French	German	Italian	Spanish
MATCH	EQUIV	VERGLEICH	CONFRONTA	COINCIDIR
MAX	MAX	MAX	MAX	MAX
MAXA	MAXA	MAXA	MAX.VALORI	MAXA
MDETERM	DETERMAT	MDET	MATR.DETERM	MDETERM
MDURATION	DUREE. MODIFIEE	MDURATION	DURATA.M	DURACION. MODIF
MEDIAN	MEDIANE	MEDIAN	MEDIANA	MEDIANA
MID	STXT	TEIL	STRINGA. ESTRAI	EXTRAIE
MIDB	MIDB	TEILB	MEDIA.B	EXTRAEB
MIN	MIN	MIN	MIN	MIN
MINA	MINA	MINA	MIN.VALORI	MINA
MINUTE	MINUTE	MINUTE	MINUTO	MINUTO
MINVERSE	INVERSEMAT	MINV	MATR.INVERSA	MINVERSA
MIRR	TRIM	QIKV	TIR.VAR	TIRM
MMULT	PRODUITMAT	MMULT	MATR. PRODOTTO	MMULT
MOD	MOD	REST	RESTO	RESIDUO
MODE	MODE	MODALWERT	MODA	MODA
MONTH	MOIS	MONAT	MESE	MES
MROUND	ARRONDI.AU. MULTIPLE	VRUNDEN	ARROTONDA. MULTIPLO	REDOND. MULT
MULTI NOMIAL	MULTI NOMIALE	POLYNOMIAL	MULTI NOMIALE	MULTINOMIAL
N	N	N	NUM	N
NA	NA	NV	NON.DISP	NOD
NEGBINOM DIST	LOI. BINOMIALE. NEG	NEGBINOM VERT	DISTRIB.BINOM. NEG	NEGBINOM DIST
NETWORK DAYS	NB.JOURS. OUVRES	NETTO ARBEITSTAGE	GIORNI. LAVORATIVI. TOT	DIAS.LAB

**Table A-3** International function names (continued)

English/ Japanese	French	German	Italian	Spanish
NOMINAL	TAUX. NOMINAL	NOMINAL	NOMINALE	TASA. NOMINAL
NORMDIST	LOI.NORMALE	NORMVERT	DISTRIB.NORM	DISTR.NORM
NORMINV	LOI.NORMALE .INVERSE	NORMINV	INV.NORM	DISTR.NORM. INV
NORMSDIST	LOI.NORMALE .STANDARD	STANDNORM VERT	DISTRIB.NORM. ST	DISTR.NORM. ESTAND
NORMSINV	LOI.NORMALE .STANDARD. INVERSE	STANDNORM INV	INV.NORM.ST	DISTR.NORM. ESTAND.INV
NOT	NON	NICHT	NON	NO
NOW	MAINTENANT	JETZT	ADESSO	AHORA
NPER	NPM	ZZR	NUM.RATE	NPER
NPV	VAN	NBW	VAN	VNA
OCT2BIN	OCTBIN	OKTINBIN	OCT.BINARIO	OCT.A.BIN
OCT2DEC	OCTDEC	OKTINDEZ	OCT.DECIMALE	OCT.A.DEC
OCT2HEX	OCTHEX	OKTINHEX	OCT.HEX	OCT.A.HEX
ODD	IMPAIR	UNGERADE	DISPARI	REDONDEA. IMPAR
ODDFPRICE	PRIX. PCOUPON. IRREG	UNREGER. KURS	PREZZO.PRIMO. IRR	PRECIO.PER. IRREGULAR.1
ODDFYIELD	REND. PCOUPON. IRREG	UNREGER. REND	REND.PRIMO. IRR	RENDTO.PER. IRREGULAR.1
ODDLPRICE	PRIX. DCOUPON. IRREG	UNREGLE. KURS	PREZZO. ULTIMO.IRR	PRECIO.PER. IRREGULAR.2
ODDLYIELD	REND. DCOUPON. IRREG	UNREGLE. REND	REND. ULTIMO.IRR	RENDTO.PER. IRREGULAR.2
OFFSET	DECALER	BEREICH. VERSCHIEBEN	SCARTO	DESREF
OR	OU	ODER	O	O

(continues)

**Table A-3** International function names (continued)

English/ Japanese	French	German	Italian	Spanish
PEARSON	PEARSON	PEARSON	PEARSON	PEARSON
PERCENTILE	CENTILE	QUANTIL	PERCENTILE	PERCENTIL
PERCENT RANK	RANG. POUR CENTAGE	QUANTILS RANG	PERCENT. RANGO	RANGO. PERCENTIL
PERMUT	PERMUTATIO N	VARIATIONEN	PERMUTA ZIONE	PERMUTA CIONES
PI	PI	PI	PI.GRECO	PI
PMT	VPM	RMZ	RATA	PAGO
POISSON	LOI.POISSON	POISSON	POISSON	POISSON
POWER	PUISSANCE	POTENZ	POTENZA	POTENCIA
PPMT	PRINCIPER	KAPZ	P.RATA	PAGOPRIN
PRICE	PRIX.TITRE	KURS	PREZZO	PRECIO
PRICEDISC	VALEUR. ENCAISSE MENT	KURSDISAGIO	PREZZO.SCONT	PRECIO. DESCUENTO
PRICEMAT	PRIX.TITRE. ECHEANCE	KURSFÄLLIG	PREZZO.SCAD	PRECIO. VENCIMIENTO
PROB	PROBABILITE	WAHRSC BEREICH	PROBABILITÀ	PROBABILI DAD
PRODUCT	PRODUIT	PRODUKT	PRODOTTO	PRODUCTO
PROPER	NOMPROPRE	GROSS2	MAIUSC.INIZ	NOMPROPIO
PV	VA	BW	VA	VA
QUARTILE	QUARTILE	QUARTILE	QUARTILE	CUARTIL
QUOTIENT	QUOTIENT	QUOTIENT	QUOZIENTE	COCIENTE
RADIANS	RADIANS	BOGENMASS	RADIANI	RADIANES
RAND	ALEA	ZUFALLSZAHL	CASUALE	ALEATORIO
RAND BETWEEN	ALEA.ENTRE. BORNES	ZUFALLS BEREICH	CASUALE.TRA	ALEATORIO. ENTRE
RANK	RANG	RANG	RANGO	JERARQUIA
RATE	TAUX	ZINS	TASSO	TASA
RECEIVED	VALEUR. NOMINALE	AUSZAHLUNG	RICEV.SCAD	CANTIDAD. RECIBIDA



**Table A-3** International function names (continued)

English/ Japanese	French	German	Italian	Spanish
REGISTER.ID	REGISTRE. NUMERO	REGISTER. KENNUMMER	IDENTIFICA TORE.REGISTRO	ID.REGISTRO
REPLACE	REPLACER	ERSETZEN	RIMPIAZZA	REEMPLAZAR
REPLACEB	REPLACERB	ERSETZENB	SOSTITUISCI.B	REEMPLAZARB
REPT	REPT	WIEDER HOLEN	RIPETI	REPETIR
RIGHT	DROITE	RECHTS	DESTRA	DERECHA
RIGHTB	DROITEB	RECHTSB	DESTRA.B	DERECHAB
ROMAN	ROMAIN	RÖMISCH	ROMANO	NUMERO. ROMANO
ROUND	ARRONDI	RUNDEN	ARROTONDA	REDONDEAR
ROUND DOWN	ARRONDI.INF	ABRUNDEN	ARROTONDA. PER.DIF	REDONDEAR. MENOS
ROUNDUP	ARRONDI.SUP	AUFRUNDEN	ARROTONDA. PER.ECC	REDONDEAR. MAS
ROW	LIGNE	ZEILE	RIF.RIGA	FILA
ROWS	LIGNES	ZEILEN	RIGHE	FILAS
RSQ	COEFFICIENT. DETERMINA TION	BESTIMMT HEITSMAS S	RQ	COEFICIENTE. R2
SEARCH	CHERCHE	SUCHEN	RICERCA	HALLAR
SEARCHB	CHERCHERB	SUCHENB	CERCA.B	HALLARB
SECOND	SECONDE	SEKUNDE	SECONDO	SEGUNDO
SERIESSUM	SOMME.SERIES	POTENZREIHE	SOMMA.SERIE	SUMA.SERIES
SIGN	SIGNE	VORZEICHEN	SEGNO	SIGNO
SIN	SIN	SIN	SEN	SENO
SINH	SINH	SINHYP	SENH	SENOH
SKEW	COEFFICIENT. ASYMETRIE	SCHIEFE	ASIMMETRIA	COEFICIENTE. ASIMETRIA
SLN	AMORLIN	LIA	AMMORT.COST	SLN
SLOPE	PENTE	STEIGUNG	PENDENZA	PENDIENTE

(continues)

**Table A-3** International function names (continued)

English/ Japanese	French	German	Italian	Spanish
SMALL	PETITE. VALEUR	KKLEINSTE	PICCOLO	K.ESIMO. MENOR
SQL.REQUEST	SQL.REQUEST	SQL.REQUEST	SQL.REQUEST	SQL.REQUEST
SQRT	RACINE	WURZEL	RADQ	RAIZ
SQRTPI	RACINE.PI	WURZELPI	RADQ.PI .GRECO	RAIZ2PI
STANDARD IZE	CENTREE. REDUITE	STANDARDI SIERUNG	NORMALIZZA	NORMALIZA CION
STDEV	ECARTYPE	STABW	DEV.ST	DESVEST
STDEVA	STDEVA	STABWA	DEV.ST.VALORI	DESVESTA
STDEVP	ECARTYPEP	STABWN	DEV.ST.POP	DESVESTP
STDEVPA	STDEVPA	STABWNA	DEV.ST.POP. VALORI	DESVESTPA
STEYX	ERREUR.TYPE. XY	STFEHLERYX	ERR.STD.YX	ERROR.TIPICO. XY
SUBSTITUTE	SUBSTITUE	WECHSELN	SOSTITUISCI	SUSTITUIR
SUBTOTAL	SOUS.TOTAL	TEILERGEBNIS	SUBTOTALE	SUBTOTALES
SUM	SOMME	SUMME	SOMMA	SUMA
SUMIF	SOMME.SI	SUMMEWENN	SOMMA.SE	SUMAR.SI
SUM PRODUCT	SOMMEPROD	SUMMEN PRODUKT	MATR.SOMMA. PRODOTTO	SUMA PRODUCTO
SUMSQ	SOMME. CARRES	QUADRATE SUMME	SOMMA.Q	SUMA. CUADRADOS
SUMX2MY2	SOMME.X2MY2	SUMMEX2MY2	SOMMA.DIFF.Q	SUMAX2 MENOSY2
SUMX2PY2	SOMME.X2PY2	SUMMEX2PY2	SOMMA. SOMMA.Q	SUMAX2 MASY2
SUMXMY2	SOMME.XMY2	SUMMEXMY2	SOMMA.Q.DIFF	SUMAX MENOSY2
SYD	SYD	DIA	AMMORT. ANNUO	SYD
T	T	T	T	T
TAN	TAN	TAN	TAN	TAN
TANH	TANH	TANHYP	TANH	TANH

**Table A-3** International function names (continued)

English/ Japanese	French	German	Italian	Spanish
TBILLEQ	TAUX. ESCOMPTE.R	TBILLÄQUIV	BOT.EQUIV	LETRA.DE.TES. EQV.A.BONO
TBILLPRICE	PRIX.BON. TRESOR	TBILLKURS	BOT.PREZZO	LETRA.DE.TES. PRECIO
TBILLYIELD	RENDEMENT. BON.TRESOR	TBILLRENDITE	BOT.REND	LETRA.DE.TES. RENDTO
TDIST	LOI.STUDENT	TVERT	DISTRIB.T	DISTR.T
TEXT	TEXTE	TEXT	TESTO	TEXTO
TIME	TEMPS	ZEIT	ORARIO	NSHORA
TIMEVALUE	TEMPSVAL	ZEITWERT	ORARIO. VALORE	HORA NUMERO
TINV	LOI.STUDENT. INVERSE	TINV	INV.T	DISTR.T.INV
TODAY	AUJOURDHUI	HEUTE	OGGI	HOY
TRANSPOSE	TRANSPOSE	MTRANS	MATR. TRASPOSTA	TRANSPONER
TREND	TENDANCE	TREND	TENDENZA	TENDENCIA
TRIM	SUPPRESPEACE	GLÄTTEN	ANNULLA. SPAZI	ESPACIOS
TRIMMEAN	MOYENNE. REDUITE	GESTUTZT MITTEL	MEDIA. TRONCATA	MEDIA. ACOTADA
TRUE	VRAI	WAHR	VERO	VERDADERO
TRUNC	TRONQUE	KÜRZEN	TRONCA	TRUNCAR
TTEST	TEST.STUDENT	TTEST	TEST.T	PRUEBA.T
TYPE	TYPE	TYP	TIPO	TIPO
UPPER	MAJUSCULE	GROSS	MAIUSC	MAYUSC
USDOLLAR	USDOLLAR	USDOLLAR	USDOLLAR	USDOLLAR
VALUE	CNUM	WERT	VALORE	VALOR
VAR	VAR	VARIANZ	VAR	VAR
VARA	VARA	VARIANZA	VAR.VALORI	VARA
VARP	VAR.P	VARIANZEN	VAR.POP	VARP

*(continues)*

**Table A-3** International function names (continued)

English/ Japanese	French	German	Italian	Spanish
VARPA	VARPA	VARIANZENA	VAR.POP. VALORI	VARPA
VDB	VDB	VDB	AMMORT.VAR	DVS
VLOOKUP	RECHERCHEV	SVERWEIS	CERCA.VERT	BUSCARV
WEEKDAY	JOURSEM	WOCHENTAG	GIORNO. SETTIMANA	DIASEM
WEEKNUM	NO.SEMAIN	KALENDER WOCHE	NUM. SETTIMANA	NUM.DE. SEMANA
WEIBULL	LOI.WEIBULL	WEIBULL	WEIBULL	DIST.WEIBULL
WORKDAY	SERIE.JOUR. OUVRE	ARBEITSTAG	GIORNO. LAVORATIVO	DIA.LAB
XIRR	TRI. PAIEMENTS	XINTZINSFUSS	TIR.X	TIR.NO.PER
XNPV	VAN. PAIEMENTS	XKAPITAL WERT	VAN.X	VNA.NO.PER
YEAR	ANNEE	JAHR	ANNO	AÑO
YEARFRAC	FRACTION. ANNEE	BRTEILJAHRE	FRAZIONE. ANNO	FRAC.AÑO
YIELD	RENDEMENT. TITRE	RENDITE	REND	RENDTO
YIELDDISC	RENDEMENT. SIMPLE	RENDITEDIS	REND.TITOLI. SCONT	RENDTO.DESC
YIELDMAT	RENDEMENT. TITRE. ECHEANCE	RENDITEFÄLL	REND.SCAD	RENDTO. VENCTO
ZTEST	TEST.Z	GTEST	TEST.Z	PRUEBA.Z

## Using the Japanese translation

- Any case-insensitive text input is also width-insensitive, so that the following are equivalent:

```
=PI ( )
= P I ( )
```

where the first line uses single-byte characters and the second line uses double-byte characters. In other words, single-byte and double-byte characters are treated the same.

- Find and Replace dialogs have a check box that appears only when running in Far Eastern languages, for which the English text is "Match Character Width". Its behavior is like that of "Match Case", but for half- vs. full-width.



# B

## Comparing Excel and BIRT Spreadsheet Designer

This appendix contains the following topics:

- About Microsoft Excel compatibility
- Working with graphical object differences
- Working with worksheet differences
- Working with formula differences
- Working with pivot range differences
- Working with VBA differences

---

## About Microsoft Excel compatibility

BIRT Spreadsheet Designer produces files that are compatible with Microsoft Excel. While BIRT Spreadsheet Designer is file-compatible with Excel, the features and functions of BIRT Spreadsheet Designer's graphical user interface (GUI) are designed to appear different from Excel's GUI.

While BIRT Spreadsheet Designer has a high degree of file compatibility with Excel, there are a few differences between Excel files and Excel files produced by BIRT Spreadsheet Designer. The features included in this appendix have some degree of incompatibility between BIRT Spreadsheet Designer and Excel.

To use some Excel features that are not available in BIRT Spreadsheet Designer, you can use a VBA template. For example, you can use a VBA template to add a surface chart or a forms object to a published report. For more information about using a VBA template, see "Including VBA functionality in a spreadsheet report," in Chapter 21, "Working with VBA."

---

## Working with graphical object differences

The following list describes some graphical object differences between BIRT Spreadsheet Designer and Excel:

- Forms objects, including buttons, check boxes, drop-down lists, list boxes, and radio buttons. BIRT Spreadsheet Designer does not support creating forms objects or reading forms objects in an imported Excel file.
- Fill patterns and textures, line styles, and line widths. BIRT Spreadsheet Designer does not support patterned lines, gradients, textures, or picture object fills.
- Picture objects. You can use a BMP, GIF, or PNG in a spreadsheet report. BIRT Spreadsheet Designer does not display Windows Metafiles (WMF). If you open an Excel file that contains a WMF, BIRT Spreadsheet Designer saves the image but does not display it.
- Object rotation. You cannot rotate a BIRT Spreadsheet Designer object. If you open an Excel file that contains a rotated object in BIRT Spreadsheet Designer, the object appears unrotated.



---

## Working with worksheet differences

The following list describes differences in the functionality of Excel worksheets and BIRT Spreadsheet Designer worksheets:

- Defined names that look like cell references. This incompatibility applies to any defined name in an Excel file that begins with the letters A through AV LH followed by 1 to 1,073,741,824, such as VAL1. When you import an Excel file containing such a name into BIRT Spreadsheet Designer, formulas containing that name cannot be entered or edited.
- Workbook protection. BIRT Spreadsheet Designer does not support protecting the workbook window. BIRT Spreadsheet Designer preserves this setting in an Excel file, but the setting has no effect when you open the file in BIRT Spreadsheet Designer.
- Rotated text. BIRT Spreadsheet Designer preserves and writes out rotated text in Excel files but the text rotation may look different in Excel.
- Rows beyond 65536; columns beyond IV. With BIRT Spreadsheet Designer, you can create a worksheet that contains a larger number of rows and columns than you can with Excel. When a BIRT Spreadsheet Designer file with cells that extend beyond Excel's row and column limits is exported to Excel, the data in the rows and columns that extend beyond Excel's limits is deleted so Excel can open the file. When formulas or defined names refer to entire rows or columns, those references are adjusted to refer to an entire row or column when reading or writing Excel files. For example, a BIRT Spreadsheet Designer row reference is A1:AV LH1. That reference changes to A1:IV1 when the file is opened in Excel.
- Validation rules. Validation rules are lost when exporting to or importing from Excel.

---

## Working with formula differences

Excel limits the number of characters you can include in a formula. Some arithmetic functions evaluate differently in Excel and BIRT Spreadsheet Designer.

### Understanding formula character limits

BIRT Spreadsheet Designer does not limit the number of characters a formula can contain. In Excel, however, a formula cannot contain more than 1024 characters. To create formulas that evaluate correctly in both BIRT Spreadsheet Designer and Excel, limit formulas to no more than 1024 characters.

## Working with differences in arithmetic functions

Some BIRT Spreadsheet Designer functions do not return exactly the same results as Excel functions. In some cases, there is inconsistency in the way the Excel functions are implemented. The following examples explain some places where BIRT Spreadsheet Designer deviates from Excel results:

- ACCRINT, ODDFPRICE, ODDFYIELD, ODDLPRICE, ODDLYIELD, PRICE, YIELD. These functions rely on the COUPDAYS, COUPDAYSNC, and COUPPCD functions listed below. In many cases, the calculations are complex and sum many interest payments. A difference in any coupon period for an interest payment might be propagated through to the answer.
- AMORDEGRC. BIRT Spreadsheet Designer and Excel XP differ in the way certain numbers are rounded. In most cases, this difference is slight, not more than 1 in the currency amount. Because of the accumulation of errors, however, there might be a small difference in final payment.
- CALL, REGISTER.ID, and SQLREQUEST functions. When you import an Excel file containing one of these functions into BIRT Spreadsheet Designer, the function returns #N/A. Future versions of BIRT Spreadsheet Designer may support GETPIVOTDATA, HYPERLINK, and SQLREQUEST.
- CELL and INFO functions. Some of the arguments to these two functions are not supported in BIRT Spreadsheet Designer.
- CHAR and CODE functions. These functions might return different results in Excel and BIRT Spreadsheet Designer because BIRT Spreadsheet Designer always uses the Unicode character set, while Excel uses the character set installed on the user's computer. In locations outside the United States, the difference is noticeable for characters above 127.
- CONVERT function. In some cases, Excel uses inaccurate measurements for the conversions. Since BIRT Spreadsheet Designer's conversions are correct according to the latest Standard International (SI) units, formulas using CONVERT may return different results in Excel and BIRT Spreadsheet Designer.
- COUNTBLANK function. This function can return different results in BIRT Spreadsheet Designer and Excel. Unlike Excel, BIRT Spreadsheet Designer does not count cells containing formulas that result in blank cells.
- COUPDAYS. For calendar basis 1, this function frequently produces incorrect numbers in Excel for semi-annual and quarterly end-of-month dates, as shown in the following examples:

In Excel, `COUPDAYS("12/30/1994", "9/30/1997", 4, 1) = 90`  
In BIRT Spreadsheet Designer, `COUPDAYS("12/30/1994", "9/30/1997", 4, 1) = 92`

The correct answer is 92, which can be seen from the following series of formulas, all of which yield the same answers in both BIRT Spreadsheet Designer and Excel XP:

```
COUPDAYBS("12/30/1994", "9/30/1997", 4, 1) = 91
COUPDAYSN("12/30/1994", "9/30/1997", 4, 1) = 1
COUPPCD("12/30/1994", "9/30/1997", 4, 1) = 34607 (or 9/30/94)
COUPNCD("12/30/1994", "9/30/1997", 4, 1) = 34699 (or 12/31/94)
```

- COUPDAYSN. This function, along with COUPDAYBS and COUPDAYBS, yields a set of answers that are always consistent for calendar basis 0 and 4. Specifically, COUPDAYBS + COUPDAYSN = COUPDAYBS. COUPDAYBS is always 90, 180, and 360 for quarterly, semi-annual, and annual, respectively. However, the following examples from Excel appear inconsistent:

```
COUPDAYBS("12/30/94", "11/29/97", 4, 0) = 31
COUPDAYSN("12/30/94", "11/29/97", 4, 0) = 60
COUPDAYBS("12/30/94", "11/29/97", 4, 0) = 90
COUPDAYBS("12/30/94", "11/29/97", 4, 4) = 31
COUPDAYSN("12/30/94", "11/29/97", 4, 4) = 58
COUPDAYBS("12/30/94", "11/29/97", 4, 4) = 90
```

The first two lines of the first example add up to 91; in the second example, the same lines add up to 89. Because BIRT Spreadsheet Designer agrees with Excel XP for COUPDAYBS and COUPDAYBS, as well as COUPPCD and COUPNCD, for these cases, BIRT Spreadsheet Designer adjusts COUPDAYSN to provide a correct and consistent set of answers.

In Excel 97, COUPDAYBS returned 32 in both of these examples. BIRT Spreadsheet Designer is consistent with Excel XP in all cases that are not obviously inconsistent.

- COUPPCD. In Excel, this function apparently does not recognize 2000 as a leap year in certain cases. The following example shows the error:

```
COUPPCD("1/1/1999", "2/28/2000", 2, 0) = "8/31/1998"
```

BIRT Spreadsheet Designer returns "8/28/1998", which is consistent with normal leap year results for both BIRT Spreadsheet Designer and Excel:

```
COUPPCD("1/1/1995", "2/28/1996", 2, 0) = "8/28/1994"
```

For non-leap years, the result is the same in both BIRT Spreadsheet Designer and Excel:

```
COUPPCD("1/1/1994", "2/28/1995", 2, 0) = "8/31/1993"
```

These results are consistent regardless of the calendar basis used. This result leads to numerous errors in related functions. Specifically, COUPDAYBS incorrectly computes the number of days in all cases for maturity of 2/28/2000. Since many of the financial functions are built on the concept of coupon periods, these errors propagate through them. The following might be

affected: ACCRINT, ODDFPRICE, ODDFYIELD, ODDLPRICE, ODDLYIELD, PRICE, YIELD.

- DATEDIF. There are some discrepancies in results when using the YD option.
- MULTINOMIAL. Because Excel's handling of Boolean values in MULTINOMIAL is inconsistent with other usage, Excel returns different results than BIRT Spreadsheet Designer for this function. For example, MULTINOMIAL(TRUE,2,3) returns the #VALUE! error in Excel, while in BIRT Spreadsheet Designer it returns 60.

## Working with array formulas

An array formula is a complex Excel calculation that acts on two or more sets of values that are known as array arguments. Certain array formulas that use aggregate functions evaluate differently in BIRT Spreadsheet Designer than they do in Excel. These discrepancies are unlikely, but if you create a report that uses an array formula, you should test it in Excel before you run the report in BIRT Spreadsheet Designer. To preview the report in Excel, choose Report>View with Excel.

## Allowing an incorrect number of arguments

Excel allows Analysis add-in functions, such as COMPLEX("j"), to have more or fewer arguments than the documentation indicates. On normal functions, this is not allowed but is caught at editing time. BIRT Spreadsheet Designer does not allow an incorrect number of arguments. When you import worksheets with an incorrect number of arguments from Excel into BIRT Spreadsheet Designer, the incorrect argument count is caught and the function result is #N/A. In this situation, Excel usually starts evaluating arguments and might issue a different error. In the COMPLEX("j") example, Excel gives a #VALUE! error.

---

## Working with pivot range differences

BIRT Spreadsheet Designer supports pivot ranges. A pivot range is similar to an Excel pivot table. You can open an Excel file that contains a pivot table in BIRT Spreadsheet Designer. A pivot range you create in BIRT Spreadsheet Designer appears as a pivot table in Excel.

The following list describes differences between pivot ranges and pivot tables:

- Data sources. BIRT Spreadsheet Designer does not support using a consolidated range as a data source for a pivot range.
- Pivot charts. BIRT Spreadsheet Designer does not support pivot charts.
- Page fields. In Excel, you can show the data for each page field on a separate worksheet. BIRT Spreadsheet Designer does not support this option. An Excel

file you open in BIRT Spreadsheet Designer shows all pivot range data on a single worksheet.

- **Print options.** BIRT Spreadsheet Designer supports setting pivot range print options, but the print option settings have no effect in BIRT Spreadsheet Designer. The pivot range print options do affect the file when you print from Excel.
- **Formatting pivot range cells.** To apply number formatting to a pivot range cell, you use PivotRange Field. To apply other formatting, such as text size and style, you must use a pivot range formatting template to apply formats to the entire pivot range. Formatting that you apply to selected pivot range cells disappears when you run the report or update the pivot range, even if you select Preserve formatting on PivotRange Options.

---

## Working with VBA differences

BIRT Spreadsheet Designer does not support developing VBA code in a report design. You can, however, use an Excel template file to add VBA functionality to BIRT Spreadsheet Designer reports. When you develop a VBA template with a spreadsheet report, follow these guidelines:

- BIRT Spreadsheet Designer does not support forms objects such as buttons. To enable users to trigger the code in a completed report, you must use custom toolbars, menu items, or VB forms.
- You cannot bind event handling code to a drawing object, a chart, or a forms control at design time. Instead, you must define event handlers at run time. For example, you can programmatically set the OnAction property of an autoshape or a button.
- You cannot call VBA code from a cell formula.

For more information about using a VBA template, see “Including VBA functionality in a spreadsheet report,” in Chapter 21, “Working with VBA.”



**absolute cell reference**

A cell reference that points to an exact location on a worksheet. When a user moves or copies a BIRT Spreadsheet cell containing an absolute reference, that cell continues to refer to the originally referenced cells.

**Related terms**

cell, cell reference, worksheet

**access control list (ACL)**

A list of security IDs for data in a report. If a security ID in the access control list matches the user ID or any security role of which the user is a member, the data is accessible to that user.

**Related terms**

data, report, security ID, security role

**Contrast with**

Actuate BIRT SmartSheet Security option

**access type**

A property specifying the shareable status of a file or folder in an Encyclopedia volume. Only the owner and an Encyclopedia volume administrator can access a file or folder that has private access type. Shared access type is a prerequisite to granting privileges to other users.

**Related terms**

administrator, Encyclopedia volume, privilege, property

**Actuate BIRT Information Object Designer**

See IO Design perspective.

**Actuate BIRT Information Designer**

See IO Design perspective.

## Actuate BIRT iServer



A stand-alone server or a cluster of servers that stores report documents and information objects in an Encyclopedia volume, manages user information, handles report requests, and analyzes and delivers report documents. BIRT iServer supports BIRT spreadsheet reports. Several options providing additional functionality for Actuate BIRT iServer require separate purchase.

### **Related terms**

Actuate BIRT iServer System options, Encyclopedia volume, information object, report, report document, spreadsheet report

### **Contrast with**

Actuate BIRT iServer System, Information Console, Management Console

## Actuate BIRT iServer System



An Actuate BIRT iServer including its options. Actuate BIRT iServer System has numerous available options, which are separately purchased products.

### **Related terms**

Actuate BIRT iServer, Actuate BIRT iServer System options

### **Contrast with**

Information Console, Management Console

## Actuate BIRT iServer System options

A set of separately licensed products for Actuate BIRT iServer. Each option extends the functionality of Actuate BIRT iServer System. For example, BIRT Spreadsheet option supports generating spreadsheet reports from BIRT Spreadsheet Designer files. BIRT SmartSheet Security option controls access to spreadsheet reports, based on user name or security role. Actuate BIRT iServer System options are separately purchased products.

### **Related terms**

Actuate BIRT iServer, Actuate BIRT SmartSheet Security option, Actuate BIRT Spreadsheet option

## Actuate BIRT Java Components

A collection of tools providing support for developing and deploying rich Web 2.0 client applications. The Actuate BIRT Java Components collection includes the Actuate Deployment Kit for spreadsheet reports. Actuate distributes BIRT Java Components as a web archive (.war) file for deployment in a servlet-container environment running BIRT Java Components applications.

### **Related terms**

Actuate BIRT Spreadsheet Deployment Kit, application, Java, report, spreadsheet report, web archive (.war) file

## Actuate BIRT SmartSheet Security option

An Actuate BIRT iServer System option that controls access to spreadsheet reports, based on user name or security role. This option requires Actuate BIRT



Spreadsheet option. Actuate BIRT iServer System options are separately purchased products.

**Related terms**

Actuate BIRT iServer System options, Actuate BIRT Spreadsheet option, security role, spreadsheet report

**Contrast with**

access control list (ACL), privilege

### **Actuate BIRT Spreadsheet API**

A Java application programming interface (API) that provides support for reading and writing Microsoft Excel spreadsheets.

**Related terms**

application programming interface (API), Java

**Contrast with**

Actuate BIRT Spreadsheet Engine and API, Actuate BIRT Spreadsheet option

### **Actuate BIRT Spreadsheet Deployment Kit**



An application that provides the ability to run and view Excel spreadsheet reports and view files and folders in a file system.

**Related terms**

application, spreadsheet report

**Contrast with**

Actuate BIRT Spreadsheet Designer

### **Actuate BIRT Spreadsheet Designer**



A tool used to design custom spreadsheets for distribution using Actuate BIRT iServer System or Actuate BIRT Spreadsheet Deployment Kit. BIRT Spreadsheet Designer accesses data from external data sources to generate richly formatted Excel spreadsheet reports. A report developer can use Actuate BIRT Spreadsheet Engine and API in a callback class to customize a spreadsheet report design.

**Related terms**

Actuate BIRT iServer System, Actuate BIRT Spreadsheet Deployment Kit, Actuate BIRT Spreadsheet Engine and API, callback class, data, data source, spreadsheet report

**Contrast with**

Actuate BIRT Spreadsheet Engine, Actuate BIRT Spreadsheet option, file types

### **Actuate BIRT Spreadsheet Engine**



A set of run-time libraries providing application programming interfaces (API) that support creating, running, and deploying spreadsheet reports. BIRT Spreadsheet Engine includes BIRT Spreadsheet Designer and Actuate BIRT Spreadsheet Engine and API.

**Related terms**

Actuate BIRT Spreadsheet Designer, Actuate BIRT Spreadsheet Engine and API, application programming interface (API), library, run time, spreadsheet report

**Contrast with**

Actuate BIRT Spreadsheet API

**Actuate BIRT Spreadsheet Engine and API**

A licensed Java application programming interface (API) that provides access to all aspects of an Actuate BIRT Spreadsheet workbook. A programmer using BIRT Spreadsheet Designer accesses the API using a Java callback class. A programmer using BIRT Spreadsheet Engine accesses the API through a Java application.

**Related terms**

Actuate BIRT Spreadsheet Designer, Actuate BIRT Spreadsheet Engine, application, application programming interface (API), callback class, Java, workbook

**Contrast with**

Actuate BIRT Spreadsheet API, Actuate BIRT Spreadsheet option

**Actuate BIRT Spreadsheet option**

A separately licensed Actuate BIRT iServer System option that supports generating spreadsheet reports from BIRT Spreadsheet Designer files. The option is a prerequisite for BIRT SmartSheet Security option. Actuate BIRT iServer System options are separately purchased products.

**Related terms**

Actuate BIRT iServer System options, Actuate BIRT SmartSheet Security option, Actuate BIRT Spreadsheet Designer, spreadsheet report

**Actuate BIRT Spreadsheet technology**

A set of tools that supports the creation and deployment of data in a spreadsheet format. Actuate BIRT Spreadsheet technology includes BIRT SmartSheet Security option, BIRT Spreadsheet API, BIRT Spreadsheet Designer, BIRT Spreadsheet Engine and API, and BIRT Spreadsheet option. Actuate BIRT iServer System options are separately purchased products.

**Related terms**

Actuate BIRT iServer System options, Actuate BIRT SmartSheet Security option, Actuate BIRT Spreadsheet API, Actuate BIRT Spreadsheet Designer, Actuate BIRT Spreadsheet Engine and API, Actuate BIRT Spreadsheet option

**Actuate Deployment Kit**

See Actuate BIRT Spreadsheet Deployment Kit.

**Actuate e.Spreadsheet API**

See Actuate BIRT Spreadsheet API.

**Actuate e.Spreadsheet Deployment Kit**

See Actuate BIRT Spreadsheet Deployment Kit.

**Actuate e.Spreadsheet Designer**

See Actuate BIRT Spreadsheet Designer.

**Actuate e.Spreadsheet Engine**

See Actuate BIRT Spreadsheet Engine.

**Actuate e.Spreadsheet Engine and API**

See Actuate BIRT Spreadsheet Engine and API.

**Actuate e.Spreadsheet option**

See Actuate BIRT Spreadsheet option.

**Actuate e.Spreadsheet Server**

See Actuate BIRT iServer.

**Actuate e.Spreadsheet technology**

See Actuate BIRT Spreadsheet technology.

**Actuate iServer**

See Actuate BIRT iServer.

**Actuate iServer System**

See Actuate BIRT iServer System.

**Actuate iServer System options**

See Actuate BIRT iServer System options.

**Actuate Java Components**

See Actuate BIRT Java Components.

**Actuate server**

See Actuate BIRT iServer.

**Actuate SmartSheet Security option**

See Actuate BIRT SmartSheet Security option.

**Actuate SQL**

A query language based on the ANSI SQL-92 standard. Information objects encapsulate Actuate SQL queries.

**Related terms**

information object, query, SQL (Structured Query Language)

## ad hoc parameter

A parameter associated with a database column that passes an expression to dynamically extend the query's WHERE or HAVING clause. An ad hoc parameter uses Query by Example (QBE) syntax to restrict the number of rows returned from the data source to the report. The tools that support ad hoc parameters are BIRT Spreadsheet Deployment Kit, BIRT Spreadsheet Designer, Information Console, and Management Console. For example, in Figure G-1 the QBE value, >100, appends the expression PURCHASEVOLUME > 100 to the query's WHERE clause:

```
WHERE PURCHASEVOLUME >100
```

Parameters

☒ Customer Parameters

☐ A

Credit Rank ☐ B

☐ C

Customer Name  [Right Arrow]

Purchase Frequency  [Right Arrow]

Purchase Volume  [Right Arrow]

☐ Office Parameters

City  [Dropdown Arrow]

State  [Dropdown Arrow]

Ad hoc parameters

**Figure G-1** Ad hoc parameters showing QBE syntax

### Related terms

Actuate BIRT Spreadsheet Deployment Kit, Actuate BIRT Spreadsheet Designer, column, database, data source, expression, Information Console, Management Console, parameter, query, Query by Example (QBE), report, row, syntax, value

### Contrast with

static parameter

## administrator

A user who is able to perform administrative tasks on a system or application.

- 1 A member of the Windows Administrators group.
- 2 In Actuate BIRT iServer System, a member of the Administrator security role.

### Related terms

Actuate BIRT iServer System, security role

## aggregate expression

An expression that uses one or more aggregate functions to produce an aggregate value. For example, the expression, max([SPEED]), produces a value that is the maximum value of the field, SPEED, in the data rows.

**Related terms**

aggregate function, aggregate value, data row, expression, field, value

**Contrast with**

aggregate row

**aggregate function**

A function that performs a calculation over a set of data rows. For example, SUM( ) calculates the sum of values of a specified numeric field over a set of data rows. Examples of aggregate functions include AVG, COUNT, MAX, MIN, and SUM.

**Related terms**

data row, field, function, value

**Contrast with**

aggregate row, aggregate value

**aggregate row**

A single row that summarizes data from a group of rows returned by a query. A SQL (Structured Query Language) query that includes an aggregate expression and a Group By clause returns one or more aggregate rows. For example, a row that totals all orders made by one customer is an aggregate row.

**Related terms**

aggregate expression, data, group, query, row, SQL (Structured Query Language)

**Contrast with**

aggregate value, data row, SQL SELECT statement

**aggregate value**

The result of applying an aggregate function to a set of data rows. For example, a set of data rows has a field, SPEED, which contains values: 20, 10, 30, 15, 40. The aggregate expression, max([SPEED]), produces the aggregate value, 40, which is the maximum value for the field.

**Related terms**

aggregate expression, aggregate function, data row, field, value

**alias**

An alternative name. For example, in a SQL SELECT statement, a name given to a database table or column.

**Related terms**

column, database, SQL SELECT statement, table

**application**

A complete, self-contained program that performs a specific set of related tasks.

**application programming interface (API)**

A set of routines, including functions, methods, and procedures, that exposes application functionality to support integration and extend applications.

**Related terms**

application, function, method, procedure

**argument** A constant, expression, or variable that supplies data to a function or method.

**Related terms**

constant, data, expression, function, method, variable

**Contrast with**

parameter

**array** A data variable consisting of sequentially indexed elements that have the same data type. Each element has a common name, a common data type, and a unique index number identifier. Changes to an element of an array do not affect other elements.

**Related terms**

data, data type, element, string, variable

**balloon help**

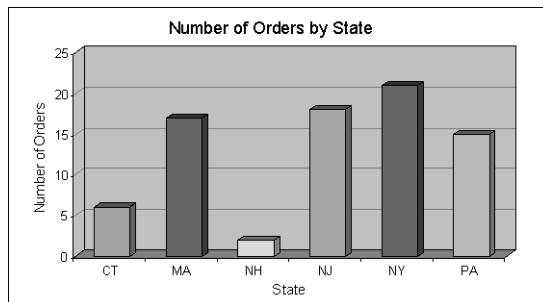
A phrase created by a report developer to explain a data item in a report. Balloon help displays when the user moves the cursor onto the item.

**Related terms**

data, report

**base chart**

The part of a chart that contains the main chart data, typically, the most important data in the chart. Every chart uses a base chart. Some charts also include study charts, which display below the base chart, or data plotted on a second *y*-axis. Figure G-2 shows a base chart.



**Figure G-2** Base chart

**Related terms**

chart, data, study chart

**Contrast with**

combination chart, dual *y*-axis chart, study chart

**base unit** A unit of time displayed on a time-scale axis in a chart.

**Related term**

chart

**Contrast with**

grid, tick

**Boolean expression**

An expression that evaluates to True or False. For example,  $\text{Total} > 3000$  is a Boolean expression. If the condition is met, the condition evaluates to True. If the condition is not met, the condition evaluates to False.

**Related term**

expression

**Contrast with**

numeric expression

**calculated column**

See computed field.

**calculated field**

In BIRT Spreadsheet Designer, a field in a data set, data range, or pivot range that uses a formula. A calculated field can perform calculations using the contents of other fields. For example, a calculated field named Forecast can estimate future orders using a formula, such as  $\text{OrderTotals} * 1.2$ .

**Related terms**

Actuate BIRT Spreadsheet Designer, data range, field, pivot range

**Contrast with**

calculated item, computed field

**calculated item**

In BIRT Spreadsheet Designer, an item in a pivot range field that uses a formula. A calculated item can perform calculations using the contents of other items in the same pivot range field. For example, a calculated item in the Product field can estimate sales for a new product based on Keyboard sales, using a formula such as  $\text{Keyboard} * 1.5$ .

**Related terms**

Actuate BIRT Spreadsheet Designer, field, formula, pivot range

**Contrast with**

calculated field

**callback class**

In BIRT Spreadsheet Designer, a Java class that a report developer writes to enhance or extend the functionality of a BIRT Spreadsheet workbook. The developer implements two methods, one called before data populates the spreadsheet and one called after data populates the spreadsheet.

**Related terms**

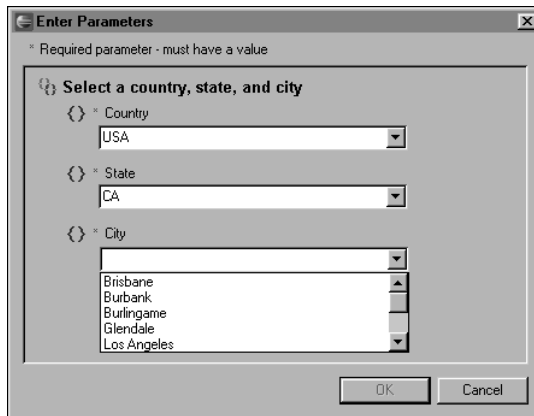
Actuate BIRT Spreadsheet Designer, class, data, Java, method, report, workbook

## cascading parameters

In Actuate BIRT spreadsheet technology, report parameters that have a hierarchical relationship, for example:

Country  
    State  
        City

In a group of cascading parameters, each report parameter displays a set of values. When a report user selects a value from the top-level parameter, the selected value determines the values that the next parameter displays, and so on. Cascading parameters display only relevant values to the user. Figure G-3 shows cascading parameters as they appear to a report user.



**Figure G-3** Cascading parameters

### Related terms

Actuate BIRT Spreadsheet technology, hierarchy, parameter, report, value

## case sensitivity

A condition in which the letter case is significant for the purposes of comparison. For example, “McManus” does not match “MCMANUS” or “mcmamus” in a case-sensitive environment.

## category

One of the discrete values that organizes the data on an area, bar, bubble, column, line, step, or stock chart axis. A category axis does not use a numeric scale. Typically, category values appear on the *x*-axis of a chart. In a pie chart, category values define which sectors appear in a pie, as shown in Figure G-4.

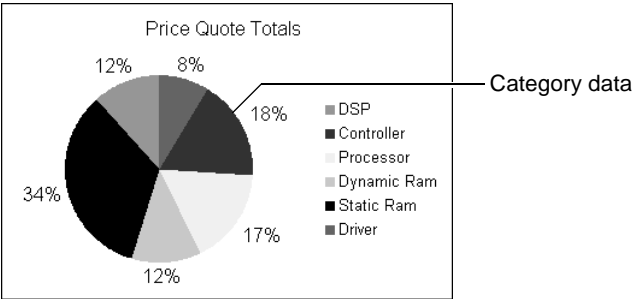
### Related terms

chart, data, hierarchy, row, value

### Contrast with

series





**Figure G-4** Category data

**cell** The intersection of a row and a column that displays a value in a spreadsheet. Figure G-5 shows a cell.

	Column 1	Column 2	Column 3
Row 1	Data	Data	Data
Row 2	Data	Data	Data
Row 3	Data	Data	Data
Row 4	Data	Data	Data

**Figure G-5** Cells in a spreadsheet

**Related terms**

column, row, value

**Contrast with**

cell reference

**cell reference**

An address, defined by letter and number, that locates a cell or a range of cells in a BIRT Spreadsheet Designer worksheet. The reference has four parts: workbook name, worksheet name, row, and column. The workbook name and worksheet name are optional. If omitted, these names default to the names of the workbook and the worksheet that use the reference. A cell reference is absolute, external, relative, or virtual. For example, to refer to the cell A1 on a worksheet named Sales in a workbook named Q2Totals.sod, use either of the following identifiers:

- '[Q2Totals.sod]Sales!A1
- A1

**Related terms**

absolute cell reference, Actuate BIRT Spreadsheet Designer, cell, column, factory, range, relative cell reference, row, virtual cell reference, workbook, worksheet

**character**

An elementary mark that represents data, usually in the form of a graphic spatial arrangement of connected or adjacent strokes, such as a letter or a digit. A

character is independent of font size and other display properties. For example, an uppercase C is a character.

**Related terms**

data, font, property

**Contrast with**

character set, string

**character set**

A mapping of specific characters to code points. For example, in most character sets, the letter A maps to the hexadecimal value 0x21.

**Related terms**

character, code point

**Contrast with**

locale

**chart**

A graphic representation of data or the relationships among sets of data, for example a bar, bubble, line, meter, pie, radar, or stock chart.

**Related term**

data

**class**

A set of methods and variables that defines the properties and behavior of an object. All objects of a given class are identical in form and behavior, but can contain different data in their variables.

**Related terms**

data, method, object, property, variable

**class declaration**

A statement that defines a class. A class declaration contains other class, method, and variable declarations.

**Related terms**

class, declaration, method, statement, variable

**class hierarchy**

A tree structure that represents the inheritance relationships among a set of classes.

**Related terms**

class, inheritance

**class name**

A unique name for a class that permits unambiguous references to its public static methods and variables.

**Related terms**

class, method, static variable, variable

**class variable**

A variable that all instances of a class share. An object-oriented environment makes only one copy of a class variable. The value of the class variable is the same for all instances of the class, for example, the taxRate variable in an Order class.

**Related terms**

class, object-oriented programming, value, variable

**code point**

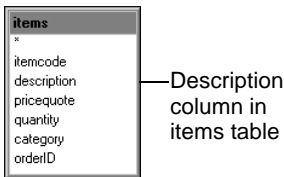
A hexadecimal value in a character set. Every character in a character set is represented by a code point. The computer uses the code point to process the character.

**Related terms**

character, character set, value

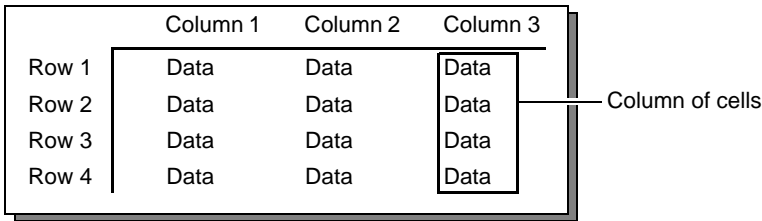
**column**

- 1 A named field in a database table or query. For each data row, the column can have a different value, called the column value. The term column refers to the definition of the column, not to any particular value. Figure G-6 shows the names of columns in a database table.



**Figure G-6** Columns in a database table

- 2 A vertical sequence of cells in a cross tab or a spreadsheet. Figure G-7 shows a column in a spreadsheet.



**Figure G-7** Column in a spreadsheet

**Related terms**

cell, data row, database, field, query, table, value

**column area**

In BIRT Spreadsheet Designer, an area that identifies the values for each column in a data range. The column area occupies the bottom row in the template area, as shown in Figure G-8.

	A	B	C	D	E
1					
2					
3					
4					
5					
6					
7					

**Figure G-8** Column area

**Related terms**

Actuate BIRT Spreadsheet Designer, column, data range, row, value

**Contrast with**

row area

**column field**

In a BIRT Spreadsheet Designer pivot range, a field defining the data that appears in columns.

**Related terms**

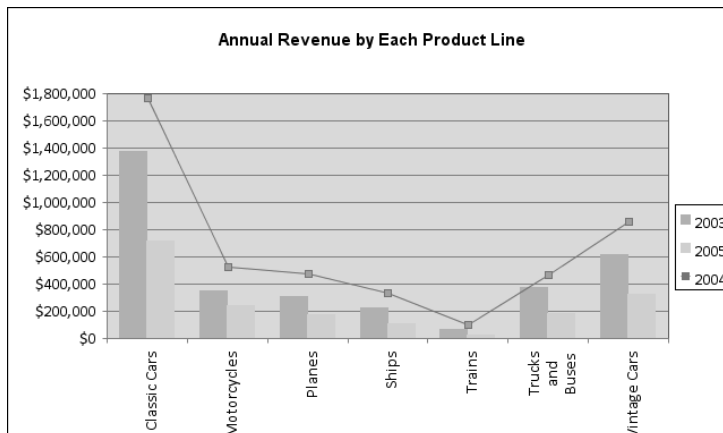
Actuate BIRT Spreadsheet Designer, column, data, field, pivot range

**Contrast with**

calculated field, calculated item, data field, row field

**combination chart**

A chart that represents multiple data series as different chart types in the same plot area. In Figure G-9, for example, the data series for 2004 appears as a line, which stands out as the year of highest annual revenue.



**Figure G-9** Combination chart

**Related terms**

Actuate BIRT Spreadsheet technology, chart, data, series

**Contrast with**

base chart, dual y-axis chart, study chart

**comma-separated values (CSV) file**

A flat file format that stores data in a tabular structure, separating the rows by new-line characters, the column values by commas, and delimiting the column values containing special characters by quotation marks.

**Related terms**

column, data, flat file, format, row, value

**Contrast with**

file types

**computed field**

A field that displays the result of an expression.

**Related terms**

expression, field

**concrete base class**

A class created to organize a hierarchy or define methods and variables that apply to derived classes. A concrete base class supports the creation of instances.

**Related terms**

class, class hierarchy, method, variable

**Contrast with**

object

**conditional format**

A format that applies to a cell when a specified condition is met.

**Related terms**

cell, format

**configuration file**

An Extensible Markup Language (XML) file containing the parameters and settings used to set run-time values for spreadsheet reports. For example, spreadsheet reports use the following configuration files:

- 1 A file that stores connection information for a database or other data source.
- 2 In Actuate BIRT iServer System, a file that specifies connection information for a database or other data source for a spreadsheet report in an Encyclopedia volume.

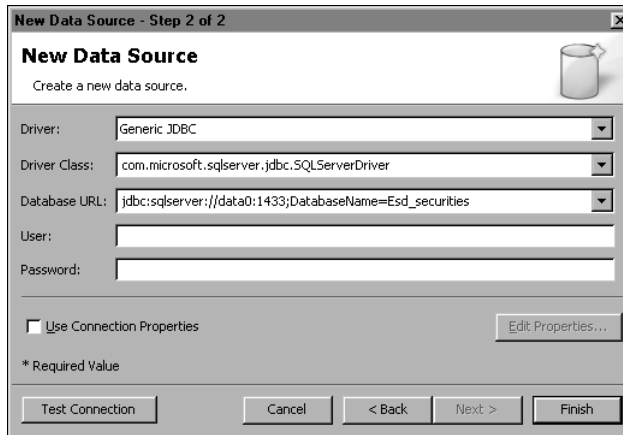
**Related terms**

connection, data source, database, Encyclopedia volume, run time, spreadsheet report

**connection**

A communication link to a database or other data source, defined by a set of connection properties. In BIRT Spreadsheet Designer, a Uniform Resource Locator (URL) used to connect a workbook to a database. Figure G-10 shows the

connection properties for a Microsoft SQL Server database.



**Figure G-10** Connection components

**Related terms**

Actuate BIRT Spreadsheet Designer, connection property, data source, database, property, uniform resource locator (URL), workbook

**connection property**

A named value used to connect to a data source. The properties vary depending on the data source type. Typical connection properties are user name and password.

**Related terms**

connection, data source, password, property, value

**Contrast with**

configuration file

**constant** An unchanging, predefined value. A constant does not change while a program is running.

**constructor code**

Code that initializes an instance of a class.

**Related term**

class

**Contrast with**

object

**content** See structured content.

**custom data source**

See open data access (ODA).

**data** Information stored in databases, flat files, or other data sources.

**Related terms**

data source, database, flat file

**Contrast with**

metadata

**Data Explorer**

In BIRT Spreadsheet Designer, a window that shows the data sets, data sources, macros, and parameters used in a report. Use Data Explorer to create, edit, or delete these items.

**Related terms**

Actuate BIRT Spreadsheet Designer, data set, data source, parameter, report

**data field** In a BIRT Spreadsheet Designer pivot range, a field that determines the calculated data or summary data that the pivot range displays. Typically, a data field uses a calculation and a database field, such as the sum of sales.

**Related terms**

Actuate BIRT Spreadsheet Designer, data, database, field, pivot range

**data label** Text that describes a data point in a chart. A data label line connects the data label to its data point.

**Related terms**

chart, data, data point

**data point** A point on a chart that corresponds to a particular pair of *x*- and *y*-axis values.

**Related terms**

chart, value

**Contrast with**

data label, data row

**data range**

In BIRT Spreadsheet Designer, a range of cells that combines and compares data in a dynamic layout similar to a cross-tab report. A data range summarizes and arranges data in multiple and dynamic hierarchies.

**Related terms**

Actuate BIRT Spreadsheet Designer, cell, data, hierarchy, layout, range, report

**Contrast with**

pivot range

**data row** row of data that a data set returns. A data set typically returns many data rows.

**Related terms**

Actuate BIRT Spreadsheet Designer, data, data set, row

**Contrast with**

data point, filter

**data set** In Actuate BIRT Spreadsheet Designer, a definition of the data to retrieve or compute from a data source.

**Related terms**

Actuate BIRT Spreadsheet Designer, data, data source

**Contrast with**

data row

**data source**

- 1 A relational database or other data repository. For example, an Extensible Markup Language (XML) file, a flat file, an information object, or a Java application can be a data source. A report or spreadsheet can include any of these types of data.
- 2 A design construct that retrieves data rows from a relational database or other data repository.

**Related terms**

application, data, data row, database, Extensible Markup Language (XML), flat file, information object, Java, report

**data type** The structure of a value that constrains its characteristics, such as the information the values can hold and permitted operations. In report development, three processes use data types: accessing data, internal processing of data, and formatting output as a report.

**Related terms**

data, report, value

**database** An integrated collection of logically related records that provides data for information application platforms. The database model most commonly used is the relational model. Other typical models are entity-relationship, hierarchical, network, object, and object-relational.

**Related terms**

Actuate BIRT Spreadsheet Designer, application, data, database, object

**database connection**

See connection.

**database management system (DBMS)**

An application that controls the storage, retrieval, and manipulation of data in a data store.

**Related terms**

application, data

**Contrast with**

relational database management system (RDBMS)

**database schema**

See schema.



**Date data type**

A Java data type used for date-and-time calculations. The base Date data type, `java.util.Date`, is a class that encapsulates a millisecond date value from January 1, 1970 00:00:00.000 GMT through the year 8099. This Date class provides accessor methods that support getting and setting the value.

**Related terms**

class, data type, Java, method, value

**DBMS (database management system)**

See database management system (DBMS).

**debug**

To detect, locate, and fix errors in a computer program. Typically, debugging involves executing specific portions of the program and analyzing the operation of those portions.

**declaration**

The definition of a class, constant, method, or variable that specifies the name and, if appropriate, the data type.

**Related terms**

class, constant, data type, method, variable

**Contrast with**

class declaration

**declarations section**

That portion of Java code that contains constant, data type, and global variable declarations.

**Related terms**

constant, data type, declaration, Java, variable

**defined name**

In BIRT Spreadsheet Designer, a string that identifies a cell, a range of cells, a formula, or a value. A defined name can also identify a constant, a formula expression, or a parameter. BIRT Spreadsheet Designer uses some reserved defined names such as `sheetname!Print_Area`.

**Related terms**

Actuate BIRT Spreadsheet Designer, cell, constant, expression, formula, parameter, range, string, value

**delete privilege**

See privilege.

**delimited data**

In BIRT Spreadsheet Designer, text used to represent data where each line of text corresponds to a data row. The data row contains fields separated by a text

character such as a comma, semicolon, space, or tab that determines column breaks within the row.

**Related terms**

Actuate BIRT Spreadsheet Designer, column, data, data row, field, row

**derived class**

See descendant class.

**descendant class**

A class that extends another class to provide additional functionality.

**Related term**

class

**design**

A report specification or the act of creating a report specification. Designing a report includes selecting data, laying out the report visually, and saving the layout in a report design file. A report developer creates and edits a report design using BIRT Spreadsheet Designer.

**Related terms**

Actuate BIRT Spreadsheet Designer, data, layout, report

**Contrast with**

file types

**design time**

The period of time in which a report developer creates a report specification.

**Related term**

report

**Contrast with**

design, run time, view time

**Design View**

In BIRT Spreadsheet Designer, a page displaying a graphical representation of a spreadsheet report on a worksheet.

**Related terms**

Actuate BIRT Spreadsheet Designer, page, spreadsheet report, worksheet

**Contrast with**

layout editor

**driver**

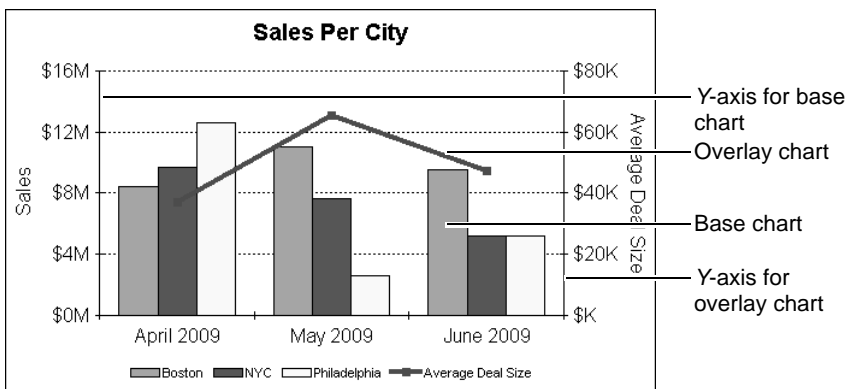
An interface that supports communication between an application and another application or a peripheral device such as a printer. To communicate with databases, BIRT Spreadsheet Designer uses drivers, such as open data access (ODA) and Java database connectivity (JDBC) drivers.

**Related terms**

Actuate BIRT Spreadsheet Designer, application, database, Java Database Connectivity (JDBC), open data access (ODA) driver

**dual y-axis chart**

A chart that uses two independent *y*-axes, as shown in Figure G-11. When one *y*-axis displays a different type of series from the other *y*-axis, the chart is called a combination chart.



**Figure G-11** Dual y-axis chart

**Related terms**

base chart, chart, combination chart, series

**Contrast with**

study chart

**element**

- 1 A single item of data.
- 2 A logical structure in an Extensible Markup Language (XML) or HyperText Markup Language (HTML) document specifying a type and optionally one or more attributes and a value. For example, the following code specifies a ConnectionParam element that has three attributes, Name, Display, and Type, and no value:

```
<ConnectionParam Name="username"
    Display="User name"
    Type="string"
/>
```

**Related terms**

data, Extensible Markup Language (XML), HyperText Markup Language (HTML), value

**ellipsis**



A button that opens tools that you use to perform tasks, such as navigating to a file.

## **encapsulation**

A technique that bundles related functions and subroutines. Encapsulation compartmentalizes the structure and behavior of a class so that parts of an object-oriented system do not depend upon or affect each other's internal details.

### **Related terms**

class, function, object, object-oriented programming

## **Encyclopedia volume**

An Actuate BIRT iServer System repository for managing data and metadata. Encyclopedia volume data includes objects such as designs and documents, stored as files in partitions. Metadata consists of information about Encyclopedia volume channels, data objects, groups, roles, users, and other configuration elements, stored in a third-party relational database management system (RDBMS) such as PostgreSQL or Oracle.

### **Related terms**

Actuate BIRT iServer System, data, design, group, information object, relational database management system (RDBMS), repository, role, security role

## **escape character (\)**

A character indicating when to take a special character literally, as in a Query by Example (QBE) expression. For example, a backslash followed by a comma represents a comma character: \,

### **Related terms**

character, Query by Example (QBE)

## **e.Spreadsheet**

See spreadsheet report.

## **e.Spreadsheet design file**

See spreadsheet object design (.sod) file.

## **e.Spreadsheet Engine**

See Actuate BIRT Spreadsheet Engine.

## **e.Spreadsheet executable file**

See spreadsheet object executable (.sox) file.

## **e.Spreadsheet option**

See Actuate BIRT Spreadsheet option.

## **e.Spreadsheet report**

See spreadsheet report.

## **e. Spreadsheet technology**

See Actuate BIRT Spreadsheet technology.

**event** An action external to a program that requires handling, such as a mouse click. An event handler in the program collects information about the event and responds.

**Related term**  
event handler

### **event handler**

A function or method that executes when an event occurs. Report items and data sources have event handlers for which a developer can provide code.

**Related terms**  
data source, event, function, method, report

**Contrast with**  
event listener

### **event listener**

An interface that detects when a particular event occurs and calls a function or method to respond to the event.

**Related terms**  
event, function, interface, method

**Contrast with**  
event handler

**exception** An abnormal situation that a program encounters. The program handles some exceptions and returns a message to the user or application running the program. In other cases, the program cannot handle the exception, and the program ends.

**Related term**  
application

### **executable file**

A file that generates report output when run in an Encyclopedia volume or a report designer. For example, a spreadsheet object executable (.sox) file generates report output.

**Related terms**  
Encyclopedia volume, file types, report, spreadsheet object executable (.sox) file

### **expression**

A combination of constants, functions, literal values, names of fields, and operators that evaluate to a single value.

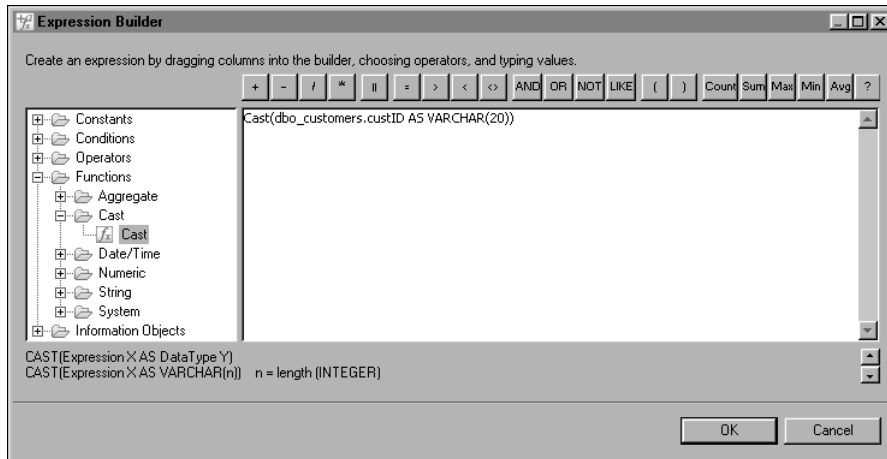
**Related terms**  
constant, expression builder, field, function, operator, value

**Contrast with**  
aggregate expression

## expression builder



A tool for selecting data fields, functions, and operators to write expressions. Figure G-12 shows the expression builder in BIRT Spreadsheet Designer.



**Figure G-12** Expression builder

### Related terms

Actuate BIRT Spreadsheet Designer, data, expression, field, function, operator

## Extensible Markup Language (XML)

A markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable. It is defined in the XML 1.0 Specification produced by the Worldwide Web Consortium (W3C). XML is widely used for the representation of data structures.

### Related terms

data, format

### Contrast with

HyperText Markup Language (HTML)

## external cell reference

A four-part cell reference having a different workbook name from the workbook that uses the reference.

### Related terms

cell reference, workbook

## factory

A process that generates a report document from an executable file.

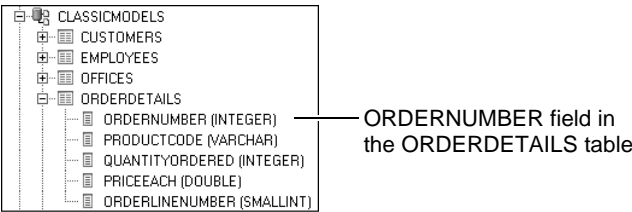
### Related terms

executable file, report document

### Contrast with

file types

**field** The smallest identifiable part of a database table structure. In a relational database, a field is also called a column. Figure G-13 shows a field in a table.



**Figure G-13** Fields and tables displayed in a query editor

**Related terms**  
column, database, query editor, table




**file types** Table G-1 lists the file types related to using BIRT Spreadsheet Designer.

**Table G-1** File types

Management Console display name	File extension	Icon
Actuate BIRT Spreadsheet Design	SOD	
Actuate BIRT Spreadsheet Document	SOI	
Actuate BIRT Spreadsheet Executable	SOX	
Actuate Information Object	IOB	
Adobe PDF File	PDF	
Comma Separated Values File	CSV	
HTML document	HTM	
HTML document	HTML	
Microsoft Excel Spreadsheet	XLS	
Microsoft Excel Spreadsheet (2007)	XLSX	
Microsoft PowerPoint Presentation	PPT	
Microsoft PowerPoint Presentation (2007)	PPTX	

(continues)

**Table G-1** File types (continued)

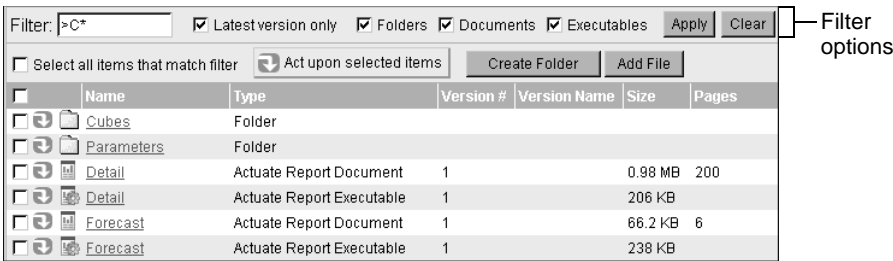
Management Console display name	File extension	Icon
Rich Text Format	RTF	
Tab Separated Values File	TSV	
Text File	TXT	

**Related terms**

information object, information object (.iob) file, Management Console, spreadsheet object design (.sod) file, spreadsheet object executable (.sox) file, spreadsheet object instance (.soi) file

**filter**

A mechanism that enables a user to reduce the number of items in a list. For example, in Information Console and Management Console, filter options appear above the list of contents. Figure G-14 shows filter options in Management Console.



**Figure G-14** Filter options in Management Console

**Related terms**

Information Console, Management Console

**flat file**

A file that contains data in the form of text.

**Related term**

data

**Contrast with**

data source

**font**

A family of characters of a given style. A font contains information that specifies posture, typeface, type size, and weight.

**Related term**

character

**footer**

A logically separate unit of information that appears after the main body of content, for example a date stamp.



**Contrast with**  
header

**format**

- 1 A specification that describes layout and properties of rich information, such as HyperText Markup Language (HTML), PDF, PostScript, PowerPoint, or RTF.
- 2 A set of standard options with which to display and print currency values, dates, numbers, strings, and times.

**Related terms**

HyperText Markup Language (HTML), layout, property, string, value

**Contrast with**

formatting toolbar, style

**formatting toolbar**

A toolbar used to change text properties. The properties available on the formatting toolbar include alignment, font, font color, font size, and font style. Figure G-15 shows a typical formatting toolbar.



**Figure G-15** Formatting toolbar

**Related terms**

font, property, toolbar

**formula**

An expression that performs a calculation in a worksheet cell. Formulas can contain cell references, defined names, functions, mathematical operators, spreadsheet-specific operators, and values.

**Related terms**

cell, cell reference, defined name, expression, function, operator, value, worksheet

**function**

A keyword used in a BIRT Spreadsheet Designer worksheet to perform a specific calculation or evaluation in a formula. For example, sum, rand, and count are spreadsheet function keywords.

**Related terms**

Actuate BIRT Spreadsheet Designer, formula, keyword, value, worksheet

**Contrast with**

method

**global variable**

A variable available at all levels in an application. A global variable stays in memory in the scope of all executing subroutines until the application terminates.

**Related terms**

application, scope, variable

**grandchild class**

See descendant class.

**grid** In a spreadsheet report, demarcation lines that extend from axis tick marks of a chart.

**Related terms**  
chart, spreadsheet report, tick

**group** A set of data rows organized by one or more common values. For example, in a sales report, a group consists of all the orders placed by a single customer.

**Related terms**  
data row, report, value  
**Contrast with**  
group key, grouped report

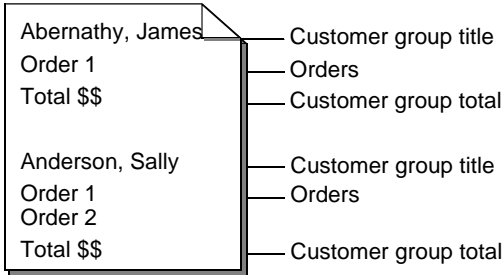
**group footer**  
See footer.

**group header**  
See header.

**group key** An expression that groups and sorts data. For example, a report developer can group and sort customers by credit rank.

**Related terms**  
data, expression, group, sort

**grouped report**  
A report that organizes data by common values. Figure G-16 shows a grouped report organized by customer name.



**Figure G-16** Grouped report

**Related terms**  
data, report, value  
**Contrast with**  
group

**header** A logically separate unit of information that appears before the main body of content. For example, a page header typically contains a document title.

**Related terms**

group, page

**Contrast with**

footer

**help** See balloon help.

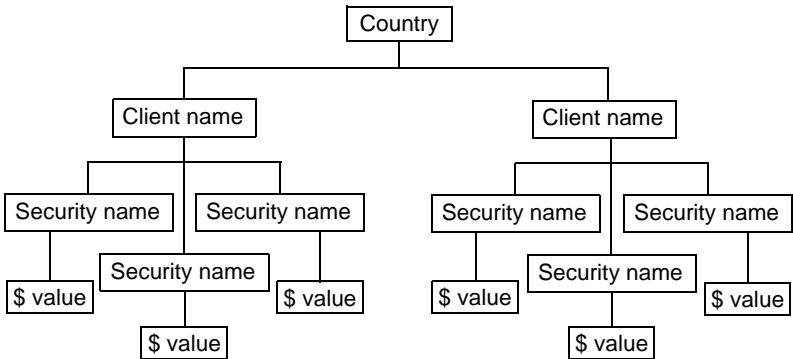
**hexadecimal number**

A number in base 16. A hexadecimal number uses the digits 0 through 9 and the letters A through F. Each place represents a power of 16. By comparison, base 10 numbers use the digits 0 through 9. Each place represents a power of 10.

**Contrast with**

character set

**hierarchy** Any tree structure that has a root and branches that do not converge. Figure G-17 shows an example data hierarchy.



**Figure G-17** Example of a data hierarchy

**home folder**

A path and folder name that is the default user working environment in an Encyclopedia volume or file system repository.

**Related terms**

Encyclopedia volume, repository

**HTML** See HyperText Markup Language (HTML).

**HTTP** See HyperText Transfer Protocol (HTTP).

**hyperlink**



An active connection in an online document that supports access to related information in the same document or an external source. The document can be an

e-mail, PDF, report, or web page. A change from the standard cursor shape to a cursor shaped like a hand indicates a hyperlink.

**Related terms**

report, web page

## **HyperText Markup Language (HTML)**

A standards-based specification that determines the layout of a web page. HTML is the markup language that a web browser parses to display a web page.

**Related terms**

layout, web page

**Contrast with**

Extensible Markup Language (XML)

## **HyperText Transfer Protocol (HTTP)**

A standard that supports request-response communication between two applications on a network. The World Wide Web Consortium (W3C) specifies the standard for HTTP.

**Related terms**

application

**identifier** A name assigned to an item in a program, for example a class, function, or variable.

**Related terms**

class, function, variable

**image** A graphic that appears in a spreadsheet. Spreadsheet reports support .bmp, .gif, .jpg, and .png.

**Related terms**

spreadsheet report

## **Information Console**

An Actuate BIRT iServer component that supports running and viewing reports stored in an Encyclopedia volume.

**Related terms**

Actuate BIRT iServer, Encyclopedia volume, report

## **information object**



A named SQL (Structured Query Language) query that simplifies access to one or more heterogeneous data sources. An information object retrieves data using database tables and views, stored procedures, and ODA data source queries as well as other information objects. A data modeler writes the query in Actuate SQL. The integration service generates a native query for each data source and retrieves the data.

**Related terms**

Actuate SQL, data, database, open data access (ODA), query, SQL (Structured Query Language), stored procedure, table, view

**Contrast with**

information object (.iob) file

**information object (.iob) file**

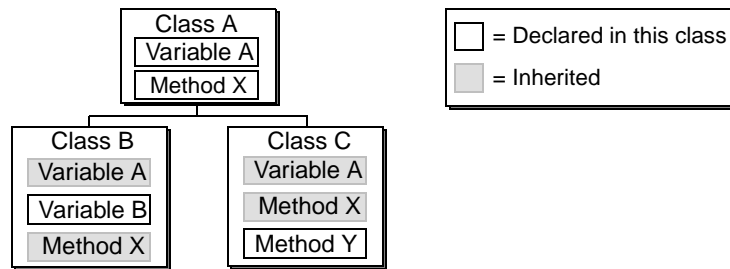
A file type that contains an Actuate SQL query. Data modelers create information objects using the IO Design perspective.

**Related terms**

Actuate SQL, file types, information object, IO Design perspective, query

**inheritance**

A mechanism whereby one class of objects can be defined as a special case of a more general class and includes the method and variable definitions of the general class, known as a base or superclass. The superclass serves as the baseline for the appearance and behavior of the descendant class, which is also known as a subclass. In the subclass, the appearance, behavior, and structure can be customized without affecting the superclass. Figure G-18 shows an example of inheritance.



**Figure G-18** Inheritance

**Related terms**

class, descendant class, method, object, variable

**Contrast with**

hierarchy, object-oriented programming

**inner join**

A type of join that returns records from two tables using matching values in the join field. Records for which there is no matching value are not included in the result set. For example, joining customer and order tables where the customer IDs are equal produces a result set that excludes records for customers who have no orders.

**Related terms**

field, join, result set, table, value

**Contrast with**

outer join

## Input Method Editor (IME) file

A Java class file that sets the keyboard mapping for a character set. BIRT Spreadsheet Designer uses this mechanism to support non-ASCII characters. Place the IME file in the `jre\lib\ext` directory to make it accessible to the Java environment.

### Related terms

Actuate BIRT Spreadsheet Designer, character, character set, class, Java

## input source

See data source.

## instance

See object.

## interface

A software component that supports access to computer resources. For example, in Java, a set of methods that provides a mechanism for classes to communicate in order to execute particular actions.

### Related terms

class, Java, method

## internationalization

The process of designing an application to work correctly in multiple locales.

### Related terms

application, locale

### Contrast with

localization

## IO Design perspective



A tool used to design information objects. Information objects use Actuate SQL syntax to retrieve data from one or more data sources. Report developers use information objects in report designs.

### Related terms

Actuate SQL, data, data source, information object, query, SQL (Structured Query Language)

### Contrast with

file types

## IOB

See information object (.iob) file.

## J2EE

See Java Platform Enterprise Edition (Java EE).

## J2SE

See Java Platform Standard Edition (Java SE).

## JAR

See Java archive (.jar) file.

## Java

An object-oriented programming language.

**Related terms**

object-oriented programming

**Java 2 Enterprise Edition (J2EE)**

See Java Platform Enterprise Edition (Java EE).

**Java 2 Runtime Standard Edition (J2SE)**

See Java Platform Standard Edition (Java SE).

**Java archive (.jar) file**

A compressed file format used to deploy Java applications.

**Related terms**

application, Java

**Java Database Connectivity (JDBC)**

A standard protocol that Java uses to access databases in a platform-independent manner. For example, BIRT Spreadsheet Engine uses JDBC to connect to databases.

**Related terms**

Actuate BIRT Spreadsheet Engine, database, database connection, Java

**Contrast with**

database management system (DBMS), open database connectivity (ODBC), schema

**Java Development Kit (JDK)**

A software development kit that defines the application programming interfaces (API) used to build Java applications. As well as software tools, the kit contains documentation and examples.

**Related terms**

application, application programming interface (API), Java

**Contrast with**

Java Platform Enterprise Edition (Java EE), Java Platform Standard Edition (Java SE), JavaServer Page (JSP)

**Java Naming and Directory Interface (JNDI)**

An application programming interface (API) that provides unified access to named components and directory services in an enterprise system.

**Related terms**

application programming interface (API)

**Java Platform Enterprise Edition (Java EE)**

A platform-independent development environment that includes application programming interfaces (API), such as Java Database Connectivity (JDBC), Remote Method Invocation (RMI), and web services. A programmer uses Java EE to develop a highly scalable, fault-tolerant, web-based application.

**Related terms**

application, application programming interface (API), Java Database Connectivity (JDBC)

**Contrast with**

Java Development Kit (JDK), Java Platform Standard Edition (Java SE), Java Virtual Machine (JVM)

**Java Platform Standard Edition (Java SE)**

A smaller-scale, platform-independent development environment defining the Java programming language and application programming interfaces (API) supporting interaction with file systems, networks, and graphical interfaces. A programmer uses Java SE to develop an application to run on a virtual machine.

**Related terms**

application, application programming interface (API), Java

**Contrast with**

Java Development Kit (JDK), Java Platform Enterprise Edition (Java EE), Java Virtual Machine (JVM)

**Java Virtual Machine (JVM)**

The Java SDK interpreter that converts Java bytecode into machine language for execution in a specified software and hardware configuration.

**Related terms**

Java, SDK (Software Development Kit)

**JavaScript**

An interpreted, platform-independent, scripting language used to embed additional processing in a web page or server.

**Related term**

web page

**Contrast with**

Java

**JavaServer Page (JSP)**

A standard Java extension that supports the generation of dynamic web pages. A JavaServer Page combines HyperText Markup Language (HTML) and JSP tags in one document. A servlet container interprets a JSP tag as a call to a Java class. The servlet container compiles the Java classes to generate a web page.

**Related terms**

class, HyperText Markup Language (HTML), Java, tag, web page

**JDK**

See Java Development Kit (JDK).

**JNDI**

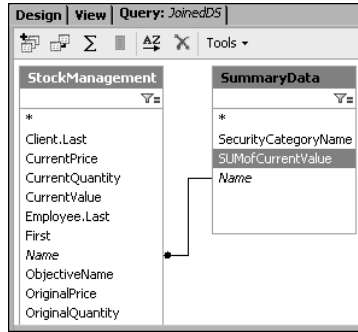
See Java Naming and Directory Interface (JNDI).

**join**

A SQL (Structured Query Language) query operation that combines records from two tables and returns them in a result set based on the values in the join fields.



Without additional qualification, join usually refers to the join in which field values are equal. For example, StockManagement and SummaryData tables are joined on a common field such as Name. The result set contains combined StockManagement and SummaryData records in which the names are equal. Figure G-19 shows joins in the query editor in BIRT Spreadsheet Designer.



**Figure G-19** Join between StockManagement and SummaryData tables

**Related terms**

field, query, query editor, result set, SQL (Structured Query Language), table, value

**Contrast with**

inner join, join condition, outer join, SQL SELECT statement

**join condition**

A condition that specifies a match in the values of related fields in two tables. Typically, the values are equal. Figure G-19 shows a join condition where the Name value in the first table equals the Name value in the second table.

**Related terms**

field, join, table, value

**joint data set**

data set that combines data from two or more data sets.

**Related terms**

data, data set

**JSP**

See JavaServer Page (JSP).

**JVM**

See Java Virtual Machine (JVM).

**keyword**

A reserved word that is recognized as part of a programming language.

**layout**

The designed appearance of a report. Designing a report entails arranging items on a page so that a report user can analyze the information easily. Figure G-20 shows a pivot range tabular layout.

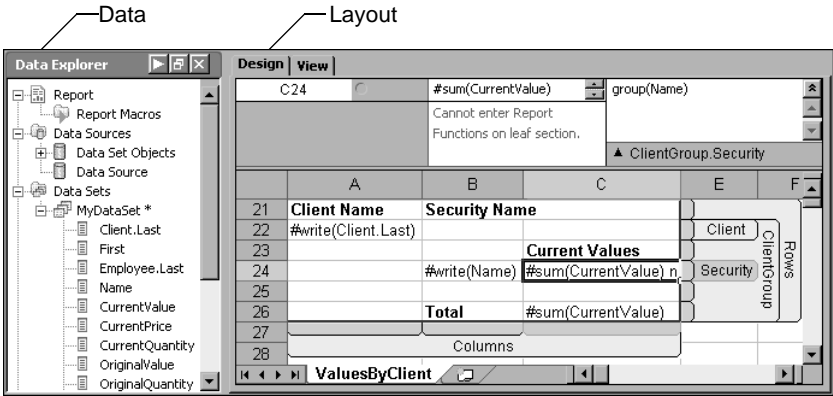
Number of orders		shipByDate			
offices city	last	Qtr1	Qtr2	Qtr3	Grand Total
Boston	Castillo	16	56	52	124
	Firrelli	21	15	42	78
	Murphy	146	28	28	202
	Patterson	75	12	20	107
	Thompson	26	59	44	129
Boston Total		284	170	186	640
NYC	Barajas	14	16	78	108
	Hernandez	89	47	2	138
	Thompson	3		22	25
	Tseng	75	17	9	101
	Vanauf		44	21	65
NYC Total		181	124	132	437
Philadelphia	Firrelli	188		24	212
	Howard	114		16	130
	Jennings	29	26	24	79
	Patterson	48		28	76
Philadelphia Total		379	26	92	497
Grand Total		844	320	410	1574

**Figure G-20** Pivot range tabular layout

**Related term**  
pivot range, report

**layout editor**

A tool in a report designer in which a report developer arranges, formats, and sizes elements. Figure G-21 shows the layout editor in BIRT Spreadsheet Designer.



**Figure G-21** Layout editor

**Related terms**  
Actuate BIRT Spreadsheet Designer, design, format, report

**Layout window**

See layout editor.

**library**

A file used when creating or running a program. For example, Windows library files are dynamic link libraries. UNIX library files are shared libraries.

**left outer join**

See outer join.

**line control**

In BIRT Spreadsheet Designer, a component that draws a line in a spreadsheet report design.

**Related terms**

Actuate BIRT Spreadsheet Designer, design, spreadsheet report

**link**

See hyperlink.

**locale**

A location and the currency format, date format, language, sorting sequence, time format, and other characteristics associated with that location. The location is not always identical to the country. There can be multiple languages and locales within one country. For example, China has two locales: Beijing and Hong Kong. Canada has two language-based locales: French and English.

**Related term**

format

**Contrast with**

localization

**localization**

The process of translating database content, printed documents, and software programs into another language. Report developers localize static text in a report so that the report displays text in another language that is appropriate to the locale configured on the user's machine.

**Related terms**

database, locale, report

**Contrast with**

internationalization

**Macro page**

In BIRT Spreadsheet Designer, a page used to add or modify macros in a report.

**Related terms**

Actuate BIRT Spreadsheet Designer, page, report

**Management Console**

A set of web pages that provide volume management functions, such as creating channels, roles, security, and users for Encyclopedia volumes.

**Related terms**

Encyclopedia volume, security role, web page

**Contrast with**

Information Console

**metadata** Information about the structure of data enabling a program to process information. For example, a relational database stores metadata that describes the data type, name, and size of objects in a database, such as tables and columns.

**Related terms**

column, data, data type, database, object, table

**method** A routine that provides functionality to an object or a class.

**Related terms**

class, object

**Contrast with**

data, function, overloaded method

**null** A value indicating that a variable or field contains no data.

**Related terms**

data, field, value, variable

**numeric expression**

A numeric constant, a simple numeric variable, a scalar reference to a numeric array, a numeric-valued function reference, or a sequence of these items, separated by numeric operators and parentheses. For example:

```
dataSetRow["PRICEEACH"] * dataSetRow["QUANTITYORDERED"]
```

**Related terms**

array, constant, function, operator, variable

**Contrast with**

Boolean expression

**object** An instance of a particular class, including its characteristics. An object has properties and methods. For example, when a user runs a spreadsheet report on BIRT iServer, the Factory process instantiates multiple persistent objects, one for each row retrieved from the data source.

**Related terms**

Actuate BIRT iServer, class, data source, method, persistent object, property, row

**Contrast with**

information object

**object reference variable**

A variable that contains a reference to an object. References can be passed to functions and subroutines as parameters.

**Related terms**

function, object, parameter, variable

**object-oriented programming**

A paradigm for writing applications using classes, not algorithms, as the fundamental building blocks. The design methodology uses four main concepts: abstraction, encapsulation, inheritance, and polymorphism.

**Related terms**

application, class, encapsulation, inheritance

**Contrast with**

object

**ODA** See open data access (ODA).

**ODBC** See open database connectivity (ODBC).

**open data access (ODA)**

A technology that handles communication between a data source and an application. ODA provides interfaces for creating data drivers to establish connections, access metadata, and execute queries to retrieve data. ODA also provides interfaces to integrate query builder tools within an application designer tool.

**Related terms**

application, connection, data, data source, interface, metadata, open data access (ODA) driver, query

**open data access (ODA) driver**

An ODA driver communicates between a data source and an application. An ODA driver establishes a connection to a data source, accesses metadata about the data, and executes queries on the data source.

**Related terms**

application, data, data source, driver, metadata, open data access (ODA), query

**open database connectivity (ODBC)**

A standard protocol used by software products as a database management system (DBMS) interface to connect applications and reports to databases.

**Related terms**

application, database, database management system (DBMS), interface, report

**Contrast with**

data source, Java Database Connectivity (JDBC)

**operator** A symbol or keyword that performs an operation on expressions.

**Related terms**

expression, keyword

**outer join** A type of join that returns records from one table even when no matching values exist in the other table. The three types of outer join are left, right, and full outer join. A left outer join returns all records from the table on the left side of the join expression, even if no matching values exist in the table on the right side. A right outer join returns all records from the table on the right side of the join expression, even if no matching values exist in the table on the left side. For example, joining customers and orders tables on customerID with the customers table on the left side of the expression returns a result set that contains all customer records,

including customers who have no orders. A full outer join is the union of the result sets of both left and right outer joins.

**Related terms**

join, result set, table, value

**Contrast with**

inner join

**output format**

A format to which spreadsheet reports can be downloaded. Example formats include Microsoft Excel and PDF.

**Related terms**

format, spreadsheet report

**overlay chart**

See dual y-axis chart.

**overloaded method**

In a single class, a method in which different sets of arguments can be specified.

**Related terms**

argument, class, method

**override**

To write new code that replaces the default code of an inherited method.

**Related terms**

inheritance, method

**package**

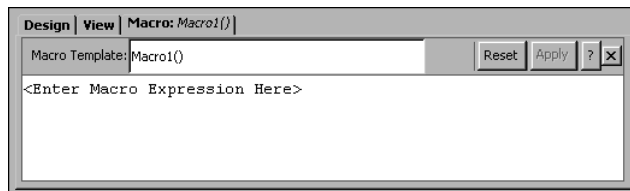
- 1 A set of functionally related Java classes organized in one directory.
- 2 A complete application, including all configuration files and programs.

**Related terms**

application, class, configuration file, Java

**page**

An area in a window that arranges and displays related information. A window can contain several pages, each of which is accessed by a tab. Figure G-22 shows an example of pages in the layout editor in BIRT Spreadsheet Designer.



**Figure G-22** Pages in the layout editor

**Related terms**

Actuate BIRT Spreadsheet Designer, layout editor, tab

**Contrast with**

JavaServer Page (JSP), web page

**page footer**

Content that appears at the bottom of each page. For example, a page footer can display a date and a page number.

**Contrast with**  
page header

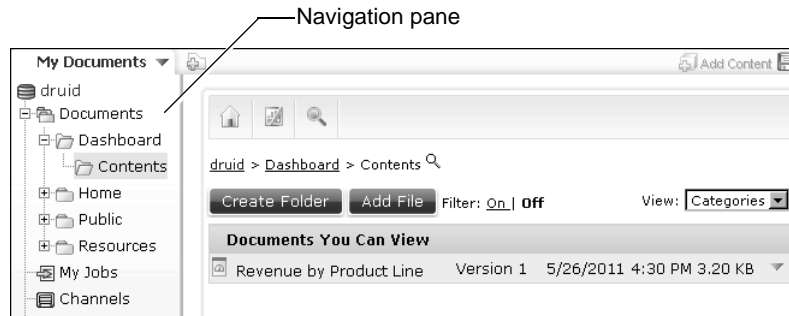
**page header**

Content that appears at the top of each page. For example, a page header can display a document title.

**Contrast with**  
page footer

**pane**

An area in a user interface. For example, Figure G-23 shows the navigation pane in Information Console.



**Figure G-23** Pane

**Related terms**

Information Console, interface

- parameter**
- 1 A report element or variable that provides input to the execution of the report. A user types or selects values for parameters on the Parameters page, as shown in Figure G-24.
  - 2 In Actuate SQL, a variable that appears in a query, for example, in a WHERE clause.
  - 3 The definition of an argument to a procedure.

**Related terms**

Actuate SQL, argument, Parameters page, procedure, query, report, variable

**Contrast with**

ad hoc parameter, cascading parameters, function, report parameter

**Figure G-24** Parameters page

## Parameters page

In Information Console and Management Console, a page that prompts a user to enter values for parameters when generating a report. Figure G-24 shows the Parameters page.

### Related terms

Information Console, Management Console, page, parameter, report, value

## password

An optional code that restricts user name access to a resource on a computer system. For example, in an Encyclopedia volume, passwords can be up to 256 characters in length and can contain any characters except control characters or spaces. Passwords are case-sensitive. Passwords can use a mixture of case, alphabetical, and numeric characters to increase security.

### Related terms

case sensitivity, Encyclopedia volume, user name

### Contrast with

security role

## persistent object

An object that is stored in a file. For example, most objects created by running a spreadsheet object executable (.sox) file, including data fields, headings, and images, are persistent. A spreadsheet object instance (.soi) file contains only persistent objects.

### Related terms

data, file types, image, object, run, spreadsheet object executable (.sox) file, spreadsheet object instance (.soi) file

## personal folder

See home folder.

## pivot range

In BIRT Spreadsheet Designer, a range of data displayed as a table in a spreadsheet report. Typically, data in a pivot range is calculated data. Pivot



ranges support summarizing and comparing data. You can change the arrangement, grouping, and sorting of pivot range data without changing the query.

**Related terms**

Actuate BIRT Spreadsheet Designer, calculated item, data, group, query, range, sort, spreadsheet report, table

**Contrast with**

calculated field

**privilege** A level of access to an item in an Encyclopedia volume. Users have privileges either directly or through security roles. The privileges include, for example, the ability to delete, execute, read, and write to an object. The user who develops an item and places it in the Encyclopedia volume and the administrator both have all privileges for that item. Table G-2 lists the privileges a user can be granted for BIRT Spreadsheet Designer items in an Encyclopedia volume.

**Table G-2** Privileges

Privilege	Enabled functionality in an Encyclopedia volume
Delete	Remove items from an Encyclopedia volume.
Execute	Run items from the Encyclopedia volume.
Grant	Extend privileges for a specific item in the volume to other users.
Read	Open, work with, print, and download an item in the volume.
Secure read	Read secure parts of a report in the Encyclopedia volume. To use secure read, the user must have BIRT SmartSheet Security option available and assigned.
Visible	View items in the Encyclopedia volume.
Write	Place and modify an item in a location in an Encyclopedia volume.

**Related terms**

Actuate BIRT SmartSheet Security option, administrator, data source, Encyclopedia volume, object, security role

**procedure** A set of commands, input data, and statements that perform a specific set of operations. For example, functions, methods, and subroutines are all procedures.

**Related terms**

data, function, method, statement

**property** A characteristic of an item that controls its appearance and behavior. For example, in BIRT Spreadsheet Designer, use the pages in the Workbook Properties window or use code to specify values for properties. For example, a report developer can specify a font size for a label or the user name and password for a database connection.

**Related terms**

Actuate BIRT Spreadsheet Designer, database connection, font, user name, value

**Contrast with method****publish**

- 1 To copy files to a shared folder to make them available to report users and developers.
- 2 In BIRT Spreadsheet Designer, to upload files to an Encyclopedia volume to make them available to users. Users can run published report executable files.

**Related terms**

Actuate BIRT Spreadsheet Designer, Encyclopedia volume, report executable file

**query**

A statement specifying the data rows to retrieve from a data source. For example, a query that retrieves data from a database typically is a SQL SELECT statement.

**Related terms**

data, data row, data source, database, SQL SELECT statement

**Query by Example (QBE)**

A syntax used to write expressions that specify data to retrieve from a data source. For example, you can use a QBE expression to specify an ad hoc parameter value. The query is modified based on the QBE expression. Figure G-25 shows QBE expressions.

Parameters

☐ Customer Parameters

Credit Rank ☒ A ☐ B ☐ C

Customer Name <H [ ]

Purchase Frequency [ ] [ ]

Purchase Volume >100 [ ]

QBE expressions for ad hoc parameters

**Figure G-25** Query by Example expressions

**Related terms**

ad hoc parameter, data, data source, expression, query, syntax, value

**query editor**

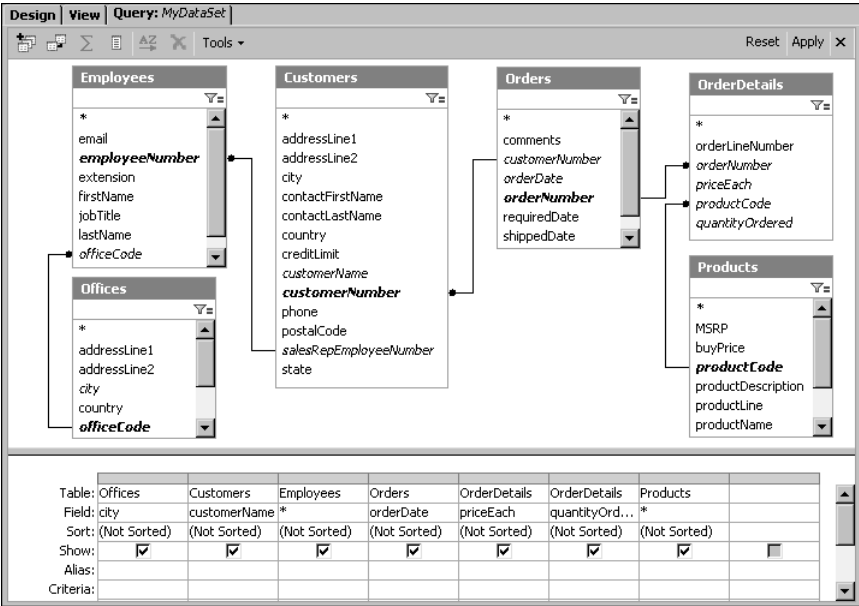
A tool used to write a statement that retrieves data from a data source. Figure G-26 shows the graphical query editor in BIRT Spreadsheet Designer. The upper pane supports selecting tables and specifying the joins between those tables. The lower pane displays selected columns.

**Related terms**

Actuate BIRT Spreadsheet Designer, data, data source, join, statement, table

**Contrast with**

SQL SELECT statement, textual query editor (TQE)



**Figure G-26** Query editor in e.Report Designer Professional

**query parameter**

See parameter.

**query synchronization**

In BIRT Spreadsheet Designer, the process of updating the query so that it is consistent with a list of source parameters.

**Related terms**

Actuate BIRT Spreadsheet Designer, parameter, query

**range**

- 1 A continuous set of values of any data type. For example, 1–31 is a numeric range.
- 2 The distance between the start and end values of the *x*-axis in a chart.
- 3 In BIRT Spreadsheet Designer, two or more cells. A string of characters identifies the location of a range, for example, A1:C3. If the starting and ending worksheets are different, the range is three-dimensional.

**Related terms**

Actuate BIRT Spreadsheet Designer, cell, chart, data type, string, three-dimensional range, value, worksheet

**Contrast with**

data range, tick interval

**read privilege**

See privilege.

**relational database management system (RDBMS)**

A database management system (DBMS) that organizes data into tabular record structures consisting of rows and columns that can be linked together by a common column. An RDBMS typically uses Structured Query Language (SQL) to enable selective retrieval, manipulation, and storage of data by an application.

**Related terms**

application, column, data, data row, database, database management system (DBMS), Structured Query Language (SQL), table

**relative cell reference**

In a BIRT Spreadsheet formula, a reference to a cell by its position in relation to the cell that contains the formula. If the position of the cell that contains the formula changes, the reference changes to match. For example, if you copy a relative reference in cell B2 to cell B3, it automatically adjusts from =A1 to =A2.

**Related terms**

Actuate BIRT Spreadsheet Designer, cell, cell reference, formula

**report**

A category of documents that presents formatted and structured content from one or more data sources, such as a database or text file. A sample spreadsheet report appears in Figure G-27.

	A	B	C	D	E	F	G
1	1900						
2		Controller					
3			description	itemcode	quantity	pricequote	
4			32 bit Programmable Controller with LCD Driver	MP1632x	13	221	
5			32 bit Programmable, Embedded Controller, 3.3v	MPL1632	13	303	
6			4 bit Programmable, Embedded Controller	MP1604	13	21	
7			8 bit Programmable Controller with 4M Static Ram	MP1608s	13	69	
8			8 bit Programmable Controller with LCD Driver	MP1608x	13	48	
9					65	\$662	
10							
11		Driver					
12			description	itemcode	quantity	pricequote	
13			32 bit Programmable Video Graphics Driver, 3.3v	MVL1632	13	150	
14			64 bit Programmable Video Graphics Driver, 3.3v	MVL1664	13	320	

**Figure G-27** Spreadsheet report

**Related terms**

data source, database, format, structured content

**report document**

A file in an Encyclopedia volume or file system that contains a report.

**Related terms**

Encyclopedia volume, report

**Contrast with**

file types

**report executable file**

A file that contains instructions for generating a spreadsheet report.

**Related term**

spreadsheet report

**Contrast with**

file types, spreadsheet object executable (.sox) file

**report parameter**

A report element that enables a user to provide a value as input to the execution of the report. Using a parameter to customize a report provides more focused information to meet specific needs. For example, parameters support selecting sales information by country and city.

**Related terms**

parameter, report, value

**Contrast with**

cascading parameters

**report script function**

A keyword used in a BIRT Spreadsheet Designer worksheet to perform a specific calculation in a data range. The report script function can evaluate a combination of data fields, filters, and simple expressions to specify and arrange the values that appear in the spreadsheet report. A data range cell or section can contain report script using multiple report script functions.

**Related terms**

Actuate BIRT Spreadsheet Designer, cell, data, data field, data range, expression, filter, function, keyword, spreadsheet report, value, worksheet

**repository** A location for rich information storage. In Actuate BIRT iServer System, the repository is an Encyclopedia volume.

**Related terms**

Actuate BIRT iServer System, Encyclopedia volume

**requester** A tool in BIRT Spreadsheet Designer that provides input or modifies parameters. Generating a spreadsheet report from a report executable file uses the values of these parameters.

**Related terms**

Actuate BIRT Spreadsheet Designer, parameter, report executable file, spreadsheet report

**Contrast with**

Parameters page

**reserved word**

See keyword.

**resource** An application component, such as a class, configuration file, image, library, or template.

**Related terms**  
application, class, configuration file, image

**result set** Data rows from an external data source. For example, the data rows that are returned by a SQL SELECT statement issued to a relational database are a result set. A stored procedure can return one or more result sets.

**Related terms**  
data row, data source, database, SQL SELECT statement, stored procedure

**right outer join**  
See outer join.

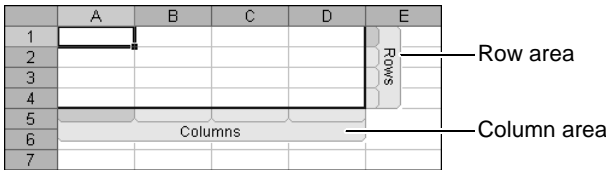
**role** See security role.

**row** **1** A record in a table.  
**2** A horizontal sequence of cells in a BIRT Spreadsheet Designer worksheet.

**Related terms**  
cell, table, worksheet

**Contrast with**  
data row

**row area** In BIRT Spreadsheet Designer, an area used to identify the values for each row section in a data range. The row area occupies the far right column in the template area, as shown in Figure G-28.



**Figure G-28** Row area

**Related terms**  
Actuate BIRT Spreadsheet Designer, column, data range, row, value

**Contrast with**  
column area

**row field** In a BIRT Spreadsheet Designer pivot range, a field that determines what data appears in rows.

**Related terms**  
Actuate BIRT Spreadsheet Designer, data, field, pivot range, row

**run** **1** To execute a program, utility, or other machine function.

- 2 To request current data in a new instance of a spreadsheet report. For example, run a spreadsheet object executable (.sox) file to generate a new spreadsheet report.

**Related terms**

data, spreadsheet report, spreadsheet object executable (.sox) file

**Contrast with**

spreadsheet object instance (.soi) file

**run time**

The period of time in which a computer program executes. For example, a report executable generates a report during run time.

**Related terms**

report, report executable file

**Contrast with**

design time, view time

**running aggregate**

An expression that calculates a value in one pass through the data rows in a data source. For example, an expression that calculates the sum of all orders in a group of data rows is a running aggregate.

**Related terms**

data row, data source, expression, value

**schema**

- 1 A database schema specifies the structure of database components and the relationships among those components. The database components are items such as tables.
- 2 An Extensible Markup Language (XML) schema defines the structure of an XML document. An XML schema consists of element declarations and type definitions that describe a model for the information that a well-formed XML document must contain. The XML schema provides a common vocabulary and grammar for XML documents that support exchanging data among applications.

**Related terms**

application, data, database, declaration, element, Extensible Markup Language (XML), object, table

**scope**

The parts of a program in which a symbol or object exists or is visible. The location of an item's declaration determines its scope. Scopes can be nested. For example, a method introduces a new scope for its parameters and local variables. A class introduces a scope for its member variables, member functions, and nested classes. Code in a method in one scope has visibility to other symbols in that same scope and, with certain exceptions, to symbols in outer scopes.

**Related terms**

class, declaration, function, method, object, parameter, variable

## SDK (Software Development Kit)

A collection of programming tools, utilities, compilers, debuggers, interpreters, and application programming interfaces (API) that a developer uses to build an application to run on a specified technology platform. For example, the Java SDK supports developers in building an application that users can download to run on any operating system. The Java Virtual Machine (JVM), the Java SDK interpreter, executes the application in the specified software and hardware configuration. BIRT Spreadsheet Engine provides an SDK.

### Related terms

Actuate BIRT Spreadsheet Engine, application, application programming interface (API), Java, Java Virtual Machine (JVM)

**search** To find a string in a document.

## secure read privilege

See privilege.

## security ID

An identifier such as an assigned name, a security role, or a user name specified in a spreadsheet report by the report designer to restrict or support access to report components using an access control list (ACL).

### Related terms

access control list (ACL), report, security role, spreadsheet report, user name

## security role

A name for a set of privilege levels. Assigning a security role to a user defines the user's privileges.

### Related term

privilege

### Contrast with

security ID, user name

**SELECT** See SQL SELECT statement.

**select** To highlight one or more items in a user interface, such as a dialog box or a layout editor. Figure G-29 shows selected cells in a spreadsheet design in the layout editor in BIRT Spreadsheet Designer.

	A	B	C	D	E	F	G
1	<b>Classic Models, Inc.</b>						
2	Revenue & Profit by Product		#write(year(orderDate))				
3			Revenue	Profit	Qty	Avg Price	Avg Profit Ea
4	ALL PRODUCTS		#formula("#formula("#formula("#IF(E4=0,"=IF(E4=0,"",D4				
5	#write(productLine)		#formula("#formula("#formula("#IF(E5=0,"=IF(E5=0,"",D5				
6	#write(productName)		#sum(reve	#sum(profi	#sum(quar	#IF(E6=0,"=IF(E6=0,"",D6	

Selected cells

**Figure G-29** Six selected cells

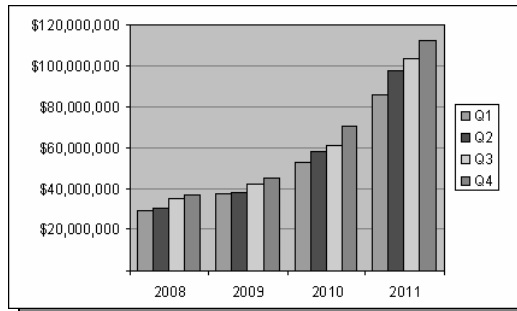


**Related terms**

Actuate BIRT Spreadsheet Designer, cell, design, interface, layout editor

**series**

A sequence of related values. In a chart, for example, a series is a set of related points. Figure G-30 shows a bar chart that displays a series of quarterly sales revenue figures over four years.



**Figure G-30** Series in a chart

**Related terms**

chart, value

**Contrast with**

category

**SmartSheets**

See Actuate BIRT SmartSheet Security option.

**SOD**

See spreadsheet object design (.sod) file.

**SOI**

See spreadsheet object instance (.soi) file.

**sort**

To specify the order in which data is processed or displayed. For example, customer names can be sorted in alphabetical order.

**Related term**

data

**sort key**

An expression used to sort data. For example, if you sort data by country, the country field is a sort key. You can sort data using one or more sort keys.

**Related terms**

data, expression, field, sort

**SOX**

See spreadsheet object executable (.sox) file.

**spreadsheet**

See spreadsheet report or worksheet.

### spreadsheet object design (.sod) file



A file containing a spreadsheet report design. A spreadsheet report developer uses BIRT Spreadsheet Designer to produce a spreadsheet object design file.

**Related terms**

Actuate BIRT Spreadsheet Designer, design, spreadsheet report

**Contrast with**

Actuate BIRT Spreadsheet Engine, file types, spreadsheet object executable (.sox) file

### spreadsheet object executable (.sox) file



A compressed file that contains a single spreadsheet file, either a spreadsheet object design (.sod) file or a Microsoft Excel (.xls) file, and other files, such as properties files and VBA template files used in a spreadsheet report design. Spreadsheet object executable files output spreadsheet reports as Microsoft Excel (.xls) files when published and run on Actuate BIRT iServer System having BIRT spreadsheet option enabled.

**Related terms**

Actuate BIRT iServer System, Actuate BIRT Spreadsheet Designer, Actuate BIRT Spreadsheet option, design, file types, publish, report, run, spreadsheet object design (.sod) file, spreadsheet report

### spreadsheet object instance

The data and formatting that represents a particular report.

**Related terms**

data, format, report

**Contrast with**

spreadsheet object instance (.soi) file

### spreadsheet object instance (.soi) file



A file that contains a viewable spreadsheet report.

**Related terms**

spreadsheet report

**Contrast with**

file types, spreadsheet object instance

### spreadsheet report

A report document created using BIRT spreadsheet technology that contains formatted and structured content. Typically, a user views a spreadsheet report as a Microsoft Excel spreadsheet. Figure G-31 shows a spreadsheet report.

**Related terms**

Actuate BIRT Spreadsheet technology, format, report document, structured content

**Contrast with**

report

	A	B	C	D	E	F	G
1	1900						
2		Controller					
3			description	itemcode	quantity	pricequote	
4			32 bit Programmable Controller with LCD Driver	MP1632x	13	221	
5			32 bit Programmable, Embedded Controller, 3.3v	MPL1632	13	303	
6			4 bit Programmable, Embedded Controller	MP1604	13	21	
7			8 bit Programmable Controller with 4M Static Ram	MP1608s	13	69	
8			8 bit Programmable Controller with LCD Driver	MP1608x	13	48	
9					65	\$662	
10							
11		Driver					
12			description	itemcode	quantity	pricequote	
13			32 bit Programmable Video Graphics Driver, 3.3v	MVL1632	13	150	
14			64 bit Programmable Video Graphics Driver, 3.3v	MVL1664	13	320	

**Figure G-31** Spreadsheet report

## SQL (Structured Query Language)

A language used to access and process data in a relational database.

### Related terms

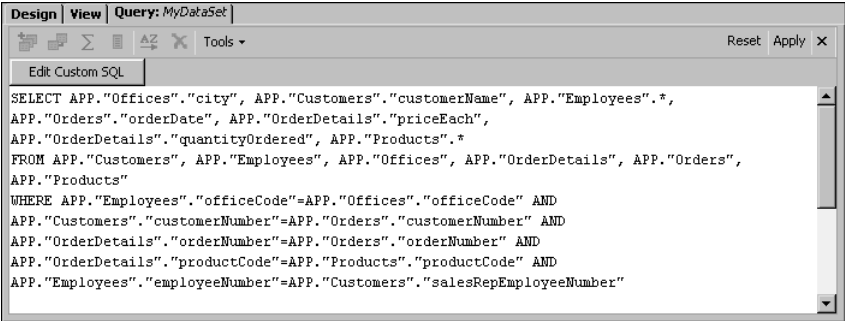
data, database

### Contrast with

query, SQL page, SQL SELECT statement

## SQL page

A page in a query editor that displays the SQL SELECT statement. For example, Figure G-32 shows the SQL page in BIRT Spreadsheet Designer.



**Figure G-32** SQL page in the BIRT Spreadsheet Designer query editor

### Related terms

Actuate BIRT Spreadsheet Designer, page, query editor, SQL SELECT statement

### Contrast with

Structured Query Language (SQL)

## SQL SELECT statement

A query statement in SQL (Structured Query Language) that provides instructions about the data to retrieve from a database. For example, the following SQL query accesses a database's customers table and retrieves the customer name and credit rank values where the credit rank is less than or equal to B. The SQL query then sorts the values by credit rank and customer name.

```
SELECT    customers.customName, customers.creditrank
FROM      customers
WHERE     customers.creditrank <= 'B'
ORDER BY  customers.creditrank, customers.customName
```

**Related terms**

data, database, query, report, sort, SQL (Structured Query Language), statement, table

**Contrast with**

query editor, SQL page

**statement** A syntactically complete unit in a programming language that expresses one action, declaration, or definition.

**Related term**

declaration

**Contrast with**

SQL SELECT statement

**static parameter**

A variable for which an end user can set an initial value when generating a report. The parameter value affects the report output. For example, if the value of a Summary parameter is True, then the report is a summary. If the value is False, then the report shows all the details. A report design can also use a static parameter to filter data rows returned by the data source. In this use, the value must match the syntax required by the data source's query language and must not contain Query by Example (QBE) syntax.

**Related terms**

data row, data source, filter, parameter, query, Query by Example (QBE), report, syntax, value, variable

**static variable**

A variable shared by all instances of a class and its descendant classes. In Java, a static variable is known as a class variable. The compiler specifies the memory allocation for a static variable. The program receives the memory allocation for a static variable as the program loads.

**Related terms**

class, class variable, descendant class, Java, variable

**stored procedure**

A named set of one or more SQL (Structured Query Language) queries that is stored in a database and can be called from an application.

**Related terms**

application, database, query, SQL (Structured Query Language)

**string** A sequence of characters, for example 'Hello world'.

**Related term**

character

**string data type**

A data type that consists of a sequence of contiguous characters including letters, numerals, punctuation marks, and spaces.

**Related terms**

character, data type, string

**Contrast with**

string expression

**string expression**

An expression that evaluates to a series of contiguous characters. Parts of the expression can include a function that returns a string, a string constant, a string literal, a string operator, or a string variable. For example, "abc"+"def" is a string expression that evaluates to "abcdef".

**Related terms**

character, constant, expression, function, operator, string, variable

**Contrast with**

string data type

**structured content**

A formatted document that displays information from one or more data sources.

**Related terms**

data source, format

**Contrast with**

report

**Structured Query Language (SQL)**

See SQL (Structured Query Language).

**study chart**

In a spreadsheet report, the part of a chart that appears below the base chart. A study chart uses a different set of axes from the base chart and typically displays different types of data.

**Related terms**

base chart, chart, data, spreadsheet report

**Contrast with**

dual y-axis chart, combination chart

**style**

A named set of formatting characteristics, such as alignment, borders, color, and font that report developers apply to a report item to control its appearance.

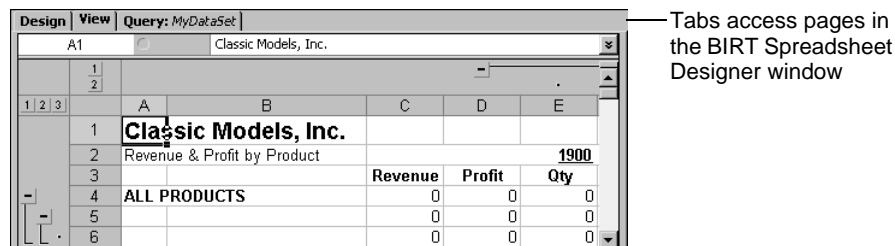
**Related terms**

font, format, report

**syntax**

The rules that govern the structure of a language.

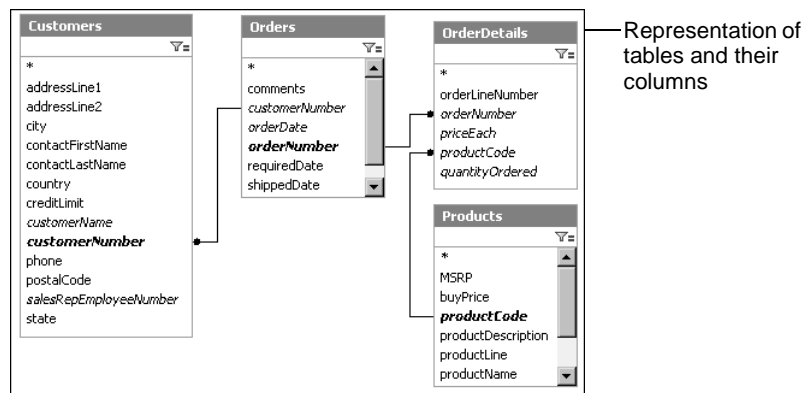
**tab** The label above or below a page in a window that contains multiple pages. Figure G-33 shows tabs that access different pages in the BIRT Spreadsheet Designer window.



**Figure G-33** Tabs in the BIRT Spreadsheet Designer window

**Related terms**  
Actuate BIRT Spreadsheet Designer, page

**table** A named set of columns in a relational database. Figure G-34 shows tables in the query editor in BIRT Spreadsheet Designer.



**Figure G-34** Tables in the BIRT Spreadsheet Designer query editor

**Related terms**  
Actuate BIRT Spreadsheet Designer, column, database, query editor

**tag** An element in a markup language that identifies how to process a part of a document.

**textual query editor (TQE)**  
A textual tool used to write a SQL SELECT statement.

**Related term**  
SQL SELECT statement

**Contrast with**  
query editor

**three-dimensional range**

In BIRT Spreadsheet Designer, a range that spans different worksheets. This range includes the same rows and columns on those multiple worksheets. The range must be within the same workbook.

**Related terms**

Actuate BIRT Spreadsheet Designer, column, range, row, workbook, worksheet

**tick**

A marker that occurs at regular intervals along the *x*- or *y*-axis of a chart. Typically, the value of each tick appears on the axis.

**Related terms**

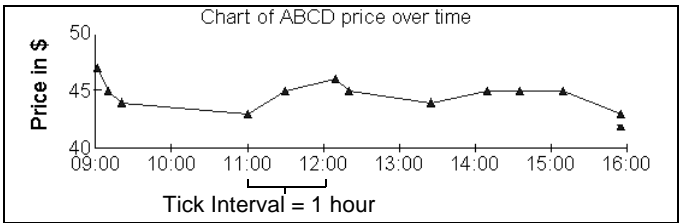
chart, value

**Contrast with**

grid, tick interval

**tick interval**

The distance between ticks on an axis. Figure G-35 shows a tick interval in a chart.



**Figure G-35** Chart displaying multiple tick intervals

**Related terms**

chart, tick

**toolbar**

A user interface component that provides access to common tasks. Different toolbars are available for different kinds of tasks. Figure G-36 shows the Formatting toolbar from BIRT Spreadsheet Designer.



**Figure G-36** BIRT Spreadsheet Designer Formatting toolbar

**Related terms**

Actuate BIRT Spreadsheet Designer, interface

**trusted execute privilege**

See privilege.

**type**

See data type.

**Unicode**

A living language standard managed by the Technical Committee of the Unicode Consortium. The current Unicode standard provides code points for more than

65,000 characters. Unicode encoding has no dependency on a platform or software program and thus provides a basis for software internationalization.

**Related terms**

character, internationalization

**uniform resource locator (URL)**

A character string that identifies the location and type of a piece of information that is accessible over the web. `http://` is the indicator that an item is accessible over the web. The URL typically includes the domain name, type of organization, and a precise location within the directory structure where the item is located.

**Related terms**

character, HyperText Transfer Protocol (HTTP), string

**URL**

See uniform resource locator (URL).

**user name**

A name that identifies a user of a resource on a computer system. For example, Management Console supports user names having the following characteristics: having up to 256 characters that are not control characters, not being case-sensitive, and optionally using space characters.

**Related terms**

case sensitivity, character, Management Console

**value**

- 1 The content of a constant, parameter, symbol, or variable.
- 2 A specific occurrence of an attribute. For example, blue is a possible value for an attribute color.

**Related terms**

constant, parameter, variable

**Varchar data type**

An Actuate SQL data type used for string calculations. The Varchar data type stores a sequence of Unicode characters. The Varchar data type supports specifying a maximum character length for the string. For example, `VARCHAR (30)` represents strings with a maximum length of 30 Unicode characters.

**Related terms**

Actuate SQL, character, data type, string, Unicode

**variable**

A named storage location for data that a program can modify. Each variable has a unique name that identifies it within its scope and contains a certain type of data.

**Related terms**

data, data type, scope

**Contrast with**

class variable, global variable, object reference variable



**variable hyperlink**

In a spreadsheet report design, a hyperlink that can change to reflect report data. When report developers add a variable hyperlink to a cell, they can include dynamic field and parameter information. For example, a variable hyperlink can be used to create a link that generates an e-mail address for each client.

**Related terms**

cell, data, design, field, hyperlink, parameter, report, spreadsheet report

**view**

**1** A predefined query that retrieves data from one or more tables in a relational database. Unlike a table, a view does not store data. Users can use views to select, delete, insert, and update data. The database uses the definition of the view to determine the appropriate action on the underlying tables. For example, a database queries a view by combining the requested data from the underlying tables.

**2** A window that displays a preview of a report.

**Related terms**

data, database, query, report, table

**view time**

The period of time in which a user examines a report.

**Related term**

report

**Contrast with**

design time, run time

**viewer**

A tool that supports viewing a report.

**Related term**

report

**virtual cell reference**

In a spreadsheet report design, a cell in the report design that expands to include multiple data cells in the report.

**Related terms**

cell, data, design, report, spreadsheet report

**Contrast with**

absolute cell reference, relative cell reference

**virtual defined name**

See defined name.

**visible privilege**

See privilege.

**volume**

See Encyclopedia volume.

**VTF**

See spreadsheet object executable (.sox) file.

**VTX** See spreadsheet object design (.sod) file.

**VTX** See spreadsheet object design (.sod) file.

**web archive (.war) file**

A file format used to bundle web applications.

**Related terms**

application, format

**Contrast with**

file types, Java archive (.jar) file

**web page** A HyperText Markup Language (HTML) page containing tags that a web browser interprets and displays.

**Related terms**

HyperText Markup Language (HTML), tag

**Contrast with**

page

**workbook** A spreadsheet object design (.sod) file that contains one or more worksheets.

**Related terms**

spreadsheet object design (.sod) file, worksheet

**worksheet** In BIRT spreadsheet technology, a document that supports storing and working with data. A worksheet is also called a spreadsheet. A worksheet consists of cells organized into columns and rows. A worksheet is stored in a workbook.

**Related terms**

Actuate BIRT Spreadsheet technology, cell, column, data, row, workbook

**write privilege**

See privilege.

**XML (Extensible Markup Language)**

See Extensible Markup Language (XML).

**XML schema**

See schema.

# Index

## Symbols

- : (colon) character
  - parameter names and 330
  - section references and 239
  - sheet references and 240
- :? in query statements 90
- :? operator 112
- ? wildcard 438
- ... button 601
- ' (single quotation mark) character
  - cell references and 240
  - report functions and 434
- " (double quotation mark) character
  - report functions and 434
  - text expressions and 434
- ( ) parentheses characters
  - creating defined names and 236
  - grouping expressions and 224
- [ ] square brackets characters
  - cell references and 240
  - report functions and 434, 435
- { } curly brackets characters 435, 436
- \* (asterisk) character 326
- \* operator 436
- \* symbol in query editor 82
- \* wildcard 438
- / character in file names 402
- / operator 436
- // characters in cells 439
- & (ampersand) character 434
- # (number sign) character
  - hyperlinks and 391
  - report functions and 214, 434
- #hyperlink function 394, 395, 396
- + operator 436, 437
- < operator 123, 436
- <> operator 123
- <= operator 123, 436
- <> operator 436
- = (equal sign) character 320
- = operator 118, 123, 436
- > operator 123, 436
- >= operator 123, 436

- || operator 130
- ~ wildcard 438
- operator 436, 437
- \$ (dollar sign) character 238

## Numerics

- 3D chart types 251
  - See also* charts with depth

## A

- ABS function 555
- absolute cell references 238, 240, 581
  - See also* cell references
- absolute index numbers 367
- absolute paths 391
- AC\_EXECUTABLE\_ENCYCLOPEDIA\_PATH parameter 396
- AC\_LAST\_REPORT\_RUN\_TIME parameter 396
- AC\_REPORT\_OUTPUT\_PATH parameter 403
- access control lists 380, 464, 581
  - See also* security IDs
- access permissions. *See* privileges
- access privileges. *See* privileges
- access restrictions 40, 380
- access types 581
- accessing
  - data 70, 103, 158
  - databases 50, 70
  - Encyclopedia volumes 104
  - expression builder 106, 115
  - external data sources 50, 51
  - information objects 101
  - JDBC drivers 70, 74
  - Prompt editor 135, 147
  - query editors 106, 153
  - SQL editor 83
- accessing protected cells 375
- accounts 105
- ACCRINT function 555, 576
- ACCRINTM function 555

- ACLs. *See* access control lists
- ACOS function 555
- ACOSH function 555
- actions, undoing 412
- activating hyperlinks 392
- Actuate Reporting System. *See* BIRT iServer System
- Actuate server. *See* iServer
- Actuate SmartSheet Security option. *See* BIRT SmartSheet Security option
- Actuate SQL 585
  - See also* queries; SQL statements
- Actuate SQL data types 116
- Actuate SQL expressions 106, 151
- Actuate SQL join algorithms 120
- Actuate SQL parameters 145
- Actuate SQL queries 153
- ad hoc data access 100
- ad hoc parameters 330, 586
  - adding to queries 90
  - assigning values to 89
  - creating 90
- Add Data Range Sections dialog 280
- add-in functions 578
- adding
  - aggregate functions 86
  - borders to chart areas 286, 289
  - calculated fields 356
  - calculated items 357
  - cell references 238
  - chart titles 265, 277, 284
  - charts 522
  - charts to designs 266
  - columns
    - to queries 81, 82
  - columns to queries 10, 19, 81, 82, 93
  - columns to reports 39, 180
  - comments 439, 536
  - computed fields 58–60, 84–85
  - data ranges 169, 184
  - data sets 104
  - defined names 236–238, 326, 459
  - drop lines 305
  - dynamic filters 134, 136
  - expressions to queries 106
  - fields to reports 25, 42, 173, 212
  - filter conditions 122–132
  - functions 435, 438
  - functions to expressions 106
  - high-low lines 305
  - hyperlinks 392, 394, 395
  - Java classes 489
  - join algorithms 120, 121
  - join conditions 117, 118, 119
  - macros 245–247
  - matrix ranges 530
  - outlines 201, 203–206
  - page breaks 370, 461
  - parameters to groups 328
  - parameters to queries 89, 90, 110, 145–146, 151
  - parameters to reports 318, 319–321
  - passwords 374
  - pivot ranges 171, 336–337, 354, 524
  - reference lines 306
  - report titles 30
  - rows to reports 39, 180
  - section references 239
  - security IDs 380, 383
  - summary values 85
  - tables to queries 9, 17, 77, 93
  - textual content 244–245, 466, 474
  - trendlines to charts 314
  - up or down bars 305
  - validation rules 247
  - VBA functionality 536, 539, 541
- addition operator 436, 437
- ADDRESS function 555
- addresses (e-mail) 390, 393
- administrators 586
- Advanced Design perspective 106, 107, 111
- aggregate columns 139, 140, 143, 144
- aggregate expressions 59, 86, 226, 241, 586
  - defined 586
- aggregate function icon 86
- aggregate functions 139, 587
  - adding to data ranges 435
  - adding to queries 86
  - compatibility with Excel 578
  - creating grant expressions and 382
  - defined 587
- aggregate rows 85, 587
- Aggregate Type property 116
- aggregate values 435, 587

- defined 587
- aggregation tools 226
- aliases 587
  - column names 115, 117, 155
  - table names 155
- aligning text 196, 290
- Alignment command 290
- Alignment dialog 290
- Alignment page (Format Cells) 196
- allocating memory 428
- alternate names. *See* aliases; display names
- AMORDEGRC function 555, 576
- AMORLINC function 555
- ampersand (&) character 434
- AND function 555
- AND operator 117, 130, 131
- and operator 436
- APIs. *See* application programming interfaces
- application programming interfaces (APIs) 478, 498, 587
- application window 5, 574
  - See also* BIRT Spreadsheet Designer
- applications 587
  - defined 587
  - deploying BIRT Spreadsheet Designer and 548
  - developing 478
  - loading 428
  - opening reports from 390
- area charts
  - See also* charts
  - creating 252
  - plotting percentages in 295
  - plotting values for 276
  - stacking series for 294
- AREAS function 555
- arguments 578, 588
  - See also* functions; parameters
- arithmetic functions 576
- arithmetic operators 436, 437
- array formulas 578
- arrays 436, 588
  - defined 588
- ASC function 555
- ascending sort order 94, 437
- ASCII characters 404
- ASCII files. *See* text files
- ASIN function 556
- ASINH function 556
- asterisk (\*) character 326, 438
- ATAN function 556
- ATAN2 function 556
- ATANH function 556
- attributes 319
  - See also* text attributes
- auto size function 182
- Auto suggest option 138
- Automatic Layout command 288
- automatic recalculation 425
- autoshares 536
- AutoSort options 365, 366
- AVEDEV function 556
- AVERAGE function 556
- average function 230, 442
- AVERAGEA function 556
- averages 86, 230, 442, 538
- AverageVBA.sod 539
- AverageVBA.sod.xls 540
- AverageVBA.xls 540
- avg function 86
- Axes command 306, 310
- axes values
  - See also* category axis; value axis
  - adding reference lines to 306
  - formatting 265, 306
  - hiding 306
  - intersecting 307
  - joining 251
  - marking segments of 251
  - plotting 274, 275
  - relating to data series 304
  - removing reference lines 306
  - reversing 309
  - scaling 307
  - setting frequency options for 308
  - setting interval options for 308, 309, 310
- Axis dialog 293
- axis labels
  - adjusting spacing between 308
  - aligning text in 290
  - displaying 250
  - formatting numeric series in 291
  - hiding 307
  - setting font attributes for 287

axis lines 306  
axis titles 250, 284, 290

## B

background colors 444  
background patterns 196  
backing up configuration files 428  
BAHTTEXT function 551  
balloon help 588  
bar charts  
    *See also* charts  
        creating 252  
        displaying multiple series in 297  
        formatting data series for 295  
        plotting percentages in 295  
        plotting values for 276  
        setting bar spacing for 296  
        setting series lines in 297  
        stacking series for 294  
        using multiple series 269  
bars (charts) 251, 293, 297  
base charts 588  
base unit (charts) 588  
base units (charts) 310  
Basic Design perspective 105, 107  
BESSELI function 556  
BESSELJ function 556  
BESSELK function 556  
BESSELY function 556  
BETADIST function 556  
BETAINV function 556  
BETWEEN operator 123  
BIN2DEC function 556  
BIN2HEX function 556  
BIN2OCT function 556  
binding VBA code to spreadsheets 541  
BINOMDIST function 556  
BIRT Information Designer. *See* IO Design perspective  
BIRT Information Object Designer. *See* IO Design perspective  
BIRT iServer 582  
    accessing connection configurations and 404, 409  
    accessing information objects and 101  
    accessing reports and 380, 398

    creating profiles for 399–403  
    debugging callback classes on 486  
    deploying callback classes to 483  
    installing JDBC drivers for 70  
    logging error and debugging messages for 483, 484  
    publishing to 399, 400, 537  
    renaming files and 402  
    selecting profiles for 401  
    uploading files to 398  
BIRT iServer Express 488  
BIRT iServer Profile dialog 399, 400  
BIRT iServer property 104  
BIRT iServer System 331, 398, 485, 582  
BIRT iServer System Options (licensing) 582  
BIRT Java Components 582  
BIRT SmartSheet Security option 582  
BIRT Spreadsheet API 478, 479, 496, 583  
BIRT Spreadsheet API Javadoc 479, 497  
BIRT Spreadsheet Deployment Kit 583  
BIRT Spreadsheet Designer 103, 104, 583  
    changing language for 550  
    changing regional settings for 415, 418, 419, 549  
    closing 485  
    compared to Excel 574–579  
    compatibility with Lotus 1-2-3 425  
    custom drivers and 50  
    customizing 478  
    deploying 546, 547, 548  
    deploying callback classes to 482  
    formatting options in 190, 195  
    limitations for 550  
    logging error and debugging messages for 483, 484  
    optimizing 428  
    overriding locale settings for 547  
    overview 4, 5  
    personalizing 412  
    restarting 485  
    running reports on 485  
    running translated versions of 547  
    security options for 40, 372  
    setting preferences for 413, 415  
    supported data sources 50  
    supported languages for 548  
    updating configurations for 428, 429

- VBA code names and 541, 542
- BIRT Spreadsheet Engine 583
- BIRT Spreadsheet Engine and API 584
- BIRT Spreadsheet option 584
- BIRT Spreadsheet reports. *See* spreadsheet reports
- BIRT Spreadsheet server. *See* BIRT iServer
- blank cells 288, 355, 421, 576
- blank lines 345
- blank values 126, 127, 326
- block totals 335, 360, 362
- bookmarks 391
- BookModel class 497
- BookModel objects 479, 496, 497
- books. *See* workbooks
- Boolean expressions 201, 224, 455, 578, 589
- Boolean values 365
- border colors 196
- Border page (Format Cells) 31, 196
- borders 196, 286, 289, 420
  - See also* outlines
- browsers. *See* web browsers
- bubble charts
  - See also* charts
  - creating 253
  - formatting data series for 298
  - labeling data values in 299
  - overlapping bubbles in 299
  - plotting values for 276
  - resizing bubbles 298
  - scaling 299
  - setting bubble size for 298
- Bubble size setting 298
- bursting. *See* sheet-level groups
- buttons 202, 322, 574

## C

- calculated columns. *See* calculated fields; computed fields
- calculated columns. *See* computed columns
- Calculated Field command 356
- calculated fields 589
  - See also* pivot ranges
  - changing expressions in 358
  - creating 356
  - defined 335

- displaying 362
- Calculated Item command 357
- Calculated Item Solve Order dialog 362
- calculated items 589
  - See also* pivot ranges
  - changing evaluation order for 362
  - changing expressions for 358
  - creating 356, 357
  - defined 335
  - displaying 362
- calculated values 25, 377
  - See also* calculations
- calculation options 424–425
- Calculation page (Workbook Properties) 424
- calculations
  - bubble size 298
  - compatibility with Excel 576, 578
  - computed fields and 58
  - customizing 359, 536
  - generating ending values and 363, 364
  - generating lists and 437
  - generating starting values and 363, 364
  - locale-specific reports and 231, 555
  - missing data and 288
  - optimizing 425
  - pivot ranges and 355, 358
  - queries and 94
  - report functions and 213
  - returning date values 445, 446, 459, 475
  - returning standard deviation for 468, 469
  - returning time values and 454, 458, 464
  - returning variance for 474
  - summary data and 226, 232, 435
- CALL function 556, 576
- callback classes 589
  - adding Java classes to 489–492
  - associating with workbooks 482
  - changing 485
  - compiling 481
  - creating 478, 479–481, 496
  - deploying 482–483
  - instantiating Logger objects and 484
  - overview 478
  - specifying location of 483
  - testing 483–488, 498
  - viewing sample code for 492, 498
- Callback page (Report Properties) 482

- callback writers 478
- CARDINALITY keyword 119
- cascading parameters 316, 329
- case conversions 457, 473
- case sensitivity 590
- categories (charts) 251
- category 590
- category axis
  - See also* x-axis values
  - adding drop lines to 305
  - adjusting intersection point for 307, 308
  - defining time-scale 310
  - displaying data on 250, 274, 295
  - hiding 306
  - marking segments of 251
  - reversing order of 309
  - scaling 307, 308
- category axis labels 250
  - See also* axis labels
- category axis titles 250, 284
  - See also* axis titles
- Category Path property 116
- category values 590
- category-scale axis 308
- CEILING function 556
- cell (defined) 591
- CELL function 556, 576
- cell references 591
  - See also* absolute cell references; relative cell references
  - adding defined names and 236
  - creating 238
  - defining for multiple sheets or books 239–240
  - deleting data and 180
  - formatting data and 197
  - hiding 420
  - identifying pivot ranges and 367
  - merging cells and 192
  - returning 443
- CellFormatting callback example 509–512
- CellFormatting.sod 509
- CellLocking callback example 507–509
- cells
  - accessing protected 375
  - adding comments to 439, 536
  - adding formulas to 575
  - adding hyperlinks to 392, 394
  - adding static content to 244, 245
  - clearing content in 181
  - conditionally formatting values in 197–201
  - counting 576
  - defined 591
  - deleting 181
  - displaying data type markers for 421
  - displaying formulas in 377, 427
  - displaying section names for 173
  - displaying zero values in 427
  - embedding hyperlinks in 454
  - entering expressions in 230
  - entering functions in 213, 214, 438
  - formatting data in 33, 450, 509, 514
  - locking 457, 507
  - merging 28, 192–195, 458
  - recalculating values in 424, 425
  - removing hyperlinks in 394
  - removing sections and 179
  - restricting access to 372
  - restricting data entry in 247
  - selecting a hyperlinked cell 392
  - setting calculation options for 424–425
  - shifting 181
  - unlocking 376
- cells function 239, 443
- centering text 33, 349
- changes, undoing 412
- changing
  - bubble size 298
  - callback classes 485
  - chart types 265, 268, 285
  - charts 284
  - code names 542
  - column aliases 115
  - column names 76, 161, 164
  - computed fields 59
  - configuration information 428
  - connections 53
  - data 39
  - data ranges 169, 178, 184, 281
  - data source connections 403, 409
  - data sources 51, 354, 403
  - data types 161, 164
  - default colors 421



- default data set 65
- doughnut start angles 301
- encryption type 379
- evaluation rules 426
- file names 402, 403
- filter conditions 133
- formulas 227
- group keys 179
- hyperlinks 393
- information object connections 102
- joins 92
- language settings 550
- pie start angles 302
- pivot range names 343
- pivot ranges 335, 343
- prompt properties 151
- queries 103, 153, 154, 502
- regional settings 415, 418, 419, 549
- report output type 413
- result sets 57
- sample data 277, 283
- section names 177, 179
- sections 179
- sort order 176
- source parameters 152
- spreadsheet reports 483
- SQL queries 64, 91
- stored procedures 98
- system-level parameters 331
- trendlines in charts 314
- worksheet names 541
- worksheet properties 426
- x-axis settings 306
- y-axis settings 306
- CHAR function 556, 576
- character attributes. *See* text attributes
- character sets 576, 592
- character strings 635, 638
- character strings. *See* strings
- Characterdelimiter property 406
- characters
  - See also* escape characters; text
  - adding comments and 439
  - as decimal separators 127
  - Auto suggest option for 138
  - column aliases and 115
  - converting case of 457, 473

- creating defined names and 236
- defined 591
- delimited text files and 158, 161, 406
- entering formulas and 356, 575
- entering functions and 434
- entering in passwords 374
- entering wildcard expressions and 438
- filter conditions and 125
- fixed-width text files and 158
- getting number of 456
- parameter names and 145
- pattern matching and 128
- predefined connections and 404
- QBE expressions and 136
- translating reports and 549
- translating spreadsheet reports and 571
- chart areas
  - alignment options for 290
  - border options for 286, 289
  - elements in 250, 284, 288
  - fill options for 286, 289
  - font attributes for 287, 290
  - formatting options for 288, 291
- Chart command 277
- chart coordinates. *See* data points
- chart function 443
- chart images 288
- chart objects 262, 277
  - See also* charts
- Chart Options command 289
- Chart Options dialog 289
- chart templates 251
- chart types
  - See also* specific type
  - changing 265, 268, 285
  - displaying multiple 267, 271
  - formatting data series and 295
  - selecting 251, 275, 277
- chart wizard 262, 277–278
- ChartDesign.sox 260
- charts 592
  - adding data series to 282, 291
  - adding descriptive elements to 265
  - adding titles to 265, 277, 284
  - adding to designs 266
  - adding to spreadsheets 522
  - aligning text on 290

## charts (*continued*)

- binding event handlers to 536
- calculating missing data for 288
- changing appearance of 284
- changing data ranges for 281
- changing x-axis settings on 306
- changing y-axis settings on 306
- comparing multiple y-axes values and 259, 292
- comparing series values in 252, 256, 257
- compatibility with Excel 578
- contrasting series values in 254
- creating 250, 260, 274, 277
- defined 592
- displaying hidden data in 287
- formatting 265, 285, 288
- hiding axes values on 306
- hiding grid lines on 306
- highlighting data series for 291, 292
- highlighting multiple values in 253, 254
- linking to 270, 278, 287
- linking to data sources 443, 465
- mapping data ranges to 275, 277, 278, 280
- moving 266
- organizing data for 262, 266
- overlapping series in 252
- plotting maximum or minimum values for 294, 295, 310
- plotting non-numeric data for 288
- previewing 281
- rearranging graphic elements in 285
- refreshing 282
- reordering data series in 294
- resizing 266
- resizing columns or rows and 182
- rotating text in 290
- selecting layout options for 288
- selecting sample data for 282
- selecting source data options for 280
- showing trends with 265, 312
- stacking series values for 294
- tracking stock data and 259
- updating 443
- updating data ranges for 281
- viewing trends in 312

charts with depth 297

check boxes 574

CHIDIST function 556

CHIINV function 556

Chinese translations 546

CHITEST function 556

CHOOSE function 556

circular references 424, 425

class declarations 592

class files 482, 483, 485

class hierarchy 592

class names 480, 592

class paths 481, 548

class variables 593

- See also* instance variables; variables

classes 592

- adding Java 489
- building callback 478, 479–481, 496
- creating grant expressions and 382
- debugging callback 483–488
- defined 592
- deploying callback 482–483
- deploying Java 491
- extending spreadsheet functionality and 478
- importing 480, 490
- viewing public methods for 497

Classic Models database 51

CLASSPATH variable 481

CLEAN function 557

Clear Outlines command 206

clearing outlines 206

clock. *See* time values

Close command 15

close values 259

- See also* stock charts

closing

- BIRT Spreadsheet Designer 485
- report designs 15
- toolbars 414

closing Information Object Query Builder 104

code

- adding VBA functionality and 536, 541
- binding to spreadsheets 541
- creating callback classes and 492, 498
- creating multi-class callbacks and 489, 490
- debugging callback classes and 483, 485
- developing workbooks and 481

- executing at run time 485
  - overriding locale settings and 547
  - testing 537
  - testing and debugging 599
  - triggering 536
- CODE function 557, 576
- code names 541, 542
- code pages 407
- code points 593
- Codepage property 407
- col function 444
- colon (:) character
  - parameter names and 330
  - section references and 239
  - sheet references and 240
- color format symbols 554
- color function 444
- color palette 421
- Color Palette page (Workbook Properties) 421
- color values 444, 450
- colors
  - changing chart 286, 289, 292
  - changing default 421
  - formatting data and 196, 197
  - setting contrasting 292
- column aliases 115, 117, 155
- column areas 593
  - See also* data ranges
- column breaks (text files) 163
- column categories 114, 116
- column charts
  - See also* charts
  - creating 252
  - displaying multiple series in 297
  - formatting data series for 295
  - plotting percentages in 295
  - plotting values for 276
  - setting bar spacing for 296
  - setting series lines in 297
  - stacking series for 294
  - using multiple series 269
- column delimiters (text files) 158, 161, 406
- column fields 594
  - See also* pivot ranges
  - adding 340, 341, 342
  - defined 335
  - hiding 350, 355
  - labeling 349
  - removing 342
  - showing top or bottom values in 366
- COLUMN function 557
- column headings
  - creating 245
  - displaying 427
  - formatting 32
  - spanning multiple columns 192, 193
- column IDs 444
- column letters 181
- column names 118, 339, 381, 434
  - See also* column headings
  - adding special characters to 76
  - changing 76, 161, 164
  - creating computed fields and 85
  - creating joins and 91
  - entering in SQL statements 76
  - setting text file 407
- column outlines 202, 204
- column section area 20
- column sections 19, 22, 39, 173
  - See also* sections
- column stubs 21, 22, 23
- column subcategories 116
- Columnnames property 407
- Columnpositions property 406
- columns 593
  - See also* fields
  - adding
    - to queries 81, 82
  - adding to queries 10, 19, 81, 82, 93
  - adding to reports 39, 180
  - categorizing 114
  - changing order of 93
  - comparing values across multiple 128
  - compatibility with Excel 575
  - defined 593
  - deleting 181
  - deleting from queries 94
  - displaying 108
  - filtering on 89
  - grouping data and 138, 140, 141
  - hiding 94, 181
  - removing from SELECT clause 142, 143
  - renaming 115

- columns (*continued*)
  - reordering multiple 93
  - resizing 182–183
  - restricting retrieval of 94
  - retrieving 41
  - returning from
    - databases 93
    - delimited text files 161
    - fixed-width text files 163
  - selecting 109, 113, 115
  - selecting delimiter characters for 406
  - setting range limits for 427
  - setting width 468
  - sorting data in 94
- COLUMNS function 557
- Columns page (SQL editor) 155
- Columntypes property 407
- COMBIN function 557
- combination chart
  - defined 594
- combination charts 594
  - See also* charts
  - creating 254
  - formatting data series for 299
  - plotting dual y-axis values and 293
  - plotting values for 276
- combo boxes 138, 322
- commands 214
  - See also* functions
- comma-separated values (CSV) files 595
- comments 439, 498, 536
- comparison operator 438
- comparisons 87, 334, 435
  - date-and-time values 127
  - filter conditions and 124, 125, 126, 128
  - numeric values 127
  - string values 127
- Compatibility page (Designer Preferences) 417
- compiling 481
- compiling errors 481
- COMPLEX function 557
- computed columns. *See* computed fields
- computed fields 117, 435, 595
  - changing 59
  - creating 58–60, 84–85
  - defined 595
  - deleting 59
  - displaying 59, 85
  - naming 59, 60
  - See also* calculated fields
- computed values. *See* calculated values; calculations
- CONCATENATE function 557
- concatenation 434
- concatenation operator 130
- Conceal Value property 149
- concrete base class 595
- Condition dialog box 220
- Conditional command 198, 200
- conditional expressions 201, 230, 436
- conditional formats 197–201, 343, 595
  - See also* formats
- Conditional Formats dialog 198, 199, 200
- conditions
  - accessing reports and 41
  - creating defined names and 238
  - creating grant expressions and 382
  - filtering data and 219, 224, 448, 455
  - formatting data and. *See* conditional formats
  - generating summary values and 230
  - grouping data and 363
  - hiding sections and 183
  - joining 222
  - parameters as 89, 90
  - queries and 87
  - retrieving data and 380, 465
  - setting values based on 455
  - sorting data and 366
  - testing 438
- cone charts 276
- CONFIDENCE function 557
- Configuration Console 486
- configuration files 428, 595
  - creating 404, 409
  - setting connection properties in 404, 409
- configurations
  - JDBC drivers 70, 74
  - predefined connections 404, 409
- configuring remote debugging 486, 487
- ConnConfigFile parameter 404
- connection information 71, 404
- Connection property 596

- Connectionproperty property 409
- connections 104, 595
  - accessing
    - information objects and 101
    - JDBC data sources and 70–73, 75
    - text file data sources and 158, 159
  - associating with specific data source 405
  - changing 53, 102, 403, 409
  - configuring 404, 409
  - creating 6, 41, 51, 52, 404
  - generating spreadsheet reports and 50
  - installing ODA drivers and 54
  - overriding 404, 405
  - setting at runtime 404
  - setting properties for 405, 406, 407
  - testing 6, 52, 400
- ConnectOptions element 405
- consolidated ranges 578
- constants 424, 596
- constructor code 596
- content. *See* structured content
- content, planning for 38, 212, 244
- context menus. *See* menus
- control types 137, 150
- conversions
  - compatibility with Excel 576
  - dates 470
  - numeric 471
  - text 457, 472, 473
- CONVERT function 557, 576
- Convert null numbers to zero setting 243
- copying
  - class files 482
  - expressions 245
  - files 398
  - privileges 401
  - report functions 245
- copying SELECT statements 83
- CORREL function 557
- COS function 557
- COSH function 557
- COUNT function 340, 557
- count function 86, 444
- COUNTA function 557
- COUNTBLANK function 557, 576
- COUNTIF function 557
- counting cells 576
- counting non-null values 86
- counting values 444
- country settings 549
- COUPDAYBS function 557, 577
- COUPDAYS function 557, 576, 577
- COUPDAYSSNC function 557, 577
- COUPNCD function 557, 577
- COUPNUM function 558
- COUPPCD function 558, 577
- COVAR function 558
- Create Calculated Field command 59, 60
- Create Cascading Group command 329
- Create Data Range command 19, 170
- Create Data Set command 8, 56, 61
- Create Data Source command 6, 52
- Create Grant Expression command 382
- Create Grant Expression dialog 382
- Create Macro command 246
- Create New Join dialog box 91
- Create Parameter Group command 328
- Create Parent command 20, 22, 173
- Create Report Parameter command 318, 319
- Create Section dialog 21
- CreatePivotRange callback example 524–530
- CreatePivotRange.sod 525
- creating
  - access control lists 380
  - ad hoc parameters 90
  - BIRT iServer profiles 399–403
  - calculated fields 356
  - calculated items 357
  - callback classes 478, 479–481, 496
  - cascading parameters 329
  - cell references 238
  - charts 250, 260, 274, 277, 522
  - column headings 245
  - column sections 22
  - computed fields 58–60, 84–85
  - conditional formats 198
  - configuration files 404, 409
  - connections 6, 41, 51, 52, 404
  - custom drivers 53
  - data filters 219, 220
  - data ranges 12, 19, 42, 170, 173, 231
  - data sets 8, 17, 56, 61, 76, 81, 96, 104
  - defined names 236–238, 326, 459
  - display names 323

## creating (*continued*)

- dynamic data filters 134, 136
- executable files 398
- filter conditions 122–132
- flat file data sources 158, 159
- grant expressions 381–383
- graphical objects 520
- hyperlinks 392, 394, 395
- join algorithms 120, 121
- joins 17, 82, 91, 117–121
- list of values 135, 137, 148
- macros 245–247
- matrix ranges 530
- outlines 201, 203–206
- page breaks 370, 461
- parameter groups 327–328, 329
- parameterized queries 96
- passwords 374
- pivot ranges 171, 336–337, 354, 524
- queries 8, 17, 103, 105, 107, 153
- report designs 38, 43
- report parameters 89, 98, 318, 319–321
- reports 6, 16
- row sections 19
- section headers 194
- section references 239
- sheet-level groups 174, 175, 186
- spreadsheet files 518
- SQL expressions 106
- static parameters 89
- summary data 85
- tables 61
- validation rules 247
- VBA templates 536
  - example for 538–541
- CreatingADataRange callback example 530–533
- credentials 102
- CRITBINOM function 558
- criteria. *See* conditions; parameters
- CUMIPMT function 558
- CUMPRINC function 558
- curly brackets characters 435, 436
- currency formats 33, 191
- currency symbols 550
- currency values 550, 576
- current date 460

- current time 460
- CURRENT\_DATE function 129
- CURRENT\_USER function 130
- custom data drivers 50, 53
- custom data sources 336
- custom data sources. *See* open data access
- customizing
  - application color palette 421
  - BIRT Spreadsheet Designer 478
  - calculations 359, 536
  - currency formats 191
  - data drivers 53, 74
  - regional settings 549
  - reports 316
- customizing queries 111
- CustomNumberFormatting callback
  - example 514–518
- CustomNumberFormatting.sod 515

## D

- data
  - See also* values
  - accessing 70, 103, 158
  - aggregating 139
  - calculating missing 288
  - changing 39
  - combining 334
  - customizing drivers for 53
  - defined 597
  - designing reports and 38
  - displaying 19, 35, 66, 112, 156, 201, 212, 317
  - filtering 110, 121–138, 144, 219, 220, 224, 317, 366, 380
  - formatting 30, 40, 190, 195, 509
  - generating sample 265, 277, 282
  - generating spreadsheet reports and 50, 51
  - grouping 23, 40, 138–144, 174, 362, 452
  - hiding 287
  - identifying type 421
  - laying out 39, 42, 168, 169
  - localizing 549, 550
  - previewing 57
  - refreshing 67
  - retrieving 41, 215, 465
  - retrieving from

- databases 70
- information objects 100, 101
- multiple data sources 51, 61
- text files 160, 162
- searching for 233, 438
- setting restrictions for 247
- sorting 109, 176, 365
- validating 247
- viewing differences in location and 252
- viewing trends in 251, 312
- data entry restrictions 247
- Data Explorer 5, 318
  - adding data sets to 56, 61, 96, 160, 162
  - adding data sources to 52, 71, 102, 159
  - changing default data set and 65
  - creating computed fields and 59, 60, 84
  - displaying fields in 59, 82, 83, 85
  - joining data sets and 61
  - previewing query results and 57
  - removing data sources from 74
  - renaming data sets and 64
- data fields 597
  - See also* pivot ranges
  - adding 340, 341, 348
  - changing expressions in 358
  - changing location of 348
  - customizing calculations for 359
  - defined 335
  - formatting 343
  - hiding 350
  - removing data in 350
  - removing from a pivot range 342
  - selecting 340
- data filters 606
  - adding conditions for 448, 455
  - adding to queries 87, 88
  - comparing values with 124, 125, 126, 127, 128
  - creating 219, 220
  - defined 606
  - defining dynamic 134, 136
  - defining parameters as 89
  - deleting 135
  - designing reports and 39
  - prompting for values and 134, 135, 136
  - reordering conditions in 224
  - returning top or bottom values with 366
  - setting with graphical editors 110, 123, 125
  - testing 130
- Data Integration Service 50, 101, 405
- data label elements (charts) 311
- data label lines (charts) 302
- data labels 597
  - adding to charts 251, 299, 311
  - displaying 311
  - formatting 311
  - formatting numeric series in 291
  - hiding 293
  - repositioning 290
  - setting font options for 290
- Data Labels command 293, 299, 311
- Data Labels dialog 293
- Data Mapping page
  - accessing 278
  - mapping to data ranges and 278, 279
  - selecting chart types and 275
- data points 597
  - adjusting spacing for 296, 297
  - changing color settings for 292
  - changing patterns for 291
  - comparing multiple series and 253, 256
  - defined 597
  - describing 311
  - displaying non-numeric data and 288
  - displaying relationship to whole 294
  - formatting 285, 291
  - identifying 251
  - linking first to last 305
  - linking highest to lowest 305
  - missing in charts 294
  - overlapping bubbles and 299
  - pairing data values and 257
  - plotting as percentages 295
  - plotting y-axis values for 274
  - setting y-error bars for 293
  - tracking stock data and 259
- Data Preview pane 112, 156
- data range area 5
- Data Range Options command 243
- Data Range Options dialog 243
- data ranges 58, 64, 597
  - adding fields to 212
  - adding multiple 184–187
  - adding static content to 244, 245, 437

- data ranges (*continued*)
  - applying grant expressions to 383
  - changing 169, 178, 184, 281
  - creating 12, 19, 42, 170, 173, 231
  - defined 168
  - displaying hidden data in 287
  - excluding values in 225
  - generating 169
  - grouping sections in 21, 23, 174–177
  - hiding column or rows in 181
  - linking to 394
  - linking to charts 278, 287
  - mapping to charts 275, 277, 278, 280
  - placing functions in 434, 435, 438
  - plotting values in 250, 262, 270, 274
  - referencing cells in 235, 239
  - removing groups from 179
  - restricting access to 380
  - returning null values in 241
  - updating 281
- data rows 597
  - See also* rows
- data rows. *See* rows
- data series 631
  - adding 282, 291
  - changing appearance of 265, 291, 295
  - changing order of 294
  - comparing 252, 256, 257
  - connecting shapes for 297
  - contrasting 254
  - defined 631
  - describing 311
  - displaying data labels for 311
  - displaying scientific data and 259
  - displaying y-error bars in 293
  - formatting data points in 285, 291
  - formatting numeric 291
  - hiding labels for 293
  - highlighting 291, 292
  - overlapping multiple 252
  - plotting data for 274
  - plotting multiple 297
  - plotting on second y-axis 292
  - relating to axes values 304
  - selecting chart types and 275
  - selecting layout options for 282
  - setting shapes for 297
  - showing relationship to whole 257
  - showing relationships between 255, 266
  - spacing data points for 296, 297
  - stacking 294
- data series labels 250
- data series line options 298
- data series markers 291, 292
- data set fields 57
- data set fields. *See* fields
- data set objects 57, 61
- Data Set Properties dialog box 67
- data sets 104, 598
  - See also* result sets
  - building for
    - JDBC data sources 56, 81, 96
    - text file data sources 160–164
  - creating 8, 17, 56, 61, 76, 81, 96
  - defining cascading parameters for 329
  - editing queries for 64
  - entering grant expressions and 381
  - filtering 317
  - importing values from 323
  - joining 61
  - naming 9, 56, 62
  - overview 56
  - renaming 64
  - reordering data rows in 177
  - retrieving data and 41, 225
  - returning null values 240
  - selecting 12
  - specifying default 64, 65
  - updating 67
- data sources 598
  - accessing 50, 51, 53
  - accessing multiple information objects and 117, 119
  - changing 51, 354, 403, 409
  - compatibility with Excel 578
  - connecting to 6, 41, 104
  - creating pivot ranges and 334, 336
  - defined 598
  - deleting 53, 74
  - designing reports and 39
  - filtering data in 122
  - joining data sets and 61
  - linking charts to 443, 465
  - naming 6, 52



- renaming 53
- retrieving data from 41, 56, 215
- retrieving distinct values from 115
- selecting 6, 9, 50, 52, 53
- data streams 50
- data type markers 420, 421
- Data Type property 116, 149
- data types
  - adding parameters and 320, 321, 330
  - assigning to parameters 145, 149
  - changing 161, 164
  - creating dynamic filters and 135
  - defined 598
  - displaying output column 155
  - filtering data and 220
  - identifying 421
- database 598
- database connections. *See* connections
- database management systems 598
- Database property 408
- database schemas 80, 81, 629
- database statements 634
- databases 122
  - accessing 50, 70
  - changing column names in 76
  - connecting to 6, 70, 71, 407, 409
  - creating queries for 76
  - localizing reports and 550
  - restricting column retrieval for 94
  - retrieving data from 438
  - returning data from 56, 76, 80
  - running sample 51
  - selecting 6
  - testing connections to 72
  - writing data to 536
- Datasource property 405
- Date data type 599
- date expressions. *See* expressions
- date fields 362
- date filters 88
- date format symbols 554
- DATE function 558
- date function 445
- date functions 576, 577
- date separators 550
- date values 127, 129
- date-and-time data series 274
- DATEDIF function 558, 578
- dates
  - adding to file names 402
  - calculating 445
  - changing null values for 243
  - comparing 87
  - converting strings to 470
  - converting to text 472
  - formatting 34
  - grouping 363
  - localizing 550
  - plotting in charts 274, 308, 310
  - returning current 460
  - returning subsets of 446, 459, 475
- DATEVALUE function 558
- DAVERAGE function 558
- day format symbol 554
- DAY function 558
- day function 446
- DAYS360 function 558
- DB function 558
- DB2 databases 71
- DBCS function 558
- DBMS (defined) 598
- DCOUNT function 558
- DCOUNTA function 558
- DDB function 558
- debugging 599
  - callback classes 483–488
  - defined 599
  - reports 483
- debugging messages 483, 484, 489
- DEC2BIN function 558
- DEC2HEX function 558
- DEC2OCT function 558
- decimal precision 424, 425
- decimal separators 127
- decimal values 472
- declarations 599
  - See also* implicit declarations
- declarations section 599
- Decrease Detail Level command 207
- Default Chart command 288
- default code names 541
- default color palette 421
- default data sets 64, 65
- Default Java Async resource group 486

- Default Java View resource group 486
- default sort order 365
- default values 110, 137, 146, 321, 322, 323, 325
- defined names 89, 599
  - assigning to parameters 326
  - compatibility with Excel 575
  - creating 236–238, 326, 459
  - deleting data and 180
  - identifying pivot ranges and 367
  - linking to 391
  - referencing cells and 235
- DEGREES function 558
- delete function 446
- delete privilege 599
- deleting
  - cells 181
  - columns 181
  - columns from queries 94
  - computed fields 59
  - conditional formats 200
  - data sources 53, 74
  - dynamic data filters 135
  - filter conditions 133
  - grant expressions 383
  - hyperlinks 394
  - information objects 113
  - join conditions 119
  - joins 91
  - outline levels 207
  - outlines 206
  - output columns 116
  - parameters 146
  - rows 181
  - sections 179, 446
  - worksheets 446
- delimited data 599
- Delimited Text File dialog box 161
- delimited text files 158, 160, 161, 406, 416
- delimiters (text files) 158, 161, 406
- delimiters property 407
- DELTA function 558
- dependent joins 120
- deploying
  - BIRT Spreadsheet Designer 546, 548
  - callback classes 482–483
  - Java classes 491
  - reports 41, 45, 398, 483
- deploying BIRT Spreadsheet Designer 547
- Deployment Kit. *See* BIRT Deployment Kit
  - and BIRT Spreadsheet Deployment Kit
- derived classes. *See* descendant classes
- descendant classes 600
- descending sort order 94, 437, 452
- Describe Query button 155
- Description property 116, 150
- design files 632
- design files. *See* spreadsheet object design files
- design time 600
- design tools 53
- Design View 600
  - See also* spreadsheet reports
- designer applications 103
- designer. *See* BIRT Spreadsheet Designer
- designing spreadsheet reports 4, 38, 41, 208, 316
- designs 600
  - See also* page layouts
  - accessing
    - information objects and 102
  - associating callback classes with 482
  - changing data sources for 51, 403
  - changing default data set for 65
  - closing 15
  - creating 38, 43
  - defined 600
  - defining default data set for 64, 65
  - installing custom drivers for 53
  - localizing reports and 550
  - plotting data values and 250, 260, 262, 266, 274
  - removing data sources from 53, 74
  - reusing 317
  - saving 15
  - selecting data sources for 50, 53
  - testing templates for 537
- detail brackets (outlines) 202
- detail buttons (outlines) 202, 208
- detail data 201
- detail dots (outlines) 202
- detail function 215, 216, 446
- detail rows 169, 203, 215, 446
- detail sections 177
- developers

- adding VBA functionality and 536, 539, 541, 579
- creating grant expressions and 382
- creating multiple-class callbacks and 489, 490
- customizing BIRT Spreadsheet Designer and 478
- deploying BIRT Spreadsheet Designer and 548
- designing reports and 316
- developing workbooks and 478, 496
- displaying messages and 483
- overriding locale settings and 547
- developing applications 478
- developing spreadsheet reports 51, 247
- development languages 612
- DEVSQL function 559
- DGET function 559
- dialog boxes 415, 418, 419, 550
- difference charts 254
  - See also* charts
- Difference from setting 359
- directories
  - accessing configuration information and 428
  - accessing example callbacks and 498
  - compiling callback classes and 481
  - creating templates and 536, 537
  - deploying BIRT Spreadsheet Designer and 548
  - deploying callback classes and 482, 483
  - deploying Java classes and 491
  - extending callback classes and 489
  - installing localized versions and 546
- directory paths 136, 155
- JDBC drivers 70
- directory paths. *See* paths
- DISC function 559
- Display Control Type property 150
- Display Format property 117, 150
- Display Length property 117, 150
- Display Name property 117, 150
- display names 148, 319, 321, 323, 541
- display options
  - outlines 203, 207
  - pivot ranges 345
  - workbooks 420
  - worksheets 427
- displaying
  - column categories 114
  - column headings 427
  - columns 108
  - computed fields 59, 85
  - custom formats 191
  - data 19, 35, 66, 112, 156, 201, 212, 317
  - data labels 311
  - data set fields 57, 82, 83
  - data type markers 421
  - debugging messages 483, 489
  - error messages 111, 355, 447, 483, 489
  - grid lines 306
  - group keys 174
  - hidden toolbars 414
  - information objects 107, 113
  - join conditions 118
  - logging messages 484
  - multiple series in bar and column charts 269
  - outlines 427
  - parameter values 44
  - parameters 108, 154
  - pivot range field list 340
  - pivot range items 350, 351, 355
  - query output 11, 154, 155, 156
  - report parameters 155, 318
  - row headings 427
  - sample formats 191
  - section groups 174
  - section names 173
  - security IDs 380, 464
  - SQL statements 78, 112
  - system parameters 331, 403
  - worksheet grids 427
  - zero values 427
- distinct function 437, 447
- distinct values 447
- Distinct values only setting 115
- distributing spreadsheet reports 41, 45, 398, 483
- #DIV/0! error 555
- division operations 459
- division operator 436
- DMAX function 559
- DMIN function 559

- Do not echo input setting 325, 326
- Do Not Prompt property 150
- Dock on Bottom Side command 414
- Dock on Top Side command 414
- docking toolbars 414
- documentation xix
- documents. *See* reports; spreadsheet reports
- DOLLAR function 559
- dollar sign (\$) character 238
- DOLLARDE function 559
- DOLLARFR function 559
- double quotation mark (") character
  - report functions and 434
  - text expressions and 434
- doughnut charts
  - See also* charts
  - changing start angle of 301
  - creating 255
  - exploding sectors in 299
  - formatting data series for 299
  - plotting values for 276
  - sizing the hole 300
- DPRODUCT function 559
- dragging and dropping 212
- drawing controls. *See* drawing objects
- drawing objects 536
  - See also* graphical objects; images
- DrawingCharts callback example 522–524
- DrawingCharts.sod 522
- DrawingObjects callback example 520–522
- DrawingObjects.sod 520
- drill-down options 354
- driver attribute 75
- driver classes 71
- Driver property 408
- drivers 600
  - accessing data with custom 53
  - accessing JDBC 70, 74
  - changing connection information for 53, 73
  - connecting to data sources and 50, 70, 71
  - customizing 53, 74
  - defined 600
  - installing ODA 54
  - running predefined 53
- drop lines (charts) 251, 305
- drop-down lists 316, 322, 574

- DSTDEV function 559
- DSTDEVP function 559
- DSUM function 559
- dual y-axis charts 259, 292
- dual y-axis charts 601
- duplicate names 115
- duplicate rows 115
- DURATION function 559
- DVAR function 559
- DVARP function 559
- dynamic data filters 134, 135, 136
- dynamic data sources 403
- dynamic hyperlinks. *See* hyperlinks
- DynamicImage callback example 505–507
- DynamicImage.sod 505

## E

- e.Reporting Server. *See* BIRT iServer
- e.Reporting System. *See* BIRT iServer System
- e.Spreadsheet API. *See* BIRT Spreadsheet API
- e.Spreadsheet Deployment Kit. *See* BIRT Spreadsheet Deployment Kit
- e.Spreadsheet Designer. *See* BIRT Spreadsheet Designer
- e.Spreadsheet Engine and API. *See* BIRT Spreadsheet Engine and API
- e.Spreadsheet Engine. *See* BIRT Spreadsheet Engine
- e.Spreadsheet option. *See* BIRT Spreadsheet option
- e.Spreadsheet reports. *See* spreadsheet reports
- e.Spreadsheet Server. *See* BIRT iServer
- e.Spreadsheet technology 584
- e.Spreadsheet technology. *See* BIRT Spreadsheet technology
- EDATE function 559
- Edit Hyperlink command 393
- editing. *See* changing
- editors 428
- EFFECT function 559
- elements 601
  - defined 601
- ellipsis (...) button 601
- e-mail addresses 390, 393
- empty cells 288, 355, 421, 576

- empty lines 345
- empty rows 241
- empty values 126, 326
- Enable drill to details setting 354
- encapsulated SQL queries 100
- encapsulation 602
- encoding 407, 638
- encoding options 161, 163
- encryption levels 379
- Encyclopedia volumes 104, 107, 113, 155, 602
  - accessing information objects in 100, 101
  - deploying to 483
  - linking to reports in 395
  - publishing to 399, 400
  - uploading files to 398
- end method 478, 479
- ending values 363, 364
- English translations 546
- EOMONTH function 559
- equal sign (=) character 320
- Equal to operator 123
- equal to operator 436
- equality operator 118
- equijoins 121
- ERF function 559
- ERFC function 559
- Error amount setting 293
- error function 447
- error messages 111
  - deploying BIRT Spreadsheet Designer and 548
  - displaying 355, 447, 483, 489
  - validating data and 247
- ERROR.TYPE function 559
- errors 112
  - callback classes and 481
  - deleting data and 180
  - deleting sections and 179
  - incorrect argument count and 578
  - international translations and 548, 555
  - pivot ranges and 355
- escape characters 554, 602
- espreadsheet directory 428
- eSpreadsheet.ini 428
- essd9.jar 481
- Euro currency symbol 550
- EUROCONVERT function 560
- eval function 448
- evaluation rules 425
- EVEN function 560
- event handlers 603
- event listeners 603
- events 603
- EXACT function 560
- example databases 51
- example databases. *See* sample databases
- Excel spreadsheets
  - See also* worksheets
  - compatibility with 574–579
  - creating locale-specific reports and 230
  - creating templates from 536, 539
  - creating validation rules and 247
  - deploying translated reports and 546, 550
  - exporting to 575
  - generating 45
  - linking to 390
  - updating code names for 541
- exceptions 481, 603
- exclude function 224, 448
- executable files 603
  - creating 398
  - generating 44, 46
  - publishing 400
  - running 317
- execute privilege 101
- executing queries 66
- executing reports 15, 403, 483, 487
- exiting Information Object Query Builder 104
- EXP function 560
- expanded folders 113
- exploding doughnut sectors 299
- exploding pie sectors 301
- EXPONDIST function 560
- exponential trendlines (charts) 313
- exporting parameters 110
- exporting worksheets 575
- expression builder 106, 114, 115, 603
  - defined 603
- Expression property 117
- expressions 603
  - See also* formulas
  - adding conditions to 436
  - adding functions to 213

## expressions (*continued*)

- adding operators to 436, 437
- aggregating data and 586
- calculating summary data and 226, 230, 233
- changing pivot range 358
- creating computed fields and 59, 60, 85
- creating hyperlinks and 390
- creating locale-specific reports and 231
- customizing views and 382
- defined 603
- defining macros for 246
- deleting data and 180
- evaluating 438, 451
- filtering data and 87, 220, 224
- formatting data and 201
- grouping 224
- handling null values in 240
- localizing reports and 555
- manipulating numeric values with 618
- manipulating string data with 635
- renaming files and 402
- renaming sections and 179
- replicating 245
- returning Boolean values from 589
- returning specific values from 438

Extensible Markup Language. *See* XML

extensible markup language. *See* XML

extensions directory 482, 491

external cell references 604

*See also* cell references

external data sources 50, 51, 336

extract function 245, 448

## F

flj.logging.properties file 484

FACT function 560

FACTDOUBLE function 560

factory 604

FALSE function 560

FDIST function 560

field names 339, 381, 434

Field Settings command 358

field types (pivot ranges) 335

field values (pivot ranges) 335, 342

fields 57, 604

*See also* columns; computed fields

*See also* columns; data set fields; page fields; row fields

adding to pivot ranges 335, 339, 341, 342, 356

adding to reports 25, 42, 173, 212

assigning functions to 42

counting values returned from 444

defined 604

deleting 181, 342

designing reports and 39

grouping on 174, 177, 179

hiding 350

importing values from 323

linking to 390

moving 342

preserving formatting changes to 343

retrieving values from 473

returning distinct values from 437

showing top or bottom values in 366

testing for null values in 455

Fields page (Delimited Text File) 161

Fields page (Positional Text File) 164

file compatibility preferences 415

file name expressions 402

file names 401, 402, 403, 537, 541, 546

file paths 136, 155

File property 406

File Protection and Encryption command 374

File Protection dialog 374, 378

file types 574, 605

files

*See also* specific type

allocating memory for 428

applying SmartSheet security to 384

compatibility with Excel 574

compiling callback classes and 481

copying 398

copying privileges for 401

creating 518

creating executable 398

creating VBA templates and 536, 537

deploying BIRT Spreadsheet Designer and 548

generating spreadsheet 46

linking to 390, 392

managing multiple versions 401

- publishing executable 400
- renaming 402, 403
- uploading 398
- viewing recently opened 413
- Filetype property 406
- FileWriting callback example 518–520
- FileWriting.sod 518
- fill colors 196, 198
- Fill Effects dialog 292
- Fill page (Conditional Formats) 199
- Fill page (Format Cells) 196
- fill patterns (charts) 284, 286, 289
- fill patterns (graphical objects) 574
- fill styles 196
- Filter command 220
- filter conditions
  - adding to SQL queries 87
  - aggregating data and 144
  - changing 133
  - creating 122–132
  - defining multiple 130, 130–132
  - deleting 133
  - excluding sets of values with 126, 127, 131
  - grouping 131
  - selecting multiple values for 125
- Filter Conditions dialog 122, 123
- filter expressions 122, 124, 129
- filter operators 123, 124, 126
- filter options 606
- filtering
  - data 110, 121–138, 144
  - error messages 111
  - string values 126
- filtering data 219, 220, 224, 317, 366, 380
  - JDBC data sources and 87
- filters 606
  - adding conditions for 448, 455
  - adding to queries 87, 88
  - comparing values with 124, 125, 126, 127, 128
  - creating 219, 220
  - defined 606
  - defining dynamic 134, 136
  - defining parameters as 89
  - deleting 135
  - designing reports and 39
  - prompting for values and 134, 135, 136
  - reordering conditions in 224
  - returning top or bottom values with 366
  - setting with graphical editors 110, 123, 125
  - testing 130
- FILTERS clause 112
- Filters page 123, 125, 131, 134
- financial functions 576
- Financial Statement database 51
- Find and Replace dialog 571
- FIND function 560
- FINDB function 560
- finding matching values 233
- FINV function 560
- first function 449
- FISHER function 560
- FISHERINV function 560
- FIXED function 560
- fixed hyperlinks 390
- fixed-width text files 158, 162, 406
- flat file data sources 159, 406
- flat files 606
- floor (charts) 251
- FLOOR function 560
- folders 107, 113, 609
  - creating 46
  - publishing to 399, 401
- font attributes 33, 196, 287, 290, 513
- font colors 196
- Font command 287
- Font dialog 287, 290
- Font page (Format Cells) 30, 196
- font scaling option 287, 290
- fontcolor function 450
- fonts 196, 551, 606
  - See also* font attributes
- footers 606
  - defined 606
- FORECAST function 560
- Forecast setting (charts) 314
- foreground colors 444
  - See also* colors
- Format AutoShape command 376
- Format Axis command 265
- Format Cells command 30, 33, 191
- Format Cells dialog
  - creating validation rules and 247
  - displaying sample formats on 191

- Format Cells dialog (*continued*)
  - formatting data and 33, 196
  - locking worksheet cells and 376
  - setting fonts and 30, 31
- Format Chart Area command 287
- Format Data Labels dialog 311
- Format Data Series command 265, 285
- format function 201, 450
- Format Object dialog 376
- Format Plot Area command 284
- Format Report command 345
- Format Sheet dialog
  - changing code names and 542
  - changing evaluation rules and 426
  - displaying formulas and 377
- format symbols 551, 554
- Format Trendlines dialog 314
- formats 407, 607
  - See also* output formats; styles
  - applying 450
  - applying conditions to 197–201
  - compatibility with Excel 579
  - customizing currency 191
  - customizing regional settings and 549
  - defined 607
  - displaying 191
  - localizing 547, 549, 551
  - preserving 343
  - removing conditional 200
  - renaming files and 402
  - reverting to default 346
  - setting calculation precision and 424
- formatting
  - charts 265, 285, 288
  - column headings 32
  - data labels 311
  - dates 34
  - null values 243
  - numbers 190, 191, 343, 407, 514
  - pivot ranges 343, 579
  - spreadsheet reports 30, 40, 190, 195
  - text 30, 32
  - text file data sources 407
- formatting options 190, 196
- formatting templates 345
- formatting toolbar 33, 607
- forms objects 536, 574
- formula bar 5, 227, 378, 420
- formula function 227, 231, 451
- formulas 50, 607
  - See also* expressions
  - adding static text and 245
  - adding to pivot ranges 355, 362, 368
  - adding validation rules and 247
  - calculating summary data and 226
  - calling VBA code and 536
  - changing 227
  - compatibility with Excel 575, 578
  - creating parameters and 320, 326
  - defining macros for 246
  - displaying 377, 427
  - entering 575
  - formatting data and 198
  - hiding 378
  - recalculating 424, 425
  - references to pivot range cells 367
  - referencing cells and 235
  - referencing pivot ranges and 368
  - removing sections and 179
  - resolving circular references in 425
  - summarizing data and 435
  - viewing data type markers for 421
  - viewing pivot range 362
- forward slash (/) character 402, 439
- French translations 546
- FREQUENCY function 560
- frequency options (charts) 308
- FTTEST function 560
- full outer joins 118
- function builder 42, 214
- function keyword 607
- function reference 442
- function signatures 106
- functions 607
  - See also* methods; nested functions; specific function
  - adding static content and 244, 245, 437
  - adding to data ranges 434, 435, 438
  - adding to expressions 106
  - aggregating data and 139
  - assigning to specific fields 42
  - calculating pivot range values and 340, 355, 358



- calculating summary data and 226, 230, 232
- compatibility with Excel 576, 578
- creating grant expressions and 381, 382
- creating hyperlinks and 394
- creating locale-specific reports and 230, 231
- defined 607
- defining cell references and 239
- displaying data type markers for 421
- displaying security IDs and 380
- entering 214, 434
- filtering data and 219, 220, 224
- formatting data and 201
- grouping across multiple sheets and 174, 186
- hiding sections and 183
- localizing reports and 551, 555
- merging cells and 192, 193
- overview 213, 439
- referencing pivot ranges and 368
- renaming files and 402
- replicating 245
- resizing columns or rows and 182, 183
- retrieving data and 215, 225, 239
- returning null values from 241
- summarizing data with 226, 232, 340
- fundamental data types. *See* data types
- FV function 560
- FVSCHEDULE function 560

## G

- GAMMADIST function 560
- GAMMAINV function 560
- GAMMALN function 560
- Gap width setting (charts) 296, 297, 305
- GCD function 561
- General format symbol 554
- General page (Workbook Properties) 421
- generating
  - data ranges 169
  - Excel spreadsheets 45
  - executable files 44, 46
  - sample data 265, 277, 282
  - spreadsheet executables 46
- Generic JDBC setting 74

- GEOMEAN function 561
- German translations 546, 549
- GESTEP function 561
- getEntry function 451
- getLock method 481
- GETPIVOTDATA function 368–370, 561
- global reporting solutions. *See* locales
- global variables 607
  - See also* variables
- Glossary 581
- grand totals 39, 335, 360, 361, 368
- Grand totals for columns setting 361
- Grand totals for rows setting 361
- grandchild classes. *See* descendant classes
- grant expressions
  - adding to reports 41, 383
  - creating 381–383
  - customizing views and 384
  - removing 383
  - testing 383
- graphical objects
  - See also* images
  - compatibility with Excel 574
  - creating 520
  - protecting 376
  - resizing columns or rows and 182
- graphical query editors 105, 106, 153
- graphical user interfaces 546, 547, 574
  - See also* application window
- graphics. *See* graphical objects; images
- graphs. *See* charts
- Greater than operator 123
- greater than operator 436
- Greater than or equal to operator 123
- greater than or equal to operator 436
- grid lines 251, 306, 308, 427
- grids 608
- Grocery database 51
- GROUP BY clause
  - creating 138, 140–141
  - removing columns from 142–143
  - restricting output for 144
- Group command 23, 174, 363, 364
- group footers 606
- group function 175, 186, 241, 452
- group keys 174, 179, 452, 453, 608
- group names 365

- group numbers 453
- Group objects 547
- Group page (Section Filtering and Grouping) 174, 177
- group sections 452, 453
- grouped reports 608
  - defined 608
- grouping
  - data 23, 40, 138–144, 174, 362, 452
  - date and time values 363
  - expressions 224
  - filter conditions 131
  - numeric values 364
  - report parameters 327
  - sections 21, 23, 174–177
- Grouping dialog 363, 364
- groupNum function 453
- groups
  - adding parameters to 328
  - adding static content and 244
  - changing group keys for 179
  - changing sort order for 176
  - creating data ranges and 231
  - defined 608
  - defining across multiple sheets 174
  - defining content 174
  - deleting 179
  - laying out reports and 39, 42, 168
  - linking to 390
  - removing items from 446
  - returning null values in 241
  - sorting 452
  - specifying conditions for 363
  - verifying 174
- groupVal function 453
- GROWTH function 561
- GUIs 546, 547, 574
  - See also* application window

## H

- HARMEAN function 561
- Has Null property 117
- HAVING clause 144
- headers 608
  - defined 608
- headers. *See* section headers

- Heading property 117, 150
- headings. *See* column headings; row headings
- help
  - See also* balloon help
- Help Text property 117, 150
- Help Text setting 326
- HEX2BIN function 561
- HEX2DEC function 561
- HEX2OCT function 561
- hexadecimal numbers 609
- hidden parameters 150
- Hide command 181, 350
- Hide Detail command 352, 353
- hiding
  - axis labels 307
  - cell references 420
  - chart axes 306
  - columns 94
  - data 287
  - data labels 293
  - detail rows 203
  - formula bar 420
  - formulas 378
  - grand totals 361
  - grid lines 306
  - parameters 326
  - pivot range items 350, 352, 353, 354, 355
  - rows or columns 181
  - scroll bars 427
  - sections 181–182, 183
  - subtotals 360
  - toolbars 414
  - user input 326
  - workbook borders 420
  - worksheet tabs 377
  - worksheets 377
- hiding column categories 114
- hierarchy 609
- high-low lines (charts) 305
- hints 285
- HLOOKUP function 561
- home folder 609
- Horizontal Alignment property 117, 150
- hour format symbol 554
- HOURLY function 561
- hour function 454
- HTML (defined) 610

- HTML elements 601
  - See also* HTML tags
- HTTP (defined) 610
- Hyperlink command 392
- Hyperlink dialog 392
- hyperlink expressions 390
- HYPERLINK function 561
- hyperlink function 454
- hyperlinks 609
  - See also* dynamic hyperlinks; static hyperlinks
  - activating 392
  - adding parameters to 396
  - changing 393
  - creating 392, 394, 395
  - deleting 394
  - embedding 454
  - overview 390
- hypertext markup language. *See* HTML
- hypertext transfer protocol. *See* HTTP
- HYPGEOMDIST function 561

## I

- identifiers 610
- IF function 561
- if function 201, 382, 455
- IMABS function 561
- image file types 574
- image files 610
- images 610
  - See also* graphical objects
  - adding 505
  - creating hyperlinks for 390
  - locking 376
  - rendering charts as 288
  - resizing columns or rows and 182
- IMAGINARY function 561
- IMCONJUGATE function 561
- IMCOS function 561
- IMDIV function 562
- IMEXP function 562
- IMLN function 562
- IMLOG10 function 562
- IMLOG2 function 562
- import statements 480, 490
- Import Values dialog 323
- IMPOWER function 562
- IMPRODUCT function 562
- IMREAL function 562
- IMSIN function 562
- IMSQRT function 562
- IMSUB function 562
- IMSUM function 562
- IN operator 123, 125
- in operator 436
- include function 224, 455
- Increase Detail Level command 204
- indented layouts 345
- INDEX function 562
- index number references 367
- Index setting 360
- Indexed property 117
- INDIRECT function 562
- inequality 436
- INFO function 562, 576
- information. *See* data
- Information Console
  - displaying reports and 398
  - publishing to 398
  - running reports and 488
  - testing callback classes and 485
- information object data sources 117
- information object files 611
- Information Object Query Builder
  - accessing expression builder in 106
  - creating joins and 118
  - creating queries and 107, 111, 112, 153
  - defining multiple conditions and 130, 131
  - exiting 104
  - filtering data and 144
  - grouping data and 140, 141, 142
  - hiding column categories in 114
  - overview 103
  - prompting for values and 135
  - selecting information objects and 113
  - starting 104
- information objects 336
  - accessing data in 100, 101
  - adding to designs 102
  - building data sets for 104
  - categorizing columns in 114
  - connecting to 101, 102, 405
  - defined 610

- information objects (*continued*)
  - defining joins for 117–121
  - deleting 113
  - displaying objects in 108
  - displaying output for 156
  - displaying parameters for 155
  - filtering data in 110, 121–138, 144
  - overview 100
  - retrieving data and 50
  - retrieving data in 103
  - selecting 107, 113
  - setting properties for 101
  - sorting data in 109
  - synchronizing parameters in 152–153
  - viewing 107, 113
- Informix databases 71
- inheritance 611
- inherited properties 151
- inner joins 78, 119, 611
  - See also* joins
- input 134, 135, 136, 147, 151
  - hiding 326
  - localizing reports and 570
  - processing text strings as 448
  - prompting for 317, 321, 322, 324
- input method editor 551
- Input Method Editor files 611
- input sources. *See* data sources
- Insert Calculated Field dialog 356
- Insert Calculated Item dialog 357
- Insert Chart button 277
- Insert Report Function dialog 214
- Insert Row After command 180, 217
- Insert Row Before command 180
- installation
  - JDBC drivers 70
  - ODA drivers 54
  - sample databases 51
- installing localized versions 546
- instances. *See* objects
- INT function 562
- integers. *See* numbers
- Integration service 121
- interactive features 536
- INTERCEPT function 562
- interfaces 612
  - defined 612
  - predefined data sources and 54
- interfaces. *See* user interfaces
- internationalization 545, 554, 612
  - See also* locales
- interval options (charts) 308, 309, 310
- INTRATE function 562
- Intrxmlopt property 408
- IO Design perspective 612
- .job files. *See* information object files
- IPMT function 562
- IRR function 562
- IS NOT NULL operator 123, 126
- IS NULL operator 123, 126
- ISBLANK function 562
- ISERR function 562
- ISERROR function 563
- iServer 104, 582
- iServer Explorer 107, 114
- iServer System options. *See* BIRT iServer System options
- iServer System. *See* BIRT iServer System
- iServer. *See* BIRT iServer
- ISEVEN function 563
- ISLOGICAL function 563
- ISNA function 563
- ISNONTEXT function 563
- isnull function 455
- isnull keyword 225
- ISNUMBER function 563
- ISODD function 563
- ISREF function 563
- ISTEXT function 563
- Italian translations 546
- items (pivot ranges) 335, 342
- iteration 424, 425
- iteration options 425

## J

- J2EE environments 612, 613
- J2SE environments 613, 614
- Japanese translations 547, 570
- .jar files. *See* Java archive files
- JAR files 481, 491, 546, 548
- Java. *See* Java programming language
- Java 2 Enterprise Edition. *See* J2EE environments

- Java 2 Runtime Standard Edition. *See* J2SE environments
  - Java archive files 613
  - Java archive files. *See* JAR files
  - Java archives. *See* .jar files
  - Java classes 382, 489, 491
  - Java Components. *See* BIRT Java Components
  - Java Database Connectivity. *See* JDBC
  - Java debugging environments 486, 487
  - Java Development Kit. *See* JDK software
  - Java factory 488
  - Java interfaces 612
  - Java Naming and Directory Interface 613
  - Java programming language 612
  - Java programs 613
  - Java resource group services 486, 487
  - Java servers 488
  - Java Virtual Machines 549
  - Java Virtual Machines. *See* JVMs
  - Java-based reports 487, 488
  - javac command line utility 481, 482
  - Javadoc 479, 497
  - javadoc directory 479
  - JavaScript 614
  - JavaServer Pages. *See* JSPs
  - JBook objects 547
  - JDBC (defined) 613
  - JDBC Connection dialog box 73
  - JDBC Connection Properties dialog box 73
  - JDBC data sources
    - accessing data in 70
    - building queries for 70, 76
    - building stored procedures for 96
    - changing connection information for 73
    - connecting to 70–73, 75, 408
    - creating 71
    - deleting 74
    - filtering data from 87
    - naming 72
    - renaming 74
    - returning data sets from 56, 81, 96
    - testing connections to 72
  - JDBC databases 336
  - JDBC drivers 70, 73, 74, 408
  - JDK software 613
  - JNDI (defined) 613
  - JNDI names 408
  - jndi property 408
  - join algorithms 120, 121
  - join conditions 117, 118, 119, 615
  - join operators 118
  - Join Properties dialog box 92
  - join types 119
  - joins 17, 226, 614
    - changing 92
    - combining multiple data sets and 61
    - creating 82, 91, 117–121
    - deleting 91
    - optimizing 119, 121
    - overview 78
    - specifying cardinality of 120, 121
  - Joins page 117, 118
  - joint data sets 611, 615
  - JSPs 614
  - JVMs 549, 614
- ## K
- key values 192, 193
  - keywords 615
  - Korean translations 547
  - KURT function 563
- ## L
- label elements (charts) 250, 287, 290, 306
  - labels 349, 370, 597
    - See also* column headings
  - language files 548
  - language preferences 415, 418, 419
  - language settings 549, 550
  - language translations 546, 554
  - LARGE function 563
  - large reports 316
  - last function 456
  - layout editor 616
    - defined 616
  - layout options 169
  - Layout window. *See* layout editor
  - layouts 615
    - See also* designs
    - creating pivot ranges and 344, 345
    - designing 38
    - structuring report data and 39, 42, 168
  - LCM function 563

- leading spaces 472
- leap years 577
- LEFT function 130, 563
- left outer joins 119
- left outer joins. *See* outer joins
- LEFTB function 563
- legends (charts) 250, 287
- LEN function 563
- len function 456
- LENB function 563
- Less than operator 123
- less than operator 436
- Less than or equal to operator 123
- less than or equal to operator 436
- lib directory 483, 491
- libraries 478, 616
  - See also* DLLs
- licenses 549
- licensing options 582
- LIKE operator 89, 123, 127
- like operator 436, 438
- line charts
  - See also* charts
  - creating 256
  - plotting percentages in 295
  - plotting values for 276
  - stacking series for 294
- line controls 617
- line elements (charts) 292, 297, 302, 304
- line numbers 111
- line styles (charts) 287, 290
- line styles (graphical objects) 574
- line widths 574
- linear trendlines (charts) 312
- LINEST function 563
- linking to bookmarks 391
- linking to charts 270, 278, 287
- links. *See* hyperlinks
- list boxes 574
- List Formulas command 362
- listing wizard 6, 169
- Listing Wizard command 12, 260
- lists 322, 437
  - See also* drop-down lists
- literal characters 128, 136
- literal values 245
  - See also* static content
- LN function 563
- loading applications 428
- local directory 548
- local parameters 151
- locale settings 547
- locales 617
  - adjusting formats for 549
  - calculating values for 230, 231
  - changing language settings for 550
  - creating passwords and 374
  - customizing currency formats for 191
  - defined 617
  - deploying BIRT Spreadsheet Designer for 546, 548
  - saving workbooks for multiple 416
  - selecting translation versions for 547
  - setting encryption options for 379
  - setting regional preferences for 415, 418, 419
  - specifying 547
  - translating reports for 546, 549, 550, 554
- localization 617
  - See also* locales
- locating matching values 233
- lock function 457
- locking
  - images 376
  - workbooks 481
  - worksheet cells 457, 507
  - worksheets 457
- locks 383
- log files 484
- LOG function 563
- log function 310
- LOG10 function 563
- logarithmic axis 310
- logarithmic trendlines 312
- LOGEST function 563
- Logger objects 483, 484
- logging error and debugging messages 483
- logging levels 484
- logging messages 484
- logical operators 131
- LOGINV function 563
- LOGNORMDIST function 563
- LOOKUP function 563
- Lotus 1-2-3 spreadsheets 426

LOWER function 563  
lower function 457  
lowercase characters 457

## M

Macro page 617  
    *See also* spreadsheet reports  
macros 245–247, 536, 538  
major units (charts) 310  
make-table queries 61  
Management Console 488  
    defined 617  
    file types displayed in 605  
management tools 617  
maps (information objects) 136  
marker shapes (charts) 291, 292  
Markers dialog 291  
markup languages 604, 610  
    *See also* elements; tags  
masking user input 326  
Match Character Width setting 571  
MATCH function 564  
matching character patterns 127  
matching values 233  
mathematical operators 436, 437  
matrix ranges 530  
MAX function 564  
max function 86, 457  
MAXA function 564  
maximum values 86  
MDTERM function 564  
MDURATION function 564  
MEDIAN function 564  
memory 121, 428  
menu bar 414  
menus  
    accessing worksheet 208  
    changing charts and 284  
    enabling VBA functionality and 536  
    setting language preferences for 415, 418, 419  
Merge and Center icon 28  
Merge Cells command 194  
merge function 192, 193, 458  
merge joins 120  
Merge labels command 349

merging cells 28, 192–195, 458  
messages 483, 489  
    *See also* e-mail; error messages  
Messages callback example 481, 483, 489, 492  
Messages.java class file 489, 492  
metadata 618  
    defined 618  
methods 618  
    *See also* functions  
    building callback classes and 478, 489  
    configuring remote debugging and 486  
    defined 618  
    defining exception handlers for 481  
    developing workbooks and 497  
    displaying messages and 483  
    overriding 620  
    retrieving objects and 497  
    setting properties and 497  
    viewing descriptions of 497  
Microsoft Excel. *See* Excel spreadsheets  
Microsoft Word documents 391  
MID function 564  
MIDB function 564  
MIN function 564  
min function 86, 458  
MINA function 564  
minimal recalculation option 425  
minimum values 86  
minute format symbol 554  
MINUTE function 564  
minute function 458  
MINVERSE function 564  
MIRR function 564  
missing data 288  
missing values 127  
mixed cell references 238  
MMULT function 564  
mock-ups 38  
MOD function 564  
mod function 459  
MODE function 564  
modifying. *See* changing  
ModifyQuery callback example 502–505  
modifyQuery.sod 502  
month format symbol 554  
MONTH function 564  
month function 459

- Move Down command 414
- Move down command 349
- Move left command 349
- Move right command 349
- Move to beginning command 349
- Move to end command 349
- Move Up command 414
- Move up command 349
- moving
  - charts 266
  - docked toolbars 414
  - pivot range fields 342
  - pivot range items 349
- moving average trendlines (charts) 313, 314
- MROUND function 564
- multi-column headings 192, 193
- multi-locale environments 545, 553
  - See also* locales
- MULTINOMIAL function 564, 578
- multiplication operations 462
- multiplication operator 436
- multi-table queries 78, 82, 91
- MyCallback example 480
- MyCallback2 example 489
- MyCallback2.java class file 489

## N

- N function 564
- #N/A error 555, 578
- NA function 564
- #NAME? error 555
- name attribute 75
- name function 459
- Name property 117, 150
- name property 408
- name references 367
- names
  - See also* defined names
  - calculating pivot range data and 356, 358
  - changing code 542
  - changing column 76, 161, 164
  - changing file 402, 403
  - changing section 177, 179
  - creating callback classes and 480
  - creating display 323
  - creating parameters and 320

- creating pivot ranges and 339, 343, 368
- deleting section 179
- developing templates and 541
- displaying data and 319
- publishing reports and 401, 403
- referencing sections and 239
- returning worksheet 467
- viewing section 173
- naming
  - calculated fields 357
  - calculated items 357
  - column sections 22
  - computed fields 59, 60
  - data sets 9, 56, 62, 104
  - data sources 6, 52
  - detail rows 217
  - flat file data sources 159
  - output columns 115
  - parameter groups 328, 329
  - parameters 145, 150, 320, 321
  - report parameters 89
  - row sections 21
  - sections 173
  - template files 536
  - trendlines 314
- negative values 292
- NEGBINOMDIST function 564
- nested loop joins 121
- nested structures 168, 173
- network administrators. *See* administrators
- NETWORKDAYS function 564
- New Data Set dialog 104
- New Data Set dialog box 56, 62
- New Data Source dialog box 52
- new lines 438
- NOMINAL function 565
- non-null values 86, 449, 456
- non-numeric data 274, 288
- NORMDIST function 565
- NORMINV function 565
- NORMSDIST function 565
- NORMSINV function 565
- NOT BETWEEN operator 123, 126
- Not equal to operator 123
- not equal to operator 436
- NOT function 565
- NOT IN operator 123



- NOT LIKE operator 123, 126, 127
  - NOT operator 131
  - not operator 436
  - NOW function 565
  - now function 460
  - NPER function 565
  - NPV function 565
  - nth function 460
  - #NULL! error 555
  - null values 117, 126, 146, 240–244, 326, 455, 618
  - #NUM! error 555
  - Number command 291
  - Number dialog 291
  - number fields 362
  - Number Format dialog 344
  - number format symbols 551
  - number formats 191, 291, 424
  - Number page (Format Cells) 191, 192
  - number sign (#) character
    - hyperlinks and 391
    - report functions and 214, 434
  - numbers
    - assigning to parameters 152
    - comparing 127
    - converting strings to 471
    - converting to text 472
    - formatting 190, 191, 343, 407, 514
    - grouping 364
    - localizing 547, 551
    - plotting in charts 274, 291
    - returning null values and 243
    - rounding 463, 576
    - setting default values and 146
    - specifying precision for 424, 425
    - truncating decimal values in 472
  - numeric data. *See* numbers
  - numeric data series 274, 291, 292
  - numeric expressions 618
    - See also* expressions
- O**
- object reference variables 618
    - See also* objects; variables
  - object rotation 574
  - object-oriented programming 619
  - objects 496, 497, 618
    - See also* persistent objects; transient objects
    - defined 618
  - OCT2BIN function 565
  - OCT2DEC function 565
  - OCT2HEX function 565
  - ODA (defined) 619
  - ODA data sources 150
    - connecting to 54
  - ODA drivers 619
    - accessing data with 53
    - defined 619
    - installing 54
    - overview 54
  - ODBC (defined) 619
  - ODBC databases 336
  - ODD function 565
  - ODDFPRICE function 565, 576
  - ODDFYIELD function 565, 576
  - ODDLPRICE function 565, 576
  - ODDLYIELD function 565, 576
  - OFFSET function 565
  - on-demand reporting 316
  - online documentation xix
  - open data access 619
  - open data access technology 619
    - See also* ODA
  - open database connectivity. *See* ODBC
  - open values 259
    - See also* stock charts
  - opening
    - configuration files 428
    - expression builder 106, 115
    - Information Object Query Builder 104
    - Prompt editor 135
    - reports 390, 395
  - operators 436, 437, 619
  - optimizing
    - joins 119, 121
  - optimizing BIRT Spreadsheet Designer 428
  - optional parameters 89, 330
  - options (licensing) 582
  - Options dialog (Format Axis) 307
  - Options dialog (Format Chart Area) 287, 288
  - Options dialog (Format Data Series) 292
  - Options page (Format Data Series) 296, 297
  - OR function 565

- OR keyword 153
- OR operator 131, 222
- or operator 436
- Oracle databases 71, 96
- ORDER BY clause 94
- outer joins 79, 91, 118, 119, 619
  - See also* joins
- outline buttons 202
- outline components 202
- outline layouts 344
- outline levels 203
- Outline Options command 203, 208
- Outline Options dialog 203, 208
- outlines
  - clearing 206
  - creating 201, 203–206
  - displaying 207, 427
  - removing levels from 207
  - testing 206
- output 156
  - changing report type for 413
  - creating charts and 266, 268
  - displaying messages and 483
  - localizing reports and unexpected 549
  - previewing query 88, 98
  - previewing query output 57
  - returning from queries 11
- output columns
  - defining 115–116
  - deleting 116
  - displaying 154, 155, 156
  - naming 115
  - setting character lengths for 117
  - setting order of 115
  - setting properties for 116, 155
- output formats 620
  - See also* formats
- output window 483
- Overlap setting (charts) 296, 297
- overlapping bubbles 299
- overlay charts. *See* dual y-axis charts
- overloaded methods 620
- override 620
- overriding
  - locale and language settings 547
  - minimal recalculation 425
- overriding connections 404, 405

- overwriting report files 401

## P

- package (defined) 620
- package statements 490
- packages 478, 490, 491, 620
- page 620
  - See also* tabs; windows
- page area (pivot ranges) 343, 347
- page breaks 370, 461
- page components 620
- page fields
  - See also* pivot ranges
  - adding 341
  - arranging 346
  - changing location of 346, 347
  - compatibility with Excel 578
  - defined 335
  - displaying items in 350
  - hiding 350, 355
  - hiding items in 351, 360
  - removing 342
  - selecting 340
- page footers 606, 621
- page headers 608, 621
- page headers and footers
  - defined 606
- page layouts 615
  - See also* designs
  - defined 615
- pageBreak function 461
- pages 620
- palette (color) 421
- panes 621
- Parameter Group dialog 328, 329
- parameter groups 327–328, 329
- Parameter Mode property 150
- parameter names 89, 320, 321, 396
- parameter types 317
- Parameter Values dialog 156
- parameterized queries
  - creating 96
  - defining parameters for 98
- parameters 621
  - adding to groups 328
  - adding to hyperlinks 396

- adding to queries 110, 145–146, 151
- adding to reports 318, 319–321
- assigning data types to 145, 149
- assigning defined names to 326
- assigning null values to 146
- assigning to parameters 152
- assigning values to 89, 320, 323
- changing 331
- changing properties for 151
- connecting to ODA data sources and 54
- creating 89, 98
- creating grant expressions and 382
- customizing reports and 316, 317
- defining static 89
- deleting 146
- designing reports and 39
- developing workbooks and 498
- displaying 318, 331
- displaying query output and 156
- exporting 110
- filtering data and 89
- grouping 327
- hiding 150, 326
- naming 89, 145, 150, 320, 321
- prompting for input and 317, 321, 322, 324
- prompting for values and 136, 147, 151
- setting properties for 147, 149
- setting values for 110, 145, 146, 151
- specifying required 150
- testing 330
- viewing 108, 154, 155
- viewing system 403
- viewing values 44
- Parameters page 622
- Parameters page (Query Design) 146, 147, 153
- Parameters page (SQL editor) 155
- parent identifier 239
- parent sections 173, 179, 239
- parentheses ( ) characters
  - creating defined names and 236
  - grouping expressions and 224
- Password property 101, 105, 406, 408
- passwords 622
  - BIRT iServer 406
  - creating 374
  - defined 622
  - encrypting data and 378
  - information objects and 101
  - JDBC data sources and 408
  - setting 374, 375, 376
- paths 136, 155
  - accessing packages and 491
  - changing configurations and 428
  - compiling callback classes and 481
  - connection configuration files 404
  - creating hyperlinks and 391
  - JDBC drivers 70
  - loading applications and 428
- pattern matching 127
- pattern matching. *See* search expressions
- pattern method 484
- patterns (worksheets) 196, 444, 450, 574
- Patterns command 286, 291, 311
- Patterns dialog 286, 287, 289, 290
- PEARSON function 566
- Percent Difference From setting 359
- Percent of column setting 360
- Percent of row setting 360
- Percent of setting 359
- Percent of total setting 360
- percentages 295, 359
- PERCENTILE function 566
- PERCENTRANK function 566
- performance 119, 121, 122
  - creating pivot ranges and 342
  - processing reports and 428
  - recalculating data and 425
- permissions. *See* privileges
- PERMUT function 566
- persistent objects 622
- personal folder. *See* home folder
- personalizing BIRT Spreadsheet Designer 412
- PHONETIC function 551
- PI function 566
- picture objects 574
  - See also* graphical objects; images
- pie charts
  - See also* charts
  - changing fill patterns in 284
  - changing start angle of 302
  - connecting sectors in 302
  - creating 257

- pie charts (*continued*)
  - exploding sectors in 301
  - formatting data series for 301
  - plotting values for 276
  - showing data label lines in 302
- pie of pie charts 302–304
- pivot charts 578
- pivot range areas 341, 347
- pivot range data sources 337, 354
- pivot range field list 340
- pivot range reports
  - printing 370, 579
  - referencing items for 343
  - selecting layout options for 344
  - sorting in 365
- pivot range toolbar 334
- pivot range wizard. *See* PivotRange Wizard
- pivot ranges 58, 623
  - See also* spreadsheet reports
  - adding fields to 335, 339, 341, 342, 356
  - adding to worksheets 171
  - building 339
  - calculating data in 355–362
  - changing 335, 343
  - changing data sources for 354
  - changing expressions for 358
  - compatibility with Excel 578
  - creating 171, 336–337, 354, 524
  - defined 168
  - filtering data for 366
  - formatting 343, 579
  - formatting data in 190
  - grouping data in 362–365
  - hiding items in 350, 352, 353, 354, 355
  - labeling fields in 349
  - laying out 344, 345
  - moving fields in 342
  - moving items in 349
  - overview 334, 335
  - printing 370, 579
  - referencing data in 343–370
  - renaming 343
  - repositioning 342
  - reverting to default formats 346
  - selecting location for 338
  - sorting data in 365
  - updating 343, 349, 354
  - viewing calculated items in 362
  - viewing detail items in 352
  - viewing errors in 355
- pivot tables 578
  - See also* pivot ranges
- PivotRange Field dialog
  - adding block totals and 362
  - changing expressions and 358
  - customizing calculations and 359
  - displaying items and 355
  - formatting numbers and 343
  - generating subtotals and 360
  - hiding items and 350, 351
  - laying out pivot ranges and 345
- PivotRange Field Layout dialog 345, 370
- PivotRange Options dialog
  - displaying errors and 355
  - generating grand totals and 361
  - generating subtotals and 360
  - preserving formats and 346
  - renaming pivot ranges and 343
  - setting drill-down options on 354
  - setting page field location and 347
  - setting print titles and 370
- PivotRange Sort and Top 10 dialog 366
- PivotRange Wizard
  - changing data sources and 355
  - creating pivot ranges and 171, 336
  - repositioning pivot ranges and 342
  - selecting worksheet lists for 337
- plot area (charts) 251, 284, 286, 306
- plug-ins 53
- PMT function 566
- POISSON function 566
- polynomial trendlines (charts) 312, 314
- Port number property 104
- port numbers 486
- Positional Text File dialog box 163
- positive values 292
- POWER function 566
- power trendlines (charts) 313
- PPMT function 566
- precision 424, 425
- predefined connections 404, 409
- predefined data drivers 53
- predefined data filters 134
- predefined data sources 53

- predefined values 596
- preferences 412
  - See also* properties
- Preferences command 547
- preserving workbook windows 575
- Preview command 57
- Preview page (BIRT Spreadsheet Designer) 57
- previewing
  - charts 281
  - conditional formats 199
  - reports 15, 30, 34, 44–45
- previewing data 57
- previewing result sets 88, 98
- PRICE function 566, 576
- PRICEDISC function 566
- PRICEMAT function 566
- primary keys 78
- print areas 498
- print titles 370
- printing pivot ranges 370, 579
- println method 483
- private access type 581
- privileges 101, 623
  - copying 401
  - viewing reports and 384, 398
- PROB function 566
- Problems pane 111
- procedures 623
- PRODUCT function 566
- product function 461
- profile settings 399, 400
- profiles
  - creating BIRT iServer 399–403
  - publishing reports and 398
  - selecting 401
  - testing connections for 400
- programmers
  - adding VBA functionality and 536, 539, 541, 579
  - creating grant expressions and 382
  - creating multiple-class callbacks and 489, 490
  - customizing BIRT Spreadsheet Designer and 478
  - deploying BIRT Spreadsheet Designer and 548
  - designing reports and 316
  - developing workbooks and 478, 496
  - displaying messages and 483
  - overriding locale settings and 547
- programming languages 612
- Progress pane 112
- Prompt editor 135, 136, 147
- prompting for input 317, 321, 322, 324
- prompting for values 134, 135, 136, 147, 151
- PROPER function 566
- properties 623
  - changing worksheet appearance and 426
  - data source connections 52, 104, 405, 406, 407, 409
  - defined 623
  - developing VBA templates and 541
  - developing workbooks and 497
  - dynamic data filters 136
  - information objects and 101
  - inheriting 151
  - JDBC connections 72, 73
  - output columns 116, 155
  - parameters 147, 149
  - setting workbook 420–424
- properties files 51
- Properties objects 409
- Protect Sheet command 374, 376
- Protect Sheet dialog 373, 374, 376
- Protect Workbook command 375
- Protect Workbook dialog 375
- protecting spreadsheet reports 372
- Protection page (Format Cells) 376
- Protection page (Format Object) 376
- prototypes 38
- public methods 497
- publish 624
- Publish and Preview Options dialog 399, 400, 401
- Publish Report command 399, 400
- publishing spreadsheet reports 398–401, 537
- PV function 566
- pyramid charts 276

## Q

- QBE (defined) 624
- QBE expressions 136, 137

- QBE syntax 134, 624
- quarter function 462
- QUARTILE function 566
- queries
  - accessing external data sources and 337
  - accessing multiple information objects and 117, 119
  - adding parameters to 110, 145–146, 151
  - aggregating data and 86
  - building data sets for 104
  - building for
    - JDBC data sources 70
    - text file data sources 158, 160, 162
  - calculating summary data and 226
  - changing 64, 91, 103, 153, 154, 502
  - converting column names for 118
  - copying 153
  - creating 8, 17, 77–83, 96, 103, 105, 107, 153
  - customizing 111
  - defined 624
  - defining output columns for 115–116
  - displaying output from 11
  - filtering data with 87
  - installing custom drivers and 54
  - joining data sets and 61, 63
  - joining tables with 78, 82, 91
  - manipulating database columns and 93
  - optimizing 119, 121
  - overview 56
  - previewing output from 57, 88, 98
  - prompting for values and 135, 136, 137, 147, 148
  - refreshing 66
  - removing parameters from 146
  - reordering columns in 93
  - restricting number of rows in 121, 144
  - retrieving data and 57, 76, 316
  - retrieving data with 41
  - returning duplicate rows and 115
  - returning single-row result sets from 463
  - running 66
  - saving 11, 83, 111, 153
  - selecting columns for 10, 19
  - selecting tables for 9, 17
  - setting dynamic filters and 112
  - sorting data with 94
  - synchronizing 625
  - unknown data types in 116
  - validating 112, 131, 135
  - viewing columns selected for 108
  - viewing errors with 111, 112
  - viewing output from 154, 155, 156
- Query by Example. *See* QBE
- Query Design 106
- query editor 42, 624
  - aggregating query results and 86
  - creating computed fields with 84–85
  - creating data sets and 81
  - creating queries with 77, 93
  - defining parameters with 89, 90
  - filtering data with 87, 88
  - hiding columns and 94
  - joining data sets and 62
  - removing columns and 94
  - selecting multiple tables for 82, 91
  - setting sort order in 94
- query editors 105, 106
- query languages 585
- query parameters. *See* parameters
- question mark (?) character 438
- quotation mark characters 245, 434
  - See also* double quotation mark character;
  - single quotation mark character
- QUOTIENT function 566

## R

- RADIANS function 566
- radio buttons 322, 574
- RAND function 566
- RANDBETWEEN function 566
- range 625
- range of values
  - defined 625
- range of values, limiting 247
- range references
  - identifying pivot ranges and 367
  - linking to charts and 281
  - setting limits for 427
- ranges
  - See also* data ranges; pivot ranges
  - laying out reports and 168
  - linking to 390, 392
  - locking cell 457, 507

- RANK function 566
- RATE function 566
- RDBMS (defined) 626
- read privilege 101, 626
- recalculating formulas 424, 425
- RECEIVED function 566
- recently opened files list 413
- records 438
  - See also* rows
  - sorting 94
- #REF! error 179, 180, 555
- reference lines (charts) 304, 306
- references. *See* cell references; range references; section references
- Refresh Data icon 354
- refreshes 66, 407, 408
- Refreshindex property 407, 408
- refreshing charts 282
- refreshing pivot ranges 346, 354
- Regional page (Preferences) 547, 550
- regional settings 415, 418, 419, 549
  - See also* locales
- REGISTER.ID function 567, 576
- relational database management systems 626
- relational databases 50, 70
- relational databases. *See* databases
- relative cell references 238, 626
  - See also* cell references
- relative index numbers 367
- relative paths 391
- releaseLock method 481
- remainder 459
- remote debugging 486, 487
- Remove Column command 95
- Remove Hyperlink command 394
- removing. *See* deleting
- renaming
  - columns 76, 161, 164
  - data sets 64
  - data sources 53
  - groups 365
  - pivot ranges 343
  - sections 177, 179
  - spreadsheet files 402, 403
  - worksheets 541
- Reorder Columns command 93
- Reorder Columns dialog box 93
- repeating values 193
- REPLACE function 567
- REPLACEB function 567
- report design files 632
- report designer applications 103
- report designs 600
  - See also* page layouts
  - accessing
    - information objects and 102
  - associating callback classes with 482
  - changing data sources for 51, 403
  - changing default data set for 65
  - closing 15
  - creating 38, 43
  - defined 600
  - defining default data set for 64, 65
  - installing custom drivers for 53
  - localizing reports and 550
  - plotting data values and 250, 260, 262, 266, 274
  - removing data sources from 53, 74
  - reusing 317
  - saving 15
  - selecting data sources for 50, 53
  - testing templates for 537
- report documents 626
- Report Encyclopedia. *See* Encyclopedia volumes
- report executable files 627
  - See also* report object executable files
- report executable files. *See* spreadsheet object executable files
- report file types 605
- report files
  - linking to 392
  - publishing 398
  - setting privileges for 384
- report function builder. *See* function builder
- Report Function command 175
- Report Function page (Format Sheet) 175, 182
- Report Functions
  - See also* functions
- report functions 245, 439, 442, 627
  - See also* functions
- report macros 245–247
- Report Parameter command 328

- Report Parameter dialog
  - creating defined names and 326
  - creating parameters and 318, 319, 321
  - entering help text on 326
  - hiding user input and 325, 326
  - selecting display type from 322, 324
  - selecting null or blank values and 326
- report parameters 627
  - defined 627
- report parameters. *See* parameters
- report ranges. *See* data ranges
- report sections. *See* sections
- report specifications. *See* report designs
- ReportCallback interface 478
- reporting server. *See* BIRT iServer
- reporting system. *See* BIRT iServer System
- reports 384, 390, 395, 483, 626
  - See also* persistent reports; temporary reports
  - See also* spreadsheet reports
  - accessing ODA data sources for 53
  - creating for
    - information objects 100
  - defined 626
  - running 403
- repositories 627
- REPT function 567
- requester 627
- Requester Page dialog
  - changing system parameters and 331
  - deploying reports and 403
  - hiding parameters and 326
  - hiding user input and 326
  - requesting null or empty values and 326
  - testing parameters and 330
- Required property 150
- required values 89
- reserved words. *See* keywords
- resizing
  - bubbles 298
  - charts 266
  - columns or rows 182–183
  - doughnut centers 301
- resource groups 486, 487
- resources 628
- restarting BIRT Spreadsheet Designer 485
- restricting access to reports 40, 380
- restricting data entry 247
- restricting user actions 372, 373, 374, 376
- result sets 219, 224, 463, 473, 628
  - aggregating values in 86
  - changing 57
  - changing column order in 115
  - defined 628
  - defining multiple conditions for 130, 130–132
  - defining output columns for 115–116
  - excluding duplicate rows from 115
  - filtering data for 87
  - generating computed fields for 117
  - handling null values in 117
  - joining tables and 78
  - missing values in 145
  - previewing 57, 88, 98
  - previewing data in 112, 156
  - removing output columns from 116
  - restricting number of rows in 121, 144
  - returning distinct values for 115
  - setting sort order for 94
  - sorting data in 94
  - viewing output columns in 154, 155, 156
- return values 498
- reversing axes values 309
- RGB color values 444, 450
- RIGHT function 567
- right outer joins 118
- right outer joins. *See* outer joins
- RIGHTB function 567
- roles 380, 630
  - See also* privileges
- rollup function 232, 462
- ROMAN function 567
- rotating
  - objects 574
  - text 575
- ROUND function 567
- round function 463
- ROUNDDOWN function 567
- rounding 424, 463, 576
- ROUNDUP function 567
- row areas 628
  - See also* data ranges
- row fields 628
  - See also* pivot ranges



- adding 341, 342
- defined 335
- hiding 350, 355
- labeling 349
- removing 342
- selecting 340
- showing top or bottom values in 366
- ROW function 567
- row function 463
- row headings 427
- row IDs 463
- row numbers 181, 463, 575
- row outlines 202, 203
- row section area 20
- row sections 19, 39, 173
  - See also* sections
- row stubs 20
- rownum function 463
- rows 628
  - See also* data rows
  - adding to reports 39, 180
  - adding to result sets 224
  - changing display order of 94
  - changing number returned 57
  - combining values from multiple 85
  - compatibility with Excel 575
  - counting 444
  - defined 628
  - defining multiple conditions for 131
  - deleting 181
  - excluding duplicate 115
  - hiding 181
  - previewing 112, 156
  - removing from result sets 224
  - reordering 177
  - resizing 182–183
  - restricting number returned 121, 144
  - retrieving from text files 161, 163
  - retrieving multiple 174
  - setting height 468
  - setting range limits for 427
- ROWS function 567
- RSQ function 567
- rules 247, 425, 575
- run 629
- Run command 44
- Run icon 44

- run time 629
- Run with parameters command 44
- Run with parameters icon 44
- Run... command 330
- running aggregates 629
- running executable files 317
- running queries 66
- running reports 15, 403, 483, 487
- Running Total setting 359
- running totals 359
- run-time connections 404
- run-time parameters 39, 89

## S

- sample data 265, 277, 282, 283
- sample databases 51
- SAP data sources 336
- Save Design As command 15
- Save Design command 15
- Save View As command 15
- saving
  - queries 11, 83
  - report designs 15
  - workbooks 416
- saving queries 111, 153
- Scale command 308
- Scale dialog 307, 308, 310
- scaling options 427
- scaling options (charts) 299, 307
- scatter charts 257, 276
- scatter charts. *See* charts
- scheduling reports 488
- schemas 629
  - See also* ROM schemas
  - configuring connections and 80, 408
  - defined 629
- scientific data sets 259
- scope 629
- scroll bars 427
- scrolling 35
- SDK package 630
- search 630
- search conditions 438
  - See also* search expressions
- search criteria. *See* search conditions
- search expressions 233

- SEARCH function 567
- SEARCHB function 567
- searching spreadsheet reports 233
- second format symbol 554
- SECOND function 567
- second function 464
- Section Filtering and Grouping dialog
  - adding detail rows and 218
  - filtering data and 220, 224
  - grouping data and 23, 174
- section headers 194
- section identifiers 239
- section names 173, 174, 179, 239
- section references 235, 239
- sectioncount function 464
- sections
  - adding 19, 22, 173
  - applying grant expressions to 383
  - changing 179
  - creating data ranges and 173, 184, 231
  - creating summary values and 227, 229
  - deleting 179, 446
  - entering functions in 214, 219, 239, 438
  - grouping 21, 23, 174–177
  - hiding 181–182, 183
  - inserting data in 25, 215
  - inserting rows in 180
  - laying out reports and 39, 168
  - merging cells in 193, 194
  - naming 21, 22, 173
  - referencing cells in 235, 239
  - renaming 177, 179
  - resizing rows or columns in 183
  - returning groups in 453
  - returning number of 464
  - sorting data in 176, 177
- sectors. *See* doughnut charts; pie charts
- secure read privilege 630
- Securities database 51
- security 40, 372
  - See also* encryption; passwords
- Security command 383
- security function 380, 464
- security IDs 630
  - adding 380
  - displaying 380, 464
  - generating sample 383
- security roles 380, 630
  - See also* privileges
- select 630
- select function 219, 225, 241, 465
- select painter. *See* query editor
- Select Rows page (Delimited Text File) 161
- Select Rows page (Positional Text File) 163
- SELECT statements 633
  - adding expressions to 106
  - copying 83
  - disabling automatic grouping and 143
  - entering manually 83
  - grouping data and 140, 141, 143, 144
  - removing columns from 142, 143
  - retrieving data with 76
  - sorting data and 94
- series 631
  - adding 282, 291
  - changing appearance of 265, 291, 295
  - changing order of 294
  - comparing 252, 256, 257
  - connecting shapes for 297
  - contrasting 254
  - defined 631
  - describing 311
  - displaying data labels for 311
  - displaying scientific data and 259
  - displaying y-error bars in 293
  - formatting data points in 285, 291
  - formatting numeric 291
  - hiding labels for 293
  - highlighting 291, 292
  - overlapping multiple 252
  - plotting data for 274
  - plotting multiple 297
  - plotting on second y-axis 292
  - relating to axes values 304
  - selecting chart types and 275
  - selecting layout options for 282
  - setting shapes for 297
  - showing relationship to whole 257
  - showing relationships between 255, 266
  - spacing data points for 296, 297
  - stacking 294
- series labels 250
- series line options 298
- series markers 291, 292

- Series order dialog 294
- Series page 282
- SERIESSUM function 567
- servers 582
- ServerUri property 101, 405
- Set Default command 65
- Set Security command 383
- setAdvancedConnectionProperties
  - method 409
- setChart function 465
- setEntry function 244, 466
- SetPrintAreas callback example 498–502
- SetPrintAreas.sod 499
- setting passwords. *See* passwords
- sfddata database 51, 530
- shadows (chart elements) 290
- Shape command 297
- shapes 536
  - See also* graphical objects
- shared access type 581
- sheet function 186, 467
- sheet-level functions 214
- sheet-level groups 174, 175, 186, 236
- sheets. *See* worksheets
- shifting cells 181
- Show bar as line setting 297
- Show Detail command 352, 354
- Show Detail dialog 352
- Show Field List command 340
- Show Field List icon 339, 340
- Show items with no data setting 355
- Show Only Full Translations setting 547
- Show Tables dialog box 62, 80, 93
- Show tee-top marker setting 293
- Show tips command 285
- side-by-side charts 266
- SIDs. *See* security IDs
- SIGN function 567
- SIN function 567
- single quotation mark (') character
  - cell references and 240
  - report functions and 434
- single-row result sets 463
- SINH function 567
- size function 175, 182, 183, 467
- Size property 150
- SKEW function 567
- slices. *See* doughnut charts; pie charts
- SLN function 567
- SLOPE function 567
- SMALL function 568
- SmartSheet security 41
- SmartSheets Security Option 379
- SmartSheets. *See* SmartSheet Security Option
- .sod files. *See* spreadsheet object design files
- Software Development Kit 630
  - See also* JDK software; SDK package
- .soi files. *See* spreadsheet object instance files
- Solve Order command 362
- sort 631
- Sort and Top 10 command 365, 366
- sort keys 631
  - selecting 94
- sort order 109
  - calculating pivot range values and 362
  - changing 176
  - creating pivot ranges and 365
  - removing 94
  - setting 94
  - specifying 366, 437
- sort-and-group-by fields. *See* group keys
- sorting
  - data 176, 365
  - groups 452
  - values in lists 324, 325, 437
- sorting data 109
  - in result sets 94
  - with queries 94
- sorting fields 177
- source code
  - adding VBA functionality and 536, 541
  - binding to spreadsheets 541
  - creating callback classes and 492, 498
  - creating multi-class callbacks and 489, 490
  - debugging callback classes and 483, 485
  - developing workbooks and 481
  - executing at run time 485
  - overriding locale settings and 547
  - testing 537
  - testing and debugging 599
  - triggering 536
- Source Data command 280
- Source Data dialog 280, 282, 284
- source data. *See* data sources

- source files 46
- source parameters 151–153
- .sox files. *See* spreadsheet object executable files
- Spanish translations 547
- special characters 356, 374, 549
- spreadsheet designer 103
- spreadsheet designer. *See* BIRT Spreadsheet Designer
- spreadsheet designer. *See* e.Spreadsheet Designer
- spreadsheet designs. *See* designs
- spreadsheet engine. *See* BIRT Spreadsheet Engine
- spreadsheet files 46, 518, 574
  - See also* specific type
- spreadsheet object design files 46, 401, 632
- spreadsheet object executable files 632
  - creating 398
  - generating 44, 46
  - publishing 400
  - running 317
- spreadsheet object instance files 45, 46, 317, 384, 632
- spreadsheet object instances 632
- spreadsheet reports 632
  - accessing text file data and 158
  - adding static content to 244–245
  - adding VBA functionality to 536, 537, 539
  - changing 483
  - changing data sources for 51, 403
  - compatibility with Excel 574
  - connecting to data sources for 52, 403
  - creating 6, 16
  - creating queries for 77–83
  - customizing 316
  - defining default data set for 64, 65
  - defining summary data for 85
  - deploying 41, 45, 398, 483
  - designing 4, 38, 41, 208, 316
  - developing 51, 247
  - extending functionality of 478, 574
  - formatting 30, 40, 190, 195
  - generating Excel files from 45
  - linking to 395
  - localizing 546, 549, 550, 554
  - managing large 316
  - modifying columns in 93
  - optimizing 428
  - overriding locale settings for 547
  - personalizing 41
  - previewing 15, 30, 34, 44–45
  - protecting 372
  - publishing 398–401, 537
  - reducing size of 316
  - restricting access to 40, 380
  - retrieving data for 50, 51, 70, 93
  - running 15
  - running sample databases for 51
  - searching 233
  - setting predefined connections for 404
  - structuring 39, 42, 168, 169
  - testing callback classes for 483
  - updating code names for 541
  - validating data for 247
- spreadsheet server. *See* BIRT iServer
- spreadsheet templates. *See* templates
- spreadsheets. *See* Excel spreadsheets; worksheets
- spreadsheets. *See* worksheets
- SQL (defined) 633
- SQL editor
  - creating queries and 83
  - opening 83
- SQL Editor button 106, 153
- SQL expressions
  - adding 106
  - converting column names to 118
  - creating joins and 119
  - defining output columns and 115
  - entering characters in 125
  - entering source parameters in 151
  - filtering data and 122, 124, 129, 135
  - generating computed fields and 117
- SQL language 633
- SQL languages 598
- SQL operators
  - filter conditions 123, 124, 126
  - joins 118
- SQL page 633
- SQL page (Stored Procedure) 98
- SQL parameters 145
- SQL Preview pane 112, 154
- SQL Server databases 71

- SQL statements 633
  - adding column names to 76
  - adding filters to 87, 88
  - adding parameters to 89, 90
  - creating 83
  - defined 633
  - defining joins with 117–121
  - defining multiple conditions in 130, 130–132
  - displaying 78, 112
  - entering manually 78
  - filtering data with 110, 121–138, 144
  - grouping data with 138–144
  - hiding columns in 94
  - removing columns in 94
  - returning distinct values for 115
  - selecting columns for 81, 82, 93
  - selecting tables for 77, 93
  - sorting data with 109
- SQL statements. *See* queries
- SQL.REQUEST function 568
- SQLREQUEST function 576
- SQRT function 568
- SQRTPI function 568
- square brackets ([ ]) characters
  - cell references and 240
  - report functions and 434, 435
- stacking chart series 294
- standard deviation 468, 469
- STANDARDIZE function 568
- start method 478, 479
- starting Query Builder 104
- starting values 363, 364
- Startrow property 407
- statement objects 634
- statements 634
- static content 244, 437
  - See also* text
- static file names 402
- static parameters 634
  - See also* global variables
  - adding to queries 89
  - assigning values to 89
  - creating 89
- static variables 634
  - See also* dynamic variables; variables
- STDEV function 568
- stdev function 468
- STDEVA function 568
- STDEVP function 568
- stdevp function 469
- STDEVPA function 568
- step charts
  - See also* charts
  - creating 258
  - plotting percentages in 295
  - plotting values for 276
  - stacking series for 294
- STEYX function 568
- stock charts
  - See also* charts
  - adjusting tick marks on 308
  - creating 259
  - plotting values for 276
- Stored Procedure Browser 97
- stored procedures 96, 150, 177, 634
- string calculations 638
- string data type 635
- string expressions 635
- strings 635
  - assigning to parameters 152
  - combining with dynamic data 245
  - comparing 87
  - comparing patterns in 127
  - concatenating 130, 434
  - converting to dates 470
  - converting to numbers 471
  - creating grant expressions and 381
  - creating QBE expressions and 136
  - entering functions and 434, 448
  - extracting subsets of 245, 448
  - filtering data and 220
  - removing whitespace in 472
  - replacing substrings in 469
  - returning length 456
  - setting default values and 146
  - testing for blank values in 127
- structured content 38, 212, 244, 635
- Structured Query Language. *See* SQL
- study charts 251, 259, 306, 635
- styles 635
  - See also* formats
  - defined 635
- subcharts 251

- SUBSTITUTE function 568
- substitute function 469
- substrings 245, 448, 469
- subtitles 192
- Subtotal dialog 229
- SUBTOTAL function 226, 229, 568
- subtotals
  - adding to worksheets 39
  - calculating 226, 232
  - changing 360
  - creating data ranges and 169
  - creating pivot ranges and 335, 360
  - displaying 203
  - filtering data and 366
  - grouping data and 364
  - hiding 360
  - including hidden items in 360
  - specifying location of 345
- Subtotals command 360
- subtraction operator 436, 437
- Sum dialog 227
- SUM function 226, 227, 340, 568
- sum function 86, 226, 470
- SUMIF function 568
- Summary Columns Before Detail setting 203
- summary data
  - See also* totals; subtotals
  - displaying 201, 203
  - entering formulas for 435
  - generating 226, 232, 340, 470
  - specifying location of 203
  - viewing formulas for 226
- summary expressions 226, 230
- summary fields 340
  - See also* data fields
- summary rows 169, 203, 226
- Summary Rows Before Detail setting 203
- summary values 85
  - See also* summary data
- SUMPRODUCT function 568
- SUMSQ function 568
- SUMX2MY2 function 568
- SUMX2PY2 function 568
- SUMXMY2 function 568
- supported languages 548
- Sybase databases 71
- SYD function 568

- symbols 549
- synchronization 625
- synchronizing source parameters 152–153
- syntax (programming languages) 636
- syntax errors 112
- system locales 550
- System page (Requester Page) 331
- system parameters 331, 403
- System Parameters tab (Requester Page) 331

## T

- T function 568
- tabbed text file-compatibility settings 416
- tab-delimited text files 158, 416
- table names 155, 339
  - creating computed fields and 85
  - creating joins and 91
- Table Options command 347
- tables 636
  - See also* databases
  - adding to queries 77, 93
  - choosing columns from 82, 93
  - creating 61
  - defined 636
  - joining 78, 82, 91
  - restricting columns retrieval from 94
  - retrieving data from 8, 41
  - retrieving subset of data in 61
  - selecting 9, 17
- tabs 636
  - See also* page
- tabs. *See* worksheet tabs
- tabular layouts 344, 345
- tags 636
  - See also* elements
- TAN function 568
- TANH function 568
- TBILLEQ function 569
- TBILLPRICE function 569
- TBILLYIELD function 569
- TDIST function 569
- template attribute 75
- template files 536, 537
- templates 251, 345, 536, 574
- testing
  - arrays 436

- BookModel objects 497
- callback classes 483–488, 498
- conditions 438
- connections 6, 52, 400
- grant expressions 383
- null values 455
- outlines 206
- report parameters 330
- reports 487
- VBA templates 537
- text
  - See also* delimited data
  - adding 244–245, 466, 474
  - aligning 196, 290
  - centering 33, 349
  - changing orientation of 290
  - converting date or number values to 472
  - converting to lowercase 457
  - converting to uppercase 473
  - formatting 30, 32
  - locking to graphical objects 376
  - plotting for charts 274, 288
  - resizing chart 287
  - returning values for 437
  - rotating 575
  - translating spreadsheet reports and 570
- text attributes 33, 196, 287, 290, 513
- text boxes 136, 321
- text editors 428
- text expressions 434
  - See also* strings
- text file data sources 336
- text files
  - accessing data in 158
  - changing data types for 161, 164
  - connecting to 159, 406
  - creating hyperlinks for 391
  - defining data sets for 160–164
  - formatting data in 407
  - retrieving data from 50, 56, 407
  - saving workbooks as 416
  - selecting 159
  - setting column breaks for 163
  - setting delimiters for 161, 406
  - setting encoding options for 161, 163
- Text Format property 117
- TEXT function 569
- text patterns. *See* search expressions
- text strings 635, 638
- text strings. *See* strings
- Textqualifier property 406
- textual data. *See* text
- textual queries
  - creating 83, 153–155
  - displaying output columns for 155
  - displaying parameters for 155
  - filtering data with 112, 125, 132
  - prompting for values and 135, 137
  - saving 153
- textual query editor 105, 106, 153, 154, 636
- 3D chart types 251
  - See also* charts with depth
- three-dimensional ranges 637
  - See also* spreadsheet reports
- tick 637
- tick interval 637
- tick mark labels 306, 307
- tick marks 251, 275, 306, 308
- tilde (~) character 438
- time 127
- time continuums 252, 258
- time fields 362
- time format symbols 554
- TIME function 569
- time function 470
- time values
  - grouping 363
  - plotting in charts 274, 310
  - returning current 460
  - returning numeric values for 470
  - returning subsets of 454, 458, 464
- time-scale axis 307, 308, 310
- timestamps
  - defining 152
  - setting default values and 146
- TIMEVALUE function 569
- TINV function 569
- tips 285
- title elements (charts) 250, 265, 290
- titles
  - adding to charts 265, 277, 284
  - creating multi-column 192, 193
  - formatting report 30
  - setting font attributes for 290

- Titles page 284
- today function 470
- TODAY function 569
- tonumber function 471
- toolbars 414, 536, 637
- ToolTips 319, 326
- Top 10 AutoShow options 367
- total fields 335, 342
  - See also* pivot ranges
- totals 86
  - adding to worksheets 39
  - calculating 226, 232, 462, 470
  - creating data ranges and 169
  - creating pivot ranges and 352, 360, 362
  - displaying 203
  - displaying grand 361, 368
  - displaying percentages of 295
  - displaying running 359
  - returning from functions 435
- toText function 471
- TQE. *See* textual query editor
- trailing spaces 128, 472
- translation settings 415, 416, 418, 419
- translations 546, 549, 550, 554
- TRANSPOSE function 569
- Treatconsecutivedelimitersasone
  - property 406
- TREND function 569
- trendline elements (charts) 251, 265, 312, 314
- TRIM function 569
- trim function 472
- TRIMMEAN function 569
- TRUE function 569
- TRUNC function 569
- trunc function 472
- truncated messages 111
- trusted execute privilege 637
- try-catch blocks 481
- TTEST function 569
- tutorials 4
- twips 183
- .txt files. *See* text files
- TYPE function 569
- type markers 420, 421
- types. *See* data types
- types. *See* data types; semantic types

## U

- unary minus operator 437
- unary plus operator 437
- Undo command 412
- undo options 412
- Undock command 414
- undocking toolbars 414
- uneven data hierarchies 232
- Unhide command 181
- Unicode character sets 576
- Unicode characters 638
- Unicode encoding 638
- Unicode file-compatibility settings 416
- Unicode standard 638
- Unicode values 58
- uniform resource locators. *See* URLs
- UNION keyword 153
- unique values 447
- universal hyperlinks. *See* hyperlinks
- unknown data types 116
- unlocking workbooks 481
- unlocking worksheet cells 376
- up or down bars (charts) 305
- updating
  - charts 443
  - class files 485
  - code names 541
  - data ranges 281
  - data sets 67
  - pivot ranges 343, 349, 354
- updating Encyclopedia volumes 113
- uploading report files 398
- upper case characters 473
- UPPER function 569
- upper function 473
- URIs 405
- URLs 104, 393, 638
  - defined 638
  - information objects and 101
  - JDBC data sources and 71, 75, 408
  - text files and 159, 406
- USDOLLAR function 569
- UseLoggedInUserCredentialsOnServer
  - property 102
- user accounts 105
- user interfaces 546, 547, 574



- See also* application window
- User name property 105
- user names 105, 130, 638
  - accessing information objects and 101
  - connecting to BIRT iServer and 406
  - connecting to JDBC data sources and 408
- User property 408
- User's ACL dialog 384
- UserName property 101, 406
- users
  - accessing reports and 40
  - creating passwords for 374
  - creating security IDs for 380, 383
  - developing interactive features for 536
  - displaying security IDs for 380, 464
  - prompting for input 317, 321, 322, 324
  - publishing reports and 398
  - restricting actions for 372, 373, 374, 376
- using
  - multiple series in bar and column charts 269

## V

- validating data 247
- Validation page (Format Cells) 247
- validation rules 247, 575
- #VALUE! error 555
- value axis
  - See also* y-axis values
  - adjusting intersection point for 307, 308
  - changing appearance of 265
  - changing intervals on 309
  - defining logarithmic 310
  - displaying data on 250, 274, 295
  - hiding 306
  - marking segments of 251
  - plotting percentages and 295
  - reversing order of 309
  - scaling 307
  - stacking series values and 294
- value axis labels 250, 291
  - See also* axis labels
- value axis titles 250, 284
  - See also* axis titles
- value formats 450
- VALUE function 569

- values 638
  - See also* data
  - applying as conditions 197
  - applying conditions to 455
  - assigning to parameters 89, 145, 146, 151, 320, 323
  - comparing 87, 334
  - computing maximum 457
  - computing minimum 458
  - counting 444
  - counting non-null 86
  - creating display names for 323
  - creating list of 135, 137, 148
  - defined 638
  - defining for pivot ranges 335
  - defining required 89
  - displaying data type markers for 421
  - displaying parameter 44
  - excluding 225
  - filtering empty or blank 126, 127
  - filtering on multiple 125, 128, 130
  - handling null 240–244
  - hiding 149
  - highlighting multiple 253, 254
  - locating matching 233
  - masking 326
  - plotting maximum or minimum 294, 295, 310
  - prompting for 134, 135, 136, 147, 151, 317, 321, 322, 324
  - recalculating 424, 425
  - retrieving 316, 473
  - returning distinct 115
  - returning non-null 449, 456
  - returning unique 447
  - selecting 137, 322, 326
  - setting at run time 89
  - setting control types for 137
  - setting default 110, 137, 146
  - setting restrictions for 247
  - showing difference between 254, 359
  - showing over time 252, 258
  - showing relationships between 250, 274
  - sorting in lists 324, 325, 437
  - specifying default 321, 322, 323, 325
  - specifying precision for 424, 425
  - testing for null 117, 126, 455

- values function 473
- VAR function 569
- var function 230, 474
- VARA function 569
- Varchar data type 638
- variable hyperlinks 390, 394, 639
- variables 638
  - customizing JDBC drivers and 75
  - defined 638
- variance 230, 474
- variance data 359
- variant data 150
- VARP function 569
- varp function 474
- VARPA function 570
- VBA code 536, 537, 541
- VBA code names 467, 541, 542
- VBA functionality 45, 536, 539, 579
- VBA page (Format Sheet) 542
- VBA templates
  - associating with reports 536, 541
  - creating 536
    - example for 538–541
  - extending spreadsheet functionality and 574
  - testing 537
- VDB function 570
- Verify password dialogs 374
- version options 401
- View page (Format Sheet) 426
- view time 639
- View with Excel command 45
- View with Excel icon 45
- viewer 639
- viewing
  - column categories 114
  - column headings 427
  - columns 108
  - computed fields 59, 85
  - custom formats 191
  - data 19, 35, 66, 112, 156, 201, 212, 317
  - data labels 311
  - data set fields 57, 82, 83
  - data type markers 421
  - debugging messages 483, 489
  - error messages 111, 355, 447, 483, 489
  - grid lines 306
  - group keys 174
  - hidden toolbars 414
  - information objects 107, 113
  - join conditions 118
  - logging messages 484
  - outlines 427
  - parameter values 44
  - parameters 108, 154
  - pivot range field list 340
  - pivot range items 350, 351, 355
  - query output 11, 154, 155, 156
  - report parameters 155, 318
  - row headings 427
  - sample formats 191
  - section groups 174
  - section names 173
  - security IDs 380, 464
  - SQL statements 78, 112
  - system parameters 331, 403
  - worksheet grids 427
  - zero values 427
- viewing options. *See* display options
- viewing scale 427
- views 80
  - accessing 41
  - creating data sets and 8
  - defined 639
  - publishing reports and 401, 403
  - saving 15
- view-time parameters 39, 317
- view-time queries 66
- virtual cell references 639
  - See also* cell references
- virtual defined names. *See* defined names
- virtual security IDs 380
- Visibility command 377
- visible privilege 639
- Visible Sheets dialog 377
- Visual Basic for Applications. *See* VBA
- VLOOKUP function 570
- .vmoptions file 428
- Volume property 101, 104, 406
- volumes. *See* Encyclopedia volumes
- .vtf files. *See* spreadsheet object executable files
- .vts files. *See* spreadsheet object design files
- .vtx files. *See* spreadsheet object design files

## W

wall (charts) 251

.war files 640

defined 640

weak encryption type 379

web archive files 640

web archive files. *See* .war files

web browsers 398

web pages 390, 392, 640

defined 640

WEEKDAY function 570

WEEKNUM function 570

WEIBULL function 570

WHERE clause 80, 87, 112, 122

whitespace (strings) 472

wildcard characters 438

wildcard expressions 436, 438

windows, preserving 575

WITH clause 145

Word documents 391

word processing applications 390

Word Wrap property 117

WordFormatting.callback example 513–514

WordFormatting.sod 513

workbook names 541, 542

Workbook Properties command 421, 542

Workbook Properties dialog 542

workbooks 64, 640

*See also* spreadsheet reports

attaching callback classes to 478, 485, 489, 496

changing appearance of 497

changing code names for 542

designing reports and 43

developing 478, 496

hiding worksheets in 377

linking to 391

locking 481

overriding locale settings for 547

overview 208

preserving state 575

protecting 372, 374

referencing multiple 239–240

restricting data entry in 247

saving 416

setting evaluation rules in 426

setting passwords for 374

setting properties for 420–424, 497

WORKDAY function 570

worksheet names 467, 541, 542

worksheet tabs

hiding 377

setting location of 420

worksheets 64, 640

*See also* Excel spreadsheets; spreadsheet reports

adding data ranges to 184, 186

adding static content to 244, 245

calculating data in 426

changing appearance of 426–427

changing code names for 542

compatibility with Excel 575, 578

counting cells in 576

creating pivot ranges and 336, 337

customizing colors for 421

deleting 446

entering functions in 213, 435

exporting 575

grouping across 174

hiding 377

hiding column or rows in 181

hiding data in 287

hiding formula bar in 420

hiding scroll bars in 427

linking to 390

linking to charts in 270

locking 457

locking cells on 457, 507

overview 208

protecting 372, 375–377

referencing multiple 239–240

renaming 541

resizing columns or rows in 182–183

restricting user actions in 372, 373, 374, 376

returning groups in 453

setting passwords for 375

setting print areas for 498

setting properties for 426

validating data in 247

workspace folders 46

write function 474

write privilege 640

## X

- x-axis settings 306
- x-axis values 250, 274
  - See also* axes values; category axis
- XIRR function 570
- .xls files 45, 401, 537
  - See also* Excel spreadsheets
- XML (defined) 604
- XML data sources 336
- XML elements 601
- XML files 390
  - setting connection properties in 404, 409
- XML parser 549
- XML schemas 408, 629
  - See also* ROM schemas
  - defined 629
- XNPV function 570
- xor operator 436

## Y

- Y Error Bars command 293

- Y Error Bars dialog 293
- y-axis settings 306
- y-axis values 250, 259, 274, 292
  - See also* axes values; value axis
- year format symbol 554
- YEAR function 570
- year function 474
- YEARFRAC function 570
- y-error bars 293
- YIELD function 570, 576
- YIELDDISC function 570
- YIELDMAT function 570

## Z

- z-axis values 251
- zero values 243, 288, 427
- Zoom command 427
- zoom settings 427
- ZTEST function 570