

Oracle® Reports Developer Reference

Release 6i

JANUARY, 2000

Part No. A73174-01

ORACLE®

Oracle® Reports Reference Release 6i

The part number for this volume is A73174-01

Copyright © 1999, Oracle Corporation. All rights reserved.

Contributors: Carol Menzigian, Frank Rovitto, Tasha Kelly

The programs are not intended for use in any nuclear, aviation, mass transit, medical, or other inherently dangerous applications. It shall be licensee's responsibility to take all appropriate fail-safe, back up, redundancy and other measures to ensure the safe use of such applications if the Programs are used for such purposes, and Oracle disclaims liability for any damages caused by such use of the Programs.

This Program contains proprietary information of Oracle Corporation; it is provided under a license agreement containing restrictions on use and disclosure and is also protected by copyright, patent and other intellectual property law. Reverse engineering of the software is prohibited.

The information contained in this document is subject to change without notice. If you find any problems in the documentation, please report them to us in writing. Oracle Corporation does not warrant that this document is error free. No part of this document may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without the express written permission of Oracle Corporation

If this Program is delivered to a U.S. Government Agency of the Department of Defense, then it is delivered with Restricted Rights and the following legend is applicable:

Restricted Rights Legend Programs delivered subject to the DOD FAR Supplement are "commercial computer software" and use, duplication and disclosure of the Programs shall be subject to the licensing restrictions set forth in the applicable Oracle license agreement. Otherwise, Programs delivered subject to the Federal Acquisition Regulations are "restricted computer software" and use, duplication and disclosure of the Programs shall be subject to the restrictions in FAR 52.227-14, Rights in Data -- General, including Alternate III (June 1987). Oracle Corporation, 500 Oracle Parkway, Redwood City, CA 94065.

Oracle is a registered trademark, and Express, Oracle Browser, Oracle Forms Developer, Oracle Graphics, Oracle Installer, Oracle Reports Developer, Oracle7, Oracle8, Oracle Web Application Server, Personal Oracle, Personal Oracle Lite, and PL/SQL are trademarks or registered trademarks of Oracle Corporation. Portions copyright © Blue Sky Software Corporation. All rights reserved. All other company or product names mentioned are used for identification purposes only and may be trademarks of their respective owners.

Table of Contents

BUILT-INS	1
Built-in packages	2
SRW.SET_ATTR attributes	3
ACTION	4
AFTCODE	5
AFTFORM_ESCAPE	6
AFTPAGE_ESCAPE	7
AFTREPORT_ESCAPE	9
BBCOLOR	11
BEFCODE	12
BEFFORM_ESCAPE	13
BEFPAGE_ESCAPE	14
BEFREPORT_ESCAPE	16
BFCOLOR	18
BOOKMARK	19
BORDERWIDTH	21
BORDPATT	22
FACE	23
FBCOLOR	24
FFCOLOR	25
FILLPATT	26
FORMATMASK	27
GCOLOR	28
GSPACING	29
HJUST	30
HYPERLINK	31
LINKTAG	34
PRINTER_INTRAY	36
STYLE	37
SZ	38
TEXT	39
WEIGHT	40
The Report Builder PL/SQL package (SRW)	41
SRW.ADD_DEFINITION	42
SRW.APPLY_DEFINITION	43

SRW.BREAK-----	44
SRW.CONTEXT_FAILURE-----	45
SRW.DO_SQL-----	46
SRW.DO_SQL_FAILURE-----	49
SRW.FILE_ESCAPE-----	50
SRW.GETERR_RUN-----	51
SRW.GET_PAGE_NUM-----	52
SRW.TEXT_ESCAPE-----	53
Set Attributes Procedures -----	54
SRW.SET_AFTER_FORM_HTML-----	55
SRW.SET_AFTER_PAGE_HTML-----	56
SRW.SET_AFTER_PRINTING_CODE-----	58
SRW.SET_AFTER_REPORT_HTML-----	59
SRW.SET_BACKGROUND_BORDER_COLOR-----	61
SRW.SET_BACKGROUND_FILL_COLOR-----	62
SRW.SET_BEFORE_FORM_HTML-----	63
SRW.SET_BEFORE_PAGE_HTML-----	64
SRW.SET_BEFORE_PRINTING_CODE-----	66
SRW.SET_BEFORE_REPORT_HTML-----	67
SRW.SET_BOOKMARK-----	69
SRW.SET_BORDER_PATTERN-----	71
SRW.SET_BORDER_WIDTH-----	72
SRW.SET_CHARSMODE_TEXT-----	73
SRW.SET_CUSTOM_SPACING-----	74
SRW.SET_DISPLAY_NAME-----	75
SRW.SET_FIELD-----	76
SRW.SET_FIELD_CHAR-----	78
SRW.SET_FIELD_DATE-----	79
SRW.SET_FIELD_NUM-----	80
SRW.SET_FILL_PATTERN-----	81
SRW.SET_FONT_FACE-----	82
SRW.SET_FONT_SIZE-----	83
SRW.SET_FONT_STYLE-----	84
SRW.SET_FONT_WEIGHT-----	85
SRW.SET_FOREGROUND_BORDER_COLOR-----	86
SRW.SET_FOREGROUND_FILL_COLOR-----	87
SRW.SET_FORMAT_MASK-----	88
SRW.SET_HYPERLINK-----	89
SRW.SET_HYPERLINK_ATTRS-----	92
SRW.SET_JUSTIFICATION-----	93
SRW.SET_LINKTAG-----	94
SRW.SET_MAXROW-----	95

SRW.SET_PAGE_NAVIGATION_HTML	97
SRW.SET_PDF_ACTION	98
SRW.SET_PRINTER_TRAY	99
SRW.SET_SPACING	100
SRW.SET_TEXT_COLOR	101
SRW.INTEGER_ERROR	102
SRW.MAXROW_INERR	103
SRW.MAXROW_UNSET	104
SRW.MESSAGE	105
SRW.NULL_ARGUMENTS	107
SRW.PROGRAM_ABORT	108
SRW.REFERENCE	109
SRW.RUN_REPORT	110
SRW.RUN_REPORT_BATCHNO	113
SRW.RUN_REPORT_FAILURE	114
SRW.SET_ATTR	115
SRW.TRACE_ADD_OPTION	120
SRW.TRACE_END	121
SRW.TRACE_REM_OPTION	122
SRW.TRACE_START	123
SRW.TRACEOPTS.MASK	124
SRW.TRUNCATED_VALUE	125
SRW.UNKNOWN_QUERY	126
SRW.UNKNOWN_USER_EXIT	127
SRW.USER_EXIT	128
SRW.USER_EXIT20	130
SRW.USER_EXIT_FAILURE	131

TRIGGERS ----- 133

Which report trigger to use ----- 133

Report trigger order of execution----- 135

Group filter	138
Formula	140
Validation trigger	142
Format trigger	144
Action trigger	148
Ref cursor query	149
After Parameter Form trigger	152
After Report trigger	153

Before Parameter Form trigger -----	154
Before Report trigger -----	155
Between Pages trigger -----	156

PROPERTIES ----- 157

Oracle8 usage notes ----- 158

About the Property Palette ----- 160

Anchor properties ----- 165

Child Edge Percent -----	167
Child Edge Type -----	168
Child Object Name -----	169
Collapse Horizontally -----	170
Collapse Vertically -----	171
Parent Edge Percent -----	173
Parent Edge Type -----	174
Parent Object Name -----	175

Boilerplate properties ----- 176

Contains HTML Tags -----	177
Minimum Widow Lines -----	178
Source File Format -----	179
Source Filename -----	180
Type -----	182
Line Stretch with Frame -----	183

Button properties ----- 185

Label Type -----	187
Text -----	188
Icon Name -----	189
Type -----	190
Multimedia File -----	191
Multimedia File Type -----	192
Multimedia Column -----	193
Multimedia Column Type -----	194
PL/SQL Trigger -----	195

Chart properties ----- 196

Chart Filename -----	198
Chart Hyperlink -----	199

Chart Parameters and Columns properties ----- 200

Chart Parameter -----	201
Report Column (for Chart Parameter) -----	202

Chart Query -----	203
Report Group-----	204
Chart Column -----	205
Report Column (for Chart Column) -----	206
Common column properties-----	207
Break Order -----	208
Column Type-----	210
Comment -----	211
Database Column Name -----	212
Datatype -----	213
File Format -----	214
Name -----	215
Read from File -----	216
Value If Null -----	218
Width-----	219
Set Break Order-----	221
Common Layout Object properties -----	222
Comments -----	224
Horizontal Elasticity -----	225
Keep With Anchoring Object -----	227
Name -----	229
Page Break After -----	231
Page Break Before-----	232
Page Protect -----	233
Base Printing On -----	235
Print Object On -----	236
Printer Code Before -----	239
Printer Code After -----	240
Vertical Elasticity -----	241
Hyperlink -----	245
Hyperlink Destination -----	247
Bookmark-----	248
Application Command Line (PDF) -----	249
Conditional Formatting -----	250
Display Name -----	251
End of Layout Section -----	252
Additional Hyperlink Attributes -----	253
Database Column properties-----	254
Field properties-----	255
Source Datatype -----	257
Format Mask -----	258
Date and Time Format Mask syntax -----	259

Number Format Mask syntax	261
Visible	265
Page Numbering	267
Source	270
Formula Column properties	272
Frame properties	273
Group properties	274
Child Dimension	275
Filter Type	276
Name	277
Comment	278
Number of Records	279
PL/SQL Filter	280
Data Link properties	281
Child Column	283
Child Query	284
Condition	285
Parent Column	286
Parent Group	287
SQL Clause	288
Matrix properties	289
Cross Product Group	290
Horizontal Repeating Frame	292
Vertical Repeating Frame	293
Database Column Object properties	294
OLE2 properties	295
Parameter properties	296
System Parameters	297
Add	298
Additional Attributes (HTML)	299
Comment	300
Datatype	301
Hide First Column	302
Initial Value	303
Input Mask	304
Name	305
Remove	306
Restrict List to Predetermined Values	307
SELECT Statement/Static Values	308
Validation Trigger	309
Value	310
Width	311

List of Values	312
Parameter Form Boilerplate properties	313
Name	314
Type	315
Parameter Form Field properties	316
Datatype	317
Name	318
Source	319
Placeholder Column properties	320
PL/SQL Formula	322
Alphabetical list of properties	323
Query properties	328
Comment	329
External Query Source File	330
Maximum Rows to Fetch	331
Name	332
SQL Query Statement	333
Type	336
Database Ref Column properties	337
Repeating Frame properties	338
Column Mode	339
Horiz. Space Between Frames	340
Maximum Records Per Page	341
Minimum Widow Records	342
Print Direction	344
Source	346
Vert. Space Between Frames	347
Report properties	348
Unit of Measurement	350
Max. Horizontal Body Pages	351
Max. Vertical Body Pages	352
Panel Print Order	353
Direction	354
Distribution	355
Previewer Title	356
Role Name	357
Width (Parameter Form window)	358
Height (Parameter Form window)	359
Number of Pages	360
Page Navigation Control Type	361
Page Navigation Control Value	362
Before Report Type	363

Before Report Value	364
After Report Type	365
After Report Value	366
Before Page Type	367
Before Page Value	368
After Page Type	369
After Page Value	370
Before Form Type	371
Before Form Value	372
After Form Type	373
After Form Value	374
Design in Character Units	375
Use Previewer Hint Line	376
Previewer Hint Line Text	377
Use Previewer Status Line	378
Previewer Status Line Text	379
Include Bitmapped Objects	380
Include Borders	381
Disable Host Menu Item	382
Disable Split Screen Key	383
Disable Zoom Key	384
Start in Zoom	385
Suppress Previewer Title	386
Ref cursor query properties	387
PL/SQL Statement	388
Section properties	389
Distribution	390
Height	391
Horizontal Panels per Page	392
Orientation	393
Report Height	394
Report Width	395
Vertical Panels per Page	396
Width	397
Summary Column properties	398
Compute At	399
Function	402
Product Order	404
Reset At	408
Source	410
Template properties	411
Alignment	412

Align Summaries with Fields	413
Background Color	414
Between Field and Labels (Horizontal)	415
Between Frame and Fields (Horizontal)	416
Between Frame and Fields (Vertical)	417
Between Master and Detail (Horizontal)	418
Between Master and Detail (Vertical)	419
Between Page and Frames (Horizontal)	420
Between Page and Frames (Vertical)	421
Between Sibling Frames (Horizontal)	422
Between Sibling Frames (Vertical)	423
Borders	424
Character Justification	425
Dash	426
Date Justification	427
Edge Foreground Color	428
Edge Background Color	429
Edge Pattern	430
Fields Per Line	431
Fill Pattern	432
Font	433
Foreground Color	434
Image	435
Inter-Field (Horizontal)	436
Inter-Field (Vertical)	437
Inter-Frame (Horizontal)	438
Inter-Frame (Vertical)	439
Justify	440
Number Justification	441
Place Labels Above Fields	442
Position	443
Style	444
Text	445
Text Color	446
Use Vertical Spacing	447
XML Properties	448
XML Tag	449
XML Tag Attributes	451
Exclude from XML Output	455
XML Prolog Type	458
XML Prolog Value	459
Outer XML Tag	461

Outer XML Attributes -----	464
Contains XML Tags -----	467
EXECUTABLES -----	469
Executable names-----	470
Executable invocation -----	471
Help on command line options-----	472
Keyword usage-----	475
Database login-----	476
Explicit login-----	477
USERID -----	478
Automatic login -----	479
Remote login-----	480
RWBLD60 -----	481
RWBLD60 Command Line Arguments -----	482
MODULE REPORT (RWBLD60)-----	484
PARAMFORM (RWBLD60)-----	485
CMDFILE (RWBLD60)-----	486
ARRAYSIZE (RWBLD60)-----	487
DESTYPE (RWBLD60)-----	488
DESNAME (RWBLD60)-----	489
DESFORMAT (RWBLD60)-----	490
COPIES (RWBLD60)-----	492
CACHELOB (RWBLD60)-----	493
CURRENCY (RWBLD60)-----	494
THOUSANDS (RWBLD60)-----	495
DECIMAL (RWBLD60)-----	496
READONLY (RWBLD60)-----	497
BUFFERS (RWBLD60)-----	498
PAGESIZE (RWBLD60)-----	499

PROFILE (RWBLD60)-----	500
RUNDEBUG (RWBLD60)-----	501
ONSUCCESS (RWBLD60)-----	502
ONFAILURE (RWBLD60)-----	503
ERRFILE (RWBLD60)-----	504
LONGCHUNK (RWBLD60)-----	505
ACCESS (RWBLD60)-----	506
ORIENTATION (RWBLD60)-----	507
BACKGROUND (RWBLD60)-----	508
MODE (RWBLD60)-----	509
PRINTJOB (RWBLD60)-----	510
TRACEFILE (RWBLD60)-----	511
TRACEMODE (RWBLD60)-----	512
TRACEOPTS (RWBLD60)-----	513
AUTOCOMMIT (RWBLD60)-----	514
NONBLOCKSQL (RWBLD60)-----	515
ROLE (RWBLD60)-----	516
BLANKPAGES (RWBLD60)-----	517
MAXIMIZE (RWBLD60)-----	518
DISABLEPRINT (RWBLD60)-----	519
DISABLEMAIL (RWBLD60)-----	520
DISABLEFILE (RWBLD60)-----	521
DISABLENEW (RWBLD60)-----	522
DELIMITER (RWBLD60)-----	523
CELLWRAPPER (RWBLD60)-----	524
DATEFORMATMASK (RWBLD60)-----	525
NUMBERFORMATMASK (RWBLD60)-----	526
DESTINATION (RWBLD60)-----	527
DISTRIBUTE (RWBLD60)-----	528
PAGESTREAM (RWBLD60)-----	529
EXPRESS_SERVER (RWBLD60)-----	530
CUSTOMIZE (RWBLD60)-----	532
SAVE_RDF (RWBLD60)-----	533
<param> (RWBLD60)-----	534
RWCGI60-----	535
RWCLI60-----	536
RWCLI60 Command Line Arguments -----	537
PARAMFORM (RWCLI60)-----	539
DESTYPE (RWCLI60)-----	540
BATCH (RWCLI60)-----	541
BACKGROUND (RWCLI60)-----	542
SERVER (RWCLI60)-----	543

JOBNAME (RWCLI60)-----	544
SCHEDULE (RWCLI60)-----	545
TOLERANCE (RWCLI60)-----	546
AUTHID-----	547
USERID-----	548
RWCON60-----	549
RWCON60 Command Line Arguments-----	551
STYPE (RWCON60)-----	552
SOURCE (RWCON60)-----	553
DTYPE (RWCON60)-----	554
DEST (RWCON60)-----	555
CMDFILE (RWCON60)-----	556
LOGFILE (RWCON60)-----	557
OVERWRITE (RWCON60)-----	558
BATCH (RWCON60)-----	559
DUNIT (RWCON60)-----	560
PAGESIZE (RWCON60)-----	561
FORMSIZE (RWCON60)-----	562
CUSTOMIZE (RWCON60)-----	563
RWMTS60-----	564
RWRQM60-----	565
RWRQV60 -----	566
RWOWS60 -----	567
RWRBE60-----	568
RWRUN60-----	569
RWRUN60 Command Line Arguments-----	570
MODULE REPORT (RWRUN60)-----	572
PARAMFORM (RWRUN60)-----	573
CMDFILE (RWRUN60)-----	574
TERM (RWRUN60)-----	575
ARRAYSIZE (RWRUN60)-----	576
DESTYPE (RWRUN60)-----	577
DESNAME (RWRUN60)-----	579
DESFORMAT (RWRUN60)-----	580
COPIES (RWRUN60)-----	582

CACHELOB (RWRUN60)	583
CURRENCY (RWRUN60)	584
THOUSANDS (RWRUN60)	585
DECIMAL (RWRUN60)	586
READONLY (RWRUN60)	587
LOGFILE (RWRUN60)	588
BUFFERS (RWRUN60)	589
BATCH (RWRUN60)	590
PAGESIZE (RWRUN60)	591
PROFILE (RWRUN60)	592
RUNDEBUG (RWRUN60)	593
ONSUCCESS (RWRUN60)	594
ONFAILURE (RWRUN60)	595
KEYIN (RWRUN60)	596
KEYOUT (RWRUN60)	597
ERRFILE (RWRUN60)	598
LONGCHUNK (RWRUN60)	599
ORIENTATION (RWRUN60)	600
BACKGROUND (RWRUN60)	601
MODE (RWRUN60)	602
PRINTJOB (RWRUN60)	603
TRACEFILE (RWRUN60)	604
TRACEMODE (RWRUN60)	605
TRACEOPTS (RWRUN60)	606
AUTOCOMMIT (RWRUN60)	607
NONBLOCKSQL (RWRUN60)	608
ROLE (RWRUN60)	609
BLANKPAGES (RWRUN60)	610
YES (RWRUN60)	611
DISABLEPRINT (RWRUN60)	612
DISABLEMAIL (RWRUN60)	613
DISABLEFILE (RWRUN60)	614
DISABLENEW (RWRUN60)	615
DESTINATION (RWRUN60)	616
DISTRIBUTE (RWRUN60)	617
DELIMITER (RWRUN60)	618
CELLWRAPPER (RWRUN60)	619
DATEFORMATMASK (RWRUN60)	620
NUMBERFORMATMASK (RWRUN60)	621
PAGESTREAM (RWRUN60)	622
EXPRESS_SERVER (RWRUN60)	623
CUSTOMIZE (RWRUN60)	625

SAVE_RDF (RWRUN60)	-----	-----	-626
<param> (RWRUN60)	-----	-----	---627

Send Us Your Comments

Reader's Comment Form – A73174-01

Oracle Corporation welcomes your comments about this manual's quality and usefulness. Your feedback is an important part of our revision process.

- Did you find any errors?
- Is the information presented clearly?
- Are the examples correct? Do you need more examples?
- What features did you like?

If you found any errors or have any other suggestions for improvement, please write the topic, chapter, and page number below:

Please send your comments to:

Oracle Applications Documentation Manager

Oracle Corporation

500 Oracle Parkway

Redwood Shores, CA 94065

U.S.A.

Or send comments by e-mail to: globedoc@us.oracle.com

Please include your name, address, and telephone number for a reply:

Thank you for your help.



Preface

Welcome to the *Oracle Reports Developer Reference*, Release 6i. This reference guide includes information to help you effectively work with Report Builder and contains detailed information about the following:

- Built ins
- Triggers
- Properties
- Executables

This preface explains how this reference is organized and introduces other sources of information that can help you use Oracle Reports Developer.

Prerequisites

You should be familiar with your computer and its operating system. For example, you should know the commands for deleting and copying files and understand the concepts of search paths, subdirectories, and path names. Refer to your Microsoft Windows 95 or NT and DOS product documentation for more information.

You should understand the fundamentals of Microsoft Windows, such as the elements of an application window. You should also be familiar with such programs as the Explorer, Taskbar or Task Manager, and Registry.

Notational Conventions

The following typographical conventions are used in this guide:

Convention	Meaning
<i>fixed-width font</i>	Text in a fixed-width font indicates commands that you enter exactly as shown. Text typed on a PC is not case-sensitive unless otherwise noted. In commands, punctuation other than brackets and vertical bars must be entered exactly as shown.
<i>lowercase</i>	Lowercase characters in a command statement represent a variable. Substitute an appropriate value.
<i>UPPERCASE</i>	Uppercase characters within the text represent command names, SQL reserved words, and keywords.
<i>boldface</i>	Boldface is used to indicate user interface items such as menu choices and buttons.
<i>C></i>	C> represents the DOS prompt. Your prompt may differ.

Related Publications

You may also wish to consult the following Oracle documentation:

Title	Part Number
Oracle Forms Developer and Oracle Reports Developer: Guidelines for Building Applications	A58766
SQL*Plus User's Guide and Reference Version 3.1	A24801

Built-Ins

Built-in packages

Oracle provides several packaged procedures which you can use when building or debugging your PL/SQL-based applications.

Your PL/SQL code can make use of the procedures, functions, and exceptions in the following client-side built-in packages:

- Developer built-in packages
- SRW built-in packages

SRW.SET_ATTR attributes

ACTION
AFTCODE
AFTFORM_ESCAPE
AFTPAGE_ESCAPE
AFTREPORT_ESCAPE
BBCOLOR
BEFCODE
BEFFORM_ESCAPE
BEFPAGE_ESCAPE
BEFREPORT_ESCAPE
BFCOLOR
BOOKMARK
BORDERWIDTH
BORDPATT
FACE
FBCOLOR
FFCOLOR
FILLPATT
FORMATMASK
GCOLOR
GSPACING
HJUST
HYPERLINK
LINKTAG
PRINTER_INTRAY
STYLE
SZ
TEXT
WEIGHT

ACTION

New Feature: It is now more convenient to set this attribute using the SRW.SET_PDF_ACTION procedure. Setting this attribute via SRW.SET_ATTR is still supported for compatibility, but it is highly recommended that you use the new, simplified procedure.

Description Is a command line that will be executed on the local machine when the object is clicked in the PDF viewer.

Note: An object that is associated with an action cannot also be the source of a Web link.

Values

Any valid command line on the local machine (e.g., c:\orawin\bin\rwrun60 userid=scott/tiger report=example.rdf or /usr/local/bin/phone smith)

ACTION restrictions

- ACTION is only available for PDF output.
- ACTION should only be set in the following triggers:
 - Format

ACTION example

Note: This example illustrates using SRW.SET_ATTR to set the ACTION attribute. It is now more convenient to set this attribute using the SRW.SET_PDF_ACTION procedure. Setting this attribute via SRW.SET_ATTR is still supported for compatibility, but it is highly recommended that you use the new, simplified procedure.

```
/* The format trigger below defines an "action" (in
** this case a call to the Reports Runtime) that
** should be executed when the user clicks on the
** boilerplate object B_2.
*/
function B_2FormatTrigger return boolean is
begin
  srw.attr.mask := srw.ACTION_ATTR;
  srw.attr.action := 'c:\orawin\bin\rwrun60' ||
  'userid=scott/tiger ' ||
  'report=example.rdf';
  srw.set_attr(0,srw.attr);
  return (TRUE);
end;
```

AFTCODE

New Feature: It is now more convenient to set this attribute using the `SRW.SET_AFTER_PRINTING_CODE` procedure. Setting this attribute via `SRW.SET_ATTR` is still supported for compatibility, but it is highly recommended that you use the new, simplified procedure.

Description Is a reference to the printer escape sequence to be executed after each line of the object. The printer escape sequence is inserted after the object is triggered to print but before it has actually printed. AFTCODE is only used when running your report in character-mode.

Values

A string of the form `&number`, where number is a number assigned to a packaged Report Builder printer escape sequence or a printer escape sequence that you created.

AFTFORM_ESCAPE

New Feature: It is now more convenient to set this attribute using the `SRW.SET_AFTER_FORM_HTML` procedure. Setting this attribute via `SRW.SET_ATTR` is still supported for compatibility, but it is highly recommended that you use the new, simplified procedure.

Description Is any text, graphics, or HTML commands that you want to appear at the bottom of the HTML Parameter Form. This attribute is useful for placing a logo or some standard links on the Parameter Form .

Values

You must specify two items for this attribute:

- `AFTFORM_ESCAPE_TYPE` can be set to either `SRW.FILE_ESCAPE` or `SRW.TEXT_ESCAPE`.
- `AFTFORM_ESCAPE_VALUE` can be a filename or a text string containing valid HTML depending upon what you specified for `AFTFORM_ESCAPE_TYPE`. If you specify a file, the File searching method is used to find it.

AFTPAGE_ESCAPE

New Feature: It is now more convenient to set this attribute using the `SRW.SET_AFTER_PAGE_HTML` procedure. Setting this attribute via `SRW.SET_ATTR` is still supported for compatibility, but it is highly recommended that you use the new, simplified procedure.

Description Is any text, graphics, or HTML commands that you want to appear at the end of pages of your document. This attribute is useful for placing a logo or some standard links at the end of each page in an HTML document.

Values

You must specify two items for this attribute:

- `AFTPAGE_ESCAPE_TYPE` can be set to either `SRW.FILE_ESCAPE` or `SRW.TEXT_ESCAPE`.
- `AFTPAGE_ESCAPE_VALUE` can be a filename or a text string containing valid HTML depending upon what you specified for `AFTPAGE_ESCAPE_TYPE`. If you specify a file, the File searching method is used to find it.

AFTPAGE_ESCAPE restrictions

- When you specify `SRW.SET_ATTR` for this attribute you must use `SRW.REPORT_ID` as the `object_id`:
`SRW.SET_ATTR(SRW.REPORT_ID, SRW.ATTR);`
- The default HTML included at the end of a page is shown below. It is not required unless you want the default separator line to appear between pages.
`<hr size=5 noshade>`
- If you want the `AFTPAGE_ESCAPE` to apply to every page of the report, you should specify it in a trigger that fires before the report begins formatting, such as the Before Report trigger.
- If you want the `AFTPAGE_ESCAPE` to apply only to the current page, you should specify it in a format trigger for an object on that page.

Graphic page separator example

Note: This example illustrates using `SRW.SET_ATTR` to set the `AFTPAGE_ESCAPE` attribute. It is now more convenient to set this attribute using the `SRW.SET_AFTER_PAGE_HTML` procedure. Setting this attribute via `SRW.SET_ATTR` is still supported for compatibility, but it is highly recommended that you use the new, simplified procedure.

```
/* The example below inserts a GIF file as a  
** page separator.
```

```
*/  
function BeforeReport return boolean is  
begin  
  if ((upper(:MODE) != 'CHARACTER') and  
      (upper(:DESFORMAT) = 'HTML')) then  
    SRW.ATTR.MASK := SRW.AFTPAGE_ESCAPE_ATTR;  
    SRW.ATTR.AFTPAGE_ESCAPE_TYPE := SRW.TEXT_ESCAPE;  
    SRW.ATTR.AFTPAGE_ESCAPE_VALUE := '<CENTER>' ||  
    '<IMG ALT="Ruler" SRC="line.gif" VSPACE=10>' ||  
    '<BR></CENTER>';  
    SRW.SET_ATTR(SRW.REPORT_ID, SRW.ATTR);  
  end if;  
  return (TRUE);  
end;
```

AFTREPORT_ESCAPE

New Feature: It is now more convenient to set this attribute using the `SRW.SET_AFTER_REPORT_HTML` procedure. Setting this attribute via `SRW.SET_ATTR` is still supported for compatibility, but it is highly recommended that you use the new, simplified procedure.

Description Is any text, graphics, or HTML commands that you want to appear at the end of your document. This attribute is useful for placing a logo or some standard links at the end of an HTML document.

Values

You must specify two items for this attribute:

- `AFTREPORT_ESCAPE_TYPE` can be set to either `SRW.FILE_ESCAPE` or `SRW.TEXT_ESCAPE`.
- `AFTREPORT_ESCAPE_VALUE` can be a filename or a text string containing valid HTML depending upon what you specified for `AFTREPORT_ESCAPE_TYPE`. If you specify a file, the File searching method is used to find it.

AFTREPORT_ESCAPE restrictions

- When you specify `SRW.SET_ATTR` for this attribute you must use `SRW.REPORT_ID` as the `object_id`:
`SRW.SET_ATTR(SRW.REPORT_ID, SRW.ATTR);`
- The default HTML included at the end of a report is shown below. If you use `AFTREPORT_ESCAPE` to replace this text, you must ensure that it contains the equivalent HTML commands.
`</body></html>`
- `AFTREPORT_ESCAPE` should be set in a trigger that fires before the report is done formatting, such as the Before Report trigger.

Links to home pages example

Note: This example illustrates using `SRW.SET_ATTR` to set the `AFTREPORT_ESCAPE` attribute. It is now more convenient to set this attribute using the `SRW.SET_AFTER_REPORT_HTML` procedure. Setting this attribute via `SRW.SET_ATTR` is still supported for compatibility, but it is highly recommended that you use the new, simplified procedure.

```
/* The example below inserts two links to home pages
** at the end of the report output.
*/
function BeforeReport return boolean is
```

```

begin
if ((upper(:MODE) != 'CHARACTER') and
    (upper(:DESFORMAT) = 'HTML')) then
    SRW.ATTR.MASK := SRW.AFTREPORT_ESCAPE_ATTR;
    SRW.ATTR.AFTREPORT_ESCAPE_TYPE := SRW.TEXT_ESCAPE;
    SRW.ATTR.AFTREPORT_ESCAPE_VALUE := '<CENTER>' ||
    '<A HREF="http://www.oracle.com/">' ||
    'Oracle Corporation</A> -' ||
    chr(10) ||
    '<A HREF="http://home.netscape.com/">' ||
    'Netscape</A> </CENTER>' ||
    '</BODY> </HTML>';
    SRW.SET_ATTR(SRW.REPORT_ID, SRW.ATTR);
end if;
return (TRUE);
end;

```

BBCOLOR

New Feature: It is now more convenient to set this attribute using the `SRW.SET_BACKGROUND_BORDER_COLOR` procedure. Setting this attribute via `SRW.SET_ATTR` is still supported for compatibility, but it is highly recommended that you use the new, simplified procedure.

Description Is the background border color of the object. This attribute is ignored in character mode.

Values

A valid color name.

BEFCODE

New Feature: It is now more convenient to set this attribute using the `SRW.SET_BEFORE_PRINTING_CODE` procedure. Setting this attribute via `SRW.SET_ATTR` is still supported for compatibility, but it is highly recommended that you use the new, simplified procedure.

Description Is a reference to the printer escape sequence to be executed before each line of the object. The printer escape sequence is inserted after the object is triggered to print but before it has actually printed. BEFCODE is only used when running your report in character-mode.

Values

A string of the form `&number`, where number is a number assigned to a packaged Report Builder printer escape sequence or a printer escape sequence that you created.

BEFFORM_ESCAPE

New Feature: It is now more convenient to set this attribute using the `SRW.SET_BEFORE_FORM_HTML` procedure. Setting this attribute via `SRW.SET_ATTR` is still supported for compatibility, but it is highly recommended that you use the new, simplified procedure.

Description Is any text, graphics, or HTML commands that you want to appear at the top of the HTML Parameter Form. This attribute is useful for placing a logo or some standard links on the Parameter Form .

Values

You must specify two items for this attribute:

- `BEFFORM_ESCAPE_TYPE` can be set to either `SRW.FILE_ESCAPE` or `SRW.TEXT_ESCAPE`.
- `BEFFORM_ESCAPE_VALUE` can be a filename or a text string containing valid HTML depending upon what you specified for `BEFFORM_ESCAPE_TYPE`. If you specify a file, the File searching method is used to find it.

BEFPAGE_ESCAPE

New Feature: It is now more convenient to set this attribute using the `SRW.SET_BEFORE_PAGE_HTML` procedure. Setting this attribute via `SRW.SET_ATTR` is still supported for compatibility, but it is highly recommended that you use the new, simplified procedure.

Description Is any text, graphics, or HTML commands that you want to appear at the beginning of pages of your document. This attribute is useful for placing a logo or some standard links at the beginning of each page in an HTML document.

Values

You must specify two items for this attribute:

- `BEFPAGE_ESCAPE_TYPE` can be set to either `SRW.FILE_ESCAPE` or `SRW.TEXT_ESCAPE`.
- `BEFPAGE_ESCAPE_VALUE` can be a filename or a text string containing valid HTML depending upon what you specified for `BEFPAGE_ESCAPE_TYPE`. If you specify a file, the File searching method is used to find it.

BEFPAGE_ESCAPE restrictions

- When you specify `SRW.SET_ATTR` for this attribute you must use `SRW.REPORT_ID` as the `object_id`:
`SRW.SET_ATTR(SRW.REPORT_ID, SRW.ATTR);`
- If you want the `BEFPAGE_ESCAPE` to apply to every page of the report, you should specify it in a trigger that fires before the report begins formatting, such as the Before Report trigger .
- If you want the `BEFPAGE_ESCAPE` to apply only to the current page, you should specify it in a format trigger for an object on that page.

Before and after page escape example

Note: This example illustrates using `SRW.SET_ATTR` to set the `BEFPAGE_ESCAPE` and `AFTPAGE_ESCAPE` attributes. It is now more convenient to set these attributes using the `SRW.SET_BEFORE_PAGE_HTML` and `SRW.SET_AFTER_PAGE_HTML` procedures. Setting this attribute via `SRW.SET_ATTR` is still supported for compatibility, but it is highly recommended that you use the new, simplified procedure.

```
/* The example below centers the document in
** the browser.
*/
function BeforeReport return boolean is
begin
if ((upper(:MODE) != 'CHARACTER') and
    (upper(:DESFORMAT) = 'HTML')) then
```

```
SRW.ATTR.MASK := SRW.BEFPAGE_ESCAPE_ATTR +
                SRW.AFTPAGE_ESCAPE_ATTR;
SRW.ATTR.BEFPAGE_ESCAPE_TYPE := SRW.TEXT_ESCAPE;
SRW.ATTR.BEFPAGE_ESCAPE_VALUE := '<CENTER>';
SRW.ATTR.AFTPAGE_ESCAPE_TYPE := SRW.TEXT_ESCAPE;
SRW.ATTR.AFTPAGE_ESCAPE_VALUE := '</CENTER>';
SRW.SET_ATTR(SRW.REPORT_ID, SRW.ATTR);
end if;
return (TRUE);
end;
```

BEFREPORT_ESCAPE

New Feature: It is now more convenient to set this attribute using the `SRW.SET_BEFORE_REPORT_HTML` and `SRW.SET_AFTER_REPORT_HTML` procedures. Setting this attribute via `SRW.SET_ATTR` is still supported for compatibility, but it is highly recommended that you use the new, simplified procedure.

Description Is any text, graphics, or HTML commands that you want to appear at the beginning of your document. This attribute is useful for placing a logo or some standard links at the beginning of an HTML document.

Values

You must specify two items for this attribute:

- `BEFREPORT_ESCAPE_TYPE` can be set to either `SRW.FILE_ESCAPE` or `SRW.TEXT_ESCAPE`.
- `BEFREPORT_ESCAPE_VALUE` can be a filename or a text string containing valid HTML depending upon what you specified for `BEFREPORT_ESCAPE_TYPE`. If you specify a file, the File searching method is used to find it.

BEFREPORT_ESCAPE restrictions

- When you specify `SRW.SET_ATTR` for this attribute you must use `SRW.REPORT_ID` as the `object_id`:

```
SRW.SET_ATTR(SRW.REPORT_ID, SRW.ATTR);
```
- The default HTML included at the beginning of a report is shown below. If you use `BEFREPORT_ESCAPE` to replace this text, you must ensure that it contains the equivalent HTML commands.

```
<html>  
<body bgcolor="#ffffff">
```
- `BEFREPORT_ESCAPE` should be set in a trigger that fires before the report starts formatting, such as the Before Report trigger.

Before and after report escape example

Note: This example illustrates using `SRW.SET_ATTR` to set the `BEFREPORT_ESCAPE` and `AFTREPORT_ESCAPE` attributes. It is now more convenient to set these attributes using the `SRW.SET_BEFORE_REPORT_HTML` procedure. Setting this attribute via `SRW.SET_ATTR` is still supported for compatibility, but it is highly recommended that you use the new, simplified procedure.

```
/* The example below sets both BEFREPORT_ESCAPE and  
** AFTREPORT_ESCAPE in the Before Report trigger.  
** The HTML for BEFREPORT_ESCAPE is located in a file
```

```

** named logo.htm. The HTML for AFTREPORT_ESCAPE is
** specified within the PL/SQL itself.
*/
function BeforeReport return boolean is
begin
if ((upper(:MODE) != 'CHARACTER') and
    (upper(:DESFORMAT) = 'HTML')) then
    SRW.ATTR.MASK := SRW.BEFREPORT_ESCAPE_ATTR +
                    SRW.AFTREPORT_ESCAPE_ATTR;
    SRW.ATTR.BEFREPORT_ESCAPE_TYPE := SRW.FILE_ESCAPE;
    SRW.ATTR.BEFREPORT_ESCAPE_VALUE := 'logo.htm';
    SRW.ATTR.AFTREPORT_ESCAPE_TYPE := SRW.TEXT_ESCAPE;
    SRW.ATTR.AFTREPORT_ESCAPE_VALUE := '<ADDRESS>' ||
    ' Questions? - ' ||
    '<A HREF=mailto:webmaster@xyztech.com>' ||
    'webmaster@xyztech.com</A>' ||
    '</ADDRESS>' ||
    '</body></html>';
    SRW.SET_ATTR(SRW.REPORT_ID, SRW.ATTR);
end if;
return (TRUE);
end;

```

BFCOLOR

New Feature: It is now more convenient to set this attribute using the `SRW.SET_BACKGROUND_FILL_COLOR` procedure. Setting this attribute via `SRW.SET_ATTR` is still supported for compatibility, but it is highly recommended that you use the new, simplified procedure.

Description Is the background fill color of the object. This attribute is ignored in character mode.

Values

A valid color name.

BOOKMARK

New Feature: It is now more convenient to set this attribute using the SRW.SET_BOOKMARK procedure. Setting this attribute via SRW.SET_ATTR is still supported for compatibility, but it is highly recommended that you use the new, simplified procedure.

Description Is a string that will appear in a frame of the master HTML document or in the PDF viewer if you open the bookmark area. Clicking on the bookmark displays the associated object at the top of the window.

Values

A text string with no indentation/ordering information. The bookmark will appear in the bookmark list according to when the object is rendered by the report.

A text string with explicit ordering/indentation of the form `x#book_mark_name`, where x is an outline number. The pound sign (#) and outline number do not appear in the bookmark window but are used to determine order and indentation. For example:

```
1#Expense Summary Section
2#Expense Detail Section
2.1#Expenses for the Administration Department
2.2#Expenses for the Engineering Department
2.3#Expenses for the Sales Department
2.3.1#Expenses for the Eastern Sales Region
2.3.2#Expenses for the Northern Sales Region
2.3.3#Expenses for the Southern Sales Region
2.3.4#Expenses for the Western Sales Region
```

BOOKMARK restrictions

- If the same outline number is used multiple times, all entries appear but the order is defined by when the objects are rendered by the report.
- If there are gaps in the numbers, one of two things will happen. If the gap is between peer level numbers, there will be no visible effect (e.g., 1.3.1 and 1.3.3, given there is no 1.3.2, will appear next to each other and at the same indentation level). If the gap is between a higher level number and a lower level number, intermediate levels will be generated as required (e.g., 1.0 followed by 2.1.1 will cause dummy 2 and 2.1 entries to be defined, whose titles will be the same as the subsequent real entry).
- BOOKMARK should only be set in the following triggers:

Format

BOOKMARK example

Note: This example illustrates using SRW.SET_ATTR to set the BOOKMARK attribute. It is now more convenient to set this attribute using the

SRW.SET_BOOKMARK procedure. Setting this attribute via SRW.SET_ATTR is still supported for compatibility, but it is highly recommended that you use the new, simplified procedure.

```
/* The format trigger below defines a bookmark for
** the boilerplate object B_2. Notice the use of
** explicit ordering information (1#) in this example.
** If you did not want the ordering information, you
** could omit 1#.
*/
function B_2FormatTrigger return boolean is
begin
  srw.attr.mask := srw.BOOKMARK_ATTR;
  srw.attr.bookmark := '1#Expense Summary Section';
  srw.set_attr(0,srw.attr);
  return (TRUE);
end;
```

Dynamic BOOKMARK example

Note: This example illustrates using SRW.SET_ATTR to set the BOOKMARK attribute. It is now more convenient to set this attribute using the SRW.SET_BOOKMARK procedure. Setting this attribute via SRW.SET_ATTR is still supported for compatibility, but it is highly recommended that you use the new, simplified procedure.

```
/* The format trigger below defines a bookmark for
** the boilerplate object B_2. Notice that the name
** of the bookmark is dynamic. CATEGORY is a column
** value that is concatenated with the string Expense
** Summary Section for each execution of the format
** trigger. In this case, CATEGORY could contain
** ordering information (e.g., 1#) or perhaps a string
** that makes the bookmark unique within the report.
*/
function B_2FormatTrigger return boolean is
begin
  srw.attr.mask := srw.BOOKMARK_ATTR;
  srw.attr.bookmark := :category ||
  ' Expense Summary Section';
  srw.set_attr(0,srw.attr);
  return (TRUE);
end;
```

BORDERWIDTH

New Feature: It is now more convenient to set this attribute using the `SRW.SET_BORDER_WIDTH` procedure. Setting this attribute via `SRW.SET_ATTR` is still supported for compatibility, but it is highly recommended that you use the new, simplified procedure.

Description Is the border width of the object.

Values

0	Means no border.
any positive integer	Means a border width of that many picas for a bit-mapped report or one character for a character mode report.

BORDPATT

New Feature: It is now more convenient to set this attribute using the `SRW.SET_BORDER_PATTERN` procedure. Setting this attribute via `SRW.SET_ATTR` is still supported for compatibility, but it is highly recommended that you use the new, simplified procedure.

Description Is the fill pattern for the border of the object. This attribute is ignored in character mode.

Values

A valid pattern name.

FACE

New Feature: It is now more convenient to set this attribute using the SRW.SET_FONT_FACE procedure. Setting this attribute via SRW.SET_ATTR is still supported for compatibility, but it is highly recommended that you use the new, simplified procedure.

Description Is the font face of a CHAR, DATE, or NUMBER field. This attribute is ignored in character mode.

Values

A valid font name on the machine where the report is run.

FBCOLOR

New Feature: It is now more convenient to set this attribute using the `SRW.SET_FOREGROUND_BORDER_COLOR` procedure. Setting this attribute via `SRW.SET_ATTR` is still supported for compatibility, but it is highly recommended that you use the new, simplified procedure.

Description Is the foreground border color of the object. This attribute is ignored in character mode.

Values

A valid color name.

FFCOLOR

New Feature: It is now more convenient to set this attribute using the `SRW.SET_FOREGROUND_FILL_COLOR` procedure. Setting this attribute via `SRW.SET_ATTR` is still supported for compatibility, but it is highly recommended that you use the new, simplified procedure.

Description Is the foreground fill color of the object. This attribute is ignored in character mode.

Values

A valid color name.

FILLPATT

New Feature: It is now more convenient to set this attribute using the SRW.SET_FILL_PATTERN procedure. Setting this attribute via SRW.SET_ATTR is still supported for compatibility, but it is highly recommended that you use the new, simplified procedure.

Description Is the fill pattern of the object. This attribute is ignored in character mode.

Values

A valid pattern name.

FORMATMASK

New Feature: It is now more convenient to set this attribute using the SRW.SET_FORMAT_MASK procedure. Setting this attribute via SRW.SET_ATTR is still supported for compatibility, but it is highly recommended that you use the new, simplified procedure.

Description Is the format mask for the DATE, or NUMBER field.

Values

Any valid format mask.

GCOLOR

New Feature: It is now more convenient to set this attribute using the SRW.SET_TEXT_COLOR procedure. Setting this attribute via SRW.SET_ATTR is still supported for compatibility, but it is highly recommended that you use the new, simplified procedure.

Description Is the global text color of the CHAR, DATE, or NUMBER field. This attribute is ignored in character mode.

Values

A valid color name.

GSPACING

New Feature: It is now more convenient to set this attribute using the SRW.SET_CUSTOM_SPACING or SRW.SET_SPACING procedures. Setting this attribute via SRW.SET_ATTR is still supported for compatibility, but it is highly recommended that you use the new, simplified procedures.

Description Is the global text leading for a CHAR, DATE, or NUMBER field, or boilerplate text. This attribute is ignored in character mode.

Values

srw.single_spacing

srw.onehalf_spacing

srw.double_spacing

srw.custom_spacing Specifies the spacing in VGS units, which are very small. If you use srw.custom_spacing, then you specify the custom number of VGS units using srw.attr.custom. For example:

```
srw.attr.mask            := SRW.GSPACING_ATTR;  
srw.attr.gspacing       := SRW.CUSTOM_SPACING;  
srw.attr.custom         := 200;  
srw.set_attr (0, srw.attr);
```

HJUST

New Feature: It is now more convenient to set this attribute using the SRW.SET_JUSTIFICATION procedure. Setting this attribute via SRW.SET_ATTR is still supported for compatibility, but it is highly recommended that you use the new, simplified procedure.

Description Is the horizontal justification of the CHAR, DATE, or NUMBER field.

Values

srw.left_hjust

srw.center_hjust

srw.right_hjust

srw.flush_hjust Is ignored in bit-mapped reports.

HYPERLINK

New Feature: It is now more convenient to set this attribute using the `SRW.SET_HYPERLINK` procedure. Setting this attribute via `SRW.SET_ATTR` is still supported for compatibility, but it is highly recommended that you use the new, simplified procedure.

Description Is a URL Web link specification.

Note: An object that is the source of a Web link cannot also be associated with an action.

Values

A valid link:

To another document on the same machine (e.g., `file:/private/mynewdoc.pdf` or `file:///C:/temp/mynewdoc.pdf`)

To another document on a different machine (e.g., `http://www.newmach.com/newdoc.pdf`)

To a destination within the current document (e.g., `#my_dest_name`)

To a destination within a local document (e.g., `file:/private/somedoc.pdf#a_dest_name`)

To a destination within a remote document (e.g.,

`http://www.newmach.com/newdoc.pdf#some_dest_name`)

To any URL (e.g., `http://www.newmach.com/newdoc.html`,

`ftp://www.reposit.com/filetoget.example`, `http://www.somemch.com/cgi-bin/webmenu?choice1`)

HYPERLINK restrictions

- HYPERLINK should only be set in the following triggers:
 - Format
 - To follow Web links from a PDF viewer to a remote server or HTML document, the PDF viewer must be configured to work with a Web browser (e.g., configured as a helper application or installed as a plug-in to your Web browser).

HYPERLINK example

Note: This example illustrates using `SRW.SET_ATTR` to set the HYPERLINK attribute. It is now more convenient to set this attribute using the `SRW.SET_HYPERLINK` procedure. Setting this attribute via `SRW.SET_ATTR` is still supported for compatibility, but it is highly recommended that you use the new, simplified procedure.

```
/* The format trigger below defines a Web link
** to a destination (mytarget) in another document
** (newdoc.pdf) for the boilerplate object B_2.
** Note: If the target were in the same document,
```

```

** you would omit http://www.newmach.com/newdoc.pdf.
*/
function B_2FormatTrigger return boolean is
begin
  srw.attr.mask := srw.HYPERLINK_ATTR;
  srw.attr.hyperlink :=
    'http://www.newmach.com/newdoc.pdf' ||
    '#mytarget';
  srw.set_attr(0,srw.attr);
  return (TRUE);
end;

```

Dynamic HYPERLINK example

Note: This example illustrates using SRW.SET_ATTR to set the HYPERLINK attribute. It is now more convenient to set this attribute using the SRW.SET_HYPERLINK procedure. Setting this attribute via SRW.SET_ATTR is still supported for compatibility, but it is highly recommended that you use the new, simplified procedure.

```

/* The format trigger below defines a Web link
** for the boilerplate object B_3. Notice how the
** destination of the link is determined dynamically
** based upon the values of SRCDIR and CF_PICKVIDEO.
** For example, if the value of SRCDIR were
** http://www.newmach.com and CF_PICKVIDEO were good.avi,
** this function would assign the following Web
** link to the object:
** http://www.newmach.com/webdemo/src/good.avi.
*/
function B_3FormatTrigger return boolean is
begin
  srw.attr.mask := srw.HYPERLINK_ATTR;
  srw.attr.hyperlink := :srcdir||:cf_pickvideo;
  if ( upper(:cf_pickvideo) like '%GOOD%' ) then
    srw.attr.mask := srw.attr.mask +
      srw.FBCOLOR_ATTR +
      srw.BBCOLOR_ATTR;
    srw.attr.fbcolor := 'green';
    srw.attr.bbcolor := 'green';
  end if;
  srw.set_attr(0,srw.attr);
  return (TRUE);
end;

/* SRCDIR is a parameter whose value is determined at
** runtime by the following After Parameter Form trigger
*/
function AfterPForm return boolean is
begin
  :srcdir := :web_server || '/webdemo/src/';
  :docsdir := :web_server || '/webdemo/docs/';
  return (TRUE);
end;

/* CF_PICKVIDEO is a formula column whose value is
** determined by the following function
*/

```

```
function CF_pickvideoFormula return Char is
begin
  if ( :avg_h_div < .80 )
    then return ('bad.avi');
    else return ('good.avi');
  end if;
end;
```

LINKTAG

New Feature: It is now more convenient to set this attribute using the SRW.SET_LINKTAG procedure. Setting this attribute via SRW.SET_ATTR is still supported for compatibility, but it is highly recommended that you use the new, simplified procedure.

Description Is an object's identifier and can be used as the destination in Web links.

Values

A valid, unique name that only makes use of the 26 upper or lower case US ASCII letters, numbers, or underscore characters. Other special characters will automatically be converted to underscore characters.

LINKTAG restrictions

- LINKTAG should only be set in the following triggers:

Format

LINKTAG example

Note: This example illustrates using SRW.SET_ATTR to set the LINKTAG attribute. It is now more convenient to set this attribute using the SRW.SET_LINKTAG procedure. Setting this attribute via SRW.SET_ATTR is still supported for compatibility, but it is highly recommended that you use the new, simplified procedure.

```
/* The format trigger below assigns an identifier
** (mytarget) to the boilerplate object B_500.
** This means that the object can now be the destination
** of a Web link.
*/
function B_500FormatTrigger return boolean is
begin
    srw.attr.mask := SRW.LINKTAG_ATTR;
    srw.attr.linktag := 'mytarget';
    srw.set_attr(0, SRW.ATTR);
    return (TRUE);
end;
```

Dynamic LINKTAG example

Note: This example illustrates using SRW.SET_ATTR to set the LINKTAG attribute. It is now more convenient to set this attribute using the SRW.SET_LINKTAG procedure. Setting this attribute via SRW.SET_ATTR is still supported for compatibility, but it is highly recommended that you use the new, simplified procedure.

```
/* The format trigger below assigns an identifier to
** the field F_Dept by concatenating some static text
** (dept_details_) with the value of the source column
** (DEPTNO). This function ensures that a unique
** identifier is assigned to each instance of F_Dept.
```

```
*/  
function F_DeptFormatTrigger return boolean is  
begin  
    srw.attr.mask := SRW.LINKTAG_ATTR;  
    srw.attr.linktag := 'dept_details_' ||  
        LTRIM(TO_CHAR(:deptno));  
    srw.set_attr(0, SRW.ATTR);  
    return (TRUE);  
end;
```

PRINTER_INTRAY

New Feature: It is now more convenient to set this attribute using the SRW.SET_PRINTER_TRAY procedure. Setting this attribute via SRW.SET_ATTR is still supported for compatibility, but it is highly recommended that you use the new, simplified procedure.

Description Is the name of a valid printer tray and can be used switch to different printer trays as your report formats.

Values

A valid, unique name as defined for your printer in the Page Setup dialog.

PRINTER_INTRAY restrictions

- PRINTER_INTRAY should only be set in the following triggers:

Between Pages

Before Report

Format

PRINTER_INTRAY example

Note: This example illustrates using SRW.SET_ATTR to set the PRINTER_INTRAY attribute. It is now more convenient to set this attribute using the SRW.SET_PRINTER_TRAY procedure. Setting this attribute via SRW.SET_ATTR is still supported for compatibility, but it is highly recommended that you use the new, simplified procedure.

```
/* The example below sets the printer tray in the Between Pages trigger.
*/
function BetweenPages return boolean is
begin
    srw.attr.mask           := SRW.PRINTER_INTRAY_ATTR;
    srw.attr.printer_intray := 'letterhead';
    srw.set_attr(SRW.REPORT_ID, srw.attr);
    return (TRUE);
end;
```

STYLE

New Feature: It is now more convenient to set this attribute using the SRW.SET_FONT_STYLE procedure. Setting this attribute via SRW.SET_ATTR is still supported for compatibility, but it is highly recommended that you use the new, simplified procedure.

Description Is the font style of the CHAR, DATE, or NUMBER field. This attribute is ignored in bit-mapped reports.

Values

srw.plain_style
srw.italic_style
srw.oblique_style
srw.underline_style
srw.outline_style
srw.shadow_style
srw.inverted_style
srw.overstrike_style
srw.blink_style

SZ

New Feature: It is now more convenient to set this attribute using the SRW.SET_FONT_SIZE procedure. Setting this attribute via SRW.SET_ATTR is still supported for compatibility, but it is highly recommended that you use the new, simplified procedure.

Description Is the font size of the CHAR, DATE, or NUMBER field. This attribute is ignored in character mode.

Values

A valid size for the named font on the machine where the report will run.

TEXT

New Feature: It is now more convenient to set this attribute using the SRW.SET_CHARMODE_TEXT procedure. Setting this attribute via SRW.SET_ATTR is still supported for compatibility, but it is highly recommended that you use the new, simplified procedure.

Description Is the character mode text characteristic for the CHAR, DATE, or NUMBER field. This attribute is ignored in bit-mapped reports.

Values

srw.plain_texta
srw.reverse_texta
srw.bold_texta
srw.reversebold_texta
srw.underline_texta
srw.underlinereverse_texta
srw.underlinebold_texta
srw.reverseboldunderline_texta

WEIGHT

New Feature: It is now more convenient to set this attribute using the `SRW.SET_FONT_WEIGHT` procedure. Setting this attribute via `SRW.SET_ATTR` is still supported for compatibility, but it is highly recommended that you use the new, simplified procedure.

Description Is the font weight of the CHAR, DATE, or NUMBER field. This attribute is ignored in character mode.

Values

`srw.ultralight_weight`

`srw.extralight_weight`

`srw.light_weight`

`srw.demilight_weight`

`srw.medium_weight`

`srw.demibold_weight`

`srw.bold_weight`

`srw.extrabold_weight`

The Report Builder PL/SQL package (SRW)

To save you time, Report Builder is shipped with a package --i.e., a collection of PL/SQL constructs--that contains many functions, procedures, and exceptions you can reference in any of your libraries or reports. The name of Report Builder's package is SRW. As a result, any time you reference a construct in the SRW package, you must prefix it with SRW, for example, SRW.DO_SQL.

Note: You cannot reference constructs in the SRW package from another product, e.g., from SQL*Plus.

Constructs found in a package are commonly referred to as "packaged"; i.e., packaged functions, packaged procedures, and packaged exceptions.

SRW.ADD_DEFINITION

Description This procedure takes an XML report definition stored in a column or variable and adds it to the document buffer. You may need to execute SRW.ADD_DEFINITION multiple times to get your entire report definition into the buffer. Once the definition is in the buffer, you can use SRW.APPLY_DEFINITION to apply it.

Syntax

```
SRW.ADD_DEFINITION(:column_name | variable);
```

Usage Notes

- To apply the XML report definition, use the SRW.APPLY_DEFINITION built-in. For example:

```
srw.add_definition (:xml_col);  
srw.apply_definition;
```

- If the XML report definition is located in a file, you can use SRW.APPLY_DEFINITION by itself to reference the file. For example:

```
srw.apply_definition('d:\xml_reps\web.xml');
```

SRW.APPLY_DEFINITION

Description This procedure takes an XML report definition stored in the document buffer or file system and applies it to the report. If applying a definition from the document buffer, SRW.APPLY_DEFINITION takes no arguments. If applying a definition from a file, SRW.APPLY_DEFINITION takes the name and path of the file as an argument.

Syntax

```
SRW.APPLY_DEFINITION[(filename.xml)];
```

Usage Notes

- If you want to apply more than one XML report definition, simply stack separate SRW.APPLY_DEFINITION statements in the order that you want them to be applied.
- If the XML report definition is located in a file, you can use SRW.APPLY_DEFINITION by itself to reference the file. For example:

```
srw.apply_definition('d:\xml_reps\web.xml');
```

- To apply an XML report definition from the document buffer you must first use SRW.ADD_DEFINITION to place the definition in the buffer. For example:

```
srw.add_definition (:xml_col);  
srw.apply_definition;
```

SRW.BREAK

Description This procedure temporarily stops report execution at the place at which SRW.BREAK was encountered, and displays (read-only) the current values of all columns and parameters. Report execution resumes when the read-only screen is accepted.

Syntax

```
SRW.BREAK ;
```

SRW.BREAK restrictions

- This procedure is not meaningful in a Before Form or After Form trigger, because data for the report is not fetched until the Before Report trigger (which fires after the After Form trigger).
- All column and parameter values are displayed in the read-only screen. No PL/SQL variables, or any other object values are displayed.

SRW.CONTEXT_FAILURE

Description This exception stops the report execution and raises the following error message:
REP-1426: Running <construct_name> from incorrect context.

Syntax

```
SRW.CONTEXT_FAILURE ;
```

Usage Notes Report Builder raises this exception when a Report Builder packaged function or procedure is called in the wrong context (see the chart below). In this chart, NO means that the function or procedure cannot be called in that context; YES means it can.

Name	Parameter	Data	Format	Report
	Form	Model	Trigger	Trigger
srw.break	NO	YES	YES	NO
srw.do_sql	YES	YES	YES	YES
srw.geterr_run	YES	YES	YES	YES
srw.get_page_num	NO	NO	YES	NO
srw.message	YES	YES	YES	YES
srw.reference	YES	YES	YES	YES
srw.run_report	YES	YES	YES	YES
srw.set_attr	NO	NO	YES	NO
srw.set_field_char	NO	NO	YES	NO
srw.set_field_date	NO	NO	YES	NO
srw.set_field_num	NO	NO	YES	NO
srw.set_maxrow	NO	YES	YES	YES
srw.user_exit	YES	YES	YES	YES

SRW.CONTEXT_FAILURE example

```
/* Suppose you want your own error message raised,  
** instead of the default error message.  
** You could handle this exception in the following way:  
*/  
  
EXCEPTION  
when SRW.CONTEXT_FAILURE then  
    srw.message(4000, 'Contact the Application  
    Development group regarding SRW.CONTEXT_FAILURE .');  
    raise srw.program_abort;
```

SRW.DO_SQL

Description This procedure executes the specified SQL statement from within Report Builder. The SQL statement can be DDL (statements that define data), or DML (statements that manipulate data). DML statements are usually faster when they are in PL/SQL, instead of in SRW.DO_SQL.

Since you cannot perform DDL statements in PL/SQL, the SRW.DO_SQL packaged procedure is especially useful for performing them within Report Builder, instead of via a user exit. For more information on DDL or DML statements, see the *ORACLE8 Server SQL Language Reference Manual*.

Syntax

```
SRW.DO_SQL (sql_statement CHAR);
```

Parameters

sql_statement	Is any valid SQL statement. Remember to precede any Report Builder object names with a colon (:).
---------------	---

SRW.DO_SQL restrictions

- In Report trigger order of execution, notice where the SET TRANSACTION READONLY occurs.
- A bind variable's value can be at most 64,000 bytes. (When the value exceeds that limit, it will be truncated to the left-most 64,000 bytes.)
- If you use a parameter as the destination of a character column for an INTO clause, you should ensure that the parameter is wide enough to contain the selected values. For example, suppose that you have the SRW.DO_SQL statement below: The destination parameter (my_ename) needs a width that is equal to the maximum width of the ENAME column. The reason for this is that the selected value contains trailing spaces up to the assumed size of the value. If the parameter is not large enough, you will get a truncation exception. If you are not sure about the maximum width of the SELECT list item, then you should use 2000 as the width for the parameter.

```
srw.do_sql('SELECT ENAME INTO :my_ename FROM EMP');
```

SRW.DO_SQL example

```
/* Suppose you want your report to create a table named CHECK
** just before the Runtime Parameter Form is displayed.
** Because CREATE TABLE is a SQL DDL statement (and PL/SQL
** cannot perform DDL statements), you need to use SRW.DO_SQL .
** Therefore, your PL/SQL could look like this in the Before Form
```

```

trigger:
*/
/* Additional Information: If you use a table created in this way for
your
** report output, the table must exist before you create your query in
the
** data model. Otherwise, Report Builder would not be able to parse
your query.
*/
FUNCTION CREATETAB RETURN BOOLEAN IS
BEGIN
SRW.DO_SQL('CREATE TABLE CHECK (EMPNO NUMBER NOT NULL
          PRIMARY KEY, SAL NUMBER (10,2)) PCTFREE 5
          PCTUSED 75');
RETURN(TRUE);
EXCEPTION
WHEN SRW.DO_SQL_FAILURE THEN
  SRW.MESSAGE(100, 'ERROR WHILE CREATING CHECK TABLE. ');
  SRW.MESSAGE(50, 'REPORT WAS STOPPED BEFORE THE RUNTIME
  PARAMETER FORM. ');
  RAISE SRW.PROGRAM_ABORT; -
END;

/* Suppose you want to create a "table of contents" by getting the
** first character of a column's value, and page number on which its
** field fires to print. Assume that you want to put the "table of
contents"
** into a table named SHIP. You could write the following construct:
*/

DECLARE
  PAGE_NO      NUMBER;
  PAGE_FOR     INDEX NUMBER;
  SORT_CHAR    CHAR(1);
  CMD_LINE     CHAR(200);
BEGIN
  SORT_CHAR := :SORT_NAME ;
  IF :CALLED = 'Y' THEN
    SRW.GET_PAGE_NUM(PAGE_FOR_INDEX);
    SRW.USER_EXIT('RWECOP PAGE_FOR_INDEX
      P_START_PAGENO');
    SRW.MESSAGE(2, TO_CHAR(:P_START_PAGENO));
  END IF;
  SRW.GET_PAGE_NUM(PAGE_NO);
  CMD_LINE := 'INSERT INTO SHIP VALUES
    ('' || SORT_CHAR || '' , '' || TO_CHAR(PAGE_NO) || '' )';
  SRW.MESSAGE(2, CMD_LINE);
  SRW.DO_SQL(CMD_LINE);

```

```
    COMMIT;
EXCEPTION
  WHEN DUP_VAL_ON_INDEX THEN
    NULL;
  WHEN SRW.DO_SQL_FAILURE THEN
    SRW.MESSAGE(1, 'FAILED TO INSERT ROW INTO SHIP
TABLE');
  WHEN OTHERS THEN
    COMMIT;
END;
```

SRW.DO_SQL_FAILURE

Description This exception stops the report execution and raises the following error message:

```
REP-1425: Error running DO_SQL package - REP-msg ORA-msg
```

where:

REP-msg Is a Report Builder message.

ORA-msg Is an optional ORACLE message, providing more information on the Report Builder message.

Syntax

```
SRW.DO_SQL_FAILURE ;
```

Usage Notes Report Builder raises this exception when the SRW.DO_SQL packaged procedure fails (e.g., if the user does not have DDL privileges, yet tries to create a table with SRW.DO_SQL).

SRW.DO_SQL_FAILURE example

```
/* Suppose you want your own error message raised,  
** instead of the default error message.  
** You could handle this exception in the following way:  
*/
```

```
EXCEPTION  
when SRW.DO_SQL_FAILURE then  
    srw.message(1000, 'Error occurred while creating  
        table CHECKS.');
```

SRW.FILE_ESCAPE

Indicates that the HTML is located in a file. When you specify SRW.FILE_ESCAPE, you should enter a filename for SRW.xxxx_ESCAPE_VALUE (where xxxx is the name of the escape, e.g., BEFREPORT).

SRW.GETERR_RUN

Description This function returns an error message if Report Builder detects an error while running the SRW.RUN_REPORT procedure.

Syntax

```
SRW.GETERR_RUN;
```

Returns An error message.

SRW.GETERR_RUN examples

```
/* Suppose you are sending parts of a report to users via Oracle*Mail.  
** For more information, see "SRW.RUN_REPORT". Also,  
** suppose that if SRW.RUN_REPORT fails, you want to display a message  
** that explains why it failed. Your PL/SQL could look like this:  
*/
```

```
BEGIN  
DECLARE TMP CHAR(100);  
begin  
    srw.run_report('batch=yes report=send.rdf  
destype=file desname=send.lis desformat=dflt');  
exception when srw.run_report_failure then  
    tmp := srw.geterr_run;  
    srw.message(1000, tmp);  
end;  
END;
```

SRW.GET_PAGE_NUM

Description This function returns the current page number. This is useful when you want to use the page number in the field's Format Trigger property.

Syntax

```
SRW.GET_PAGE_NUM (page_num);
```

Parameters

page_num	Is the variable in which you want to place the current page number.
----------	---

Returns The current page number.

SRW.GET_PAGE_NUM restrictions

- SRW.GET_PAGE_NUM is only meaningful in a format trigger. It has no effect when entered in other places.

SRW.GET_PAGE_NUM example

```
/* Suppose you want to perform a computation based upon a page number.  
** In the field's Format Trigger, you could use SRW.GET_PAGE_NUM  
function:  
*/
```

```
BEGIN  
DECLARE PAGE_NUM NUMBER;  
begin  
    srw.get_page_num (page_num);  
    srw.set_field_num (0, page_num + 3);  
end;  
END;
```

SRW.TEXT_ESCAPE

Indicates that the HTML is specified in SRW.xxxx_ESCAPE_VALUE (where xxxx is the name of the escape, e.g., BEFREPORT). When you specify SRW.TEXT_ESCAPE, you should enter a quoted string for SRW.xxxx_ESCAPE_VALUE.

Set Attributes Procedures

SRW.SET_AFTER_FORM_HTML
SRW.SET_AFTER_PAGE_HTML
SRW.SET_AFTER_PRINTING_CODE
SRW.SET_AFTER_REPORT_HTML
SRW.SET_BACKGROUND_BORDER_COLOR
SRW.SET_BACKGROUND_FILL_COLOR
SRW.SET_BEFORE_FORM_HTML
SRW.SET_BEFORE_PAGE_HTML
SRW.SET_BEFORE_PRINTING_CODE
SRW.SET_BEFORE_REPORT_HTML
SRW.SET_BOOKMARK
SRW.SET_BORDER_PATTERN
SRW.SET_BORDER_WIDTH
SRW.SET_CHARMODE_TEXT
SRW.SET_CUSTOM_SPACING
SRW.SET_DISPLAY_NAME
SRW.SET_FIELD
SRW.SET_FIELD_CHAR
SRW.SET_FIELD_DATE
SRW.SET_FIELD_NUM
SRW.SET_FILL_PATTERN
SRW.SET_FONT_FACE
SRW.SET_FONT_SIZE
SRW.SET_FONT_STYLE
SRW.SET_FONT_WEIGHT
SRW.SET_FOREGROUND_BORDER_COLOR
SRW.SET_FOREGROUND_FILL_COLOR
SRW.SET_FORMAT_MASK
SRW.SET_HYPERLINK
SRW.SET_HYPERLINK_ATTRS
SRW.SET_JUSTIFICATION
SRW.SET_LINKTAG
SRW.SET_MAXROW
SRW.SET_PAGE_NAVIGATION_HTML
SRW.SET_PDF_ACTION
SRW.SET_PRINTER_TRAY
SRW.SET_SPACING
SRW.SET_TEXT_COLOR

SRW.SET_AFTER_FORM_HTML

Description This procedure inserts any text, graphics, or HTML commands that you want to appear at the bottom of the HTML Parameter Form. This attribute is useful for placing a logo or some standard links on the Parameter Form.

Syntax

```
SRW.SET_AFTER_FORM_HTML(type,'string');
```

Parameters

type	Is SRW.FILE_ESCAPE or SRW.TEXT_ESCAPE. It indicates whether the string parameter is a filename or the text to be inserted.
string	Is a filename or the text to be inserted depending on what was specified for the type parameter.

Property Palette To define this attribute using the Property Palette, set the After Form Type and After Form Value properties.

SRW.SET_AFTER_FORM_HTML example

```
/* This example inserts two links to home pages
** at the bottom of the HTML Parameter Form.
*/
function BeforeForm return boolean is
begin
if ((upper(:MODE) != 'CHARACTER') and
    (upper(:DESFORMAT) = 'HTML')) then
    SRW.SET_AFTER_FORM_HTML(SRW.TEXT_ESCAPE,'<CENTER>' ||
        '<A HREF="http://www.oracle.com/">' ||
        'Oracle Corporation</A> -' ||
        'chr(10)' ||
        '<A HREF="http://home.netscape.com/">' ||
        'Netscape</A> </CENTER>' ||
        '</BODY> </HTML>');
end if;
return (TRUE);
end;
```

SRW.SET_AFTER_PAGE_HTML

Description This procedure inserts any text, graphics, or HTML commands that you want to appear at the end of pages of your document. This attribute is useful for placing a logo or some standard links at the end of each page in an HTML document.

If you use HTML page streaming, this procedure applies to all pages of your report output. To specify HTML for only the first (header) or last (footer) pages of your report, use the SRW.SET_BEFORE_REPORT_HTML or SRW.SET_AFTER_REPORT_HTMLPL/SQL procedures, respectively.

Syntax

```
SRW.SET_AFTER_PAGE_HTML(type,'string');
```

Parameters

type	Is SRW.FILE_ESCAPE or SRW.TEXT_ESCAPE. It indicates whether the string parameter is a filename or the text to be inserted.
string	Is a filename or the text to be inserted, depending on what you specified for the type parameter.

Property Palette To define this attribute using the Property Palette, set the After Page Type and After Page Value properties.

SRW.SET_AFTER_PAGE_HTML restrictions

- The default HTML included at the end of a page is shown below. It is not required unless you want the default separator line to appear between pages.

```
<hr size=5 noshade>
```
- If you want SRW.SET_AFTER_PAGE_HTML to apply to every page of the report, you should specify it in a trigger that fires before the report begins formatting, such as the Before Report trigger.
- If you want SRW.SET_AFTER_PAGE_HTML to apply only to the current page, you should specify it in a format trigger for an object on that page.

SRW.SET_AFTER_PAGE_HTML example

```
/* The example below inserts a GIF file as a
** page separator.
*/
function BeforeReport return boolean is
begin
if ((upper(:MODE) != 'CHARACTER') and
    (upper(:DESFORMAT) = 'HTML')) then
```

```
SRW.SET_AFTER_PAGE_HTML(SRW.TEXT_ESCAPE, '<CENTER>' ||  
'<IMG ALT="Ruler" SRC="line.gif" VSPACE=10>' ||  
'<BR></CENTER>');  
end if;  
return (TRUE);  
end;
```

SRW.SET_AFTER_PRINTING_CODE

Description This procedure inserts a reference to the printer escape sequence to be executed after each line of the object. The printer escape sequence is inserted after the object is triggered to print but before it has actually printed. Printer codes are only used when running your report in character-mode.

Syntax

```
SRW.SET_AFTER_PRINTING_CODE('code');
```

Parameters

code	Is a valid printer code that is defined in the .prt file specified for DESFORMAT.
------	---

Property Palette To define this attribute using the Property Palette, set the Printer Code After property.

SRW.SET_AFTER_REPORT_HTML

Description This procedure inserts any text, graphics, or HTML commands that you want to appear at the end of your document. This attribute is useful for placing a logo or some standard links at the end of an HTML document.

If you use HTML page streaming, this procedure applies to the first page of your report output. To specify HTML to apply to the entire report (e.g., background color or images, or any other <body> HTML attributes), you must use the SRW.SET_BEFORE_PAGE_HTML.PL/SQL procedure.

Syntax

```
SRW.SET_AFTER_REPORT_HTML(type, 'string');
```

Parameters

type	Is SRW.FILE_ESCAPE or SRW.TEXT_ESCAPE. It indicates whether the string parameter is a filename or the text to be inserted.
string	Is a filename or the text to be inserted, depending on what you specified for the type parameter.

Property Palette To define this attribute using the Property Palette, set the After Report Type and After Report Value properties.

SRW.SET_AFTER_REPORT_HTML restrictions

- The default HTML included at the end of a report is shown below. If you use SRW.SET_AFTER_REPORT_HTML to replace this text, you must ensure that it contains the equivalent HTML commands.
`</body></html>`
- SRW.SET_AFTER_REPORT_HTML should be set in a trigger that fires before the report is done formatting, such as the Before Report trigger.

SRW.SET_AFTER_REPORT_HTML example

```
/* The example below inserts two links to home pages
** at the end of the report output.
*/
function BeforeReport return boolean is
begin
if ((upper(:MODE) != 'CHARACTER') and
    (upper(:DESFORMAT) = 'HTML')) then
    SRW.SET_AFTER_REPORT_HTML(SRW.TEXT_ESCAPE, '<CENTER>' ||
        '<A HREF="http://www.oracle.com/">' ||
```

```
'Oracle Corporation</A> -' ||  
chr(10) ||  
'<A HREF="http://home.netscape.com/">' ||  
'Netscape</A> </CENTER>' ||  
'</BODY> </HTML>');  
end if;  
return (TRUE);  
end;
```

SRW.SET_BACKGROUND_BORDER_COLOR

Description This procedure specifies the background border color of the object. This attribute is ignored in character mode. Depending upon the pattern being used, you may not see the background color. For example, if the pattern is solid, the background color does not show through the pattern. Only the foreground color shows through in this case.

Syntax

```
SRW.SET_BACKGROUND_BORDER_COLOR('color');
```

Parameters

color Is a valid color from the color palette.

Usage Notes

- This procedure does not apply to Windows platforms because they do not support a border pattern. As a result, the border foreground color is the only one visible on Windows.

Property Palette (Templates only) To define this attribute for templates using the Property Palette, set the Edge Background Color property.

User Interface To set the background border color in the user interface, refer to Changing colors .

SRW.SET_BACKGROUND_FILL_COLOR

Description This procedure specifies the background fill color of the object. This attribute is ignored in character mode. Depending upon the pattern being used, you may not see the background color. For example, if the pattern is solid, the background color does not show through the pattern. Only the foreground color shows through in this case.

Syntax

```
SRW.SET_BACKGROUND_FILL_COLOR('color');
```

Parameters

color Is a valid color from the color palette.

Property Palette (Templates only) To define this attribute for templates using the Property Palette, set the Background Color property.

User Interface To set the background fill color in the user interface, refer to Changing colors .

SRW.SET_BEFORE_FORM_HTML

Description This procedure inserts any text, graphics, or HTML commands that you want to appear at the top of the HTML Parameter Form. This attribute is useful for placing a logo or some standard links on the Parameter Form .

Syntax

```
SRW.SET_BEFORE_FORM_HTML(type, 'string');
```

Parameters

type	Is SRW.FILE_ESCAPE or SRW.TEXT_ESCAPE. It indicates whether the string parameter is a filename or the text to be inserted.
string	Is a filename or the text to be inserted, depending on what you specified for the type parameter.

Property Palette To define this attribute using the Property Palette, set the Before Form Type and Before Form Value properties.

SRW.SET_BEFORE_FORM_HTML example

```
/* The example below sets both the before and after
** form escapes in the Before Form trigger.
** The HTML for the before escape is located in a file
** named logo.htm. The HTML for the after report escape
** is specified within the PL/SQL itself.
*/
function BeforeForm return boolean is
begin
if ((upper(:MODE) != 'CHARACTER') and
    (upper(:DESFORMAT) = 'HTML')) then
    SRW.SET_BEFORE_FORM_HTML(SRW.FILE_ESCAPE, 'logo.htm');
    SRW.SET_AFTER_FORM_HTML(SRW.TEXT_ESCAPE, <B> ||
        '<FONT FACE="Arial, Helvetica">' ||
        '<FONT COLOR="#FF0000">' ||
        'Click <IMG SRC="RUN.GIF" HEIGHT=18 WIDTH=18>' ||
        'to run the report using the parameters you have' ||
        'specified above. ' ||
        '</FONT></FONT></B>');
end if;
return (TRUE);
end;
```

SRW.SET_BEFORE_PAGE_HTML

Description This procedure inserts any text, graphics, or HTML commands that you want to appear at the beginning of pages of your document. This attribute is useful for placing a logo or some standard links at the beginning of each page in an HTML document.

If you use HTML page streaming, this procedure applies to all pages of your report output (e.g., background color or images, or any other <body> HTML attributes). To specify HTML for only the first (header) or last (footer) pages of your report, use the SRW.SET_BEFORE_REPORT_HTML or SRW.SET_AFTER_REPORT_HTML PL/SQL procedures, respectively.

Syntax

```
SRW.SET_BEFORE_PAGE_HTML(type, 'string');
```

Parameters

type	Is SRW.FILE_ESCAPE or SRW.TEXT_ESCAPE. It indicates whether the string parameter is a filename or the text to be inserted.
string	Is a filename or the text to be inserted, depending on what you specified for the type parameter.

Property Palette To define this attribute using the Property Palette, set the Before Page Type and Before Page Value properties.

SRW.SET_BEFORE_PAGE_HTML restrictions

- If you want the SRW.SET_BEFORE_PAGE_HTML to apply to every page of the report, you should specify it in a trigger that fires before the report begins formatting, such as the Before Report trigger .
- If you want the SRW.SET_BEFORE_PAGE_HTML to apply only to the current page, you should specify it in a format trigger for an object on that page.

SRW.SET_BEFORE_PAGE_HTML example

```
/* The example below centers the document in
** the browser.
*/
function BeforeReport return boolean is
begin
if ((upper(:MODE) != 'CHARACTER') and
    (upper(:DESFORMAT) = 'HTML')) then
    SRW.SET_BEFORE_PAGE_HTML(SRW.TEXT_ESCAPE, '<CENTER>');
```

```
    SRW.SET_AFTER_PAGE_HTML(SRW.TEXT_ESCAPE, '</CENTER>');  
end if;  
return (TRUE);  
end;
```

SRW.SET_BEFORE_PRINTING_CODE

Description This procedure inserts a reference to the printer escape sequence to be executed before each line of the object. The printer escape sequence is inserted after the object is triggered to print but before it has actually printed. Printer codes are only used when running your report in character-mode.

Syntax

```
SRW.SET_BEFORE_PRINTING_CODE('code');
```

Parameters

code	Is a valid printer code that is defined in the .prt file specified for DESFORMAT.
------	---

Property Palette To define this attribute using the Property Palette, set the Printer Code Before property.

SRW.SET_BEFORE_REPORT_HTML

Description This procedure inserts any text, graphics, or HTML commands that you want to appear at the beginning of your document. This attribute is useful for placing a logo or some standard links at the beginning of an HTML document.

If you use HTML page streaming, this procedure applies to the first page of your report output. To specify HTML to apply to the entire report (e.g., background color or images, or any other <body> HTML attributes), you must use the SRW.SET_BEFORE_PAGE_HTML PL/SQL procedure.

Syntax

```
SRW.SET_BEFORE_REPORT_HTML(type, 'string');
```

Parameters

type	Is SRW.FILE_ESCAPE or SRW.TEXT_ESCAPE. It indicates whether the string parameter is a filename or the text to be inserted.
string	Is a filename or the text to be inserted, depending on what you specified for the type parameter.

Property Palette To define this attribute using the Property Palette, set the Before Report Type and Before Report Value properties.

SRW.SET_BEFORE_REPORT_HTML restrictions

- The default HTML included at the beginning of a report is shown below. If you use SRW.SET_BEFORE_REPORT_HTML to replace this text, you must ensure that it contains the equivalent HTML commands.

```
<html>
<body bgcolor="#ffffff">
```

- SRW.SET_BEFORE_REPORT_HTML should be set in a trigger that fires before the report starts formatting, such as the Before Report trigger.

SRW.SET_BEFORE_REPORT_HTML example

```
/* The example below sets both the before and after
** report escapes in the Before Report trigger.
** The HTML for the before escape is located in a file
** named logo.htm. The HTML for the after report escape
** is specified within the PL/SQL itself.
*/
function BeforeReport return boolean is
begin
```

```
if ((upper(:MODE) != 'CHARACTER') and
    (upper(:DESFORMAT) = 'HTML')) then
  SRW.SET_BEFORE_REPORT_HTML(SRW.FILE_ESCAPE, 'logo.htm');
  SRW.SET_AFTER_REPORT_HTML(SRW.TEXT_ESCAPE, '<ADDRESS>' ||
  ' Questions? - ' ||
  '<A HREF=mailto:webmaster@xyztech.com>' ||
  'webmaster@xyztech.com</A>' ||
  '</ADDRESS>' ||
  '</body></html>');
end if;
return (TRUE);
end;
```

SRW.SET_BOOKMARK

Description This procedure associates a bookmark with the object and specifies a string that will appear in a bookmark frame of the master HTML document or the PDF document. If you open the bookmark area and click on the bookmark, the object appears at the top of the window.

Syntax

```
SRW.SET_BOOKMARK('bookmark');
```

Parameters

bookmark Is one of the following:

- A text string with no indentation/ordering information. The bookmark will appear in the bookmark list according to when the object is rendered by the report.
- A text string with explicit ordering/indentation of the form `x#book_name`, where `x` is an outline number. The pound sign (#) and outline number do not appear in the bookmark window but are used to determine order and indentation. For example:

```
1#Expense Summary Section
2#Expense Detail Section
2.1#Expenses for the Administration
Department
2.2#Expenses for the Engineering
Department
2.3#Expenses for the Sales Department
2.3.1#Expenses for the Eastern Sales
Region
2.3.2#Expenses for the Northern Sales
Region
2.3.3#Expenses for the Southern Sales
Region
2.3.4#Expenses for the Western Sales
Region
```

Property Palette To define this attribute using the Property Palette, set the Bookmark property.

SRW.SET_BOOKMARK restrictions

- If the same outline number is used multiple times, all entries appear but the order is defined by when the objects are rendered by the report.
- If there are gaps in the numbers, one of two things will happen. If the gap is between peer level numbers, there will be no visible effect (e.g., 1.3.1 and 1.3.3, given there is no 1.3.2, will appear next to each other and at the same indentation level). If the gap is between a higher level number and a lower level number, intermediate levels will be generated as required (e.g., 1.0 followed by 2.1.1 will cause dummy 2 and 2.1 entries to be defined, whose titles will be the same as the subsequent real entry).
- SRW.SET_BOOKMARK should only be set in the following triggers:
 - Format

SRW.SET_BOOKMARK example

```
/* The format trigger below defines a bookmark for
** the boilerplate object B_2. Notice the use of
** explicit ordering information (1#) in this example.
** If you did not want the ordering information, you
** could omit 1#.
*/
function B_2FormatTrigger return boolean is
begin
  srw.set_bookmark('1#Expense Summary Section');
  return (TRUE);
end;
```

Dynamic SRW.SET_BOOKMARK example

```
/* The format trigger below defines a bookmark for
** the boilerplate object B_2. Notice that the name
** of the bookmark is dynamic. CATEGORY is a column
** value that is concatenated with the string Expense
** Summary Section for each execution of the format
** trigger. In this case, CATEGORY could contain
** ordering information (e.g., 1#) or perhaps a string
** that makes the bookmark unique within the report.
*/
function B_2FormatTrigger return boolean is
begin
  srw.set_bookmark(:category ||
  ' Expense Summary Section');
  return (TRUE);
end;
```

SRW.SET_BORDER_PATTERN

Description This procedure specifies the fill pattern for the border of the object. This attribute is ignored in character mode.

Syntax

```
SRW.SET_BORDER_PATTERN('pattern');
```

Parameters

pattern Is the fill pattern for the border (e.g., solid).

Usage Notes

- This procedure does not apply to Windows platforms because they do not support a border pattern.

Property Palette (Templates only) To define this attribute for templates using the Property Palette, set the Edge Pattern property.

User Interface To set the border attributes in the user interface, refer to [Changing object border attributes](#) .

SRW.SET_BORDER_WIDTH

Description This procedure specifies the width of the object's border in picas.

Syntax

```
SRW.SET_BORDER_WIDTH(width);
```

Parameters

width	Is zero or any positive integer. Zero indicates no border and a positive integer indicates the width of the border in picas.
-------	--

User Interface To set the border attributes in the user interface, refer to [Changing object border attributes](#) .

SRW.SET_CHARMODE_TEXT

Description This procedure specifies the font style of a CHAR, DATE, or NUMBER field. This attribute is ignored in bit-mapped reports.

Syntax

```
SRW.SET_CHARMODE_TEXT('style');
```

Parameters

style Is one of the following:

```
SRW.REVERSE_TEXTA  
SRW.BOLD_TEXTA  
SRW.REVERSEBOLD_TEXTA  
SRW.UNDERLINE_TEXTA  
SRW.UNDERLINEREVERSE_TEXTA  
SRW.REVERSEBOLDUNDERLINE_TEXTA  
A  
SRW.PLAIN_TEXTA
```

User Interface To set the font attributes in the user interface, refer to [Changing text attributes](#) .

SRW.SET_CUSTOM_SPACING

Description This procedure specifies the global text leading for a CHAR, DATE, or NUMBER field, or boilerplate text. This attribute is ignored in character mode.

Syntax

```
SRW.SET_CUSTOM_SPACING(spacing);
```

Parameters

spacing	Is a number that represents the leading in VGS units, which are very small.
---------	---

SRW.SET_DISPLAY_NAME

Description This procedure specifies text that displays in a popup as the cursor moves over an image object in reports output in HTML or HTMLCSS format.

Syntax

```
SRW.SET_DISPLAY_NAME('text_string');
```

Parameters

`text_string` Is any text string that only makes use of the 26 upper or lower case US ASCII letters, numbers, or underscore characters. Other special characters will automatically be converted to underscore characters.

Property Palette To define this attribute using the Property Palette, set the Display Name property.

SRW.SET_DISPLAY_NAME restrictions

- SRW.SET_DISPLAY_NAME should only be set in the following triggers:
 - Format

SRW.SET_DISPLAY_NAME example

```
/* The format trigger below assigns a text string
** ('Click here to fill your shopping basket') to the image
** object BASKET. In the HTML report, this popup will display
** when the cursor moves over the BASKET object.
*/
function BASKETFormatTrigger return boolean is
begin
  SRW.SET_DISPLAY_NAME('Click here to fill your shopping basket');
  return (TRUE);
end;
```

SRW.SET_FIELD

Description This procedure sets the value of a character, number, or date field. This is useful when you want to conditionally change a field's value.

Syntax

```
SRW.SET_FIELD (object_id, text CHAR|number NUM|date DATE);
```

Parameters

object_id	Is always 0. (The object must always set its own attributes.)
text	Is the character, number, or date string you want the field to display.
number	
date	

Usage Notes

- For date values, you need to convert the second argument to a date value with the TO_DATE function. For example:

```
srw.set_field(0, to_date('01-JAN-99'));
```

- If you omit quotes around the value, the value is assumed to be a number. For example:

```
srw.set_field(0, 99);
```

Alternatively, you can use the quotes if you also use the TO_NUMBER function. For example:

```
srw.set_field(0, to_number('99'));
```

- If you use quotes without a function, the value is assumed to be a character string. Alternatively, you can use the TO_CHAR function for consistency with number and date values:

```
srw.set_field(0, to_char('my string'));
```

SRW.SET_FIELD Example

```
/* Suppose you want to conditionally change the
** number of a field, based on each employee's salary.
** In the format trigger for the field, you could
** type the following:
*/
```

```
FUNCTION CHGFIELD RETURN BOOLEAN IS
TMP    NUMBER;
BEGIN
    if :sal >= 2000 then
        tmp := :sal * 1.08;
        srw.set_field (0, tmp);
```



```
else
  srw.set_field (0, 2500);
end if;
RETURN (TRUE);
END;
```

SRW.SET_FIELD_CHAR

Description This procedure sets the value of a character field. This is useful when you want to conditionally change a field's character value.

Syntax

```
SRW.SET_FIELD_CHAR (object_id, text CHAR);
```

Parameters

object_id	Is always 0. (The object must always set its own attributes.)
text	Is the character string you want the field to display.

SRW.SET_FIELD_CHAR restrictions

- SRW.SET_FIELD_CHAR is only meaningful in the format trigger of a field of Datatype Character. It has no affect when entered in other places.

SRW.SET_FIELD_CHAR example

```
/* Suppose you want to conditionally change the value of a
** Character field, based on each employee's salary.
** In the format trigger for the field, you could type the following:
*/
```

```
FUNCTION CHGFIELD RETURN BOOLEAN IS
BEGIN
    if :sal >= 2000000 then
        srw.set_field_char (0, 'HIGH SALARY');
    end if;
RETURN (TRUE);
END;
```

SRW.SET_FIELD_DATE

Description This procedure sets the value of a date field. This is useful when you want to conditionally change a field's date value.

Syntax

```
SRW.SET_FIELD_DATE (object_id, date DATE);
```

Parameters:

object_id	Is always 0. (The object must always set its own attributes.)
date	Is the date you want the field to display.

SRW.SET_FIELD_DATE restrictions

- SRW.SET_FIELD_DATE is only meaningful in a format trigger for a date field. It has no affect when entered in other places.

SRW.SET_FIELD_DATE example

```
/* Suppose you want to conditionally change the date of the reunion
** invitation, based on each student's graduation year. In the format
** trigger for the field, you could type the following:
*/
```

```
FUNCTION CHGFIELD RETURN BOOLEAN IS
BEGIN
  if :graduation >= 1975 then
    srw.set_field_date (0, '02-JUL-95');
  else
    end if;
RETURN (TRUE);
END;
```

SRW.SET_FIELD_NUM

Description This procedure sets the value of a number field. This is useful when you want to conditionally change a field's number value.

Syntax

```
SRW.SET_FIELD_NUM (object_id, number NUM);
```

Parameters

object_id	Is always 0. (The object must always set its own attributes.)
number	Is the number you want the field to display.

SRW.SET_FIELD_NUM restrictions

- SRW.SET_FIELD_NUM is only meaningful in the format trigger of a field of Datatype NUMBER. It has no affect when entered in other places.

SRW.SET_FIELD_NUM example

```
/* Suppose you want to conditionally change the number of a field,  
** based on each employee's salary. In the format trigger for the  
** field, you could type the following:  
*/
```

```
FUNCTION CHGFIELD RETURN BOOLEAN IS  
TMP    NUMBER;  
BEGIN  
    if :sal >= 2000 then  
        tmp := :sal * 1.08;  
        srw.set_field_num (0, tmp);  
    else  
        srw.set_field_num (0, '2500');  
    end if;  
RETURN (TRUE);  
END;
```

SRW.SET_FILL_PATTERN

Description This procedure specifies the fill pattern of the object. This attribute is ignored in character mode.

Syntax

```
SRW.SET_FILL_PATTERN('pattern');
```

Parameters

pattern Is the fill pattern for the object (e.g., solid).

Property Palette (Templates only) To define this attribute for templates using the Property Palette, set the Fill Pattern property.

User Interface To set the border attributes in the user interface, refer to [Changing patterns](#).

SRW.SET_FONT_FACE

Description This procedure specifies font face for a CHAR, DATE, or NUMBER field. This attribute is ignored in character mode.

Syntax

```
SRW.SET_FONT_FACE('face');
```

Parameters

face Is the font face, e.g., arial.

Property Palette (Templates only) To define this attribute for templates using the Property Palette, set the Font property.

User Interface To set the font attributes in the user interface, refer to Changing text attributes .

SRW.SET_FONT_SIZE

Description This procedure specifies font size for a CHAR, DATE, or NUMBER field. This attribute is ignored in character mode.

Syntax

```
SRW.SET_FONT_SIZE(size);
```

Parameters

size Is the font size, e.g., 9 (point).

Property Palette (Templates only) To define this attribute for templates using the Property Palette, set the Font property.

User Interface To set the font attributes in the user interface, refer to Changing text attributes .

SRW.SET_FONT_STYLE

Description This procedure specifies font style for a CHAR, DATE, or NUMBER field. This attribute is ignored in character mode.

Syntax

```
SRW.SET_FONT_STYLE(style);
```

Parameters

style	Is one of the following:
	SRW.ITALIC_STYLE
	SRW.OBLIQUE_STYLE
	SRW.UNDERLINE_STYLE
	SRW.OUTLINE_STYLE
	SRW.SHADOW_STYLE
	SRW.INVERTED_STYLE
	SRW.BLINK_STYLE
	SRW.PLAIN_STYLE

Usage Notes

- If you use this procedure multiple times on the same field, the font styles will accumulate. For example, the following would cause the field to be underlined and italic:

```
SRW.SET_FONT_STYLE(SRW.UNDERLINE_STYLE);  
SRW.SET_FONT_STYLE(SRW.ITALIC_STYLE);
```

To negate all of the accumulated styles, use the PLAIN style.

Property Palette (Templates only) To define this attribute for templates using the Property Palette, set the Font property.

User Interface To set the font attributes in the user interface, refer to [Changing text attributes](#) .

SRW.SET_FONT_WEIGHT

Description This procedure specifies font weight for a CHAR, DATE, or NUMBER field. This attribute is ignored in character mode.

Syntax

```
SRW.SET_FONT_WEIGHT(weight);
```

Parameters

weight Is one of the following:

SRW.ULTRALIGHT_WEIGHT
SRW.EXTRALIGHT_WEIGHT
SRW.LIGHT_WEIGHT
SRW.DEMILIGHT_WEIGHT
SRW.DEMIBOLD_WEIGHT
SRW.BOLD_WEIGHT
SRW.EXTRABOLD_WEIGHT
SRW.MEDIUM_WEIGHT

Property Palette (Templates only) To define this attribute for templates using the Property Palette, set the Font property.

User Interface To set the font attributes in the user interface, refer to [Changing text attributes](#) .

SRW.SET_FOREGROUND_BORDER_COLOR

Description This procedure specifies the foreground border color of the object. This attribute is ignored in character mode.

Syntax

```
SRW.SET_FOREGROUND_BORDER_COLOR('color');
```

Parameters

color Is a valid color from the color palette.

Property Palette (Templates only) To define this attribute for templates using the Property Palette, set the `Edge Foreground Color` property.

User Interface To set the foreground border color in the user interface, refer to `Changing colors` .

SRW.SET_FOREGROUND_FILL_COLOR

Description This procedure specifies the foreground fill color of the object. This attribute is ignored in character mode.

Syntax

```
SRW.SET_FOREGROUND_FILL_COLOR('color');
```

Parameters

color Is a valid color from the color palette.

Property Palette (Templates only) To define this attribute for templates using the Property Palette, set the Foreground Color property.

User Interface To set the foreground fill color in the user interface, refer to Changing colors .

SRW.SET_FORMAT_MASK

Description This procedure specifies the format mask for the DATE or NUMBER field.

Syntax

```
SRW.SET_FORMAT_MASK('mask');
```

Parameters

mask	Is a valid format mask. See Date Format Mask Syntax or Number Format Mask Syntax
------	--

Property Palette To define this attribute using the Property Palette, set the Format Mask property.

SRW.SET_HYPERLINK

Description This procedure specifies a URL Web link specification.

Note: An object that is the source of a Web link cannot also be associated with an action.

Syntax

```
SRW.SET_HYPERLINK('hyperlink');
```

Parameters

- | | |
|-----------|------------------|
| hyperlink | Is a valid link: |
|-----------|------------------|
- To another document on the same machine (e.g.,
file:/private/mynewdoc.pdf or
file:///C:/temp/mynewdoc.pdf)
 - To another document on a different machine (e.g.,
http://www.newmach.com/newdoc.pdf)
 - To a destination within the current document (e.g.,
#my_dest_name)
 - To a destination within a local document (e.g.,
file:/private/somedoc.pdf#a_dest_name)
 - To a destination within a remote document (e.g.,
http://www.newmach.com/newdoc.pdf#some_dest_name)
 - To any URL (e.g.,
http://www.newmach.com/newdoc.html,
ftp://www.reposit.com/filetoget.example,
http://www.somemch.com/cgi-bin/webmenu?choice1)

Property Palette To define this attribute using the Property Palette, set the Hyperlink property.

SRW.SET_HYPERLINK restrictions

- SRW.SET_HYPERLINK should only be set in the following triggers:
 - format
- To follow Web links from a PDF viewer to a remote server or HTML document, the PDF viewer must be configured to work with a Web browser (e.g., configured

as a helper application or installed as a plug-in to your Web browser).

SRW.SET_HYPERLINK example

```
/* The format trigger below defines a Web link
** to a destination (mytarget) in another document
** (newdoc.pdf) for the boilerplate object B_2.
** Note: If the target were in the same document,
** you would omit http://www.newmach.com/newdoc.pdf.
*/

function B_2FormatTrigger return boolean is
begin
  SRW.SET_HYPERLINK('http://www.newmach.com/newdoc.pdf' ||
    '#mytarget');
  return (TRUE);
end;
```

Dynamic SRW.SET_HYPERLINK example

```
/* The format trigger below defines a Web link
** for the boilerplate object B_3. Notice how the
** destination of the link is determined dynamically
** based upon the values of SRCDIR and CF_PICKVIDEO.
** For example, if the value of SRCDIR were
** http://www.newmach.com and CF_PICKVIDEO were good.avi,
** this function would assign the following Web
** link to the object:
** http://www.newmach.com/webdemo/src/good.avi.
*/

function B_3FormatTrigger return boolean is
begin
  SRW.SET_HYPERLINK(:srcdir||:cf_pickvideo);
  if ( upper(:cf_pickvideo) like '%GOOD%' ) then
    SRW.SET_FOREGROUND_BORDER_COLOR('green');
    SRW.SET_BACKGROUND_BORDER_COLOR('green');
  end if;
  return (TRUE);
end;

/* SRCDIR is a parameter whose value is determined at
** runtime by the following After Parameter Form trigger
*/

function AfterPForm return boolean is
begin
  :srcdir := :web_server || '/webdemo/src/';
  :docsdir := :web_server || '/webdemo/docs/';
  return (TRUE);
end;

/* CF_PICKVIDEO is a formula column whose value is
** determined by the following function
*/

function CF_pickvideoFormula return Char is

begin
```

```
if ( :avg_h_div < .80 )
  then return ('bad.avi');
  else return ('good.avi');
end if;
end;
```

SRW.SET_HYPERLINK_ATTRS

Description This procedure applies additional HTML commands to the hyperlink specified in the Property Palette or SRW.SET_HYPERLINK procedure.

Syntax

```
SRW.SET_HYPERLINK_ATTRS('string');
```

Parameters

HTML_string Is a string containing the HTML commands to be applied.

Property Palette To define this attribute using the Property Palette, set the Additional Hyperlink Attributes property.

SRW.SET_HYPERLINK_ATTRS restrictions

- SRW.SET_HYPERLINK_ATTRS should only be set in the following triggers:
 - format

SRW.SET_HYPERLINK_ATTRS example

```
/* The format trigger below generates a status
** of information at the bottom of your Web browser
** when the mouse cursor passes over a Shopping Basket
** object.
*/
function BASKETFormatTrigger return boolean is
begin
SRW.SET_HYPERLINK_ATTRS('onMouseover "window.status='Click here to see
what is in your shopping basket';return true" onMouseout
="window.status=' ';return true"');
return (TRUE);
end;
```

SRW.SET_JUSTIFICATION

Description This procedure specifies the horizontal justification of the CHAR, DATE, or NUMBER field.

Syntax

```
SRW.SET_JUSTIFICATION(justification);
```

Parameters

justification Is one of the following:

SRW.LEFT_HJUST
SRW.RIGHT_HJUST
SRW.FLUSH_HJUST
SRW.CENTER_HJUST.

Property Palette (Templates only) To define this attribute in templates using the Property Palette, set the Character Justification property for character fields, Date Justification property for date fields, Number Justification property for number fields.

User Interface To set the horizontal justification in the user interface, refer to Changing text attributes .

SRW.SET_LINKTAG

Description This procedure specifies an object's identifier and can be used as the destination in Web links.

Syntax

```
SRW.SET_LINKTAG('linktag');
```

Parameters

linktag	A valid, unique name that only makes use of the 26 upper or lower case US ASCII letters, numbers, or underscore characters. Other special characters will automatically be converted to underscore characters.
---------	--

Property Palette To define this attribute using the Property Palette, set the Hyperlink Destination property.

SRW.SET_LINKTAG restrictions

- SRW.SET_LINKTAG should only be set in the following triggers:
 - format

SRW.SET_LINKTAG example

```
/* The format trigger below assigns an identifier
** (mytarget) to the boilerplate object B_500.
** This means that the object can now be the destination
** of a Web link.
*/
function B_500FormatTrigger return boolean is
begin
    SRW.SET_LINKTAG('mytarget');
    return (TRUE);
end;
```

Dynamic SRW.SET_LINKTAG example

```
/* The format trigger below assigns an identifier to
** the field F_Dept by concatenating some static text
** (dept_details_) with the value of the source column
** (DEPTNO). This function ensures that a unique
** identifier is assigned to each instance of F_Dept.
*/
function F_DeptFormatTrigger return boolean is
begin
    SRW.SET_LINKTAG('dept_details_' ||
        LTRIM(TO_CHAR(:deptno)));
    return (TRUE);
end;
```

SRW.SET_MAXROW

Description This procedure sets the maximum number of records to be fetched for the specified query. This is useful when your report formats (i.e., displays) fewer records than the query (or queries) that fetch them. Thus, with `SRW.SET_MAXROW`, you can conditionally restrict data that is fetched for your report, enabling you to improve the report's performance.

Syntax

```
SRW.SET_MAXROW (query_name CHAR, maxnum PLS_INTEGER);
```

Parameters

<code>query_name</code>	Is the query whose fetched records will be limited.
<code>maxnum</code>	Is maximum number of records you want the query to fetch.

Property Palette To define this attribute using the Property Palette, set the `Maximum Rows to Fetch` property.

SRW.SET_MAXROW restrictions

- `SRW.SET_MAXROW` is only meaningful in a Before Report trigger (i.e., after the query is parsed). If `SRW.SET_MAXROW` is called after the Before Report trigger (i.e., after the queries have been executed), the `SRW.MAXROW_UNSET` packaged exception is raised.
- Because this procedure causes only the specified number of records to be fetched, the "unfetched" records of the query are not used in computations, etc.
- If you specify that 0 records should be fetched, the query will still be parsed.

SRW.SET_MAXROW examples

```
/* Suppose your report has two queries, Q_Stocks and Q_Bonds.
** Suppose also, that you have a user-created parameter, named
** WHICHDATA, that enables users to specify which data they want
** the report to display: either stocks or bonds. In the
** Before Report trigger, you could use the SRW.SET_MAXROW
** procedure to ensure that only one query's data is fetched:
*/
```

```
FUNCTION FETCHIT RETURN BOOLEAN IS
BEGIN
    if :whichdata != 1 then
        srw.set_maxrow ('Q_Stocks', 0);
    else
```

```
        srw.set_maxrow ('Q_Bonds', 0);  
    end if;  
RETURN (TRUE);  
END;
```

SRW.SET_PAGE_NAVIGATION_HTML

Description This procedure specifies the scripting for navigation controls in page-streamed HTML/HTMLCSS report output.

Syntax

```
SRW.SET_PAGE_NAVIGATION_HTML(type, 'string');
```

Parameters

type	Is SRW.FILE_ESCAPE or SRW.TEXT_ESCAPE. It indicates whether the string parameter is a filename or the HTML script text to be inserted.
string	Is a filename or the HTML scripting code to be inserted, depending on what was specified for the type parameter. If left blank, the default Report Builder JavaScript is used to define navigation controls on each page of report output.

Usage Notes

If you choose to provide your own script for navigation controls, you must include two variables in your script:

- &TotalPages (total number of pages)
- &file_name (the name of the output destination file; e.g., myreport).

Report Builder assigns values to these variables during formatting.

The height of the page navigation control frame is hard coded to 70 points without a scroll bar. The width of the frame is variable. If you want to use icons and images in this frame, keep these dimensions in mind.

Property Palette To define this attribute using the Property Palette, set the Page Navigation Control Type and Page Navigation Control Value properties.

SRW.SET_PDF_ACTION

Description This procedure specifies a command line that will be executed on the local machine when the object is clicked in the PDF viewer.

Note: An object that is associated with an action cannot also be the source of a Web link.

Syntax

```
SRW.SET_PDF_ACTION('action');
```

Parameters

action	Is any valid command line on the local machine (e.g., c:\orawin\bin\rwrun60 userid=scott/tiger report=example.rdf or /usr/local/bin/phone smith).
--------	--

Property Palette To define this attribute using the Property Palette, set the Application Command Line (PDF) property.

SRW.SET_PDF_ACTION restrictions

- SRW.SET_PDF_ACTION is only available for PDF output.
- SRW.SET_PDF_ACTION should only be set in the following triggers:

Format

SRW.SET_PDF_ACTION example

```
/* The format trigger below runs a  
** report named example.rdf when the  
** button object U_Report is clicked  
** in the PDF viewer.  
*/  
  
function U_ReportFormatTrigger return boolean is  
begin  
  SRW.SET_PDF_ACTION ||  
  ('c:\orawin\bin\rwrun60' ||  
  'userid=scott/tiger ' ||  
  'report=example.rdf');  
end;
```

SRW.SET_PRINTER_TRAY

Description This procedure specifies the name of a valid printer tray and can be used to switch to different printer trays as your report formats.

Syntax

```
SRW.SET_PRINTER_TRAY('tray');
```

Parameters

tray	Is the name of a valid printer tray and can be used to switch to different printer trays as your report formats.
------	--

Usage Notes

- If you use SRW.SET_PRINTER_TRAY in the Format Trigger of an object, the printer tray will be changed on the page where the object is printed. The printer tray you specify remains in effect until you change it in a subsequent trigger.

SRW.SET_PRINTER_TRAY restrictions

- SRW.SET_PRINTER_TRAY should only be set in the following triggers:

After Report trigger

Before Report

Between Pages

Format

If you use SRW.SET_PRINTER_TRAY in the Before or After Form triggers, you will get an error for incorrect context.

SRW.SET_PRINTER_TRAY example

```
/* The example below sets the printer
** tray in a Between Pages trigger.
*/
function BetweenPages return boolean is
begin
  SRW.SET_PRINTER_TRAY('letterhead');
  return (TRUE);
end;
```

SRW.SET_SPACING

Description This procedure specifies the global text leading for a CHAR, DATE, or NUMBER field, or boilerplate text. This attribute is ignored in character mode.

Syntax

```
SRW.SET_SPACING(spacing);
```

Parameters

spacing Is one of the following:

```
SRW.SINGLE_SPACING  
SRW.DOUBLE_SPACING  
SRW.ONEHLF_SPACING
```

SRW.SET_TEXT_COLOR

Description This procedure specifies the global text color of the CHAR, DATE, or NUMBER field. This attribute is ignored in character mode.

Syntax

```
SRW.SET_TEXT_COLOR('color');
```

Parameters

color Is a valid color from the color palette.

Property Palette (Templates only) To define this attribute in templates using the Property Palette, set the Text Color property.

User Interface To set the text color in the user interface, refer to Changing colors .

SRW.INTEGER_ERROR

Description This exception stops the report execution and raises the following error message:

```
REP-1417: Invalid integer argument passed to <SRW.MESSAGE or  
SRW.SET_MAXROW>.
```

Syntax:

```
SRW.INTEGER_ERROR;
```

Usage Notes Report Builder raises this exception when SRW.MESSAGE or SRW.SET_MAXROW is called with a non-integer as a result of an internal error. (If an internal error is not present, PL/SQL will catch the type mismatch via the PL/SQL exception VALUE_ERROR.)

SRW.INTEGER_ERROR example

```
/* Suppose you want your own error message raised,  
** instead of the default error message.  
** You could handle this exception in the following way:  
*/  
  
EXCEPTION  
when SRW.INTEGER_ERROR then  
    srw.message(4000, 'Contact the Application  
        Development group regarding SRW.INTEGER_ERROR.');
```

```
raise srw.program_abort;
```

SRW.MAXROW_INERR

Description This exception stops the report execution and raises the following error message:

```
REP-1424: Internal error while executing SRW.SET_MAXROW.
```

If it is raised, you should contact Oracle Customer Support.

Syntax

```
SRW.MAXROW_INERR;
```

Usage Notes Report Builder raises this exception when it detects an internal error while executing the SRW.SET_MAXROW packaged procedure.

SRW.MAXROW_INERR restrictions

- When you handle SRW.MAXROW_INERR, you should always raise SRW.PROGRAM_ABORT to end the report's execution (because the report has an internal problem).

SRW.MAXROW_INERR example

```
/* Suppose you want your own error message raised,  
** instead of the default error message.  
** You could handle this exception in the following way:  
*/
```

```
EXCEPTION  
when SRW.MAXROW_INERR then  
    srw.message(1000, 'Contact Oracle''s customer  
        support: SRW.MAXROW_INERR');  
    raise srw.program_abort;
```

SRW.MAXROW_UNSET

Description This exception temporarily stops the report execution and raises the error message below. After the message is raised and you accept it, the report execution will continue.

```
REP-1423: Cannot set maximum rows of the query after it started
executing.
```

Syntax

```
SRW.MAXROW_UNSET;
```

Usage Notes Report Builder raises this exception when the SRW.SET_MAXROW packaged procedure is called after the query's records have already been fetched.

SRW.MAXROW_UNSET example

```
/* Suppose you want your own error message raised,
** instead of the default error message.
** You could handle this exception in the following way:
*/
```

```
EXCEPTION
when SRW.MAXROW_UNSET then
    srw.message(1000, 'Data was fetched before
        SRW.SET_MAXROW was called.');
```

SRW.MESSAGE

Description This procedure displays a message with the message number and text that you specify. The message is displayed in the format below. After the message is raised and you accept it, the report execution will continue.

```
MSG-msg_number: msg_text.
```

Syntax

```
SRW.MESSAGE (msg_number NUMBER, msg_text CHAR);
```

Parameters

<code>msg_number</code>	Is a number from one to ten digits, to be displayed on the message line. Numbers less than five digits will be padded with zeros out to five digits. For example, if you specify 123, it will be displayed as SRW-00123.
<code>msg_text</code>	Is at most 190 minus the <code>msg_number</code> alphanumeric characters to be displayed on the message line.

SRW.MESSAGE restrictions

- You cannot trap nor change Report Builder error messages.
- SRW.MESSAGE does not terminate the report execution; if you want to terminate a report after raising a message, use SRW.PROGRAM_ABORT.
- Any extra spaces in the message string will be displayed in the message; extra spaces are not removed by Report Builder.

SRW.MESSAGE examples

```
/* Suppose you have a user exit named MYEXIT to which you want to
** pass the values of the SAL column. Suppose, also, that you want
** to raise your own error if the user exit is not found (e.g., because
** it is not linked, compiled, etc.). To do these things, you could
** write the following PL/SQL in the Format Trigger of the F_SAL field:
*/
/* This trigger will raise your message as follows:
** MSG-1000: User exit MYEXIT failed. Call Karen Smith x3455.
*/
```

```
FUNCTION FOO RETURN BOOLEAN IS
```

```
BEGIN
    srw.reference(:SAL);
    srw.user_exit('myexit sal');
EXCEPTION
    when srw.unknown_user_exit then
        srw.message(1000, 'User exit MYEXIT failed.
Call Karen Smith x3455. ');
        raise srw.program_abort;
RETURN (TRUE);
END;
```

SRW.NULL_ARGUMENTS

Description This exception stops the report execution and raises the following error message:

```
REP-1418: Passed null arguments to <SRW.DO_SQL or SRW.MESSAGE or
SRW.RUN_REPORT or SRW.SET_MAXROW or SRW.USER_EXIT>.
```

Syntax

```
SRW.NULL_ARGUMENTS ;
```

Usage Notes Report Builder raises this exception when one of its packaged functions or procedures is called with a missing argument. This exception is useful when you pass argument values to a packaged function or procedure, and you want to ensure that the values passed are not NULL (e.g., when a formula column calls SRW.USER_EXIT, and the user exit string is passed from a PL/SQL library).

The following could raise this exception:

- SRW.DO_SQL
- SRW.MESSAGE
- SRW.RUN_REPORT
- SRW.SET_MAXROW
- SRW.USER_EXIT

SRW.NULL_ARGUMENTS example

```
/* Suppose you want your own error message raised,
** instead of the default error message.
** You could handle this exception in the following way:
*/
```

```
EXCEPTION
when SRW.NULL_ARGUMENTS then
    srw.message(4000, 'Contact Application Development
    regarding SRW.NULL_ARGUMENTS. ');
    raise srw.program_abort;
```

SRW.PROGRAM_ABORT

Description This exception stops the report execution and raises the following error message:

```
REP-1419: PL/SQL program aborted.
```

SRW.PROGRAM_ABORT stops report execution when you raise it.

Syntax

```
SRW.PROGRAM_ABORT;
```

Usage Notes You must raise the exception from within your PL/SQL.

SRW.PROGRAM_ABORT examples

```
/* Suppose you want to put a border around the salary if it is greater
than 0.
** Suppose, also, that if the report fetches a salary less than 0, you
want to
** raise a customized error message (i.e., "FOUND A NEGATIVE SALARY. .
."),
** then terminate the report execution. To do so, you could write the
** following format trigger for F_SAL.
*/
```

```
FUNCTION foo return boolean is
BEGIN
if :sal >= 0 then
    srw.attr.mask          := SRW.BORDERWIDTH_ATTR;
    srw.attr.borderwidth  := 1;
    srw.set_attr (0, srw.attr);
else
    srw.message(100, 'FOUND A NEGATIVE SALARY.
CHECK THE EMP TABLE.');
```

```
    raise srw.program_abort;
end if;
RETURN (TRUE);
END;
```

SRW.REFERENCE

Description This procedure causes Report Builder to add the referenced object to the PL/SQL construct's dependency list. This causes Report Builder to determine the object's value just before firing the PL/SQL construct. This is useful when you want to ensure that a column value passed to a user exit is the most recently computed or fetched value.

Syntax

```
SRW.REFERENCE (:object CHAR|DATE|NUMBER);
```

Parameters

object	Is the Report Builder parameter or column whose value needs to be ascertained before the construct fires.
--------	---

SRW.REFERENCE restrictions

- The colon is required before the object name.
- SRW.REFERENCE is unnecessary when the object is already referenced in the current PL/SQL construct.

SRW.REFERENCE example

```
/* Suppose you want to pass the temperature and pressure values
** to a user exit called SUPERHEAT. Suppose, also, that if the
** temperature is too low, you want to raise a customized error message.
** To do so, you could write the following formula:
*/
```

```
FUNCTION EXIT RETURN BOOLEAN IS
BEGIN
  if :temp > 0 then
    srw.reference (:temp); -- unnecessary reference
    srw.reference (:pressure);
    srw.user_exit('superheat temp pressure');
  else srw.message(1000, 'Temperature is below
    normal. Is machine off?');
    raise srw.program_abort;
  end if;
RETURN(TRUE);
END;
```

SRW.RUN_REPORT

Description This procedure invokes RWRUN60 with the string that you specify. This procedure is useful for:

- running drill-down reports (i.e., calling a report from a button's action trigger)
- sending parts of a report to different recipients (e.g., to send a report via e-mail to each manager with just his or her group's data)
- sending parts of a report to different printers (e.g., to send each manager's report to his or her printer)
- running multiple reports from a single "driver" report

SRW.RUN_REPORT executes the specified RWRUN60 command.

Syntax

```
SRW.RUN_REPORT (command_line CHAR);
```

Parameters

command_line Is a valid RWRUN60 command.

SRW.RUN_REPORT restrictions

- If you want parameter values that are entered on the Runtime Parameter Form to be passed in the RWRUN60 string, you must call SRW.RUN_REPORT after the before form trigger.
- The string that is specified for or passed to this procedure must follow the syntax and case-sensitivity rules for your operating system.
- No userid should be specified for the SRW.RUN_REPORT procedure. The userid is inherited by the "calling" report.
- If the parent report that invokes SRW.RUN_REPORT is run in batch, then DESTYPE can only be File, Printer, Sysout, or Mail. Otherwise, DESTYPE can be File, Printer, or Mail.
- If SRW.RUN_REPORT is used in the PL/SQL for a button, the Runtime Parameter Form will not appear by default when the button is selected. If you want the Runtime Parameter Form to appear, you must specify PARAMFORM=YES in the call to SRW.RUN_REPORT.
- If you do not specify a path, Report Builder will use its file path search order to

find the report.

SRW.RUN_REPORT example

```
/* Suppose you have the following two reports:
** MGR_RUN, which queries manager names, and invokes a second report
named MAIL_IT
** MAIL_IT, which queries employee names for the manager that MGR_RUN
passes it,
** and sends the report output to the manager via e-mail.
** The description of MGR_RUN could be as follows:
** Query:
    SELECT ENAME, EMPNO FROM EMP WHERE JOB='MANAGER'
** Group Filter:
*/

FUNCTION FOO RETURN BOOLEAN IS
BEGIN
    srw.run_report('report=MAIL_IT
        desname='||:ename ||' desformat=dfmt batch=yes
        mgr_no='|| TO_CHAR(:empno) ');
    RETURN (TRUE);
EXCEPTION
    when srw.run_report_failure then
        srw.message(30, 'Error mailing reports. ');
        raise srw.program_abort;
END;

/* This PL/SQL invokes MAIL_IT, specifies that MAIL_IT's output
** should be sent to the manager via Oracle Mail, and passes the
** manager number, so that the MAIL_IT report can query only the
** manager's employees.
** Note: EMPNO's values must be converted to characters
** (TO_CHAR in the PL/SQL above), because SRW.RUN_REPORT
** requires a character string.
** Layout: None is needed, because this report only fetches data,
** then passes it to a second report.
** The description of MAIL_IT could be as follows:
** Query:
    SELECT DEPTNO, ENAME, SAL FROM EMP WHERE MGR=:MGR_NO
** Layout: Master/Detail
*/

/* Suppose that you have three reports that you almost always run
together.
** The reports are named SALARY, COMMISS, and TAXES. To run these
reports
** with one RWRUN60 command, you create a driver report named PAYROLL.
** The description of PAYROLL could be as follows:
** Query:
    SELECT DEPTNO FROM DEPT
```

```

** Before Report Trigger:
*/

FUNCTION FOO RETURN BOOLEAN IS
BEGIN
    srw.run_report('batch=yes report=SALARY
        destype=file desformat=dflt desname=salary.lis');
    srw.run_report('batch=yes report=COMMISS
        destype=file desformat=dflt desname=comiss.lis');
    srw.run_report('batch=yes report=TAXES
        destype=file desformat=dflt desname=comiss.lis');
    RETURN (TRUE);
END;

/* Layout: Tabular
** When you run PAYROLL from the designer or RWRUN60, the other three
** reports will all be run. (Note that, in this case, the query and
** the layout for Payroll could be anything. They are only used here
** in order to make it possible to run PAYROLL.)
*/

```

SRW.RUN_REPORT_BATCHNO

Description This exception stops the report execution and raises the following error message:

```
REP-1429: SRW.RUN_REPORT cannot be invoked with batch=no.
```

Syntax

```
SRW.RUN_REPORT_BATCHNO;
```

Usage Notes Report Builder raises this exception when the SRW.RUN_REPORT procedure is called with a command line string containing BATCH=NO.

SRW.RUN_REPORT_BATCHNO example

```
/* Suppose you want your own error message raised,  
** instead of the default error message (above).  
** You could handle this exception in the following way:  
*/  
  
EXCEPTION  
when SRW.RUN_REPORT_BATCHNO then  
    srw.message(4000, 'Contact Application Dev. Services  
    regarding SRW.RUN_REPORT_BATCHNO.');
```

```
raise srw.program_abort;
```

SRW.RUN_REPORT_FAILURE

Description If the SRW.RUN_REPORT packaged procedure fails, this exception stops the report execution and raises the following error message:

```
REP-1428: Error while running SRW.RUN_REPORT.
```

Syntax

```
SRW.RUN_REPORT_FAILURE;
```

Usage Notes Report Builder raises this exception when the SRW.RUN_REPORT packaged procedure fails.

SRW.RUN_REPORT_FAILURE example

```
/* Suppose you want your own error message raised, instead of the
default
** error message (above). The following PL/SQL code raises a customized
** error message when SRW.RUN_REPORT_FAILURE is raised.
*/
```

```
EXCEPTION
  WHEN srw.run_report_failure then
    srw.message(30, 'Error mailing reports.');
```

SRW.SET_ATTR

New Feature: It is now more convenient to set attributes using the Set Attributes Procedures. SRW.SET_ATTR is still supported for compatibility, but it is highly recommended that you use the simplified procedures.

Description SRW.SET_ATTR applies attribute settings such as font, size, or color to layout objects. You specify formatting attributes with SRW.SET_ATTR in three steps:

- 1 Specify the attributes you wish to set (e.g., the border) by setting SRW.ATTR.MASK.
- 2 Specify a value for each of the attributes that you specified in the previous step (e.g., width of 1 character).
- 3 Apply the mask to the object by calling the SRW.SET_ATTR procedure.

Note: In most cases, you can define Web links (HTML or PDF) and HTML attributes in an object's Property Palette. If you require more complex implementation, such as conditional settings, you must use SRW.SET_ATTR:

- To specify Web links, use the ACTION, BOOKMARK, HYPERLINK, and LINKTAG attributes.
- To define HTML document and page headers and footers, use the BEFPAGE_ESCAPE, AFTPAGE_ESCAPE, BEFREPORT_ESCAPE, AFTREPORT_ESCAPE attributes.
- To define HTML Parameter Form headers and footers, use the BEFFORM_ESCAPE and AFTFORM_ESCAPE.

Syntax

```
SRW.ATTR.MASK := SRW.attr_ATTR [+ SRW.attr_ATTR. . .];  
SRW.ATTR.attr := value; [SRW.ATTR.attr := value;. . .]  
SRW.SET_ATTR (object_id, SRW.ATTR);
```

Parameters

attr	Is the name of the attribute that you are setting (e.g., BORDERWIDTH).
object_id	Is 0 or SRW.REPORT_ID. When setting the value of BEFREPORT_ESCAPE, AFTREPORT_ESCAPE, BEFPAGE_ESCAPE, and AFTPAGE_ESCAPE it is SRW.REPORT_ID. In all other cases, it is 0.
value	Is a valid value for the attribute (e.g., 1 would be a valid value for BORDERWIDTH).

SRW.SET_ATTR restrictions

- If you specify an attribute (e.g., SRW.FILLPATT_ATTR), you must specify a value for it. If you do not specify a value for a specified attribute, you will get unpredictable results for that attribute.
- If you specify a value that is not provided by the Report Builder package, it must be a string (i.e., enclosed by single quotes). For example, a value for SRW.ATTR.FORMATMASK must be enclosed by single quotes, because you may specify any value. A value for SRW.ATTR.HJUST must not be enclosed by single quotes, because you must specify one of the values that Report Builder provides.
- Attributes for text can only be applied to fields via SRW.SET_ATTR. They cannot be applied to boilerplate text. To change the text attributes of boilerplate text, use the Format menu or the tool palette in the Layout editor.

SRW.SET_ATTR character mode example

```
/* Here's an example of using all of the formatting
** attributes valid for character-mode reports.
** Notice that there are two values applied to the
** horizontal justification attribute (HJUST).
** Note: For the printer codes &1 and &2 to be
** meaningful, the printer definition file for
** this report must define what the printer should
** do for &1 and &2.
*/
```

```
function F_SALFormatTrigger return boolean is
begin
  IF :SAL > 2000 THEN
    SRW.ATTR.MASK      := SRW.BEFCODE_ATTR      +
                        SRW.AFTCODE_ATTR      +
                        SRW.TEXT_ATTR         +
                        SRW.FILLPATT_ATTR      +
                        SRW.BORDERWIDTH_ATTR   +
                        SRW.FORMATMASK_ATTR    +
                        SRW.HJUST_ATTR;

    SRW.ATTR.BEFCODE   := '&1';
    SRW.ATTR.AFTCODE   := '&2';
    SRW.ATTR.TEXT      := SRW.BOLD_TEXTA;
    SRW.ATTR.FILLPATT  := 'TRANSPARENT';
    SRW.ATTR.BORDERWIDTH := 1;
    SRW.ATTR.FORMATMASK := 'DD-MON-YY';
    SRW.ATTR.HJUST     := SRW.CENTER_HJUST +
                        SRW.FLUSH_HJUST;
  END IF;
```



```

    SRW.SET_ATTR(0, SRW.ATTR);
    RETURN(TRUE);
end;

```

SRW.SET_ATTR format mask example

```

/* If the salary is 2000, the following code
** segment sets the format mask to -99990.
*/
function F_SALFormatTrigger return boolean is
begin
    if :sal = 2000 then
        srw.attr.mask      := SRW.FORMATMASK_ATTR;
        srw.attr.formatmask := '-99990';
        srw.set_attr (0, srw.attr);
    end if;
    RETURN (TRUE);
end;

```

SRW.SET_ATTR set field color example

```

/* If the salary is 2000, this function sets the
** following attributes: border width to 1, foreground
** border color to "red," background border color
** to "blue," border pattern to "checker," foreground
** fill color to "yellow," background fill color to
** "green," and the fill pattern to "crisscross."
**
** Note: When you run the report with this PL/SQL
** construct in character mode, only the border width
** (of 1) and fill pattern (of solid) will be set;
** all other settings will be ignored. When you run
** the report with this PL/SQL construct in bitmap,
** all of the attributes will be set. (Be aware that
** the border background color cannot be set on some
** GUIs.)
*/
function F_SALFormatTrigger return boolean is
begin
    if :sal = 2000 then
        srw.attr.mask      := SRW.BORDERWIDTH_ATTR +
            SRW.FBCOLOR_ATTR      +
            SRW.BBCOLOR_ATTR      +
            SRW.BORDPATT_ATTR      +
            SRW.FFCOLOR_ATTR      +
            SRW.BFCOLOR_ATTR      +
            SRW.FILLPATT_ATTR;
        srw.attr.borderwidth := 1;
        srw.attr.fbcolor     := 'red';
        srw.attr.bbcolor     := 'blue';
        srw.attr.bordpatt    := 'checker';
    end if;
end;

```

```

        srw.attr.ffcolor      := 'yellow';
        srw.attr.bfcolor     := 'green';
        srw.attr.fillpatt    := 'crisscross';
        srw.set_attr (0, srw.attr);
    end if;
    RETURN(TRUE);
end;
```

SRW.SET_ATTR set field text attributes example

```

/* If the salary is 2000, this function
** sets the following text attributes: text
** to bold, font to TIMES, size to 18 points,
** style to underline, weight to bold, text color
** to "blue," and the justification to center.
**
** Note: When you run the report with this PL/SQL
** construct in character mode, only the text will
** be set to bold (SRW.TEXT_ATTR); all other
** attributes will be ignored. When you run the
** report with this PL/SQL construct in bitmap,
** all of the attributes--except SRW.TEXT_ATTR--will
** be set.
*/
```

```

function F_SALFormatTrigger return boolean is
begin
    if :sal = 2000 then
        srw.attr.mask      := SRW.TEXT_ATTR      +
                               SRW.FACE_ATTR    +
                               SRW.SZ_ATTR      +
                               SRW.WEIGHT_ATTR  +
                               SRW.STYLE_ATTR   +
                               SRW.GCOLOR_ATTR  +
                               SRW.HJUST_ATTR;

        srw.attr.text      := SRW.BOLD_TEXTA;
        srw.attr.face      := 'times';
        srw.attr.sz        := 18;
        srw.attr.weight    := SRW.BOLD_WEIGHT;
        srw.attr.style     := SRW.UNDERLINE_STYLE;
        srw.attr.gcolor    := 'blue';
        srw.attr.hjust     := SRW.CENTER_HJUST;

        srw.set_attr (0, srw.attr);
    end if;
    RETURN (TRUE);
end;
```

SRW.SET_ATTR vertical spacing example

```

/* If the salary is 2000, this function sets
** the vertical spacing between lines to a custom
```

```
** size of 200 VGS units. (One inch is equivalent
** to 8192 VGS units.)
**/
```

```
function F_SALFormatTrigger return boolean is
begin
  if :sal = 2000 then
    srw.attr.mask      := SRW.GSPACING_ATTR;
    srw.attr.gspacing  := SRW.CUSTOM_SPACING;
    srw.attr.custom    := 200;
    srw.set_attr (0, srw.attr);
  end if;
  RETURN (TRUE);
end;
```

SRW.SET_ATTR example (switching printer trays)

```
/* The example below sets the printer tray in a Between Pages trigger.
**/
function BetweenPages return boolean is
begin
  srw.attr.mask      := SRW.PRINTER_INTRAY_ATTR;
  srw.attr.printer_intray := 'letterhead';
  srw.set_attr(SRW.REPORT_ID, srw.attr);
return (TRUE);
end;
```

SRW.TRACE_ADD_OPTION

Description SRW.TRACE_ADD_OPTION enables you to specify a new trace option after you have begun logging trace information with SRW.TRACE_START. The following example specifies three tracing options for the mask and then invoking the mask in a call to SRW.TRACE_ADD_OPTION:

```
SRW.TRACEOPTS.MASK := SRW.TRACE_ERR +  
                    SRW.TRACE_BRK +  
                    SRW.TRACE_PRF;  
SRW.TRACE_ADD_OPTION(SRW.TRACEOPTS);
```

Syntax

```
SRW.TRACE_ADD_OPTION (SRW.TRACEOPTS);
```

Parameters

SRW.TRACEOPT	Applies the trace options mask previously
S	defined by SRW.TRACEOPTS.MASK. Note
	that SRW.TRACE_APPEND and
	SRW.TRACE_REPLACE are not trace
	options and are not available for use with
	SRW.TRACE_ADD_OPTION.

SRW.TRACE_END

Description SRW.TRACE_END ends the logging of trace information to a file during report execution. To start the logging, use SRW.TRACE_START.

Syntax:

```
SRW.TRACE_END ( );
```

SRW.TRACE_REM_OPTION

Description SRW.TRACE_REM_OPTION removes trace options.

The following example specifies three tracing options for the mask and then invoking the mask in a call to SRW.TRACE_REM_OPTION:

```
SRW.TRACEOPTS.MASK := SRW.TRACE_ERR +  
                    SRW.TRACE_BRK +  
                    SRW.TRACE_PRF;  
SRW.TRACE_REM_OPTION(SRW.TRACEOPTS);
```

Syntax

```
SRW.TRACE_REM_OPTION (SRW.TRACEOPTS);
```

Parameters

SRW.TRACEOPTS	Removes the trace options in the mask previously defined by SRW.TRACEOPTS.MASK. Note that SRWTRACE_APPEND and SRWTRACE_REPLACE are not trace options and are not available for use with SRW.TRACE_REM_OPTION.
---------------	---

SRW.TRACE_START

Description SRW.TRACE_START enables you to start logging tracing information to a file during report execution. To stop the logging, use SRW.TRACE_END. You can also specify what information you want included in the log file. This information can be useful in debugging a report or finding performance bottlenecks. You specify which tracing options should be applied by defining a mask. To set tracing options in PL/SQL, you do the following:

- 1 Specify the options you want turned on.
- 2 Apply the options to the object by using SRW.TRACEOPTS when you call the SRW.TRACE_START procedure.

The following example specifies three tracing options and then invoking them in a call to SRW.TRACE_START.

```
SRW.TRACEOPTS.MASK := SRW.TRACE_ERR +
                    SRW.TRACE_BRK +
                    SRW.TRACE_PRF;
SRW.TRACE_START(test.dat, SRW.TRACE_APPEND, SRW.TRACEOPTS);
```

Syntax

```
SRW.TRACEOPTS.MASK := SRW.TRACE_opts [+ SRW.TRACE_opts . . .];
SRW.TRACE_START (filename, {SRW.TRACE_REPLACE|SRW.TRACE_APPEND},
SRW.TRACEOPTS);
```

Parameters

filename	Is the name of the file in which Report Builder stores logging information.
SRW.TRACE_APPEND	Adds the new information to the end of the file.
SRW.TRACE_REPLACE	Adds the new information to the end of the file.
SRW.TRACE_OPTS	Applies the trace options mask previously defined by SRW.TRACEOPTS.MASK.
opts	Is a valid tracing option.

SRW.TRACEOPTS.MASK

The options that you can apply to a trace mask are:

Mask Option	Description
SRW.TRACE_ ALL	Includes all possible trace information in the log file.
SRW.TRACE_ APP	Includes trace information on all the report objects in the log file.
SRW.TRACE_ BRK	Lists breakpoints in the log file.
SRW.TRACE_ DST	Lists distribution lists in the log file. You can use this information to determine which section was sent to which destination.
SRW.TRACE_ ERR	Includes error messages and warnings in the log file.
SRW.TRACE_ PLS	Includes trace information on all the PL/SQL objects in the log file.
SRW.TRACE_ PRF	Includes performance statistics in the log file.
SRW.TRACE_ SQL	Includes trace information on all the SQL in the log file.

SRW.TRACE_START restrictions

- Logging information can only be generated when running a *.rdf* file. You cannot specify logging when running a *.rep* file.
- Only one log session is allowed at one time. SRW.TRACE_START automatically ends any other log sessions.

SRW.TRUNCATED_VALUE

Description This exception will stop the report execution and raise one of two error messages. If the error resulted from a PL/SQL assignment, this message will be displayed:

```
REP-1435: Value of column <parameter/column name> was truncated.
```

If the error resulted from a user exit IAF PUT, this message will be displayed:

```
REP-1416: User exit <name> IAF PUT: value of column <parameter/column name> was truncated.
```

If the SRW.TRUNCATED_VALUE exception is handled, the value will be truncated and remain assigned to the parameter or column.

Syntax

```
SRW.TRUNCATED_VALUE ;
```

Usage Notes Reports raises this exception when a user exit or PL/SQL construct attempts to assign a value to a parameter or column which is larger than the object's maximum width.

SRW.TRUNCATED_VALUE example

```
/* Suppose you want your own error message raised, instead of the
** default error message (above). You could handle this exception
** in the following way:
*/
```

```
EXCEPTION
when SRW.TRUNCATED_VALUE then
    srw.message(2000, 'Contact Dan Brown:
    SRW.TRUNCATED_VALUE.');
```

SRW.UNKNOWN_QUERY

Description This exception will stop the report execution and raise the following error message:

```
REP-1427: Nonexistent query name referenced in SRW.SET_MAXROW.
```

Syntax

```
SRW.UNKNOWN_QUERY;
```

Usage Notes Reports raises this exception when the SRW.SET_MAXROW packaged procedure is called with an unknown query.

SRW.UNKNOWN_QUERY example

```
/* Suppose you want your own error message raised,  
** instead of the default error message (above).  
** You could handle this exception in the following way:  
*/  
  
EXCEPTION  
when SRW.UNKNOWN_QUERY then  
    srw.message(4000, 'Contact Per Jensen:  
    SRW.UNKNOWN_QUERY.');
```

```
    raise srw.program_abort;
```

SRW.UNKNOWN_USER_EXIT

Description If your user exit is unknown (e.g., because it is not linked), this exception stops the report execution and raises the following error message:

```
REP-1415: Unknown user exit.
```

Syntax

```
SRW.UNKNOWN_USER_EXIT;
```

Usage Notes Reports raises this exception when the user exit specified for the SRW.USER_EXIT or SRW.USER_EXIT20 packaged procedure cannot be located (e.g., because the user exit was not linked).

SRW.UNKNOWN_USER_EXIT example

Suppose you want your own error message raised, instead of the default error message (above). The following PL/SQL will raise a customized error message for SRW.UNKNOWN_USER_EXIT.

```
EXCEPTION
  WHEN srw.unknown_user_exit then
    srw.message(200, 'PASS USER EXIT WAS UNKNOWN.
      CHECK IF IT'S LINKED.');
```

SRW.USER_EXIT

Description This procedure calls the user exit named in `user_exit_string`. It is useful when you want to pass control to a 3GL program during a report's execution.

Syntax

```
SRW.USER_EXIT (user_exit_string CHAR);
```

Parameters

`user_exit_string` Is the name of the user exit you want to call and any columns or parameters that you want to pass to the user exit program.

SRW.USER_EXIT restrictions

- User exits are not portable. If your report must be portable, and you need to add conditional logic to it, use PL/SQL.
- If the user exit string passes a column or parameter to the user exit program, `SRW.REFERENCE` must be called before this procedure.

SRW.USER_EXIT example

```
/* Suppose you have a user exit named STORE to which you want
** to pass salary values from Report Builder. To do so, you
** could write the following formula. For more information on
** how to call user exits, see Calling a user exit.
*/
```

```
FUNCTION FOO RETURN BOOLEAN IS
BEGIN
  IF :SAL >= 0 THEN
    SRW.REFERENCE(:SAL);
    SRW.USER_EXIT('STORE SAL');
  ELSE
    SRW.MESSAGE(100, 'FOUND A NEGATIVE SALARY. CHECK THE
EMP TABLE.');
```

```
  END IF;
  EXCEPTION
    WHEN SRW.UNKNOWN_USER_EXIT THEN
      SRW.MESSAGE(200, 'STORE USER EXIT WAS UNKNOWN.
CHECK IF IT'S LINKED.');
```

```
    WHEN SRW.USER_EXIT_FAILURE THEN
      SRW.MESSAGE(200, 'STORE USER EXIT FAILED.
CHECK ITS CODE.');
```

```
RETURN(TRUE);  
END;
```

SRW.USER_EXIT20

Description This procedure is the same as SRW.USER_EXIT, except that it calls the user exit with five arguments instead of just two. This enables you to share user exits with other Oracle products, such as Oracle Forms Developer, which pass five arguments instead of two.

SRW.USER_EXIT passes the following when calling a user exit:

- the character string that you specify
- a pointer to the length of the character string

SRW.USER_EXIT20 passes the following when calling a user exit:

- the character string that you specify
- a pointer to the length of the character string
- an error string (This argument is not used by Report Builder and will always pass a value of ''.)
- a pointer to the length of the error string (This argument is not used by Report Builder and will always pass a value of 0.)
- an in query argument (This argument is not used by Report Builder and will always pass a value of 0.)

Syntax

```
SRW.USER_EXIT20 (user_exit_string CHAR);
```

Parameters

`user_exit_string` Is the name of the user exit you want to call and any columns or parameters that you want to pass to the user exit program.

SRW.USER_EXIT_FAILURE

Description This exception is raised when the user exit you called has failed. When called, it stops the report execution and raises the following error message:

```
REP-1416: User exit <exit_name> failed.
```

Syntax

```
SRW.USER_EXIT_FAILURE;
```

Usage Notes Reports raises this exception when the SRWERB buffer is non-empty.

SRW.USER_EXIT_FAILURE example

```
/* Suppose you want your own error message raised, instead of the
** default error message (above). The following PL/SQL code will
** raise a customized error message for SRW.USER_EXIT_FAILURE .
*/
```

```
EXCEPTION
  when SRW.USER_EXIT_FAILURE then
    SRW.MESSAGE(200, 'PASS user exit failed.
    Check its code.');
```


Triggers

Which report trigger to use

As a general rule, any processing that will affect the data retrieved by the report should be performed in the Before Parameter Form or After Parameter Form triggers. (These are the two report triggers that fire before anything is parsed or fetched.) Any processing that will not affect the data retrieved by the report can be performed in the other triggers.

Report Builder has five global report triggers. You cannot create new global report triggers. The trigger names indicate at what point the trigger fires:

Before Report	Fires before the report is executed but after queries are parsed.
After Report	Fires after you exit the Previewer, or after report output is sent to a specified destination, such as a file, a printer, or an Oracle Office userid. This trigger can be used to clean up any initial processing that was done, such as deleting tables. Note, however, that this trigger always fires, whether or not your report completed successfully.
Between Pages	Fires before each page of the report is formatted, except the very first page. This trigger can be used for customized page formatting. In the Previewer, this trigger only fires the first time that you go to a page. If you subsequently return to the page, the trigger does not fire again.
Before Parameter Form	Fires before the Runtime Parameter Form is displayed. From this trigger, you can access and change the values of parameters, PL/SQL global variables, and report-level columns. If the Runtime Parameter Form is suppressed, this trigger still fires. Consequently, you can use this trigger for validation of command line parameters.

After Parameter Form Fires after the Runtime Parameter Form is displayed. From this trigger, you can access parameters and check their values. This trigger can also be used to change parameter values or, if an error occurs, return to the Runtime Parameter Form. Columns from the data model are not accessible from this trigger. If the Runtime Parameter Form is suppressed, the After Parameter Form trigger still fires. Consequently, you can use this trigger for validation of command line parameters or other data.

Report trigger order of execution

The order of events when a report is executed is as follows:

Before Parameter Form trigger is fired.

- 1 Runtime Parameter Form appears (if not suppressed).
- 2 After Parameter Form trigger is fired (unless the user cancels from the Runtime Parameter Form).
- 3 Report is "compiled."
- 4 Queries are parsed.
- 5 Before Report trigger is fired.
- 6 SET TRANSACTION READONLY is executed (if specified via the READONLY argument or setting).
- 7 The report is executed and the Between Pages trigger fires for each page except the last one. (Note that data can be fetched at any time while the report is being formatted.) COMMITs can occur during this time due to any of the following-- user exit with DDL, SRW.DO_SQL with DDL, or if ONFAILURE=COMMIT, and the report fails.
- 8 COMMIT is executed (if READONLY is specified) to end the transaction.
- 9 After Report trigger is fired.
- 10 COMMIT/ROLLBACK/NOACTION is executed based on what was specified via the ONSUCCESS argument or setting.

Cautions

- In steps 4 through 9, avoid DDL statements that would modify the tables on which the report is based. Step 3 takes a snapshot of the tables and the snapshot must remain valid throughout the execution of the report. In steps 7 through 9, avoid DML statements that would modify the contents of the tables on which the report is based. Queries may be executed in any order, which makes DML statements unreliable (unless performed on tables not used by the report).
- If you specify READONLY, you should avoid DDL altogether. When you execute a DDL statement (e.g., via SRW.DO_SQL or a user exit), a COMMIT is automatically issued. If you are using READONLY, this will prematurely end the transaction begun by SET TRANSACTION READONLY.

Report trigger restrictions

- If you are sending your report output to the Runtime Previewer or Live Previewer, you should note that some or all of the report triggers may be fired before you see the report output. For example, suppose that you use `SRW.MESSAGE` to issue a message in the Between Pages trigger when a condition is met. If there are forward references in the report (e.g., a total number of pages displayed before the last page), Report Builder may have to format ahead to compute the forward references. Hence, even though you have not yet seen a page, it may already have been formatted and the trigger fired.
- In report triggers, you can use the values of report-level columns and parameters. For example, you might need to use the value of a parameter called `COUNT1` in a condition (e.g., `IF :COUNT1 = 10`). Note, though, that you cannot reference any page-dependent columns (i.e., a column with a Reset At of Page) or columns that rely on page-dependent columns.
- In the Before and After Parameter Form, and Before and After Report triggers, you can set the values of parameters (e.g., give them a value in an assignment statement, `:COUNT1 = 15`). In the Before and After Report triggers, you can also set the values of report-level, placeholder columns.
- In the Between Pages trigger, you cannot set the values of any data model objects. Note also that the use of PL/SQL global variables to indirectly set the values of columns or parameters is not recommended. If you do this, you may get unpredictable results.
- If you run a report from Report Builder Runtime (i.e., not the command line or `SRW.RUN_REPORT`), you should commit database changes you make in the Before Parameter Form, After Parameter Form, and Validation triggers before the report runs. When running in this way, these triggers will share the parent process' database connection. When the report is actually executed, however, it will establish its own database connection.
- A lexical reference cannot be used to create additional bind variables after the After Parameter Form trigger fires. For example, suppose you have a query like the following (note that the WHERE clause is replaced by a lexical reference):

```
SELECT ENAME, SAL FROM EMP
&where_clause
```

If the value of the `WHERE_CLAUSE` parameter contains a reference to a bind variable, you must specify the value in the After Parameter Form trigger or earlier. You would get an error if you supplied the following value for the parameter in the Before Report trigger. If you supplied this same value in the After Parameter Form trigger, the report would run.

```
WHERE SAL = :new_bind
```

Group filter

Description A group filter is a PL/SQL function that determines which records to include in a group, if the Filter Type property is PL/SQL. The function must return a boolean value (TRUE or FALSE). Depending on whether the function returns TRUE or FALSE, the current record is included or excluded from the report. You can access group filters from the Object Navigator, the Property Palette (the PL/SQL Filter property), or the PL/SQL Editor.

Definition Level group

On Failure Excludes the current record from the group.

Group filter example

```
function filter_comm return boolean is
begin
    if :comm IS NOT NULL then
        if :comm < 100 then
            return (FALSE);
        else
            return (TRUE);
        end if;
    else
        return (FALSE); -- for rows with NULL commissions
    end if;
end;
```

Group filter restrictions

- Group filters cannot be added to groups if Filter Type is First or Last.
- Group filters cannot be added to cross-product groups.
- In a group filters, you can read the values of Report Builder columns and parameters of the correct frequency (look at the rule below), but you cannot directly set their values. For example, you can use the value of a parameter called COUNT1 in a condition (e.g., IF :COUNT1 = 10), but you cannot directly set its value in an assignment statement (e.g., :COUNT1 = 10). Note also that the use of PL/SQL global variables to indirectly set the values of columns or parameters is not supported. If you do this, you may get unpredictable results. You also cannot reference any page-dependent columns (i.e., Reset At of Page) or columns that rely on page-dependent columns in a group filter.
- The function that you enter for a group filter can only depend upon the following columns:
 - a database column owned by the group's query or a query above it in the data model hierarchy

- computed columns (formulas or summaries) that depend on unrelated queries (i.e., computed columns that do not depend upon columns in the group, the group's ancestors, or the group's descendants)

Formula

Description Formulas are PL/SQL functions that populate formula or placeholder columns. You can access the PL/SQL for formulas from the Object Navigator, the PL/SQL Editor, or the Property Palette (i.e., the PL/SQL Formula property).

A column of datatype Number can only have a formula that returns a value of datatype NUMBER. A column of Datatype Date can only have a formula that returns a value of datatype DATE. A column of Datatype Character can only have a formula that returns a value of datatype CHARACTER, VARCHAR, or VARCHAR2.

Definition Level column

On Failure No value is returned for the column.

Formula for adding values example

```
function salcomm return NUMBER is
begin
    return(:sal + :comm);
end;
```

Formula with condition example

```
function calcomm return NUMBER is
    temp number;
begin
    if :comm IS NOT NULL then
        temp := :sal + :comm;
    else
        temp := :sal;
    end if;
    return (temp);
end;
```

Formula restrictions

- In a formula, you can read and assign values to the column itself, placeholder columns, and parameters of the correct frequency (look at the rule below). For example, you can use the value of a column called COMP in a condition (e.g., IF :COMP = 10) and you can directly set its value in an assignment statement (e.g., :COMP := 15).
- A formula can only make reference to columns that are in the same or a higher group in the group hierarchy. For example, a formula for a report-level column can only reference other report-level columns.
- Formulas are calculated such that any column referenced in the formula will be calculated first. To do so, Report Builder builds a dependency list, to guarantee proper ordering of calculations. Note that circular dependencies, in which a column references another column which in turn references the first column, either

directly or indirectly, are not allowed.

- A user exit may only change the value of a placeholder column.
- When using `SRW.DO_SQL`, we recommend that you do not read database values that are updated or inserted in the same report. There is no guarantee of the exact time Report Builder will fetch records from the database for formatting the output. Report Builder does internal "data look-ahead" to optimize performance. Thus, a particular record might already have been accessed before an update is issued to the same record. Report Builder builds internal dependency lists which guarantee that events, such as invocation of user exits, calculation of summaries, etc., happen in the correct order. However, Report Builder cannot guarantee these events will be synchronized with its internal data access or with the formatting of data.

Validation trigger

Description Validation triggers are PL/SQL functions that are executed when parameter values are specified on the command line and when you accept the Runtime Parameter Form. (Notice that this means each validation trigger may fire twice when you execute the report.) Validation triggers are also used to validate the Initial Value property of the parameter. The function must return a boolean value (TRUE or FALSE). Depending on whether the function returns TRUE or FALSE, the user is returned to the Runtime Parameter Form. You can access validation triggers from the Object Navigator, the PL/SQL Editor, or the Property Palette (Validation Trigger property).

Definition Level parameter

On Failure The user is returned to the parameter value in the Runtime Parameter Form where they can either change it or cancel the Runtime Parameter Form.

Validation trigger example

```
/* This function prevents the runtime user from
** sending report output anywhere except a printer.
** The user will be returned to the Runtime Parameter
** Form unless PRINTER is specified as the destination
** type (DESTYPE).
*/
function DESTYPEValidTrigger return boolean is
begin
  IF UPPER(:DESTYPE) = 'PRINTER' THEN
    RETURN(TRUE);
  ELSE
    RETURN(FALSE);
  END IF;
end;
```

Validation trigger restrictions

- The PL/SQL in a validation trigger can be a maximum of 32K characters. The upward limit may vary between operating systems.
- In a validation trigger, you can read and assign values to Report Builder parameters. You cannot read or assign values to columns. For example, you can use the value of a parameter called COUNT1 in a condition (e.g., IF :COUNT1 = 10) and you can directly set its value in an assignment statement (e.g., :COUNT1 = 15). In some cases, though, the Validation Trigger may fire more than once for the same parameter. As a result, it is usually best to assign parameter values in the After Parameter Form trigger. Note also that the use of PL/SQL global variables to indirectly set the values of columns is not supported. If you do this, you may get unpredictable results.

- You should not use DDL in Validation triggers.
- For reports that are spawned processes of other reports (e.g., run with BACKGROUND=YES), you should commit database changes you make in the Before Parameter Form, After Parameter Form, and Validation triggers before the report runs. Spawned processes use their parent process' database connection for the Before Parameter Form, After Parameter Form, and Validation triggers. When the spawned process runs the report, though, it establishes its own database connection. Any database changes not committed by the time the child report runs will therefore be lost.

Format trigger

Description Format triggers are PL/SQL functions executed before the object is formatted. The trigger can be used to dynamically change the formatting attributes of the object. The function must return a Boolean value (TRUE or FALSE). Depending on whether the function returns TRUE or FALSE, the current instance of the object is included or excluded from the report output. You can access format triggers from the Object Navigator, the Property Palette, or the PL/SQL Editor.

Definition Level layout object

On Failure Excludes the current instance of the object from the output.

Usage Notes

- Format triggers do not affect the data retrieved by the report. For example, if a format trigger returns FALSE for a field, the data for the field is retrieved even though the field does not appear in the output.
- If a format trigger suppresses report output on the last page of the report, the last page will still be formatted and sent to the appropriate output and the page will be included in the total number of pages.

Format trigger example (highlighting a value)

```
/* Suppose that you are building a banking report and
** would like it to indicate if a customer is overdrawn.
** To do so, you give the repeating frame around the
** customer information a format trigger that causes
** it to have a border only if a customer's account
** balance is less than 0 (or the required minimum
**balance).
*/
function my_formtrig return BOOLEAN is
begin
if :bal < 0 then
    srw.attr.mask := SRW.BORDERWIDTH_ATTR;
    srw.attr.borderwidth := 1;
    srw.set_attr (0, srw.attr);
end if;
return (true);
end;
```

Format trigger example (suppressing labels)

```
/* Suppose that you are building a master/detail report
** and, if no detail records are retrieved for a master
** record, you do not want the boilerplate labels to
** appear. To do this, you first create a summary
** column called MYCOUNT with a Function of Count in
** the source group of the master repeating frame.
** In the format trigger for the group frame that
** surrounds the detail repeating frame and its labels,
```

```

** you enter the following:
*/
function my_formtrig return BOOLEAN is
begin
if :mycount = 0 then
return (false);
else
return (true);
end if;
end;

```

Format trigger example (suppressing values)

```

/* Suppose that you are building a salary report and
** you only want to see salaries above $2500. In the
** report summary of all salaries, however, you want all
** salaries to be included. To do this, you create a
** data model for the report without restricting the records
** retrieved. Then, you enter the following format trigger
** for the repeating frame. The report output will only
** show salaries greater than $2500, but the summary that
** calculates the sum of all salaries will include all the
** salaries retrieved from the database.
*/
function my_formtrig return BOOLEAN is
begin
if :sal > 2500 then
return (true);
else
return (false);
end if;
end;

```

Format trigger example (placing a comma between fields)

```

/* Suppose that you want to create a comma-separated
** list for the names of employees. First, you create
** a field inside a repeating frame with a Print
** Direction of Across. Next to the field, you create
** a boilerplate text object that contains a comma
** followed by a space. To ensure that the comma does
** not appear after the last name in the list, enter the
** following format trigger for the boilerplate object.
** LASTNAME is a summary column with a Source of ENAME,
** a Function of Last, and Reset At of Report.
*/
function my_formtrig return BOOLEAN is
begin
if :ename <> :lastname then
return (true);
else
return (false);
end if;
end;

```

Format trigger restrictions

Caution: The PL/SQL in a format trigger is executed each time that Report Builder attempts to format the layout object. As a result, format triggers should only contain PL/SQL program units that set formatting attributes (e.g., color and highlighting). You should not perform other actions, such as inserting data in a table, because you cannot predict when or how many times the trigger will fire. For example, if you have Page Protect set for an object, the object might not be formatted on the logical page where the trigger is fired. In addition, the trigger may be executed more than once.

- If a `f2format` trigger returns false for an object on the last page of your report and no other objects are formatted on the last page, the last page be a blank page.
- Comments inserted directly into the PL/SQL code must use the PL/SQL comment delimiters.
- In a format trigger, you can read the values of Report Builder columns and parameters of the correct frequency (look at the rule below), but you cannot directly set their values. For example, you can use the value of a parameter called `COUNT1` in a condition (e.g., `IF :COUNT1 = 10`), but you cannot directly set its value in an assignment statement (e.g., `:COUNT1 = 10`). (This restriction also applies to user exits called from the format trigger.) Note also that the use of PL/SQL global variables to indirectly set the values of columns or parameters is not supported. If you do this, you may get unpredictable results.
- You cannot reference columns or variables in the format trigger of an object that have a different frequency than the object. For example, if you create a master/detail report, the parent group's repeating frame cannot have a format trigger that relies on a value in the child group. For each parent, there may be multiple children. Therefore, at the parent record level, Report Builder cannot determine which of the child records to use. You also cannot reference any page-dependent columns (i.e., `Reset At of Page`) or columns that rely on page-dependent columns in a format trigger. The reason for this is that it would result in a circular dependency. That is, the value of a page-dependent column cannot be computed until the report is formatted, but the report cannot be formatted until the format trigger is executed.
- If a format trigger returns false and the object does not format, this can cause other objects not to print. For example, if a repeating frame does not format, any objects (fields, boilerplate, frames, or other repeating frames) it encloses would not format either.
- For repeating frames, the format trigger is executed for each instance of the repeating frame. To create a format trigger that acts upon all instances of a repeating frame at once, create a frame around the repeating frame and enter a

format trigger for the frame. If the format trigger returns FALSE for every instance of a repeating frame on a logical page, the repeating frame will occupy no space on the logical page and anchors to other objects will collapse (if specified).

- The PL/SQL in a format trigger must return consistent results for the same object. For example, say you have a frame whose format trigger returns FALSE when a certain condition is met. If the frame spans two pages, the format trigger actually fires twice (once for each page on which the frame formats). The condition in your PL/SQL must return the same result both times the format trigger fires. Otherwise, only part of the frame will be formatted (e.g., the part of the frame on the first page formats, but the part on the second page does not).
- If the format trigger on a repeating frame in a matrix report returns FALSE, an entire row or column of the matrix will not format. For example, if an instance of the across dimension repeating frame does not format, the entire column will not format in the matrix.
- If you want to conditionally change the cell of a matrix, you should put a frame around the field inside the matrix and use the format trigger for the frame.

Action trigger

Description Action triggers are PL/SQL procedures executed when a button is selected in the Runtime Previewer. The trigger can be used to dynamically call another report (drill down) or execute any other PL/SQL. You can access action triggers from the Object Navigator, the Property Palette (PL/SQL Trigger property), or the PL/SQL Editor.

Definition Level button

Usage Notes

- PL/SQL action triggers cannot be tested in the Live Previewer because the buttons are not active there. You must use the Runtime Previewer (choose **View** → **Runtime Preview** from the Live Previewer).
- You cannot use the PL/SQL Interpreter to debug action triggers because it is not available from the Runtime Previewer and buttons cannot be activated in the Live Previewer. To get around this, you can move your action trigger code to a report trigger to test it from the Live Previewer.

Action trigger example

```
/* When the button is clicked, the action trigger
** defined by this procedure displays the Parameter
** Form for a report named web4.rdf, then displays
** the resulting report output on your screen.
*/
procedure U_1ButtonAction is
begin
    srw.run_report('module=web4.rdf destype=Screen paramform=yes');
end;
```

Ref cursor query

Description A ref cursor query uses PL/SQL to fetch data for the report. In a ref cursor query, you specify a PL/SQL function that returns a cursor value from a cursor variable.

Definition Level query

On Failure No data is returned to the query .

Usage Notes

- When you make a ref cursor query the child in a data link, the link can only be a group to group link. It cannot be a column to column link.
- If you use a stored program unit to implement ref cursors, you receive the added benefits that go along with storing your program units in the Oracle database.

Package with ref cursor example

```
/* This package spec defines a ref cursor
** type that could be referenced from a
** ref cursor query function.
** If creating this spec as a stored
** procedure in a tool such as SQL*Plus,
** you would need to use the CREATE
** PACKAGE command.
*/
PACKAGE cv IS
  type comp_rec is RECORD
    (deptno number,
     ename varchar(10),
     compensation number);
  type comp_cv is REF CURSOR return comp_rec;
END;
```

Package with ref cursor and function example

```
/* This package spec and body define a ref
** cursor type as well as a function that
** uses the ref cursor to return data.
** The function could be referenced from
** the ref cursor query, which would
** greatly simplify the PL/SQL in the
** query itself. If creating this spec
** and body as a stored procedure in a
** tool such as SQL*Plus, you would need
** to use the CREATE PACKAGE and CREATE
** PACKAGE BODY commands.
*/
PACKAGE cv IS
  type comp_rec is RECORD
    (deptno number,
     ename varchar(10),
     compensation number);
```

```

    type comp_cv is REF CURSOR return comp_rec;
    function emprefc(deptno1 number) return comp_cv;
END;
PACKAGE BODY cv IS
function emprefc(deptno1 number) return comp_cv is
    temp_cv cv.comp_cv;
begin
    if deptno1 > 20 then
        open temp_cv for select deptno, ename,
            1.25*(sal+nvl(comm,0)) compensation
            from emp where deptno = deptno1;
        else
            open temp_cv for select deptno, ename,
                1.15*(sal+nvl(comm,0)) compensation
                from emp where deptno = deptno1;
            end if;
        return temp_cv;
    end;
END;

```

Ref cursor query example

```

/* This ref cursor query function would be coded
** in the query itself. It uses the cv.comp_cv
** ref cursor from the cv package to return
** data for the query.
*/
function DS_3RefCurDS return cv.comp_cv is
    temp_cv cv.comp_cv;
begin
    if :deptno > 20 then
        open temp_cv for select deptno, ename,
            1.25*(sal+nvl(comm,0)) compensation
            from emp where deptno = :deptno;
        else
            open temp_cv for select deptno, ename,
                1.15*(sal+nvl(comm,0)) compensation
                from emp where deptno = :deptno;
            end if;
        return temp_cv;
    end;
end;

```

Ref cursor query calling function example

```

/* This ref cursor query function would be coded
** in the query itself. It uses the cv.comp_cv
** ref cursor and the cv.emprefc function from
** the cv package to return data for the query.
** Because it uses the function from the cv
** package, the logic for the query resides
** mainly within the package. Query
** administration/maintenance can be
** done at the package level (e.g.,
** modifying SELECT clauses could be done
** by updating the package). You could also
** easily move the package to the database.
** Note this example assumes you have defined
** a user parameter named deptno.

```

```
*/  
function DS_3RefCurDS return cv.comp_cv is  
  temp_cv cv.comp_cv;  
begin  
  temp_cv := cv.emprefc(:deptno);  
  return temp_cv;  
end;
```

After Parameter Form trigger

Description The After Parameter Form trigger fires after the Runtime Parameter Form is displayed. From this trigger, you can access parameters and check their values. This trigger can also be used to change parameter values or, if an error occurs, return to the Runtime Parameter Form. Columns from the data model are not accessible from this trigger. If the Runtime Parameter Form is suppressed, the After Parameter Form trigger still fires. Consequently, you can use this trigger for validation of command line parameters or other data.

Definition Level report

On Failure

Returns to the Runtime Parameter Form. If the Form is suppressed, then returns to place from which you ran the report .

After Report trigger

Description The After Report trigger fires after you exit the Runtime Previewer, or after report output is sent to a specified destination, such as a file, a printer, or a mailid. This trigger can be used to clean up any initial processing that was done, such as deleting tables. Note, however, that this trigger always fires, whether or not your report completed successfully.

Definition Level report

On Failure

Does not affect formatting because the report is done. You can raise a message, though, to indicate that the report did not run correctly. For example, you could put the system time in a variable in the Before Report trigger and then compare it against the system time in the After Report trigger. If the report took longer than a certain time to run, you could raise an error.

Usage Notes

- The After-Report trigger does not fire when you are in the Live Previewer.

Before Parameter Form trigger

Description The Before Parameter Form trigger fires before the Runtime Parameter Form is displayed. From this trigger, you can access and change the values of parameters, PL/SQL global variables, and report-level columns. If the Runtime Parameter Form is suppressed, this trigger still fires. Consequently, you can use this trigger for validation of command line parameters.

Definition Level report

On Failure

Displays an error message and then returns to the place from which you ran the report .

Before Report trigger

Description The Before Report trigger fires before the report is executed but after queries are parsed and data is fetched.

Definition Level report

On Failure

Displays an error message and then returns to the place from which you ran the report .

Between Pages trigger

Description The Between Pages trigger fires before each page of the report is formatted, except the very first page. This trigger can be used for customized page formatting. In the Runtime Previewer or Live Previewer, this trigger only fires the first time that you go to a page. If you subsequently return to the page, the trigger does not fire again.

Definition Level report

On Failure

Displays an error message when you try to go to the page for which the trigger returned FALSE. The pages subsequent to the page that returned FALSE are not formatted. If the trigger returns FALSE on the last page, nothing happens because the report is done formatting. The Between Pages trigger does not fire before the first page. If the trigger returns FALSE on the first page, the first page is displayed, but, if you try to go to the second page, an error message is displayed.

Properties

Oracle8 usage notes

Column objects and Ref columns

- To reference the attribute columns or methods of a column object, you must qualify the name with a table alias or correlation variable. For example: `select e.empno, e.address.street, e.address.city from emp e`
- You can move column objects or Ref columns as a whole or move their individual attribute columns. When you move an attribute column outside of its associated column object, it is detached from the column object and is treated like any other column. Only the name is updated to indicate the association with the column object (e.g., ADDRESS_ZIP).
- Column objects cannot be the source for a layout object, but their attribute columns can be used as the source for a layout object. For example, suppose you have a column object named ADDRESS that is composed of STREET, CITY, STATE, and ZIP. ADDRESS cannot be the source for a field, but STREET, CITY, STATE or ZIP can be the source for a field.
- A column object or Ref column must reside in the lowest group in which one of its attribute columns resides. This occurs when the frequency of the column object or Ref column is the same as the attribute column with the highest frequency. If the column object or Ref column is in a group above one of its attribute columns, then its frequency will be too low and an error is raised. For example, suppose the column object ADDRESS is in a group named G_MASTER and one of its attribute columns is ZIP. If you drag ZIP into a child of G_MASTER called G_DETAIL, then ADDRESS must also be moved into G_DETAIL.
- Column objects cannot be used as bind variables (i.e., :column_name), but their attribute columns can be used as bind variables.
- You cannot break on a column object or a Ref column. As a result, a column object or Ref column cannot be the only column in a break group. You can, however, break on the attribute columns of a column object. For example, you could drag the column object and all of its attribute columns into the break group and assign a break order to as many of the attribute columns as you like.
- You cannot use column objects or Ref columns in data model links. You can only use their individual attribute columns in links. For example, suppose you have a column object named ADDRESS composed of STREET, CITY, STATE, and ZIP. You can create a link using any one of the attribute columns of ADDRESS (e.g., CITY).

Summaries, formulas, and placeholders

- The datatype of a formula or placeholder column must be one of the Oracle7 built-in datatypes (e.g., CHAR or VARCHAR). The datatypes that are new with Oracle8 are not yet supported.
- Summary columns that have a column object or Ref column as their source can only use the First, Last, and Count functions.

Varying arrays and nested tables

- Varying array and nested table types are not supported yet. If you select a column object of these types in a query, a special icon will appear next to it in the Data Model view indicating that it is not supported.

Large objects (LOBs)

- LOBs are not supported yet. If you select a LOB in a query, a special icon will appear next to it in the Data Model view indicating that it is not supported.

About the Property Palette

The Property Palette has an outline-style interface that enables you to view, locate, and set object properties. The Property Palette features:

- expandable and collapsible nodes
- in-place property editing
- search features
- multi-selection
- complex property dialogs
- the ability to invoke multiple instances of the Property Palette

When you select an object in an editor or in the navigator, the Property Palette is updated to show the properties of that object. By default, the list of properties in the palette is synchronized whenever you select a different object. You can turn synchronization On or Off for a specific palette by clicking the **Freeze/Unfreeze** tool) on the Property Palette toolbar.

When you need to compare the properties of objects, you can pin the current Property Palette and invoke additional Property Palettes as needed.

The Property Palette has the following features:

Property List

The property list is a 2-column list of property names and property values. When no objects are selected, no properties are displayed in the list. Properties are grouped under functional or object headings, such as General Info properties and Field properties. You can select properties in the list by clicking and by navigating with the Up/Down arrow keys. You can set properties in the property list pane by selecting the property and then typing or double-clicking as required.

Name

The name field displays the name of the object currently selected in the navigator or an editor. The **Name** field is not displayed when no objects are selected. When multiple objects are selected at the same time, a list of names is displayed at the top of the Property Palette.

Set Property Controls

When a property is selected in the properties list, a text field, pop-list, combo box, or **More** button is displayed beside the property, as follows:

A **text field** is displayed when the current property can be set by entering a text value. For properties that require longer text values, there is an iconic button next to the text field that invokes a modal editor.

A **poplist** is displayed when the current property is either True/False, or has a fixed set of valid values. You can cycle through the values by double-clicking repeatedly.

A **combo box** is displayed when the current property can be set by typing a value or selecting from a poplist.

A **More...** button is displayed when the current property requires more complex values. Selecting the button invokes a dialog where you enter the required information.

The Property Palette is where you set the properties of objects you create in documents.

About making multiple selections in the Property Palette

You can select multiple objects at the same time by using Shift+Click or Ctrl+Click in the navigators or the editors. When two or more objects are selected, a list of object names is displayed at the top of the Property Palette .

The Intersection/Union command on the property palette toolbar determines which properties are displayed in the property list when more than one object is selected. Toggling between Intersection and Union changes the list of properties displayed in the property palette, but does not affect the setting of any property.

Intersection The default. Only properties common to all selected objects are displayed.

Union All properties of every object selected are displayed.





Properties that are common to two or more objects in a multiple selection are listed only once, and the property setting is displayed as follows:

- If the property has the same setting for all of the selected objects that have it in common, the common setting is displayed.
- If the property is set differently for the objects that have it in common, the string ***** is displayed.

About toolbar commands in the Property Palette

The toolbar provides quick access to commands. Click once on the icon for the command you want to execute.

Inherit	Applies only to Sections, Headings, Body, and Summaries in templates. Sets the current property to its default setting. You can select multiple properties and click Inherit.
Localize	Applies only to Sections, Headings, Body, and Summaries in templates. Makes the current property local (i.e., not inherited). You can select multiple properties and click Localize.

Find field	Is a string that you want to search for in the Property Palette.
Find forward	Searches forward for the string in the Find field.
	
Find backward	Searches backward for the string in the Find field.
	
Intersection/ Union	Toggles the Property Palette between union and intersection display modes. This option determines which properties appear when more than one object is selected at a time (a multiple selection). In intersection mode (the default), only properties that the selected objects have in common are displayed. In union mode, the properties of all objects in the current selection are displayed.
	
Freeze/Unfreeze	Toggles Property Palette synchronization On and Off. When Freeze is Off (the default), the property list is updated to display the properties of objects you select in the navigators and other windows. When Freeze is On, the property list is pinned and does not get updated, allowing you to compare it to other property lists.
	

Displaying the Property Palette

To display the Property Palette for an object, do one of the following:

- Double-click the object. To invoke a second Property Palette, you must pin the current one and double-click an object icon.
- Click the object, then choose **Tools**→**Property Palette**.
- From the pop-up menu (right-click in Windows), choose **Property Palette**.
- Double-click the properties icon for the object in the Object Navigator.

Closing the Property Palette

To close the Property Palette:

- Double-click the Close box in the upper left corner.

Setting properties using the Property Palette

To set properties using the Property Palette:

- 1 Display the Property Palette for the object whose properties you want to set.
- 2 Select the property you want to set.



You can select properties by clicking or by navigating with the Up/Down arrow keys.

When a property is selected, a text field, poplist, combo box, or **More...** button is displayed next to the property.

- 1 Set the property to the desired setting .

Setting properties of multiple selections

To set the properties of multiple selections:

- 1 In the Object Navigator or the editors, select the objects whose properties you want to set. The objects can be of different types and can even be in different documents.
- 2 Choose **Tools**→**Property Palette**.
- 3 In the Property Palette, click  (the Union tool) to see all properties of all objects or leave the tool as  (the Intersection tool) to see only the properties the objects have in common.
- 4 Set the properties as desired.

Any change you make to a property setting is applied to all of the objects in the current selection to which that property applies.


Setting properties of multiple selections example

A report includes several fields, each of which displays the date. The Datatype property of each field is DATE, which causes dates to be displayed in the default ORACLE format DD-MON-YY.

To use a different date format throughout the application, you need to set the Format Mask property for each field that displays dates. Rather than setting each field's Format Mask property individually, select all of the items, then set the Format Mask property once to change the format mask for each item selected.

Comparing the properties of one object to another

To compare the properties of one object to another:

- 1 In the Object Navigators or the editors, double-click the first object so that its properties are displayed in the Property Palette.
- 2 In the Property Palette, click  to turn synchronization Off and "freeze" the Property Palette.
- 3 In the Object Navigator, click the second object, then choose **Property Palette** from the pop-up menu.

A second Property Palette is displayed. If the second window is on top of the first, drag it to move it alongside the first window.

Anchor properties

Child Edge Percent
Child Edge Type
Child Object Name
Collapse Horizontally
Collapse Vertically
Comments
Name
Parent Edge Percent
Parent Edge Type
Parent Object Name

Anchor restrictions

- An object may be anchored to only one other object.
- Matrix objects, anchors, and the margin cannot be anchored to anything (i.e., they may not be the parent or child object for an anchor).
- A repeating frame that is the vertical or horizontal repeating frame for a matrix cannot be anchored to another object, but other objects may be anchored to it (i.e., it can be the parent but not the child object for an anchor).
- Nothing can be anchored to a hidden object (an object with Visible set to *No*).
- Moving an anchor also causes the two objects it anchors together to move.
- Objects cannot be anchored together in such a way that they have a circular dependency. For example:
 - Assume that object A and object B are anchored together and object A is the parent. Object B cannot be the parent for another anchor between object A and object B. In addition, object B cannot be the parent for an anchor to object C, if object C is the parent for an anchor to object A.

See the online help for details and illustrations.
 - Assume that frame A contains field B and field B is above frame A in the Layout Model. Furthermore, frame A is anchored to field C, which is not inside of frame A, and field C is the parent. Field C cannot be the parent for an anchor to field B. Because it is inside of frame A, field B cannot be anchored to

field C, if field C is the parent.

See the online help for details and illustrations.

- To copy an anchor, you must select the anchor and the two objects it anchors together. If you select the anchor by itself, nothing will be copied to the paste buffer. If you select the anchor and one of the objects, only the object is placed in the clipboard.
- An anchor cannot be resized.
- An anchor must always be on top of the objects it anchors together (i.e., it must be one or more layers above the parent and child objects). Report Builder prevents you from moving the anchor to a layer below its parent and child objects.
- You cannot use *Align*, *Align Objects*, or *Size Objects* from the *Arrange* menu on anchors.

Child Edge Percent

Description The Child Edge Percent property is the percentage down or across (from top to bottom or left to right) the child object's edge where the anchor is located. This setting is the position of the anchor on the edge when the object is populated.

Values An integer from 0 through 100.

Applies to anchors

Child Edge Type

Description The Child Edge Type property is the edge of the child object on which the anchor is located.

Values Top, Bottom, Left, and Right

Applies to anchors

Child Object Name

Description The Child Object Name property is the name of the child object. This field is read-only.

Applies to anchors

Collapse Horizontally

Description The Collapse Horizontally property indicates whether the anchor should have zero width if the parent object does not print for some reason (e.g., because of its Print Object On, Base Printing On, or Format Trigger properties). If the parent object does not print, the child object moves horizontally into the space vacated by the parent.

Applies to anchors

Collapse Vertically

Description The Collapse Vertically property indicates whether the anchor should have zero height if the parent object does not print for some reason (e.g., because of its Print Object On, Base Printing On, or Format Trigger properties). If the parent object does not print, the child object moves vertically into the space vacated by the parent.

Applies to anchors

Centering fields horizontally example

Suppose you want to horizontally center a field within a repeating frame that has its Horizontal Elasticity set to *Variable*. You also want the field to be a fixed distance from the top edge of the repeating frame. Since you don't know how large or small the formatted repeating frame will be, you need to use anchors to center the field.

To center the field horizontally within the repeating frame do the following:

- Place the field inside the repeating frame such that the x coordinate of the center of the top of the field is the same as the x coordinate of the center of the top of the repeating frame. See the figure below.
- Draw an anchor from the center of the top of the field to the center of the top of the repeating frame. **Tip**

The field will now be centered horizontally within the repeating frame.

See the online help for details and illustrations.

Centering fields vertically example

Suppose you want to vertically center a field within a repeating frame that has its Vertical Elasticity set to *Variable*. You also want the field to be a fixed distance from the top edge of the repeating frame. Since you don't know how large or small the formatted repeating frame will be, you need to use anchors to center the field.

To center the field vertically within the repeating frame, draw an anchor from the center of the left edge of the field to the center of the left edge of the repeating frame. **Tip**

The field will now be centered vertically within the repeating frame.

See the online help for details and illustrations.

Collapse Vertically example (field)

Suppose that you have anchored Field B to Field A and Collapse Vertically is set to *Yes*. See the online help for details and illustrations.

Assume that on the first logical page of the report, Field A prints and takes up so much room that Field B cannot fit on the logical page. Therefore, Field B prints on the second

logical page, where Field A does not appear. Since Collapse Vertically is set to *Yes*, the anchor collapses in the y direction. Field B maintains its relative positioning to Field A in the x direction even though Field A does not actually appear on the logical page. See the online help for details and illustrations.

Collapse Vertically and Horizontally example (field)

Suppose that you have anchored Field B to Field A, and you want Field B to move both vertically and horizontally into the space where Field A would have printed on the logical page. Do the following:

- Draw an anchor from the top of Field B to the bottom of Field A.
- Set both Collapse Horizontally and Collapse Vertically to *Yes*.

See the online help for details and illustrations.

On the first logical page of the report, Field A prints and takes up so much room that Field B cannot fit on the logical page. Therefore, Field B prints on the second logical page, where Field A does not appear. The anchor collapses and Field B moves into the position that Field A would have appeared in had it appeared on the logical page as painted.

See the online help for details and illustrations.

When Fields A and B do print on the same logical page, the anchor does not collapse and the fields maintain their relative position to one another.

Parent Edge Percent

Description The Parent Edge Percent property is the percentage down or across (from top to bottom or left to right) the parent object's Edge where the anchor is located. This setting is the position of the anchor on the Edge when the object is populated.

Values An integer from 0 through 100.

Applies to anchors

Parent Edge Type

Description The Parent Edge Type property is the edge of the parent object on which the anchor is located.

Values Top, Bottom, Left, and Right

Applies to anchors

Parent Object Name

Description The Parent Object Name property is the name of the parent object. This field is read-only.

Applies to anchors

Boilerplate properties

Additional Hyperlink Attributes
Application Command Line (PDF)
Base Printing On
Bookmark
Comments
Contains HTML Tags
Display Name
Format Trigger
Horizontal Elasticity
Hyperlink
Hyperlink Destination
Keep With Anchoring Object
Line Stretch with Frame
Minimum Widow Lines
Name
Page Break After
Page Break Before
Page Protect
Print Object On
Printer Code After
Printer Code Before
Source File Format
Source Filename
Type
Vertical Elasticity

Boilerplate restrictions

- The maximum size of boilerplate text is 64K of text per paragraph and 64K paragraphs. This limit may vary according to your operating system.
- Boilerplate must be in front of the repeating frame that contains it.

Caution: When you resize boilerplate text, ensure the text fits in the object. If font descends (the space left for the lower parts of letters like g and q) do not fit, the text line will appear in the Report Editor, but, when the report is run, the line may not appear in the output, depending upon the Horizontal and Vertical Elasticity settings.

Contains HTML Tags

Description The Contains HTML Tags property indicates that the boilerplate or field object includes HTML tags. Report Builder fixes the object both vertically and horizontally, and formats the object as a normal text object (i.e., any field references are resolved). If you need to pass the object width and height as parameters, you can use the ObjectWidth and ObjectHeight variables, as shown in the example.

Applies to boilerplate and fields

Required/Optional optional

Default No

Contains HTML Tags restrictions

- Contains HTML Tags can only be used on text boilerplate, link file boilerplate with a source file format of text, and fields with a source datatype of character.

ObjectWidth and ObjectHeight output to HTML example

In a boilerplate text object, you can type the following Java Applet, called NervousText.class, which takes the object width and height as parameters:

```
<base href=http://cagney.uk.oracle.com/java/NervousText/>
<applet code="NervousText.class" width=&ObjectWidth
height=&<ObjectHeight>>
<param name=text value="&deptno">
</applet>
```

This is output to HTML as:

```
<base href=http://cagney.uk.oracle.com/java/NervousText/>
<applet code="NervousText.class" width=84 height=72>
<param name=text value="10">
</applet>
```

Minimum Widow Lines

Description The Minimum Widow Lines property is the minimum number of lines of the boilerplate or field text that should appear on the logical page where the object starts to print. If the number of lines specified for this property cannot fit on the logical page, then all lines of text are moved to the next page.

Note: This property applies only to the initial lines of text in the boilerplate or field object. It does not provide widow protection for individual paragraphs.

Values Any whole, positive number from 0 through 32K.

Applies to boilerplate and fields

Required/Optional optional

Default 0 (the text that can fit on the logical page should be printed before continuing on the next page)

Minimum Widow Lines restrictions

- Minimum Widow Lines can only be used on text boilerplate, link file boilerplate with a source file format of text, and fields with a source datatype of character.

Minimum Widow Lines example (set to 1 vs. 2)

With Minimum Widow Lines set to 1, you might get a situation like the one shown in the figure, where the first line of the field is printed at the bottom of a logical page and the rest is printed on the next logical page. With Minimum Widow Lines set to 2 (or more), you can avoid having one line of a field at the bottom of the logical page by itself, as shown in the second figure.

See the online help for details and illustrations.

Source File Format

Description The Source File Format property is the format of the external file to which the boilerplate is linked.

Values

Text	Is text in ASCII format
Image	Is a bitmapped image
CGM	Is a line drawing in CGM format
Oracle Drawing Format	Is a line drawing in Oracle format
Image URL	Is a URL link to an image

Note: All the other image formats are provided in case you want to open the report in Oracle Reports Developer Version 2.0. If you are planning to open/run the report in Version 2.0, select the specific image format of the boilerplate. If not, you can choose from the above choices.

Applies to external boilerplate

Required/Optional required

Source Filename

Description The Source Filename property is the name of the external file or image URL to which the boilerplate is linked. Linking to a file is useful for including the contents of files that are likely to change. The boilerplate is automatically updated to reflect the latest contents of the file when you do any of the following:

- accept the property sheet
- paste the object
- import the object (if it's in Oracle Format)
- open the report
- run the report

The contents of a file linked to a boilerplate object will appear in the Report Editor, but cannot be edited within the Report Editor.

Values Enter a valid filename or image URL not to exceed 1K in length. You can prefix a path to the filename. If a path is not prefixed to the filename, Report Builder uses its file path search order to find the file.

Applies to external boilerplate

Required/Optional optional

Default blank

Usage Notes

- To ensure the portability of your reports, you can take advantage of the Report Builder-specific environment variable called `REPORTS30_PATH`. You can use this variable to specify the default directory or directories where you want Report Builder to search for external files you use in your reports (e.g., external queries, boilerplate, and PL/SQL). This prevents the need to hardcode directory paths in your report.
- You can also include files in your report using Read from File.

Source Filename restrictions

- The maximum size of the file you specify depends upon your operating system.
- The source filename may be operating-system dependent. If you move the report to another operating system, you may need to update the filename.

- This field appears only for objects created with the Link File tool.
- If you specify a file that contains a layout object and its properties (i.e., a file that contains an exported layout object), the file is treated as an anonymous graphical object (i.e., the properties of the object will be ignored). If you want the object's properties to be used, then you should import it instead of linking the file.

Type

Description The Type property is the format (e.g., text) of the boilerplate object. This field is read-only.

Applies to boilerplate

Line Stretch with Frame

Description The Line Stretch with Frame property is the name of the frame or repeating frame to which the boilerplate line is associated. When set, this property anchors the line at both endpoints to the associated frame. Thus, the line stretches the same amount as the associated frame stretches when the report is formatted. If the line is vertical, the top endpoint is anchored to the top of the frame and the bottom endpoint is anchored to the bottom of the frame. If the line is horizontal, the left endpoint is anchored to the left side of the frame and the right endpoint is anchored to the right side of the frame.

Values Select a frame or repeating frame from the list of those currently in the report. *Null* means the property is not set, and the boilerplate line will remain fixed in size when the report is formatted.

Applies to boilerplate lines

Required/Optional optional

Default Null

Usage Notes

- Line Stretch with Frame can be set for any boilerplate line, regardless of its position in the layout.
- If you use this property to associate a line with a repeating frame, the line must be enclosed completely within the repeating frame.

Line Stretch with Frame example (grid)

Suppose you want to separate the rows and columns of a group left report with a grid. See the online help for details and illustrations.

Using anchors, boilerplate lines, and Line Stretch with Frame, you would edit the layout as follows:

- Expand the innermost group frame (GF2) and detail repeating frame (RF2) to enclose the two group fields, so that all the fields are enclosed in the detail repeating frame.
- Anchor each of the group fields to the master repeating frame (RF1).
- Create boilerplate line L1 under the fields in the detail repeating frame (RF2).
- Create boilerplate lines L2 through L5 and associate them with the surrounding group frame (GF1) using Line Stretch with Frame.

See the online help for details and illustrations.

When the report is formatted, the explicit anchors cause the group fields to appear at the same frequency as the master repeating frame. Line L1 repeats for each record in the detail repeating frame, and lines L2 through L5 stretch to cover the whole length of the outermost group frame.

Line Stretch with Frame restrictions

- Line Stretch with Frame supports only vertical and horizontal lines. If set for a diagonal line, it becomes a vertical or horizontal line when formatted.
- When this property is set, the `implicit anchoring algorithm` does not apply.

Button properties

Additional Hyperlink Attributes
Application Command Line (PDF)
Base Printing On
Bookmark
Comments
Format Trigger
Horizontal Elasticity
Hyperlink
Hyperlink Destination
Icon Name
Keep With Anchoring Object
Label Type
Multimedia Column
Multimedia Column Type
Multimedia File
Multimedia File Type
Name
Page Break After
Page Break Before
Page Protect
PL/SQL Trigger
Print Object On
Printer Code After
Printer Code Before
Text
Type
Vertical Elasticity

Button restrictions

- Buttons do not appear in printed output. They also do not appear when reports containing them are run in character mode.
- Buttons appear in the Live Previewer view with the characteristics of native buttons on your platform. Therefore, changes made in the Layout Model view that affect the appearance of the button (e.g., changing borders to dashed lines) may not appear in your output. Changing the border to a dashed line appears in the Layout Model view but not the Live Previewer view.

- The ability to assign different colors and patterns to user-defined buttons is platform-specific.
- You cannot rotate a button.
- A button is always be displayed on the uppermost layer of objects in the Live Previewer view. However, it may not truly be on the top layer of the Layout Model view; for example, if it is covered by a repeating frame in the Layout Model view, it will appear above the objects within the repeating frame in the Live Previewer view, but will not repeat with the repeating frame.

Label Type

Description The Label Type property is the type of identifier you want to appear on the button.

Values

Text	Means that you want to use a text string as the button label. Use the Text property to enter the text string for the label.
Icon	Means that you want to use an icon as the button label. Use the Icon Name property to specify the name of the file containing the icon for the label (the file must reside in the directory specified by the TK25_ICON environment variable).

Applies to buttons

Required/Optional required

Default Text

Label Type restrictions

- An icon used as a label should be in the standard icon format of your platform, with an extension of *.ico*. Icons are not portable.

Text

Description The Text property is the text label that will appear on the button if you selected Text for the Label Type. The specified text string is center-justified both horizontally and vertically on the button.

Values Any text string up to 2K in length; however, practical length is governed by the size of the button.

Applies to buttons

Required/Optional optional

Default blank

Usage Notes

- You can change the font, size, weight, and style of the label text using some of the choices on the Format menu. However, certain menu choices do not affect button text, such as Spacing and Justification .

Text restrictions

- If the text does not fit on the button, it will not wrap to a second line, but will be clipped on both the left and the right.

Icon Name

Description The Icon Name property is the name of the file containing the icon that will appear on the button if you selected Icon for the Label Type.

Values Any valid icon file in the standard icon format of your platform, with an extension of *.ico*. The file must reside in the directory specified by the TK25_ICON environment variable.

Applies to buttons

Required/Optional optional

Default blank

Type

Description The Type property is the action to be performed when the button is pressed.

Values

Multimedia File	Means that the action is contained in a file in your operating system, specified by the Multimedia File property.
Multimedia Column	Means that the action is contained in a database column, specified by the Multimedia Column property.
PL/SQL	Means that the action is governed by a PL/SQL program unit associated with the button, specified by the PL/SQL Trigger property.

Applies to buttons

Required/Optional required

Default Multimedia File

Type restrictions

- If you specify PL/SQL Trigger and your PL/SQL uses SRW.RUN_REPORT, the Runtime Parameter Form will not appear by default when the button is selected. If you want the Runtime Parameter Form to appear, you must specify PARAMFORM=YES in the call to SRW.RUN_REPORT.
- If you specify PL/SQL Trigger and you raise SRW.PROGRAM_ABORT in the button's action trigger, it will close any Previewer window that was raised by the button.

Multimedia File

Description The Multimedia File property is the name of the multimedia (sound, video, or image) file.

Values Any valid sound, video, or image file.

Applies to buttons

Required/Optional Required, if Type is Multimedia File.

Default blank

Usage Notes

- To ensure the portability of your reports, you can take advantage of the Report Builder-specific environment variable called `REPORTS30_PATH`. You can use this variable to specify the default directory or directories where you want Report Builder to search for external files you use in your reports (e.g., external queries, boilerplate, and PL/SQL). This prevents the need to hardcode directory paths in your report.

Multimedia File restrictions

- If you select Multimedia File, you must indicate the filename in the text field provided. If you do not give a path for a file, Report Builder uses its file path search order to find the file.

Multimedia File Type

Description The Multimedia File Type property is the format of the file specified by the Multimedia File property.

Values Image, Video, and Sound

Applies to buttons

Required/Optional required

Default Image

Multimedia Column

Description The Multimedia Column property is the name of a database column in the report that contains a multimedia (sound, video, or image) object.

Values Any valid database column in the report containing a multimedia object.

Applies to buttons

Required/Optional Required, if Type is Multimedia Column.

Default blank

Multimedia Column restrictions

- If you specify a Multimedia Column for the action of a button and later delete that column from your report (i.e., select an image column, specify it as the Multimedia Column for a button, and later delete that column or change its format to a non-multimedia format), your button no longer has access to data to display, and Multimedia Column becomes undefined.

If the button's property sheet is open at the time the Multimedia Column becomes undefined, you must choose another column before you can accept the property sheet. If the property sheet is closed at the time the Multimedia Column becomes undefined, you will not be alerted until you run the report and try to activate the button. An error will result.

Multimedia Column Type

Description The Multimedia Column Type property is the format of the multimedia object stored in the database column specified by the Multimedia Column property.

Values Image, Video, and Sound

Applies to buttons

Required/Optional required

Default When Multimedia Column is specified, the proper multimedia column type is automatically selected. Otherwise, the default is Image.

PL/SQL Trigger

Description The PL/SQL Trigger property is a PL/SQL procedure that is executed when a button is selected in the Live Previewer view. The trigger can be used to dynamically call another report (drill down) or execute any other PL/SQL.

Applies to buttons

Required/Optional Required, if the button Type is PL/SQL

Default blank

Chart properties

Additional Hyperlink Attributes
Application Command Line (PDF)
Base Printing On
Bookmark
Chart Filename
Chart Hyperlink
Comments
Format Trigger
Horizontal Elasticity
Hyperlink
Hyperlink Destination
Keep With Anchoring Object
Name
Page Break After
Page Break Before
Page Protect
Parameters and Columns
Print Object On
Printer Code After
Printer Code Before
Vertical Elasticity

Chart in a group report example

Suppose that you have a group report with the following data model:

See the online help.

In Graphics Builder, you have created a Graphics Builder display (empsal.ogd) that shows a bar chart of salaries for each employee and you would like to include it in your report. Using the Chart tool in the Layout Model view, you first draw a chart object inside the repeating frame for the master repeating frame (R_Dept). This means that the bar chart will appear once for each department.

You then go to the Property Palette for the chart object and set the Display Name property to empsal.ogd. Display Name can optionally contain a path, but in this case, no path was specified because we want Report Builder to use the path specified for the REPORTS60_PATH environment variable.

You then set properties in the Property Palette as follows:

Source	G_Emp
Display Query	Q_OG_Emp
Report	ENAME/OG_NAME

Column/Display SAL/OG_SALARY
Column

Source is the group (G_Emp) that is summarized by the display.

Display Query is the name of a query in the display. Q_OG_Emp contains the following SELECT statement:

```
SELECT ENAME OG_NAME, SAL OG_SALARY FROM EMP
```

Since you specified Q_OG_Emp for Display Query, this query will retrieve the values of ENAME and SAL from Report Builder instead of the database.

Report Column and **Display Column** map the report's columns to the display's columns. You select ENAME and SAL from the list under Report Column. You must enter the names of the display columns that correspond to ENAME and SAL in Display Column. (Display Column cannot be blank if Report Column is specified.)

Chart restrictions

- A Graphics Builder display/chart that passes data can be placed anywhere in the Layout Model view unless its source group is a dimension of a matrix. If a Graphics Builder display/chart source group is a dimension group (i.e., a group within a cross-product group), then it must appear inside the repeating frame(s) of its source group's ancestors (if any).
- A Graphics Builder display/chart that passes parameters must be placed inside of a repeating frame whose source group is the group (or a descendant of the group) that contains the columns or parameters it is passing. If a column being passed is a summary owned by a cross-product group, then the Graphics Builder display/chart must be placed inside of the set of repeating frames whose sources are the groups in the summary column's Product Order.

Chart Filename

Description The Chart Filename property is the name of the file in which the Graphics Builder display/chart is located.

Values Enter any valid filename (not to exceed 512 bytes in length) of a Graphics Builder display/chart (the *.ogd* extension is optional).

Applies to chart objects

Required/Optional required

Default blank

Usage Notes

- You can optionally prefix a path to the filename. If a path is not prefixed to the filename, Report Builder uses its file path search order to find the file.
- To ensure the portability of your reports, you can take advantage of the Report Builder-specific environment variable called `REPORTS60_PATH`. You can use this variable to specify the default directory or directories where you want Report Builder to search for external files you use in your reports (e.g., external queries, boilerplate, and PL/SQL). This prevents the need to hardcode directory paths in your report.

Chart Filename restrictions

- You can only reference displays/charts stored in files; those stored in the database cannot be referenced in Chart Filename.

Chart Hyperlink

Description The Chart Hyperlink property is a hyperlink that you can specify to be different for each chart section. Note

Values

Any object in your report output, another report, or any valid hyperlink destination

Applies to

chart objects

Required/Optional optional

Default blank

Usage notes

- You must specify the Hyperlink Destination property if you want to set a Chart Hyperlink property for a chart that specifies another object as its destination.
- You cannot use the Additional hyperlink attributes property to apply additional HTML attributes to a link defined by the Chart Hyperlink property .
- You can set another report as the Chart Hyperlink property by running the second report from your report output. See d2kcfg.htm in your ORACLE_HOME for more information.
- You cannot set this property using PL/SQL since the format trigger is executed only once for the chart. Instead, you can specify &<column_name> as the Chart Hyperlink property .

Chart Parameters and Columns properties

Chart Column

Chart Parameter

Chart Query

Report Column (for Chart Column)

Report Column (for Chart Parameter)

Report Group

Chart Parameter

Description The Chart Parameter property is a parameter in the Graphics Builder display/chart that corresponds to the column or parameter specified in Report Column.

Note: One report column or parameter can correspond to multiple chart parameters, but each chart parameter can only correspond to one report column or parameter.

Values Enter the name (not to exceed 1K in length) of a valid parameter in your Graphics Builder display/chart. You can also enter a list of valid parameter names in your Graphics Builder display/chart, separating each one with a comma (e.g. parm1 , parm2 , parm3) .

Applies to chart objects

Required/Optional optional

Default Report Builder parameter name

Report Column (for Chart Parameter)

Description The Report Column (for Chart Parameter) property is a list of the columns and parameters in your report that could be used to supply the value of a parameter in a Graphics Builder display/chart. Selecting a column or parameter means that there is a corresponding parameter in the display/chart and that the display/chart parameter will get its value from the selected report column or parameter. If the parameter in the display/chart has a different name than the column or parameter in the report, you must specify its name in the corresponding Chart Parameter field.

Values Select one or more valid columns or parameters in your report.

Applies to chart objects

Required/Optional optional

Default deselected

Report Column (for Chart Parameter) restrictions

- A page-level summary column (i.e., a summary with a Reset At or Compute At of *Page*) cannot be passed to a Graphics Builder display/chart. Similarly, any summary or formula column that depends upon a page-level summary cannot be passed to a display/chart.

Chart Query

Description The Chart Query property is the name of the query in the Graphics Builder display/chart that should get its records from Report Builder. If you specify a query name, that query will retrieve its records through Report Builder instead of the database. This enables your display/chart to reflect any filtering or summarizing that your report performs on the data. Chart Query ensures that the data used in the display/chart is the same as the data in the report.

Note: If you specify a query name here, you must also specify which report columns or parameters should have their values used by the display/chart query.

Values Enter any valid query name (not to exceed 512 bytes in length) of the Graphics Builder display/chart specified in Chart Filename.

Applies to chart objects

Required/Optional optional

Report Group

Description The Report Group property is the group in the report that is summarized by the Graphics Builder display/chart.

Values Select any group name from the list of values.

Applies to chart objects

Required/Optional Required, if value entered for Chart Query.

Report Group restrictions

- You cannot specify a cross-product group for Report Group.
- Any columns that you pass to the Graphics Builder display/chart via Report Column (for Chart Column) must be in the report group or its ancestor.
- If you select a Report Group but not a Chart Query, the Report Group is deselected when you accept the property sheet.

Chart Column

Description The Chart Column property is a column or list of columns in the Graphics Builder display/chart that corresponds to the column or parameter specified in Report Column.

Note: One report column or parameter can correspond to multiple chart columns, but each chart column can only correspond to one report column or parameter.

Values Enter the name (not to exceed 1K in length) of a valid column in your Graphics Builder display/chart. You can also enter a list of valid column names in your Graphics Builder display/chart, separating each one with a comma (e.g. `col1,col2,col3`).

Applies to chart objects

Required/Optional optional

Default Report Builder column name

Report Column (for Chart Column)

Description The Report Column (for Chart Column) property is a list of the columns and parameters in your report that could be used by the query specified in Chart Query. Selecting a column or parameter means that there is a corresponding column(s) in the Graphics Builder display/chart and that the display/chart column(s) will get its value from the selected report column or parameter. If the column(s) in the display/chart has a different name than the column or parameter in the report, you must specify its name in the corresponding Chart Column field.

Values Select one or more valid columns or parameters in your report.

Applies to chart objects

Required/Optional Required, if value entered for Chart Query.

Report Column (for Chart Column) restrictions

- Any columns that you pass to the Graphics Builder display/chart via Report Column (for Chart Column) must be in the Report Group or its ancestor.
- A page-level summary column (i.e., a summary with a Reset At or Compute At of *Page*) cannot be passed to a Graphics Builder display/chart. Similarly, any summary or formula column that depends upon a page-level summary cannot be passed to a Graphics Builder display/chart.

Common column properties

Break Order
Comment
Column Type
Datatype
File Format
Name
Read from File
Set Break Order
Value If Null
Width

Break Order

Description The Break Order property is the order in which to display the column's values. This property applies only to columns that identify distinct values of user-created groups (i.e., break groups). The order of column values in a default group is determined by the ORDER BY clause of the query. For column values in user-created groups, however, you must use Break Order to specify how to order the break column's values.

Values

Ascending

Descending

Applies to columns

Required/Optional Required, if Set Break Order is set to *Yes*.

Default *Ascending*

Break Order example (descending)

Suppose that you create a report with the following query:

```
SELECT DEPTNO, JOB, ENAME, SAL FROM EMP
ORDER BY SAL
```

You then create two groups, G_DEPT and G_JOB. G_DEPT contains the DEPTNO column and G_JOB contains the JOB column. If you specify a Break Order of *Descending* for the DEPTNO column and *Ascending* for the JOB column, your output would appear similar to that below (assuming you use a Tabular style):

Deptno	Job	Ename	Sal
30	CLERK	JAMES	950.00
	MANAGER	BLAKE	2850.00
	SALESMAN	WARD	1250.00
		MARTIN	1250.00
		TURNER	1500.00
		ALLEN	1600.00
20	ANALYST	SCOTT	3000.00
		FORD	3000.00
	CLERK	SMITH	800.00
		ADAMS	1100.00
	MANAGER	JONES	2975.00
10	CLERK	MILLER	1300.00
	MANAGER	CLARK	2450.00
	PRESIDENT	KING	5000.00

Break Order example (ORDER BY)

Suppose that you created a Group Left report with DEPTNO and ENAME. In the query, you have ORDER BY DEPTNO. You have also specified a Break Order of *Descending* for the DEPTNO column. The output would appear similar to that below:

```
Deptno      Ename
-----
      30    JAMES
          WARD
          MARTIN
          TURNER
          ALLEN
          BLAKE
      20    SMITH
          ADAMS
          JONES
          SCOTT
          FORD
      10    MILLER
          CLARK
          KING
```

Notice that the Break Order property takes precedence over the ORDER BY clause (i.e., the department numbers are in descending order).

Break Order restrictions

- Break Order has no effect on columns that belong to the lowest group of a particular query. Break Order only effects columns in groups that are above the lowest child group of a query.
- Every group above the lowest child group of a query must have at least one column with Break Order set.
- Break Order cannot be specified for columns of Datatype *Long* or *Long Raw*.
- Break Order only affects the ordering of the column on which you are breaking. It does not affect the ordering of the columns within the break group. For example, suppose that you break on DEPTNO and for each department you list ENAME. If the Break Order is *Ascending*, then department numbers are printed in ascending order, but the names within each department print in the order specified by the query. (You would need to use an ORDER BY clause in your SELECT statement to change their ordering.)
- A summary column cannot be a break column and cannot have Break Order set.
- A formula column that depends upon a summary column cannot be a break column and cannot have Break Order set.

Column Type

Description The Column Type property is the category into which the column falls.

Values

Database - Object

Database - Ref

Database - Scalar

Database - Unkown indicates that the column is of an unsupported type.

Formula

Placeholder

Summary

Comment

Description The Comment property is a text field in which you can document the column.

Values Enter any text not to exceed 64K.

Applies to columns

Required/Optional optional

Default blank

Database Column Name

Description The Database Column Name property is the full name of the column used by the Oracle server to identify the column. For example, suppose that you have a column object named ADDRESS that contains a column object named COUNTRY that contains a column named CITY. The Database Column Name for CITY would be ADDRESS.COUNTRY.CITY.

Applies to columns

Required/Optional required

Datatype

Description The Datatype property is the type of the column's contents.

Values

Character

Date

Long

Long Raw

Number

Raw

Varchar

Varchar2

Trusted Oracle datatypes (MLSLABEL and ROWLABEL) are supported by Report Builder. When running on an operating system with Trusted Oracle, Report Builder treats these columns as Trusted Oracle datatypes. This means that the columns can be properly sorted. When running on an operating system without Trusted Oracle (e.g., Windows), Report Builder treats these columns as character datatypes.

Applies to columns

Required/Optional required

Default For database columns, the default Datatype is derived from the database. For summary columns with a Function of Count, the default Datatype is *Number*. For other summary columns, the default is derived from Source. For formula and placeholder columns, the default Datatype is *Number*.

Datatype restrictions

- You can only edit the Datatype of a formula or placeholder column. For formula and placeholder columns, you can set the Datatype to *Character*, *Date*, and *Number*.
- A column of Datatype *Long*, *Long Raw*, or *Raw* is treated as text unless you specify otherwise in the File Format field.
- If a column has an alias and you change your query, causing the datatype of the column to change, the change in datatype will be automatically reflected in Datatype when you accept the query.

File Format

Description The File Format property is the format of the object being retrieved from the database or identified by the filename stored in the column.

Value

Text	Is text in ASCII format
Image	Is a bitmapped image
CGM	Is a line drawing in CGM format
Oracle Format (Drawing)	Is a line drawing in Oracle format
Sound	Is a sound object in longraw format
Video	Is a video object in longraw format
OLE2	Is a linked object to be embedded in your report
Image URL	Is a URL link to an image

Applies to columns

Name

Description The Name property is the name Report Builder uses to identify the current column. Enter any valid column name not to exceed 30 bytes.

Applies to columns

Required/Optional required

Default For a database column, the name as referenced in the SELECT list of the SELECT statement. For a summary column, CS_ *n*, where *n* is a number that is unique among summary names for the report. For a formula column, CF_ *n*, where *n* is a number that is unique among formula names for the report. For a placeholder column, CP_ *n*, where *n* is a number that is unique among placeholder names for the report.

Name restrictions

- Name is only editable for summary, formula, and placeholder columns.
- Report Builder uses the column name, alias, or text of the expression from the query as the default name for a database column.

Read from File

Description The Read from File property indicates that the column contains the names of files or a URL for an image. When you set Read from File to *Yes*, you must specify the format of the files in the File Format property.

Read from File is useful if you have image, graphic, or text files that are referenced by a column. If you set Read from File to *Yes*, the contents of the files will be imported and used as the values of the column. If Read from File is set to *No*, the column will get its values from the database. As a result, if a column contains filenames and Read from File is set to *No*, the filenames will be printed instead of the file contents.

You can use Read from File with database, summary, formula, or placeholder columns (assuming that the column's values are filenames). You can also include files in your report as boilerplate.

Applies to columns

Required/Optional optional

Usage Notes

- You can prefix paths to the filenames in the column. If a path is not prefixed to the filename, Report Builder uses its file path search order to find the file.
- To ensure the portability of your reports, you can take advantage of the Report Builder-specific environment variable called REPORTS60_PATH. You can use this variable to specify the default directory or directories where you want Report Builder to search for external files you use in your reports (e.g., external queries, boilerplate, and PL/SQL). This prevents the need to hardcode directory paths in your report.

Read from File example (image)

Suppose that for each of your employees you have a *.bmp* file that contains a picture of the employee. In your employee table, you have a column named PICTURE that contains the name of the *.bmp* file with the employee's picture.

To generate a report that displays the employees' pictures, you select the PICTURE column in your query. On the Column property sheet for the PICTURE column, you set Read from File to *Yes* and specify *Image* as the File Format. Report Builder will display the files pointed to by the PICTURE column wherever you have referenced the PICTURE column in your layout.

Note: If Read from File is set to *No*, the filenames will be printed instead of the pictures.

Read from File example (formula column)

Suppose that you want to create an employee listing with some text from a file next to each employee's information. If the employee's salary is greater than \$2000 you want text from a file called *salary1.txt* to appear next to the employee. If the employee's salary is less than \$2000 you want the text from a file called *salary2.txt* to appear next to the employee.

You create a formula column with a Datatype of *Character* and Read from File set to *Yes* (with a File Format of *Text*). In the Formula field for the column, you enter the following PL/SQL:

```
if :sal > 2000 then
  return('/home/jsmith/txt/salary1.txt');
else
  return('/home/jsmith/txt/salary2.txt');
end if;
```

Read from File restrictions

- The size of each file may be at most 4 gigabytes. (On some operating systems the file may be at most 64K.)
- Read from File may only be set to *Yes* if the Datatype of the column is character-compatible (i.e., *Character*, *VARCHAR*, or *VARCHAR2*).
- If you set Read from File to *Yes* on a summary or formula column, the summary or formula is calculated before the file is read. Similarly, for placeholder columns with Read from File set to *Yes*, the column's value is calculated first.

Value If Null

Description The Value if Null property is a value to be substituted for any null values of the column. For example, if you enter X in this field, then an X will be displayed for null values fetched for the column. If left blank, no substitution will be done for null values.

Values Enter any valid value that conforms to the column's Datatype. Value if Null cannot exceed 1K in length.

Applies to columns

Required/Optional optional

Default blank

Usage Notes

- Entering a string for Value if Null on a break column causes Report Builder to prefetch all of the rows in the break column's group. This could degrade performance when you run the report. To avoid prefetching, use the NVL function in your query rather than entering a string for Value if Null.

Value If Null restrictions

- For columns with a character datatype (e.g., VARCHAR), if the width of the string you enter for Value if Null exceeds the Width of the column, the string is truncated.
- If a column is referenced via a lexical reference in a query, that column's Value if Null is used to validate the query.
- The value entered for Value If Null must be the same datatype as the column. For example, if the column's Datatype is *Number*, you cannot specify "XX" in Value If Null, you must specify a number.
- If you enter a date value in Value If Null, it is validated against Report Builder' internal mask for dates (i.e., DD-MON-YY). (Number and character values are not validated.)

Width

Description The Width property is the maximum number of characters that the values of the column can occupy. This Width refers to the width that Report Builder uses internally for the column. It effects such things as the calculation of summaries and formulas. The size used to display the column's data, however, is determined by the size of the associated field in the Layout Model view and its Vertical and Horizontal Elasticity properties.

Values

Enter any number from 1 through 64K.

Enter two numbers in the form a,b where a indicates the number of digits to the left of the decimal point and b indicates the number of digits to the right of the decimal point. (This form can only be used for defining the width of numbers.)

Applies to columns

Required/Optional required

Default For database columns, the default Width is derived from the database. For summary and formula columns, the default for Width is 10, 0. If a column has a Type of *Summary*, Report Builder updates the Width based upon what you enter as the column's Source and Function.

Usage Notes

- If the Column Type is Database Scalar and the Datatype is Character, Date, Number, or Raw, Width is uneditable. However, this does not mean you cannot control the width of the column's values in the report output. You can change the width of the values in the output by modifying the field's width in the Default Layout dialog box or the Field property sheet.
- If the Column Type is Database Scalar or Placeholder and the Datatype is Long or Long Raw, you can enter a value in Width. Note that if you do not specify a width for a Long or Long Raw column, Report Builder allocates the maximum width for the column. You can also enter a Width for a column of Type Formula.
- If the Column Type is Database Ref, Width is uneditable.

Width restrictions

Caution: If the value of a column of Type Database and Datatype Long or Long Raw exceeds the Width of the column, the value is truncated.

- If the value of a column of Type Placeholder or Formula exceeds the Width of the column, an exception is raised.
- If a column has an alias and you change your query, causing the width of the column to change, the change in width will be automatically reflected in Width

when you accept the query.

Set Break Order

Description The Set Break Order property is whether to set the order in which to display the column's values, using the Break Order property.

Applies to columns

Required/Optional optional

Default *Yes*

Common Layout Object properties

Additional Hyperlink Attributes
Application Command Line (PDF)
Base Printing On
Bookmark
Comments
Conditional Formatting
Display Name
Format Trigger
Horizontal Elasticity
Hyperlink
Hyperlink Destination
Keep With Anchoring Object
Name
Page Break After
Page Break Before
Page Protect
Print Object On
Printer Code Before
Printer Code After
Vertical Elasticity

Common Layout Object restrictions

- You must ensure that properties on the same or different objects do not conflict with each other. Because of the power of the layout properties, it is possible to specify properties such that they conflict with other properties. You should avoid the following combinations of layout properties on the same object. If you try to use these combinations on the same object, you will get an error:
 - Base Printing On of *Enclosing* and Print Object On of *Last Page*
 - Base Printing On of *Enclosing* and Print Object On of *All But Last Page*
 - Print Object On of *All Pages* and Page Break Before set to *Yes*
 - Print Object On of *All But First Page* and Page Break Before set to *Yes*
 - Print Object On of *All But Last Page* and Page Break Before set to *Yes*
 - Keep With Anchoring Object and Page Break Before set to *Yes* (for an object that is located after its anchoring object) or Page Break After set to *Yes* (for an object that is located before its anchoring object)

- Similarly, combinations of properties on different objects can cause unpredictable results or an infinite report. For example, suppose you have two boilerplate objects anchored together inside of a frame. The parent object has a Print Object On of *All Pages* and a Base Printing On of *Enclosing*. The child object has a Print Object On of *First Page*, a Base Printing On of *Anchoring*, and Page Break Before set to *Yes*. The figure below illustrates this layout. When you run the report, the child object does not appear in the output. Because it has Page Break Before set to *Yes*, the child can never satisfy its Print Object On (*First Page*) and it's Base Printing On (*Anchoring*). Note, however, that the object underneath the child object (B_Plate3) does appear.

See the online help for details and illustrations.

Comments

Description The Comments property contains a text field in which you can document the object. Enter any text not to exceed 64K.

Applies to objects

Required/Optional optional

Default blank

Horizontal Elasticity

Description The Horizontal Elasticity property is how the horizontal size of the object will change at runtime to accommodate the objects or data within it:

- For frames and repeating frames, elasticity defines whether the size of the frame or repeating frame should vary with the objects inside of it.
- For objects containing text, elasticity defines whether the field or boilerplate should vary with the size of the text. Fixed size text will wrap within the defined size of the object and may be truncated if there is not enough room. Number or date data will appear as asterisks if the data cannot fit within the defined size.
- For images, drawings, and chart objects, Report Builder uses proportional scaling. The elasticity options for images, drawings, and chart objects determine the scaling factor.

See the online help for details and illustrations.

Values

Contract	Means the horizontal size of the object decreases, if the formatted objects or data within it are wide enough, but it cannot increase to a width greater than that shown in the Report Editor. Truncation of data may occur; look at the examples. (You can think of this option as meaning "only contract, do not expand.")
Expand	Means the horizontal size of the object increases, if the formatted objects or data within it are wide enough, but it cannot decrease to a width less than that shown in the Report Editor. (You can think of this option as meaning "only expand, do not contract.")
Fixed	Means the width of the object is the same on each logical page, regardless of the size of the objects or data within it. Truncation of data may occur; look at the examples. The width of the object is defined to be its width in the Report Editor.
Variable	Means the object may expand or contract horizontally to accommodate the objects or data within it (with no extra space), which means the width shown in the Report Editor has no effect on the object's width at runtime.

Applies to layout objects

Required/Optional required

Default varies according to the object

Usage Notes

- If you create a Graphics object in your report with a Vertical and Horizontal Elasticity of *Fixed*, Report Builder scales the display to fit within the dimensions you defined for the object.

Horizontal Elasticity example (field)

Suppose that you have a field with the following characteristics:

Characteristic	Value
width	1 inch
Horizontal Elasticity	Contract
height	1 inch
Vertical Elasticity	Fixed

See the online help for details and illustrations.

Note that when the data is less than one inch wide the size of the field decreases to the minimum size necessary to hold the data. It is also important to note here that if the data was more than one inch long, it would be truncated because the field cannot expand vertically or horizontally.

Horizontal Elasticity example (frame)

Suppose that you have a frame named M_1 with the following characteristics:

Characteristic	Value
Horizontal Elasticity	Variable
height	7 inches
Vertical Elasticity	Fixed

Assume further that the Page Width you defined in the report property sheet is eight inches. M_1 contains a repeating frame with one field, F_Ename. The first value of F_Ename is four inches wide, the next value is 3 inches, the next is four inches, and the last is four inches. The following diagram shows how the frame might be formatted: See the online help for details and illustrations.

Note that the first two instances of field F_Ename print on the first logical page, but, the second two instances, because there is not enough room on the first logical page, print on the second logical page. It is also important to note that if the Horizontal Elasticity was *Contract* or *Fixed*, instead of *Variable*, the frame would not print unless it could fit entirely on one logical page.

Keep With Anchoring Object

Description The Keep with Anchoring Object property indicates whether to keep an object and the object to which it is anchored on the same logical page. Setting Keep with Anchoring Object to *Yes* means that if the object, its anchoring object, or both cannot fit on the logical page, they will be moved to the next logical page.

If you set Keep with Anchoring Object to *Yes* for a repeating frame, it means the first instance of the repeating frame must be able to fit on the same page as its anchoring object. Otherwise, the Keep With Anchoring Object condition is not satisfied. If you set Keep With Anchoring Object to *Yes* for any layout object other than a repeating frame, it means the object must be able to format entirely on the same page as its anchoring object.

The anchor between the two objects may be explicit or implicit. Consequently, Keep With Anchoring Object may have an effect even if you have not explicitly created an anchor between two objects.

Applies to layout objects

Required/Optional optional

Default No

Keep With Anchoring Object example

Suppose that you have created three boilerplate objects:

See the online help for details and illustrations.

B_2 and B_3 each has Keep With Anchoring Object set to *Yes*. On the logical page where Report Builder first tries to print these objects, there is enough room for B_1 and B_2 but not for B_3. Since B_3 has Keep With Anchoring Object set to *Yes* and its anchoring object is B_2, B_2 and B_3 would be moved to the next logical page. Since B_2 also has Keep With Anchoring Object set to *Yes* and its anchoring object is B_1, B_1 would also be moved to the next logical page.

Keep With Anchoring Object restrictions

- Keep With Anchoring Object applies only to the first logical page on which the object is triggered to be formatted. Keep With Anchoring Object is ignored on subsequent pages.
- If moving the object and its anchoring object to the next logical page would leave the current logical page blank or only containing repeated objects (e.g., a page heading), Keep With Anchoring Object will not be applied. This prevents unnecessary blank pages in your report.
- If the anchoring object is a repeating frame and the current object is outside of the repeating frame, then anchoring object refers to the repeating frame as a whole (all

the instances of the repeating frame).

- If two objects are anchored to their enclosing frame and the frame has Vertical and/or Horizontal Elasticity of *Contract* or *Variable*, then all three objects must be able to complete formatting on the same page to satisfy the Keep With Anchoring Object condition. If the two objects are enclosed by a repeating frame instead of a frame, then this rule applies to each instance of the repeating frame. In the figure, B_1, B_2, and M_1 would all have to be able to complete formatting on the same page in order to satisfy the Keep With Anchoring Object condition.

See the online help for details and illustrations.

Name

Description The Name property is the name that Report Builder uses to identify the current object. Enter any valid name not to exceed 30 bytes.

Applies to objects

Required/Optional required

Default

boilerplate	B_ <i>n</i> , where <i>n</i> is a number that is unique among boilerplate object names for the report.
button objects	U_ <i>n</i> , where <i>n</i> is a number that is unique among field names for the report.
chart objects	D_ <i>n</i> , where <i>n</i> is a number that is unique among Graphics object names for the report.
database columns	the column name as referenced in the SELECT statement of the SQL query (display only).
fields	F_ <i>n</i> , where <i>n</i> is a number that is unique among field names for the report.
formula columns	CF_ <i>n</i> , where <i>n</i> is a number that is unique among formula names for the report.
frames	M_ <i>n</i> , where <i>n</i> is a number that is unique among frame names for the report.
queries	Q_ <i>n</i> , where <i>n</i> is a number that is unique among query names for the report.
groups	G_ <i>queryname</i> , where <i>queryname</i> is the query's name and a number that is unique among groups. If the query has a default name (e.g., Q_1), the default group name is of the form: G_ <i>n</i> , where <i>n</i> is a number that is unique among group names for the report. If the group is not associated with a query (i.e., a cross product group), the default group name is also G_ <i>n</i> .
matrix objects	X_ <i>n</i> , where <i>n</i> is a number that is unique among matrix names for the report.
OLE2 objects	B_ <i>n</i> , where <i>n</i> is a number that is unique among OLE2 object names for the report.
parameters	P_ <i>n</i> , where <i>n</i> is a number that is unique among column names for the report. If you create a parameter by making a bind reference in your query, the name that you used in the query is the default.

placeholder columns	CP_ <i>n</i> , where <i>n</i> is a number that is unique among placeholder names for the report.
repeating frames	R_ <i>n</i> , where <i>n</i> is a number that is unique among repeating frame names for the report.
summary columns	CS_ <i>n</i> , where <i>n</i> is a number that is unique among summary names for the report.

Page Break After

Description The Page Break After property indicates that you want all children of the object to be moved to the next page. In other words, any object that is a child object of an anchor (implicit or explicit) to this object will be treated as if it has Page Break Before set to *Yes*. Note that this does not necessarily mean that all the objects below the object with Page Break After will move to the next page.

Applies to layout objects

Required/Optional optional

Default No

Page Break After restrictions

- Page Break After delays only the formatting of an object's child objects. The formatting of sibling objects is not delayed because there is no hierarchy (i.e., parent-child relationship) by which Report Builder can determine the order of formatting for siblings. As a result, an object must be anchored (implicitly or explicitly) to another object as the parent for Page Break After to have an effect. See the online help for details and illustrations.
- For repeating frames, Page Break After sets a logical page break after the repeating frame as a whole. To have a page break after each instance of the repeating frame you should also set Maximum Records per Page to *1* on the repeating frame's property sheet.
- Objects below an object with Page Break After set to *Yes* may not move to the next page. If an object is not a child of the object with Page Break After set to *Yes*, then that object, if it can fit, may print on the same page as the object with Page Break After set to *Yes*. See the online help for details and illustrations.
- If an object has no external objects anchored to it (implicitly or explicitly), then Page Break After has no effect.

Page Break Before

Description The Page Break Before property indicates that you want the object to be formatted on the page after the page on which it is initially triggered to print. Note that this does not necessarily mean that all the objects below the object with Page Break Before will move to the next page.

Applies to layout objects

Required/Optional optional

Default No

Page Break Before restrictions

- For repeating frames, Page Break Before sets a logical page break before the repeating frame as a whole. To have a page break before each instance of the repeating frame, you should also set Maximum Records per Page to *1* on the repeating frame's property sheet.
- Objects below an object with Page Break Before set to *Yes* may not move to the next page. If an object is set to print on a page but is moved to the next page because of Page Break Before, other objects may be placed in the space where the object was originally set to print, if there is sufficient room. See the online help for details and illustrations.

Page Protect

Description The Page Protect property indicates whether to try to keep the entire object and its contents on the same logical page. Setting Page Protect to *Yes* means that if the contents of the object cannot fit on the current logical page, the object and all of its contents will be moved to the next logical page.

Note: Using Page Protect may cause objects below the page protected object(s) to appear above the page protected object(s).

Applies to layout objects

Required/Optional optional

Default No

Page Protect example (small object)

Suppose that you have created a frame named M_1 with a number of fields that display various columns and summaries. For ease of reading, you want to keep all of the fields in M_1 on the same logical page so that readers can see the columns and their summaries together.

Specify Page Protect for M_1. If M_1 and all its fields cannot fit on the first logical page that Report Builder attempts to print them, Report Builder will try to print M_1 and all its objects on the next logical page.

See the online help for details and illustrations.

Page Protect example (large object)

Suppose that you have created a frame named M_1 with a number of fields that display various columns and summaries. For ease of reading, you want to keep all of the fields in M_1 on the same logical page so that readers can see the columns and their summaries together.

Specify Page Protect for M_1. If M_1 and all its fields cannot fit on the first logical page that Report Builder attempts to print them, Report Builder will try to print M_1 and all its objects on the next logical page.

Suppose further, though, that M_1 and all its fields cannot fit on the second logical page on which Report Builder tries to print it. Report Builder will print as much of M_1 as it can on the logical page and print the rest of it on the following logical page.

Page Protect example (group report)

Suppose that you have a group report. If at all possible, you would like to have all of the details and the master appearing on the same page. To do this, you specify Page Protect for the master repeating frame (the outermost repeating frame). If the details and the master cannot fit on the first page on which they are triggered to print, they will be triggered to print on the next page instead.

Page Protect restrictions

- Page Protect applies only to the first logical page on which the object is triggered to be formatted. Page Protect is ignored on subsequent pages.
- For repeating frames, Page Protect refers to each instance of the repeating frame. Each instance of the repeating frame and its enclosed objects will be kept together by Page Protect. The only exception is that page protection is not applied to the first instance on a page other than the first page of the repeating frame. For example, in the diagram below, page protection is applied to instances 1, 2, and 4, but not to instance 3. Instance 3 is the first instance to format on the second page of the repeating frame. Notice that, in this case, if page protection had been applied to instance 3, it would have started on page 3 instead of page 2. This would have left page 2 almost completely blank. See the online help for details and illustrations.
- Page Protect is not supported for repeating frames that are part of a matrix. When repeating frames are related via matrix object, setting Page Protect will not effect them.
- If moving the object and its contents to the next logical page would leave the current logical page blank or only containing repeated objects (e.g., a page heading), Page Protect will not be applied. This prevents unnecessary blank pages in your report.
- If Page Protect causes an object to be moved to the next logical page when formatting, an object alongside the page protected object may or may not be moved as well. If the object alongside the page protected object is not anchored (implicitly or explicitly) to the page protected object, and that object can fit in the space where the page protected object could not fit, the object will print on that logical page. Otherwise, the object will be moved along with the page protected object to the next logical page.

Base Printing On

Description The Base Printing On property is the object on which to base the Print Object On property of the current object. For example, if you specify a Print Object On of *All Pages* and a Base Printing On of *Anchoring Object*, the current object will be triggered to print on every logical page on which its anchoring object (parent object) appears.

Values

Anchoring Object Is the parent object to which the current object is implicitly or explicitly anchored.

Enclosing Object Is the object that encloses the current object.

Applies to layout objects

Required/Optional required

Default Anchoring Object

Print Object On

Description The Print Object On property is the frequency with which you want the object to appear in the report. The Print Object On options indicate the logical page(s) on which the object should be triggered to print with regard to the Base Printing On object.

Note: Just because the object is triggered to print on a logical page does not mean it will print on that logical page. Other settings (e.g., Page Break Before) or the amount of space available on the page may cause Report Builder to print an object on a page other than the one on which it was initially triggered to print.

In applying these options, Report Builder considers the first page of an object to be the first logical page on which some part of the object is printed. Likewise, the last page is considered to be the last logical page on which some part of the object is printed. For example, if you specify a Print Object On of *First Page* and a Base Printing On of *Enclosing Object*, the object will be triggered to print on the first logical page on which its enclosing object appears.

Values

All Pages	Means the object and all of its contents will be printed on all logical pages of the Base Printing On object. The object will be repeated on any overflow pages of the Base Printing On object and will be truncated at the logical page boundary, if necessary.
All but First Page	Means the object and all of its contents will be printed on all logical pages of the Base Printing On object except the first logical page. The object will be formatted only on overflow pages of the Base Printing On object and will be truncated at the logical page boundary, if necessary.
All but Last Page	Means the object and all of its contents will be printed on all logical pages of the Base Printing On object except the last logical page. The object will be repeated on any overflow pages of the Base Printing On object except the last one and will be truncated at the logical page boundary, if necessary.
*Default	Means that Report Builder will use object positioning to set the Print Object On to either * <i>First Page</i> or * <i>Last Page</i> for you. (The asterisk indicates that Report Builder specified the setting for you.)
First Page	Means that the object and all of its contents will only be printed on the first logical page of the Base Printing On object. The object will be formatted and will overflow to

subsequent pages, if necessary.

Last Page Means that the object and all of its contents will only be printed on the last logical page of the Base Printing On object. The object will be formatted after the Base Printing On object and will overflow to subsequent pages, if necessary.

Applies to layout objects
Required/Optional required
Default varies according to object

Print Object On example (all)

Suppose that you have created a summary column named Col_Sum that sums the values of a column in a group named Group_1. Col_Sum is the source of a field (F_1) that is enclosed in a repeating frame (R_1), which contains the values from Group_1. If you specify a Print Object On of *All Pages* and a Base Printing On of *Enclosing Object* for F_1, it prints in every instance of R_1 on every logical page.

Print Object On example (overflow)

See the online help for details and illustrations.

Because it is outside of repeating frame C, boilerplate object D's Base Printing On setting refers to the repeating frame as a whole. Since they are inside the repeating frame, boilerplate objects A and B's Base Printing On setting refer to each instance of repeating frame C.

Object D appears on all logical pages on which repeating frame C appears. Object A, however, only prints when an instance of repeating frame C overflows. The Print Object On of *All But First Page* in this case means to trigger the object to print on every logical page of each instance of the repeating frame, except for the very first page on which an instance prints.

Since repeating frame C only contains two objects (boilerplate objects A and B), it only overflows to a second page when boilerplate object B overflows. Looking at the Print Object On setting of objects A and B (*All But First Page* and *First Page*), you would think that these two objects could never appear on the same page. In this scenario, however, they can only appear on the same page.

Print Object On restrictions

Caution: If its contents cannot fit within the logical page on which it starts to print, an object with a Print Object On of *All Pages*, *All But First Page*, or *All But Last Page* cannot overflow onto subsequent pages. In this case, the contents of the object will be truncated and the object will print with the same, truncated contents on every logical page it is triggered to print.

- If its contents cannot fit within the logical page on which it starts to print, an object with a Print Object On of *First Page* or *Last Page* can overflow onto

subsequent pages. As a result, even though it is triggered to print only on the first or last page, it may overflow beyond the first or last page. As a result of overflow, a Print Object On of *First Page* and a Print Object On of *All But First Page* are not necessarily mutually exclusive. For example, if one object has a Print Object On of *First Page* and another has *All But First Page* and they both have the same Base Printing On setting, you might think that the two objects could never appear on the same page. However, if the object with *First Page* overflows onto the second page, the two objects could appear on the same page.

- An object that is implicitly or explicitly anchored to its enclosing frame or repeating frame cannot have a Print Object On of *Last Page* or *All But Last Page*. An object that is anchored to another object within its frame or repeating frame cannot have a Print Object On of *Last Page* or *All But Last Page* with a Base Printing On of *Enclosing Object*. See the online help for details and illustrations.
- If the Base Printing On setting of an object is a repeating frame that encloses the object, then it refers to each instance of the repeating frame to which the object is anchored.
- If the Base Printing On setting of an object is a repeating frame and the object is outside of the repeating frame, then it refers to the repeating frame as a whole. For example, a Print Object On of *First Page* means the first logical page on which the repeating frame is triggered to print.
- Whether the Base Printing On setting refers to a repeating frame as a whole or the instances of a repeating frame, the Print Object On setting always refers to logical pages. For example, suppose the Base Printing On of an object is a repeating frame and the object is outside of the repeating frame. In this case, the Base Printing On would refer to the instances of the repeating frame. A Print Object On of *First Page*, however, does not mean the first instance of the repeating frame. It means the first logical page on which each instance of the repeating frame is triggered to print.
- Objects in the margin region are restarted on every physical page. As a result not all Print Object On settings make sense for margin objects. Because objects are restarted on every page, it is as if you are always on the first page of the object. *First Page* is the same as *All Pages*. *Last Page* and *All But Last Page* are invalid because "last" is never reached. *All But First Page* causes the object to never appear because the object never goes beyond "first."

Printer Code Before

Description Printer Codes are references to printer escape sequences that define special printing instructions (e.g., special font sizes or highlighting) for the object. The Printer Code Before property references the printer escape sequence to be executed before each line of the object. The printer escape sequences are inserted after the object is triggered to print but before it has actually printed.

Values Enter a valid printer code reference not to exceed 256 bytes in length. Entries may be of the form &number, where number is a number assigned to a packaged Report Builder printer code or a printer code you've created.

Applies to layout objects

Usage Notes

- In most cases, you should be able to accomplish any special formatting by writing PL/SQL for the layout objects.

PL/SQL To define this attribute using PL/SQL, use the `SRW.SET BEFORE PRINTING CODE` procedure.

Printer Code After

Description Printer codes are references to printer escape sequences that define special printing instructions (e.g., special font sizes or highlighting) for the object. The Printer Code After property references the printer escape sequence to be executed after each line of the object. The printer escape sequences are inserted after the object is triggered to print but before it has actually printed.

Values Enter a valid printer code reference not to exceed 256 bytes in length. Entries may be of the form &number, where number is a number assigned to a packaged Report Builder printer code or a printer code you've created.

Applies to layout objects

Usage Notes

- In most cases, you should be able to accomplish any special formatting by writing PL/SQL for the layout objects.

PL/SQL To define this attribute using PL/SQL, use the `SRW.SET AFTER PRINTING CODE` procedure.

Printer Code Special Font example

Suppose that in your printer definition file you created two printer codes named &123 and &456. Code &123 causes the printer to print a special font and code &456 causes the printer to print the normal font. Assume that you want all fields in a repeating frame (R_1) to be printed using the special font. You could either use the printer code for each individual field or use it for the entire repeating frame. The latter case is easier to create and maintain.

On the property sheet for the R_1 repeating frame, you enter &123 for Printer Code Before, and &456 for Printer Code After. When R_1 is printed, the printer code will be executed before and after each line of R_1 is printed.

See the online help for details and illustrations.

Printer Code restrictions

- Printer codes are only used when running your report in character-mode. They are not used when running it in bitmap.

Vertical Elasticity

Description The Vertical Elasticity property is how the vertical size of the object may change at runtime to accommodate the objects or data within it:

- For frames and repeating frames, elasticity defines whether the size of the frame or repeating frame should vary with the objects inside of it.
- For objects containing text, elasticity defines whether the field or boilerplate should vary with the size of the text. Fixed size text will wrap within the defined size of the object and may be truncated if there is not enough room. Number or date data will appear as asterisks if the data cannot fit within the defined size.
- For images, drawings, and chart objects, Report Builder uses proportional scaling. The elasticity options for images, drawings, and chart objects determine the scaling factor. See the online help for details and illustrations.

Values

Contract	Means the vertical size of the object decreases, if the formatted objects or data within it are short enough, but it cannot increase to a height greater than that shown in the Report Editor. Truncation of data may occur; look at the examples. (You can think of this option as meaning "only contract, do not expand.")
Expand	Means the vertical size of the object increases, if the formatted objects or data within it are tall enough, but it cannot decrease to a height less than that shown in the Report Editor. (You can think of this option as meaning "only expand, do not contract.")
Fixed	Means the height of the object is the same on each logical page, regardless of the size of the objects or data within it. Truncation of data may occur; look at the examples.
Variable	The height of the object is defined to be its height in the . Means the object may expand or contract vertically to accommodate the objects or data within it (with no extra space), which means the height shown in the Report Editor has no effect on the object's height at runtime.

Applies to layout objects

Required/Optional required

Default varies according to the object

Usage Notes

- If you create a chart object in your report with a Vertical and Horizontal Elasticity

of *Fixed*, Report Builder scales the Graphics Builder display to fit within the dimensions you defined for the object.

Vertical Elasticity example (expand)

Suppose that you have a field with the following characteristics:

Characteristic	Value
width	1 inch
Horizontal Elasticity	Fixed
height	2 inches
Vertical Elasticity	Expand

See the online help for details and illustrations.

Vertical Elasticity example (variable)

Suppose that you have a frame named M_2 with the following characteristics:

Characteristic	Value
width	5 inches
Horizontal Elasticity	Fixed
Vertical Elasticity	Variable

Assume further that the Page Height for the report is twelve inches. Because of the size of the objects within it, M_2 requires eight inches vertically to format. Another frame, called M_1, precedes M_2 on the logical page and takes up six inches.

The following diagram shows how the frame might be formatted.

See the online help for details and illustrations.

Note that all of M_2 cannot fit on the first logical page. Because the Vertical Elasticity is *Variable*, the first five inches of M_2 are printed on the first logical page and the last two inches are printed on the second logical page. Had the Vertical Elasticity been *Fixed*, all of M_2 would have printed on the second logical page.

Vertical Elasticity restrictions

Caution: If the contents of an object exceed its defined height, its Vertical and Horizontal Elasticity determine whether the contents overflow onto the next page or are truncated. Images, drawings, and chart objects cannot overflow. See the rules below for details about images, drawings, and Graphics Builder displays. See the online help for details and illustrations.

Additional Information: If you want your data truncated, you should use the SQL SUBSTR function in your query. For example, if you want only the first four characters of a column, then use SUBSTR in your SELECT statement to retrieve only those characters from the database.

- When formatting an object, if Vertical Elasticity is *Fixed*, Report Builder will only format it on a logical page if the page has enough space to contain the entire object.
- If Vertical Elasticity is *Expand*, Report Builder will only format the object on a logical page if the page has enough space to contain the defined height of the object (i.e., its minimum size). If formatting is not complete, the object overflows to the next page.
- If Vertical Elasticity is *Variable* or *Contract*, Report Builder will format as much of the object as possible on the first logical page and, if necessary, complete the object on the following logical page(s).
- If a number cannot fit in its field, then asterisks will appear in the field instead of the value. To get the value to appear, you either need to resize the object or change its elasticity to *Expand* or *Variable*.
- The following elasticity rules apply to objects (e.g., fields or boilerplate) that contain an image, drawing, or Graphics Builder display:
 - If the elasticity of the layout object is variable (i.e., both Vertical and Horizontal Elasticity are *Variable*), then the contents will appear the same size as they do in their source (e.g., the source file).
 - If the elasticity of the layout object is fixed (i.e., both Vertical and Horizontal Elasticity are *Fixed*), then proportional scaling is performed on the contents. In some cases, this may lead to empty space or "gaps" between the contents and the border of the object. For example, if an object is fixed both horizontally and vertically and the image is too large for the object, then the image will only fill the object in one direction. In the other direction, the image will be shorter than the object. To avoid these gaps, you can either change the elasticity (e.g., make it variable horizontally, vertically, or both) or you can manually resize the object.
 - If an object containing an image, drawing, or a Graphics Builder display is *Fixed* in only one direction, Report Builder will always attempt to have the contents fill up the *Fixed* dimension.
 - If the elasticity of the object is *Contract* in one direction and *Fixed* in the other, the contents may be clipped. To avoid clipping of the contents, you can either change the elasticity (e.g., make it *Variable* both horizontally and vertically) or you can manually resize the object.
 - If the elasticity of the object is variable (i.e., Vertical or Horizontal Elasticity is *Variable*) and the object cannot fit in its enclosing object (e.g., a frame or the logical page), the report will infinitely loop (because the object can never

format).

- If a value to be displayed in the report output relies on the formatting of the report, then the field or boilerplate object that displays the value must have a Vertical and Horizontal Elasticity of *Fixed*. (This is sometimes referred to as page-dependent referencing.) If the elasticity is not fixed, Report Builder will change it to be fixed at runtime.

A typical case of page-dependent referencing is a field that has a Source of *&Total Logical Pages*. The total number of logical pages cannot be determined until the entire report has been formatted. But Report Builder must format the field before the end of the report and cannot wait until the report is completely formatted to determine how big to make the field. Consequently, the field's elasticity must be fixed, so that Report Builder knows how much space to reserve for it before its source value is determined.

Following are cases of page-dependent referencing, where a field or boilerplate object's elasticity must be fixed because its source value depends on the formatting of the report:

- If a summary column has a Reset At or Compute At of *Page*, then the field or boilerplate object that displays its value must have fixed elasticity.
- If a formula or summary column relies on a summary column that has a Reset At or Compute At of *Page*, then the field or boilerplate object that displays its value must have fixed elasticity.
- If the Source of a field is *&Logical Page Number*, *&Panel Number*, *&Physical Page Number*, *&Total Logical Pages*, *&Total Panels*, or *&Total Physical Pages*, then any object that displays the field (either the field itself or a boilerplate object) must have fixed elasticity.
- If a field has a page number as its Source and, in the Page Numbering dialog box, Reset At is a repeating frame, then any layout object that displays its value must have fixed elasticity.

Note: If a field containing a page-dependent reference has Visible set to *No*, then the field may have Horizontal Elasticity of *Variable*, *Expand*, or *Contract*. Any boilerplate object that references the field, though, must be fixed in elasticity.

Hyperlink

Description The Hyperlink property is a URL link specification that displays a document or destination within a document when the object is clicked in the Web browser or PDF viewer.

Values Any valid link to a:

- destination within the current document (e.g., #my_dest_name)
- destination within a local document (e.g., file:/private/somedoc.pdf#a_dest_name)
- document on the local machine (e.g., file:/private/mynewdoc.pdf or file:///C:/temp/mynewdoc.pdf)
- document on a remote machine (e.g., http://www.newmach.com/newdoc.pdf, http://www.newmach.com/newdoc.html, ftp://www.reposit.com/filetoget.example, http://www.somemch.com/cgi-bin/webmenu?choice1)
- destination within a remote document (e.g., http://www.newmach.com/newdoc.pdf#some_dest_name)

Applies to layout objects

Default blank

Usage Notes

- If a Hyperlink property is specified for a frame, the property will be transferred to all the child (interior) objects that do not have a Hyperlink property specified. If the property cannot be transferred to the child objects, the frame's Hyperlink value is lost.
- A report output in PDF format can include both hyperlinks and application command line links. If the Application Command Line property is set for an object, it will be applied to the object. Otherwise, the Hyperlink property is applied.
- If a Hyperlink property is specified for an object in a frame, clicking the object will by default replace the frame with an entire window displaying the destination of the hyperlink. To replace only the frame, include the following HTML in the Additional Hyperlink Attributes property setting: target= *filename*, where *filename* is the name of frame to replace.

PL/SQL To define this attribute using PL/SQL, use the SRW.SET HYPERLINK procedure.

Hyperlink restrictions

- To follow Web links from a PDF viewer to a remote server or HTML document, the PDF viewer must be configured to work with a Web browser (e.g., configured as a helper application or installed as a plug-in to your Web browser).

Hyperlink Destination

Description The Hyperlink Destination property is a unique identifier for an object, which can be used as the destination of a Web link.

Values A valid, unique name that only makes use of the 26 upper or lower case US ASCII letters, numbers, or underscore characters. Other special characters will automatically be converted to underscore characters.

Applies to layout objects

Required/Optional Required, if you want to set a Hyperlink property for an object that specifies this object as its destination.

Default blank

Usage Notes

- If a Hyperlink Destination property is specified for a frame, the property is transferred to the visible object nearest to the upper left corner of the frame.

PL/SQL To define this attribute using PL/SQL, use the `SRW.SET LINKTAG` procedure.

Bookmark

Description The Bookmark property is a Web link that will appear in a bookmark frame of the master HTML document or in the bookmark area of the PDF viewer. Clicking on the bookmark displays the associated object at the top of the window.

Values

A text string with no indentation/ordering information. The bookmark will appear in the bookmark list according to when the object is rendered by the report.

A text string with explicit ordering/indentation of the form `x#book_mark_name`, where x is an outline number. The pound sign (#) and outline number do not appear in the bookmark window but are used to determine order and indentation.

Applies to layout objects

Default blank

Usage Notes

- If a Bookmark property is specified for a frame, the property is transferred to the visible object nearest to the upper left corner of the frame.

PL/SQL To define this attribute using PL/SQL, use the `SRW.SET_BOOKMARK` procedure.

Bookmark example (with indentation/ordering)

```
1#Expense Summary Section
2#Expense Detail Section
2.1#Expenses for the Administration Department
2.2#Expenses for the Engineering Department
2.3#Expenses for the Sales Department
2.3.1#Expenses for the Eastern Sales Region
2.3.2#Expenses for the Northern Sales Region
2.3.3#Expenses for the Southern Sales Region
2.3.4#Expenses for the Western Sales Region
```

Bookmark restrictions

- If the same outline number is used multiple times, all entries appear but the order is defined by when the objects are rendered by the report.
- If there are gaps in the numbers, one of two things will happen. If the gap is between peer level numbers, there will be no visible effect (e.g., 1.3.1 and 1.3.3, given there is no 1.3.2, will appear next to each other and at the same indentation level). If the gap is between a higher level number and a lower level number, intermediate levels will be generated as required (e.g., 1.0 followed by 2.1.1 will cause dummy 2 and 2.1 entries to be defined, whose titles will be the same as the subsequent real entry).

Application Command Line (PDF)

Description The Application Command Line (PDF) property is a command line that will be executed on the local machine when the object (in a PDF document) is clicked in the PDF viewer.

Values Any valid command line on the local machine (e.g., c:\orawin\bin\rwrn60 userid=scott/tiger report=example.rdf or /usr/local/bin/phone smith)

Applies to layout objects

Default blank

- A report output in PDF format can include both hyperlinks and application command line links. If the Application Command Line property is set for an object, it will be applied to the object. Otherwise, the Hyperlink property is applied.

PL/SQL To define this attribute using PL/SQL, use the `SRW.SET_PDF ACTION` procedure.

Application Command Line (PDF) restrictions

- The Application Command Line property is only available for PDF output.

Conditional Formatting

Description The Conditional Formatting property uses the values specified in the Conditional Formatting dialog box to set output formatting for the selected layout object when the specified condition(s) evaluate to TRUE.

Applies to layout objects

Required/Optional Optional

Display Name

Description The Display Name property is text that displays in a popup as the cursor moves over an image object in reports output in HTML or HTMLCSS format.

Values A text string not to exceed 256 characters.

Applies to image objects in HTML reports.

Default blank

Usage Notes

- When you output your report in HTML or HTMLCSS format, the Display Name property value is inserted into the IMG tag using the HTML ALT tag.
- If there is a hyperlink defined against the image object, you can also use the Additional Hyperlink Attributes property to generate a status line of information.

PL/SQL To define this attribute using PL/SQL, use the `SRW.SET DISPLAY NAME` procedure.

End of Layout Section

#Description The End of Layout Section property determines where the Live Previewer will stop searching for aligned objects. This property is used primarily in character mode and is used to restrict the scope of search for matching objects.

Values

- | | |
|-----|---|
| Yes | Means Live Previewer will stop searching for aligned objects at the end of the frame. |
| No | Means Live Previewer will search for all objects aligned in the same direction. |

Applies to frames

Required/Optional required

Default varies according to the object

Additional Hyperlink Attributes

Description The Additional Hyperlink Attributes property specifies additional HTML attributes to be applied to the link defined by the Hyperlink property.

Values Any valid HTML tags for the link defined by the Hyperlink property.

Applies to layout objects with a link defined by the Hyperlink property. If no link is defined, this property has no effect.

Default blank

Usage Notes

- If a Hyperlink property is specified for an object in a frame, clicking the object will by default replace the frame with an entire window displaying the destination of the hyperlink. To replace only the frame, include the following HTML in the Additional Hyperlink Attributes property setting: `target=filename`, where *filename* is the name of frame to replace.

PL/SQL To define this attribute using PL/SQL, use the `SRW.SET HYPERLINK ATTR` procedure.

Database Column properties

Break Order
Column Type
Comment
Database Column Name
Datatype
File Format
Name
Read from File
Value If Null
Width

Database Column restrictions

- Only Report Builder can create database columns (it creates one for each item in the SELECT lists of your queries). To create a database column, you must add to the SELECT list of your query and Report Builder will then create the database column for you.
- Database columns can only be deleted by removing the associated item from the SELECT list.

Field properties

Additional Hyperlink Attributes
Application Command Line (PDF)
Base Printing On
Bookmark
Comments
Contains HTML Tags
Format Mask
Format Trigger
Horizontal Elasticity
Hyperlink
Hyperlink Destination
Keep With Anchoring Object
Minimum Widow Lines
Name
Page Break After
Page Break Before
Page Numbering
Page Protect
Print Object On
Printer Code After
Printer Code Before
Source
Source Datatype
Vertical Elasticity
Visible

Field restrictions

- The maximum size of text in a field is 64K of text per paragraph and 64K paragraphs. This limit may vary according to your operating system.
- The contents of fields are wrapped on word boundaries, i.e., on spaces, hyphens, and tabs. Other punctuation marks are considered to be the last character of the word they follow. New line characters in the field's contents force new lines. There is no limit to the number of lines that can be created.
- In the Layout Model view, a field must be placed inside of a repeating frame whose source group is the group (or a descendant of the group) that contains the field's source column. If a field's source is a summary owned by a cross-product group, then the field must be placed inside of the set of repeating frames whose sources

are the groups in the summary column's Product Order .

- A field must be in front of the repeating frame that contains it.
- If a number cannot fit in its field, then asterisks will appear in the field instead of the value. To get the value to appear, you either need to resize the object or change its elasticity to *Expand* or *Variable*.

Source Datatype

Description The Source Datatype property is a read-only field that displays the Datatype of the field's Source.

Applies to fields

Usage Notes

- A field of Datatype *Long*, *Long Raw*, or *Raw* is treated as text unless you specify otherwise in the Format field of the Column Property Palette.

Format Mask

Description The Format Mask property defines how you want to display Date and Number values in the field. You can enter custom masks for numbers and dates or you can select from the list of values. Character values have no format mask. The values displayed initially in the list are determined by the Datatype of the field (e.g., for a field of Datatype *Number*, only number masks are shown in the list). If you enter a mask not in the list for Format Mask and accept the Field Property Palette, the new format mask will appear in the list of values.

Values Any valid number or date format mask not to exceed 1K. Blank means Report Builder will format the values based on the Datatype of the field's Source.

Applies to fields

Required/Optional optional

Default blank

Usage Notes

- If you leave this field blank and the Datatype of the field's Source is *Date*, the format mask is the default for your language (e.g., in the United States, the default is 'DD-MON-RR'); if the Datatype of the field's Source is *Number*, the values are formatted according to the width of the Source (see table below). See the online help for details and illustrations.

This default behavior is the same as in SQL*ReportWriter Version 1.1.

- Custom format masks are only valid during the current Report Builder session. If you quit Report Builder and then invoke it again, the list of values for the Format Mask field does not contain the format masks you created in the previous Report Builder session (unless you open the report(s) that used these masks). To include masks in the list that will be saved between sessions, use Edit Masks in the Tools Options dialog box (**Tools-->Tools Options**).

PL/SQL To define this attribute using PL/SQL, use the `SRW.SET FORMAT MASK` procedure.

Date and Time Format Mask syntax

The following tables describe the format masks for dates and the suffixes you can add to them, and time:

Format Mask	Explanation
SCC or CC	Century, abbreviated; 'S' prefixes "BC" with (-)
SYYYYY or YYYYY	Year; 'S' prefixes "BC" date with a (-)
I or IY or IYY	Last 1, 2, or 3 digit(s) of year
Y or YY or YYY	Last 1, 2, or 3 digit(s) of year
Y,YYY	Year with comma
SYEAR or YEAR	Year, spelled out; 'S' prefixes "BC" date with (-)
RRRR or SRRRR	Year; 'S' prefixes "BC" date with a (-)
RR	Last 2 digit(s) of year
BC, AD, or B.C., A.D.	Century indicator
Q	Quarter of year (Jan-Mar=Quarter 1)
MM	Month in digits (Jan = 01)
MONTH or MON	Name of month, or 3-letter abbreviation
WW, IW	Week in year
W	Week in Julian days
J	Julian day; the number of days since January 1,4712 BC
DDD, DD, or D	Day in year, month, or week
DAY	Day of week fully spelled out (e.g., MONDAY)
DY	Name of day, 3-letter abbreviation (e.g., MON)
AM, PM, or A.M., P.M.	Meridian indicator
HH or HH12	Hour of day (1-12)
HH24	Hour of day (0-23)
MI	Minute
SS; SSSS	Second in minute; seconds in day
FM	Toggles fill mode which replaces multiple spaces before or between dates, numbers, or words with a single space

The following suffixes may be added to the format masks:

Suffix	Explanation
TH	Suffix number ("DDth" for "4th")
SP	Spelled out number ("DDSP" for "FOUR")
SPTH or THSP	Spelled and suffixed number ("DDSPTH" for "FOURTH")

Number Format Mask syntax

The following table describes the tokens you can use in creating a number format mask:

Format Token	Explanation
0	Prints one digit.
N	Prints one digit, unless it is a leading zero to the left of the decimal point or a trailing zero to the right of the decimal point.
*	Prints one digit, unless it is a leading zero to the left of the decimal point, in which case an asterisk (*) is printed. Trailing zeros to the right of the decimal point are printed.
9	Prints one digit, unless it is a leading zero to the left of the decimal point, in which case a space is printed. Trailing zeros to the right of the decimal point are printed.
+	Prints a leading plus (+) for positive values and a leading minus (-) for negative values. This token must lead the mask.
-	Prints a leading minus (-) for negative values and nothing for positive values. This token must lead the other tokens.
MI	Prints a minus (-) after negative values and a space after positive values. This token must trail the other tokens.
S	Prints a minus (-) for negative values and a plus (+) for positive values (wherever the S appears in the mask). This token must lead or trail the other tokens.
PR	Prints angle brackets (<>) around negative values and spaces around positive values. This token must trail the other tokens.
()	Prints parentheses around negative values and spaces around positive values. The parentheses must surround the other tokens.
DB	Prints a "DB" after positive values. This token must trail the other tokens.
CR	Prints a "CR" after negative values. This token must trail the other tokens.
Y	Causes no sign symbol to be printed for negative or positive values.
V	Causes the number to be multiplied by 10 ^N , where N is the number of 0, 9, *, and S tokens that appear to the right of the V.
EEEE	Causes the number to be printed in scientific notation. All

	digit tokens refer to digits of the mantissa. There must be exactly one digit to the left of the decimal point (displayed or implicit). The token EEEE prints as E followed by a plus (+), if the ordinate is positive or zero, and a minus (-), if the ordinate is negative, and two digits representing the ordinate (e.g., E-99).
"string"	Prints the string between the double quotes. To have double-quotes inside the string, type double-quotes back to back ("").
. (period)	Prints a period (.) to separate the integral and fractional parts of a number.
D	Prints the local decimal character to separate the integral and fractional parts of a number.
, (comma)	Prints a comma (,) as the group/thousands separator.
G	Prints the local group/thousands separator.
\$	Prints \$.
L	Prints the local currency symbol.
C	Prints the ISO currency symbol.
%	Prints %.
" "	Prints a blank space. (Do not include quotes in mask.)
v	Prints a blank space for all zero values, regard-less of other tokens.
K	Prints a blank space.
<>	Delineates the beginning and ending of the decimal-aligned region (i.e., that part of the number that you want aligned with the decimal point in the format mask). Angle brackets indicate that the number should always occupy the same amount of space. If necessary, values are padded with blanks to the left or right of the decimal point.
RN, rn	Prints values in uppercase or lowercase Roman numerals, respectively. You cannot enter any other tokens with this token.

Format Masks examples

The following tables illustrate some sample format masks:

Sample Date Format	Date Displayed
MM/DD/RR	03/04/85
DD MON RRRR	04 MAR 1985
Mon. DD, RRRR	Mar. 4, 1985
Day Month DD fmHH:MI	Monday March 4 11:35 AM
AM	

Dy Mon ddth fmHH24:MI:SS Mon Mar 4th 23:35:22
 Day "the" ddthsp "of" Month Monday the fourth of March

Sample Number Format	Number	Number Displayed
-0000	7934	"7934"
	-7934	"-7934"
-00000	7934	"07934"
-NNNN	7639	"7639"
	535	"535"
-NNN	7639	"*****"
_.****	7902	"7902"
_.*****	7902	"*7902"
+NNNN	100	"+100"
	-99	"-99"
(NNNN)	1600	" 1600 "
	-800	"(800)"
NNNNPR	1600	" 1600 "
	-800	"<800>"
NNNNMI	1600	"1600 "
	-800	"800-"
NNNVNN	343	"34300"
N.NNEEEE	7369	"7.37E+03"
"SRW"-0000	7782	"SRW7782"
-\$NNNN.NN	800	"\$800"
	1600	"\$1600"
-%NNN	10	"%10"
-NN NNN.NN	3000	"3 000"
+KKNNNNN.00	1950	"+ 1950.00"
	900	"+ 900.00"
\$<NNNNN.00>	1200	"\$ 1200.00"
	500	"\$ 500.00"
\$<NNNNN.00> DB	1200	"\$ 1200.00 DB"
	-500	"\$ 500.00"
\$<NNNNN.00> CR	1200	"\$ 1200.00"
	-500	"\$ 500.00 CR"

* The quotation marks will not appear in your output. They are used here to make it clear where there are leading or trailing spaces.

Format Mask restrictions

- For number format masks, if the actual value is longer than the specified format mask, the value will appear as a string of asterisks in the report output, regardless of

the field's width. For example, if a fixed field's width is 8, the value is 1234567, and the format mask is <NNNNNN>, your output will be *****.

Similarly, if the number format mask causes the value to be larger than the field width, asterisks will appear in place of the value. For example, if a fixed field's width is 6, the value is 1234, and the format mask is -9999999, your output will be *****. This occurs because the format token 9 prints a blank for leading zeros to the left of the decimal. As a result, the value is too long to be displayed in a field of width 6.

- If you do not specify a sign token in the format mask, positive values are preceded by a space and negative values are preceded by a minus (-). This behavior is different from the default behavior in SQL*ReportWriter Version 1.1. Specifying the Y format mask token causes the Version 1.1 default behavior.
- After you create a format mask it will display in the list of values only if it is an appropriate format mask for the Datatype of the Source --i.e., format masks for numbers are displayed when the Source is a number, and format masks for dates are displayed when the Source is a date.
- Format masks that print spaces for zeros (e.g., 9) increase the number of bytes per page taken up by your output.

Format restrictions

- Sound and Video sources can only be accessed from user-created buttons.

Visible

Description The Visible property indicates whether or not the field should be formatted. If set to *No*, the field is hidden (not formatted). This is used for fields that you only want to reference in boilerplate text. Hidden fields are most often used for form letter reports, where fields are referenced in boilerplate text. To reference a hidden field, type *&fieldname* within the boilerplate text object.

Applies to fields

Required/Optional optional

Default Yes

Usage Notes

- A reference to a hidden field can appear anywhere in boilerplate text. If you want to place the field reference next to other text with no spaces in between, enclose the field name in angle brackets (e.g., *&<fieldname>Kg*). Otherwise, Report Builder will assume that all of the text between the ampersand and the next space is part of the field name. Note, however, that you can put references right next to each other with no spaces in between and without angle brackets (e.g., *&field1&field2&field3*).

Hidden Field Reference example

Suppose that you are building a form letter report and you want to insert the name, address, and phone number of the addressee in the letter. You create a field called NAME1 for the name of the addressee, ADDRESS for the address, and PHONE for the phone number. These fields are only to appear within boilerplate text, so you select No. Within the boilerplate text where you want these values to appear, you insert references as follows:

```
Dear &NAME1,  
Our records show that your address and phone are the  
following:  
&ADDRESS  
&PHONE
```

In the output, the letter would look something like this:

```
Dear J. Smith,  
Our records show that your address and phone are the  
following:  
234 Summer Lane  
San Francisco, CA 11104  
415-363-1234
```

Hidden Field restrictions

The following properties and attributes of a hidden field are applied to references to the hidden field:

- Format menu selections (Font, Size, Weight, Style, Justification)
- Horizontal Sizing
- Format Mask
- Page Numbering attributes (in Page Numbering dialog box).
- Format Trigger

Page Numbering

Description The Page Numbering property determines how page numbers are calculated for fields that have a Source of *&Physical Page Number*, *&Total Number of Physical Pages*, *&Logical Page Number*, or *&Total Number of Logical Pages*. See the online help for details and illustrations.

Note: The settings for each page are based on the logical or physical page as specified by the field's Source.

Page Numbering example (pages in body)

Suppose that you want to number the physical pages of the Main Section of your report:

- 1 Create a field named F_PAGE and set the Source property to *&Physical Page Number*.
- 2 In the Field Property Palette for F_PAGE, set the Page Numbering property by filling out the Page Numbering dialog box as follows:
 - Check Main Section and uncheck Header Section and Trailer Section because you only want to number the Main Section pages.
 - Type 1 in the Start At setting.
 - Type 1 in the Increment By setting.
 - Type *&Report* under Reset At because you want to number all the pages in the Main Section consecutively.
- 3 Position F_PAGE in the Layout Model view.

Page Numbering example ("Page *n* of *m* pages")

Suppose that you want to have the page number at the top of each page of your report. Furthermore, you want the page number to be of the form "Page *n* of *m* pages":

- 1 Create a field named F_PAGE and set the Source property to *&Physical Page Number*.
- 2 In the Field Property Palette, set the Page Numbering property by filling out the Page Numbering dialog box as follows:
 - Check Main Section and uncheck Header Section and Trailer Section because you only want to number the Main Section pages.
 - Type 1 in the Start At setting.
 - Type 1 in the Increment By setting.

- Type `&Report` under Reset At because you don't want the page numbers to reset to zero until the report is done.
- 3 Create a field called `F_TOTPGE` and set the Source property to `&Total Number of Physical Pages`.
 - 4 Do either of the following:
 - Hide these fields (i.e., set the Visible property to No) and reference them within boilerplate text:
`Page &F_PAGE of &F_TOTPGE`
 - Alternatively, position `F_PAGE` and `F_TOTPGE` beside each other. Insert boilerplate to the left of `F_PAGE` that contains the text "Page". Insert boilerplate between `F_PAGE` and `F_TOTPGE` that contains the text "of".

Page Numbering example (repeating frame instances)

Suppose that you have a repeating frame (`R_1`) with instances that span several pages and you want to number the physical pages that each instance spans. Assume also that you want the page number to be of the form "Page *n* of *m* pages":

- 1 Create a field named `F_PAGE` and set the Source property to `&Physical Page Number`.
- 2 In the Field Property Palette for `F_PAGE`, set the Page Numbering property by filling out the Page Numbering dialog box as follows:
 - Check Main Section and uncheck Header Section and Trailer Section because you only want to number the Main Section pages.
 - Type 1 in the Start At setting.
 - Type 1 in the Increment By setting.
 - Type `&R_1` under Reset At because you want the page numbers to reset at each repeating frame instance.
- 3 Create a field called `F_TOTPGE` and set the Source property to `&Total Number of Physical Pages`.
- 4 In the Field Property Palette for `F_TOTPGE`, set the Page Numbering property by filling out the Page Numbering dialog box as follows:
 - Type `&R_1` under Reset At because you want the page numbers to reset at each repeating frame instance.
- 5 Set the Print Condition of `F_PAGE` and `F_TOTPGE` such that they will print on all logical pages of `R_1`:

- set the Print Condition Type property to All
 - set the Print Condition Object property to Enclosing Object
- 6 Do either of the following:
- Hide these fields (i.e., set the Visible property to No) and create a boilerplate text object inside R_1 with the following text:
Page &F_PAGE of &F_TOTPGE
 - Alternatively, position F_PAGE and F_TOTPGE beside each other, inside R_1, using the Layout Model view. Insert boilerplate to the left of F_PAGE that contains the text "Page". Insert boilerplate between F_PAGE and F_TOTPGE that contains the text "of".

Page Numbering restrictions

If the Reset At value is a repeating frame, do not have the field appear on pages where the repeating frame does not appear. Because the value of the field is determined by the repeating frame, the results will be unpredictable on pages where the repeating frame does not appear. To avoid this situation, you can do any one of the following:

- Place the field inside the repeating frame at which its value is reset. This ensures that the field will only appear when the repeating frame appears.
- Create a Format Trigger for the field such that it will only be displayed on pages where its value makes sense (i.e., pages on which one instance of the repeating frame appears).

Similarly, if the field is not inside the repeating frame it resets at, and multiple instances appear on the same page, Report Builder cannot determine the value of the field and the results are again unpredictable. To avoid this situation, you can do any one of the following:

- Place the field inside the repeating frame at which its value is reset.
- Set Maximum Records Per Page to 1 for the repeating frame at which the the field's value is reset.

Source

Description The Source property is the place where the field gets its values (usually a column or a parameter).

Note: If Source is a page number value (e.g., *&Logical Page Number*), you can select *Page Numbering* to control how the page numbering is done.

Values Any valid column or parameter name.

&Current Date	Is the operating system date when the report runs, after the Runtime Parameter Form is finished.
&Logical Page Number	Is the current page number based upon numbering the output by logical pages.
&Panel Number	Is the current panel number within the logical page.
&Physical Page Number	Is the current page number based upon numbering the output by physical pages.
&Total Logical Pages	Is the total number of pages based upon numbering the output by logical pages.
&Total Panels	Is the total number of panels within a logical page.
&Total Physical Pages	Is the total number of pages based upon numbering the output by physical pages.

Applies to fields

Required/Optional required

Source example

Assume that you have a report with a data model like the one in the figure. Assume further that *G_Dept* is the Source for repeating frame *R_Dept* and *G_Emp* is the Source for repeating frame *R_Emp*.

See the online help for details and illustrations.

Source restrictions

- If a field is enclosed in a repeating frame, the column used as its Source must be in the source group or an ancestor of the source group of the field's enclosing repeating frame.

The figure shows a group tree. If the source group of the repeating frame were 6, then the field Source could be 1, 3, or 6. If the field has no enclosing repeating frame, the column must be a report-level column, i.e., it must be owned by the report. See the online help for details and illustrations.

- If a column has a *Reset At of Page*, then any fields that use it as a source must be contained in repeating frames that use the column's group as their source. In

addition, any boilerplate objects that reference the field must also be contained in repeating frames that use the column's group as their source.

- If you specify a Source for a field and later delete that source from your report (i.e., create a summary column, specify it as the source for a field, and later delete that summary column), your field no longer has access to data to display, and *UNDEFINED* appears in the Source field.
- If the Source of a field is *&Logical Page Number*, *&Panel Number*, *&Physical Page Number*, *&Total Logical Pages*, *&Total Panels*, or *&Total Physical Pages*, then any layout object that displays the field (either the field itself or a boilerplate object) must have fixed sizing.
- A field cannot have a Source of *Sound* or *Video*.

Formula Column properties

Break Order
Column Type
Comment
Datatype
File Format
Name
PL/SQL Formula
Product Order
Read from File
Set Break Order
Value If Null
Width

Formula Column restrictions

- You can set placeholder or formula columns to any of the following Datatypes:
Character, Number, and Date.

Frame properties

Additional Hyperlink Attributes
Application Command Line (PDF)
Base Printing On
Bookmark
Comments
Format Trigger
Horizontal Elasticity
Hyperlink
Hyperlink Destination
Keep With Anchoring Object
Name
Page Break After
Page Break Before
Page Protect
Print Object On
Printer Code After
Printer Code Before
Vertical Elasticity

Group properties

Child Dimension
Comment
Filter Type
Name
Number of Records
PL/SQL Filter

Group restrictions

- Any group that is not the lowest group owned by a query must contain at least one column with Break Order specified.
- For rules regarding cross-product groups, see "Matrix Restrictions" .

Child Dimension

Description Child Dimension indicates whether a group is a child dimension group in a nested matrix data model. By identifying which groups are child dimensions, you eliminate empty rows and/or columns in your matrix.

Values

- | | |
|-----|--|
| Yes | Identifies the group as a child dimension in a nested matrix report. |
| No | Identifies the group as not a child dimension in a nested matrix report. |

Applies to dimension groups in a single-query, matrix data model

Required/Optional optional

Default No

Usage Notes

- When Child Dimension is set to No, the lines between groups inside a cross product group are dotted. When you set Child Dimension to Yes, the line between that group and its parent will become solid.

Child Dimension example (set to No)

For the matrix below, the group containing deptno (G_DEPTNO) had Child Dimension set to No. Note how all rows are printed even if they contain no values.



Child Dimension example (set to Yes)

For the matrix below, the group containing deptno (G_DEPTNO) had Child Dimension set to Yes. Note how only rows that contain values are printed.



Child Dimension restrictions

- Child Dimension is only valid for dimension groups in single-query, matrix data models.
- Child Dimension is not valid for the topmost group in the data model.
- In the example data model below, Child Dimension would not be valid for G_1, G_Cross, and G_YEAR. It would be valid for G_DEPTNO and G_JOB.



Filter Type

Description Filter Type specifies the type of filter Report Builder should use on the records being fetched for the group.

Values

None	Means that you are not using a filter for the group.
First	Means that the first <i>n</i> records retrieved should be included in the group, where <i>n</i> is a number specified in the Number of Records property.
Last	Means that the last <i>n</i> records retrieved should be included in the group, where <i>n</i> is a number specified in the Number of Records property.
PL/SQL	Means that a PL/SQL function for determining which records to include in the group will be defined (or referenced).

Applies to groups

Required/Optional optional

Default None

Filter Type example

Suppose that you have a group named EMP that contains two columns (ENAME and SAL). The parent group of EMP is DEPT, which contains one column (DEPTNO). You specify a Filter Type of *First* and Number of Records of 3 for DEPT and a Filter Type of *First* and Number of Records of 5 for EMP. This will return the first three department numbers retrieved by Report Builder with the first five names and salaries retrieved for each department.

Filter Type restrictions

- If FilterType is *First* or *Last*, you must specify a positive whole number in the associated field. The number cannot exceed 64K in size.
- A cross-product group cannot have a filter on it.
- Records excluded from the group by filter are not used in computations. If you want to use data in calculations but not have it appear in the report output, use a Format trigger to restrict what appears in the output.

Name

Description The Name property is the name Report Builder uses to identify the current group. Enter any valid group name not to exceed 30 bytes.

Applies to groups

Required/Optional required

Default *G_queryname*, where *queryname* is the query's name and a number that is unique among groups. If the query has a default name (e.g., Q_1), the default group name is of the form: *G_n*, where *n* is a number that is unique among group names for the report. If the group is not associated with a query (i.e., a cross product group), the default group name is also *G_n*.

Comment

Description The Comment property contains text that documents the group. For example, you could describe the purpose of a break group so that someone else would be able to quickly understand what it does. Enter any text not to exceed 64K.

Applies to groups

Required/Optional optional

Default blank

Number of Records

Description The Number of Records property is the number of records to include in the group when the Filter Type is set to *First* or *Last*. For example, if the Filter Type is *First*, and the Number of Records is *20*, then the first 20 records retrieved will be included in the group.

Values Enter a positive whole number, not to exceed 64K in size.

Applies to groups

Required/Optional Required, when the Filter Type property is set to *First* or *Last*.

Default 0

PL/SQL Filter

Description The PL/SQL Filter property is a button that displays the PL/SQL Editor in which you can enter a PL/SQL function for determining which records to include in the group.

Applies to groups

Required/Optional Required, when the Filter Type property is set to *PL/SQL*.

Data Link properties

Child Column
Child Query
Condition
Parent Column
Parent Group
SQL Clause

Data Model Link restrictions

- If you are creating a link with columns specified, then you must have both a parent and a child column and the columns must have compatible datatypes. For example, if the parent column is a character datatype (e.g., VARCHAR) then the child column also must be a character datatype (e.g., CHAR).

▪

WHERE Clause Link example

Suppose that you are creating a master/detail report with the following two queries:

```
Q_1:  SELECT DISTINCT ENAME, EMPNO
      FROM EMP
      WHERE JOB = 'SALESMAN'
Q_2:  SELECT REPID, AMOUNT
      FROM SALES
      ORDER BY REPID
```

To join these two queries, you create a link such that the link's Property Palette settings are as follows:

SQL Clause	Parent Column	Condition	Child Column
WHERE	EMPNO	=	REPID

This creates a link as if you wrote one query as follows:

```
SELECT DISTINCT ENAME, EMPNO, REPID, AMOUNT
FROM EMP, SALES
WHERE EMPNO = REPID (+) AND JOB = 'SALESMAN'
ORDER BY REPID
```

Tip

Using a Tabular style, your output would look something like the figure below.

Ename	Empno	Repid	Amount
ALLEN	7499	7499	3000
	7499		810
	7499		847
	7499		24
	7499		1500

			7499	340
			7499	240
			7499	400
			7499	180
			7499	280
			7499	250
MARTIN		7654	7654	16569
		7654		2301
		7654		3306
		7654		5600
TURNER		7844	7844	350
		7844		485
		7844		293
		7844		50
		7844		1703

START WITH Link example

Suppose that you wanted to create a report that listed employees by their position in the management hierarchy. Following are the queries you might use:

```
Q_1:  SELECT EMPNO FROM EMP
      WHERE JOB = 'MANAGER'

Q_2:  SELECT ENAME, JOB, EMPNO, MGR, LEVEL
      FROM EMP
      CONNECT BY PRIOR EMPNO = MGR
```

To join these two queries, you create a link such that the link's Property Palette settings are as follows:

SQL Clause	Parent Column	Condition	Child Column
START WITH	EMPNO	=	EMPNO1

Using a tabular default layout, your output would look something like the figure below.

Empno	Ename	Job	Empno1	Mgr	Level
7566	JONES	MANAGER	7566	7839	1
	SCOTT	ANALYST	7788	7566	2
	ADAMS	CLERK	7876	7788	3
	FORD	ANALYST	7902	7566	2
	SMITH	CLERK	7369	7902	3
7698	BLAKE	MANAGER	7698	7839	1
	ALLEN	SALESMAN	7499	7698	2
	WARD	SALESMAN	7521	7698	2
	MARTIN	SALESMAN	7654	7698	2
	TURNER	SALESMAN	7844	7698	2
	JAMES	CLERK	7900	7698	2
7782	CLARK	MANAGER	7782	7839	1
	MILLER	CLERK	7934	7782	2

Child Column

Description The Child Column property is the name of a column in the child query that relates to a column in the parent group (i.e., parent column). The Child Column must be a database column, it cannot be a summary or formula. It also cannot depend upon a lexical reference (see "Child Column Restrictions").

Applies to links

Child Column restrictions

- A Report Builder link should not depend upon a lexical reference. That is, neither the child column of a link or its table name should be determined by a lexical reference. To achieve this functionality, you need to create a link with no columns specified and then enter the SQL clause (e.g., WHERE) for the link directly in the query. For example, your parent and child queries might be written as follows:

```
Parent Query:  SELECT DEPTNO FROM EMP
Child Query:   SELECT &PARAM_1 COL_1, &PARAM2
               COL_2
               FROM EMP
               WHERE &PARAM_1 = :DEPTNO
```

Note how the WHERE clause makes a bind reference to DEPTNO, which was selected in the parent query. Also, this example assumes that you have created a link between the queries in the Data Model view with no columns specified.

Child Query

Description The Child Query property is the name of the query defined as the child when you created the link in the Data Model view. This field is read-only.

Applies to links

Child Query restrictions

- A Child Query can have only one Parent Group. Multiple parents for the same child are not allowed.

Condition

Description The Condition property is a SQL operator that defines the relationship between Parent Column and Child Column. For details on SQL operators, see the *ORACLE8 Server SQL Language Reference Manual*.

Values

=	(equal to)
<	(less than)
<=	(less than or equal to)
<>	(not equal to)
>	(greater than)
>=	(greater than or equal to)
Like	Means that the condition is true when the value in one column matches the pattern in the other column. The pattern can contain % and _ as wildcard characters.
Not Like	Means that the condition is true when the value in one column does not match the pattern in the other column. The pattern can contain % and _ as wildcard characters.

Applies to links

Required/Optional required, if a Parent Column is specified.

Default = (equals sign)

Parent Column

Description The Parent Column property is a column in the parent group that relates to a column in the child query (i.e., child column). Parent Column can be a database, summary, or formula column in the parent group. If Parent Column is a summary or formula column, it cannot depend, directly or indirectly, on a column owned by the child query. This field is read-only.

Applies to links

Parent Group

Description The Parent Group property is the name of the group defined as the parent when you created the link in the Data Model view. This field is read-only.

Applies to links

Usage Notes

- You can create a link from a parent group to a child query without specifying any columns by which to link. This establishes a relationship between the two objects; however, all of the detail records will appear for each master record in the parent group.

Parent Group restrictions

- A Parent Group cannot have the Child Query as one of its ancestors. In other words, your links cannot create a circular dependency (i.e., a situation where the child query depends on the parent group and the parent group depends on the child query).

SQL Clause

Description The SQL Clause property is the type of clause that relates the parent group to the child query. For details, see the *ORACLE8 Server SQL Language Reference Manual*.

Values

HAVING

START WITH

WHERE

Applies to links

Required/Optional required, if Parent Column(s) is specified.

Default WHERE

Usage Notes

- If you use a SQL Clause of START WITH, then the SELECT Statement of the Child Query should contain a CONNECT BY clause (without a START WITH clause, since that is contained in this property). Similarly, if you use a SQL Clause of HAVING, then the SELECT Statement of the Child Query should contain a GROUP BY clause (without a HAVING clause, since that is contained in this property).

Matrix properties

Additional Hyperlink Attributes
Application Command Line (PDF)
Bookmark
Comments
Cross Product Group
Horizontal Repeating Frame
Hyperlink
Hyperlink Destination
Name
Vertical Repeating Frame

Cross Product Group

Description The Cross Product Group property is the group that contains the source groups of the Horizontal and Vertical Repeating Frames. The cross product group correlates values between one or more groups to produce the data in the matrix.

Values Enter a valid cross product group name.

Applies to matrices

Required/Optional required

Default blank

Matrix example

Suppose that you have a group named Group1 that contains a column called C_DEPTNO, which gets its values from the database column DEPTNO. A group called Group2, contains column C_JOB, which gets its values from the database column JOB, and column C_DEPTNO1, which is used for linking to Group1's query. A group called Group3 contains a column called SUMSAL, which is a summary of the database column SAL.

			Job	
		Analyst	Clerk	Manager
Dept	10		\$1300	\$2450
	20	\$6000	\$1900	\$2975
	30		\$ 950	\$2850

In this example:

- The Vertical Repeating Frame is the repeating frame that contains Group2 (the job titles).
- The Horizontal Repeating Frame is the repeating frame that contains Group1 (the department numbers).
- The Cross Product Group is Group4 (the group that is the parent of Group1 and Group2).

If you need to build a more complex matrix, you can do so by adding more columns to Group1 and Group2. For example, instead of having Group1 just contain department numbers, it could also contain the locations (LOC) of the departments. The matrix might then look something like the one below.

			Job	
Loc	Dept	Analyst	Clerk	Manager
New York	10		\$1300	\$2450
Dallas	20	\$6000	\$1900	\$2975
Chicago	30		\$ 950	\$2850

Matrix restrictions

- The down repeating frames must be below the across repeating frames in a matrix.
- A matrix object must always be on top of the repeating frames that form it (i.e., it must be one or more layers above its horizontal and vertical repeating frames). Report Builder prevents you from moving the matrix below its horizontal and vertical repeating frames.
- Moving a matrix also causes its two repeating frames to move.
- A matrix object cannot be anchored to another object and other objects cannot be anchored to it (i.e., a matrix object cannot be the parent or child object for an anchor).
- To copy a matrix, you must select the matrix and its two repeating frames. If you select the matrix by itself, nothing will be copied to the paste buffer. If you select the matrix and one of the repeating frames, only the repeating frame is placed in the paste buffer.
- A matrix object can only be resized by resizing its associated repeating frames.
- You cannot use Align, Align Objects, or Size Objects from the Arrange menu on matrix objects.
- The source groups of the repeating frames that make up the dimensions of a matrix must be from the same cross-product group.
- Repeating frames whose source groups are in the same "family" hierarchy (i.e., are descendants or ancestors of each other) must have the same Print Direction. Parent-child relationships within a cross-product group are used to create nesting in the matrix. As a result, the repeating frames associated with such groups must print in the same direction on the page.
- You can put a border on a matrix object just as you would any other object, but the width will always be the minimum width possible. You cannot widen the border due to the closeness of the objects in a matrix layout.

Horizontal Repeating Frame

Description The Horizontal Repeating Frame property is the repeating frame whose values produce the row headings and summaries, and help compute the cross product for the matrix.

Values Enter any valid repeating frame name.

Applies to matrices

Required/Optional required

Default blank

Horizontal and Vertical Repeating Frame restrictions

- The source groups of the repeating frames specified in the Horizontal and Vertical Repeating Frame properties must be within the same cross-product group.

Vertical Repeating Frame

Description The Vertical Repeating Frame property is the repeating frame whose values produce the column headings and summaries, and help compute the cross product for the matrix.

Values Enter any valid repeating frame name.

Applies to matrices

Required/Optional required

Default blank

Database Column Object properties

Comment
Column Type
Datatype
Name

OLE2 properties

Additional Hyperlink Attributes
Application Command Line (PDF)
Base Printing On
Bookmark
Comments
Format Trigger
Horizontal Elasticity
Hyperlink
Hyperlink Destination
Keep With Anchoring Object
Name
Page Break After
Page Break Before
Page Protect
Print Object On
Printer Code After
Printer Code Before
Vertical Elasticity

Parameter properties

Add
Add
Additional Attributes (HTML)
Comment
Datatype
Hide First Column
Initial Value
Input Mask
List of Values
Name
Remove
Restrict List to Predetermined Values
SELECT Statement/Static List of Values
Validation Trigger
Value
Width

System Parameters

BACKGROUND	Is whether the report should run in the foreground or the background.
COPIES	Is the number of report copies that should be made when the report is printed.
CURRENCY	Is the symbol for the currency indicator (e.g., "\$").
DECIMAL	Is the symbol for the decimal indicator (e.g., ".").
DESFORMAT	Is the definition of the output device's format (e.g., landscape mode for a printer). This parameter is used when running a report in a character-mode environment, and when sending a bitmap report to a file (e.g. to create PDF or HTML output).
DESNAME	Is the name of the output device (e.g., the file name, printer's name, mail userid).
DESTYPE	Is the type of device to which to send the report output (screen, file, mail, printer, or screen using PostScript format).
MODE	Is whether the report should run in character mode or bitmap.
ORIENTATION	Is the print direction for the report (landscape, portrait, default).
PRINTJOB	Is whether the Print Job dialog box should appear before the report is run.
THOUSANDS	Is the symbol for the thousand's indicator (e.g., ",").

Add

Description The Add button places the value in the Value field to the list of static values.

Applies to parameters

Required/Optional required, if you want to add to the static values

Additional Attributes (HTML)

Description The Additional Attributes (HTML) property contains JavaScript event handlers that trigger various input or select events (e.g., raising error messages for invalid entries or formatting input values) in the Runtime Parameter Form when the report is run via the Web.

Values Any valid JavaScript event handler

Applies to parameter form fields

Default blank

Comment

Description The Comment property contains text that documents the parameter. For example, you could describe the purpose of the parameter so that someone else would be able to quickly understand what it does. Enter any text not to exceed 64K.

Applies to parameters

Required/Optional optional

Default blank

Datatype

Description The Datatype property is the type of values the parameter contains (e.g., numbers).

Values

Character

Date

Number

Applies to parameters

Required/Optional required

Default Character

Datatype restrictions

- Any values supplied for a parameter must conform to the parameter's Datatype, even at runtime.
- Datatype cannot be edited for packaged parameters (e.g., DESTYPE).
- Parameters with a Datatype of *Character* are not padded, unless you explicitly pad them in a SQL statement.

Hide First Column

Description The Hide First Column property is a check box that indicates whether the first column in your SELECT list should be displayed in the parameter's list of values. This setting is useful for hiding values that the user does not know about or are confidential.

Applies to parameters

Default checked

Hide First Column example

You want to create a report that lists the employees in a department that the user specifies at runtime. The tables you will be using have DEPTNO as the primary/foreign key. The people using the report do not know departments by number, they only know them by name. The report query would be as follows:

```
SELECT * FROM EMP
WHERE DEPTNO = :MYDEPT
```

In the Data/Selection tab of the Parameter property sheet, you choose SELECT Statement and enter the following SQL statement to populate the list of values:

```
SELECT DEPTNO, DNAME FROM DEPT
```

You then check Hide First Column. This means that DEPTNO will not appear in the list of values when you run the report. Only DNAME will appear. When the user selects a DNAME value from the list, the corresponding DEPTNO value will be used for the parameter MYDEPT.

Initial Value

Description The Initial Value property is the default value for the parameter. This value is used unless it is overridden on the command line or the Runtime Parameter Form. Enter any value appropriate for the Datatype of the parameter. Initial Value cannot exceed 1K in length. If Initial Value is left blank, the parameter has no default value.

Applies to parameters

Required/Optional optional

Default For DESTYPE, *Screen*. For DESFORMAT, *dflt*. For COPIES, 1. For all other parameters, blank.

Initial Value restrictions

- Initial Value is validated according to the Datatype and Width of the parameter. In addition, Initial Value is validated against format masks:
 - If you specify one, the mask in Format Mask is used.
 - If Format Mask is blank and the value is a date, Report Builder' internal mask (e.g., DD-MON-YY) is used. (Number and character values are not validated unless you specify a Format Mask.)
- If no value is specified and the parameter is referenced in SQL or PL/SQL, a null is used to parse the SQL or PL/SQL. If a null causes the SQL or PL/SQL construct to parse incorrectly, an error will be raised.

Input Mask

Description The Input Mask property is the format mask Report Builder uses to validate the value supplied for the parameter on the command line or the Runtime Parameter Form. It is also used to validate the Initial Value of the parameter. Enter any valid format mask not to exceed 128 bytes or select a value from the list.

Applies to parameters

Required/Optional optional

Default blank. This implies that when the Datatype is *Date*, the Input Mask is the default for your language (e.g., in the United States, the default is 'DD-MON-YY'), and when the Datatype is *Number*, the Input Mask is -9999.

Input Mask restrictions

- The values displayed initially in Input Mask list of values are determined by the datatype of the parameter (e.g., for a parameter of Datatype *Number*, only number masks are shown in the list). If you enter a new format mask and accept the property sheet, the new value will appear in the list. However, these format masks are only valid during the current Report Builder session. If you quit Report Builder and then invoke it again, the list of values will not contain the format masks you created in the previous Report Builder session (unless you open the report(s) that used these masks). To add new format masks to the list that will be saved between sessions, use the Edit Masks button in the Preferences dialog.
- The Input Mask property does not apply to parameters of Datatype *Character*.

Name

Description The Name property is the name that Report Builder uses to identify the current parameter. Enter any valid parameter name not to exceed 30 bytes.

Applies to parameters

Required/Optional required

Default P_*n*, where *n* is a number that is unique among column names for the report. If you create a parameter by making a bind reference in your query, though, the name that you used in the query is the default.

Name restrictions

- You cannot rename system parameters.

Remove

Description The Remove button removes the selected value(s) from the list of static values.

Applies to parameters

Required/Optional required, if you want to remove static values

Restrict List to Predetermined Values

Description The Restrict List to Predetermined Values is a checkbox that indicates whether a value for the parameter must be in the list of values. For example, if you check Restrict List to Predetermined Values and the Initial Value does not match a value in the list of values, an error will be generated when you accept the Parameter property sheet. Similarly, if you set a value for the parameter on the command line or in a command script and it does not match a value in the list of values, an error will be generated when the report is run. If you check Restrict List to Predetermined Values, a list of values will appear in the runtime parameter form. If you do not check it, a combo box will appear (i.e., you can enter values as well as select from the list).

Applies to parameters

Default checked

SELECT Statement/Static Values

Description The SELECT Statement/Static Values property is a radio button that indicates whether you want to use a SELECT (SQL query) statement or a static list to populate the list of values for the parameter.

Values

SELECT Statement	Indicates that you want to enter a SELECT statement to populate the list of values for the parameter. If you select this option, then you must enter a SELECT statement in the area underneath the SELECT Statement radio button.
Static List	Indicates that you want to manually enter values to populate the list of values for the parameter. If you select this option, you must create a list of values manually. To do this, you enter a value in the Value field and then choose the Add button. Repeat this process for each value you want in the list. To remove value(s), select them in the list and then choose the Remove button.

Applies to parameters

Required/Optional required

Validation Trigger

Description The Validation Trigger property is a button that displays the PL/SQL Editor in which you can enter PL/SQL to validate and, if necessary, modify the parameter.

Applies to parameters

Value

Description The Value property is a value that will be added to the static list of values for the parameter when you choose the Add button. Enter a valid value for the parameter and then choose the Add button to add it to the static list of values.

Applies to parameters

Required/Optional required, if you want to add a value to the static list.

Width

Description The Width property is the maximum number of characters of the parameter value when entered on the command line and the Runtime Parameter Form. Enter a number from 1 through 64K.

Applies to parameters

Required/Optional required

Default 40 characters

Width restrictions

- Report Builder truncates any parameter value supplied at runtime that is longer than Width.
- Width cannot be edited for packaged parameters (e.g., DESNAME).

List of Values

Add
Hide First Column
Remove
Restrict List to Predetermined Values
SELECT Statement/Static List of Values
Value

Parameter Form Boilerplate properties

Comments

Name

Type

Name

Description The Name property is the name that Report Builder uses to identify the current boilerplate.

Values Enter any valid parameter form boilerplate name.

Applies to parameter form boilerplate

Required/Optional required

Default PB_*n*, where *n* is a number that is unique among parameter form boilerplate names for the report.

Type

Description The Type property displays the format (e.g., text) of the boilerplate object. This field is read-only.

Parameter Form Field properties

Comments
Datatype
Name
Source

Datatype

Description The Datatype property is a read-only field that displays the datatype of the parameter form field's Source.

Name

Description The Name property is the name that Report Builder uses to identify the current field.

Values Enter any valid parameter form field name.

Applies to parameter form fields

Required/Optional required

Default PF_*n*, where *n* is a number that is unique among parameter form field names for the report.

Source

Description The Source property is the parameter from which the parameter form field gets its value. The list of values contains all parameters in the report.

Values Enter any valid parameter name.

Applies to parameter form fields

Required/Optional required

Default blank

Source restrictions

- The Source of a parameter form field must be a data model parameter.
- You cannot specify more than one parameter form field with the same Source.

Placeholder Column properties

Break Order
Column Type
Comment
Datatype
File Format
Name
PL/SQL Formula
Product Order
Read from File
Set Break Order
Value If Null
Width

Placeholder Column restrictions

- You can set placeholder or formula columns to any of the following Datatypes: *Character, Number, and Date.*
- You can only assign the value of a placeholder column from one group. You cannot assign its value in multiple groups.

Placeholder for Special Summary example

Suppose that you wanted to create a master/detail report that would list the highest paid employee(s) for each department. The output below shows a sample of what you would get:

```
Dname ACCOUNTING          Loc NEW YORK          Highest Paid
                          Employee(s) :
  Ename      Job          Sal
  -----
  CLARK      MANAGER      2450.00      KING
  KING      PRESIDENT      5000.00
  MILLER     CLERK          1300.00
Dname RESEARCH          Loc DALLAS          Highest Paid
                          Employee(s) :
  Ename      Job          Sal
  -----
  SMITH     CLERK          800.00      SCOTT and FORD
  JONES     MANAGER      2975.00
  SCOTT     ANALYST      3000.00
  ADAMS     CLERK          1100.00
  FORD     ANALYST      3000.00
```

To get the names of the highest paid employee(s), you create a placeholder column named HIGHEST_PAID_EMP of Datatype *Character*. The value of HIGHEST_PAID_EMP is set by a formula column named SET_HIGHEST_PAID_EMP of Datatype *Character*. R_COUNT is a summary column with a Function of *Count*. SET_HIGHEST_PAID_EMP contains the following PL/SQL function in Formula:

```
function set_plch_high_emp return character is
begin
  /* Check if we are at first record in child, if so,
  then reset placeholder */
  if (:r_count <= 1) then
    :highest_paid_emp := ' ';
  end if;
  /* Does this child record compare to the max sal
  for the dept ? If yes, and placeholder already
  set (i.e. we have more than one max sal) set the
  placeholder to the concatenated string. Else
  set the placeholder to the employee's name */
  if (:sal = :max_dept_sal) then
    if (:highest_paid_emp <> ' ') then
      :highest_paid_emp := :highest_paid_emp||' and '
      ||:ename;
    else
      :highest_paid_emp := :ename;
    end if;
  end if;
  return (:highest_paid_emp);
end;
```

PL/SQL Formula

Description The PL/SQL Formula property is a button that displays the PL/SQL Editor in which you can enter your formula.

Applies to formula and placeholder columns

Alphabetical list of properties

Add
Additional Hyperlink Attributes
After Form Type
After Form Value
After Page Type
After Page Value
After Report Type
After Report Value
Align Summaries with Fields
Alignment
Application Command Line (PDF)
Background Color
Base Printing On
Before Form Type
Before Form Value
Before Page Type
Before Page Value
Before Report Type
Before Report Value
Between Field and Labels (Horizontal)
Between Frame and Fields (Horizontal)
Between Frame and Fields (Vertical)
Between Master and Detail (Horizontal)
Between Master and Detail (Vertical)
Between Page and Frames (Horizontal)
Between Page and Frames (Vertical)
Between Sibling Frames (Horizontal)
Between Sibling Frames (Vertical)
Bookmark
Borders
Break Order
Character Justification
Chart Column
Chart Filename
Chart Hyperlink
Chart Parameter
Chart Query
Child Column

Child Edge Percent
Child Edge Type
Child Object Name
Child Query
Collapse Horizontally
Collapse Vertically
Column Mode
Column Type
Comments
Compute At
Condition
Conditional Formatting
Cross Product Group
Dash
Datatype (column)
Datatype (parameter)
Datatype (parameter form field)
Date Justification
Design in Character Units
Direction
Distribution (section)
Distribution (report)
Disable Host Menu Item
Disable Split Screen Key
Disable Zoom Key
Display Name
Edge Background Color
Edge Foreground Color
Edge Pattern
External Query Source File
Fields Per Line
File Format
Fill Pattern
Font
Foreground Color
Filter Type
Format Mask
Format Trigger
Function
Height (Parameter Form window)
Height (physical page of section)
Hide First Column

Horiz. Space Between Frames
Horizontal Elasticity
Horizontal Panels per Page
Horizontal Repeating Frame
Hyperlink
Hyperlink Destination
Icon Name
Image
Include Bitmapped Objects
Include Borders
Initial Value
Input Mask
Inter-Field (Horizontal)
Inter-Field (Vertical)
Inter-Frame (Horizontal)
Inter-Frame (Vertical)
Justify
Keep With Anchoring Object
Label Type
Line Stretch with Frame
List of Values
Max. Horizontal Body Pages
Max. Vertical Body Pages
Maximum Records Per Page
Maximum Rows to Fetch
Minimum Widow Lines
Minimum Widow Records
Multimedia Column Type
Multimedia Column
Multimedia File Type
Multimedia File
Name
Number of Pages
Number of Records
Number Justification
Orientation
Page Break After
Page Break Before
Page Navigation Control Type
Page Navigation Control Value
Page Numbering
Page Protect

Panel Print Order
Parent Column
Parent Edge Percent
Parent Edge Type
Parent Group
Parent Object Name
PL/SQL Filter
PL/SQL Formula (formula/placeholder column)
PL/SQL Statement
PL/SQL Trigger
Place Labels Above Fields
Previewer Hint Line Text
Previewer Status Line Text
Print Direction
Print Object On
Printer Code After
Printer Code Before
Product Order
Read from File
Remove
Report Column (for Chart Column)
Report Column (for Chart Parameter)
Report Group
Report Height
Report Width
Reset At
Role Name
Restrict List to Predetermined Values
SELECT Statement/Static List of Values
Set Break Order
Source Datatype
Source File Format
Source Filename
Source (summary column)
Source (repeating frame)
Source (parameter form field)
Source (field)
SQL Clause
SQL Query Statement
Start in Zoom
Style
Suppress Previewer Title

Text (button)
Text (template)
Text Color
Title
Type (boilerplate)
Type (button)
Type (query)
Unit of Measurement
Use Previewer Hint Line
Use Previewer Status Line
Use Vertical Spacing
Validation Trigger
Value If Null
Value
Vert. Space Between Frames
Vertical Elasticity
Vertical Panels per Page
Vertical Repeating Frame
Visible
Width (column)
Width (parameter)
Width (Parameter Form window)
Width (physical page of section)

Query properties

Comment
External Query Source File
Maximum Rows to Fetch
Name
SQL Query Statement
Type

Comment

Description The Comment property contains text that documents the query . For example, you could describe the purpose of the query so that someone else would be able to quickly understand what it does. Enter any text not to exceed 64K.

Applies to queries

Required/Optional optional

Default blank

Usage Notes

- You can also insert comments directly into your SQL SELECT statement using Oracle's standard SQL comment delimiters (e.g., /* comment */). For more information, see the *ORACLE8 Server SQL Language Reference Manual*.

External Query Source File

Description The External Query Source File property is the name of a query stored in the database or a file whose SELECT statement should be used as the SELECT statement for the current query. Enter a valid external query source file name not to exceed 1K. The external query source file you reference cannot exceed 32K in length. If it does, it will be truncated and a warning raised. If the query is stored in a file, you can prefix a path to the query name. If a path is not prefixed to the filename, Report Builder uses its file path search order to find the file. If the query is stored in the database, you can prefix DB: to it to indicate that it is stored in the database. For example:

DB:myquery

If you prefix DB: to the query name, the Report Builder will go directly to the database to find the query. If you are not connected to the database, it will not be able to find the query and will issue a message to that effect.

Note: To ensure the portability of your reports, you can take advantage of the environment variable called REPORTS30_PATH. You can use this variable to specify the default directory or directories where you want to search for external files you use in your reports (e.g., external queries, boilerplate, and PL/SQL). This prevents the need to hardcode directory paths in your report.

Applies to queries

Required/Optional optional

Default blank

Usage Notes

- The SELECT statement of the external query source file you reference from this field is validated when you accept the query's property sheet.
- If you update the external query source file, you must ensure that the update does not invalidate the report definition. For example, if the report has a column for DEPTNO and you delete DEPTNO from the SELECT list of the external query source file, the report cannot run.
- When you specify an external query source file name, the SELECT statement of the external query appears in the SQL Query Statement field as read-only. To modify it, you must open the external query source file.

Maximum Rows to Fetch

Description The Maximum Rows to Fetch property is the maximum number of rows of data that the query should retrieve. Report Builder will only retrieve the number of rows for the query that you specify in this property. When you are designing a report, you often must run it many times to see how it looks. If your report retrieves a lot of data, this can be very time consuming. This property can improve performance by reducing the amount of data retrieved and formatted.

Values A whole number from 1 through 32K means that only that number of rows will be retrieved. If left blank, all rows are retrieved.

To restrict the rows retrieved by some, but not all, of a query's groups, use Filter in the Group property sheet.

Applies to queries

Required/Optional optional

Default blank

Usage Notes

- Rows that were not retrieved because they were beyond the maximum will not be used in any Report Builder-specified calculations. For example, if you set Maximum Rows to Fetch to 5 on a child query, then a summary of the child contained in the parent group will only use the five records that are retrieved by the query. If you want to use data in calculations but not have it appear in the report output, use a Format Trigger to restrict what appears in the output.

PL/SQL To define this attribute using PL/SQL, use the `SRW.SET MAXROW` procedure.

Name

Description The Name property is the identifier that Report Builder uses to refer to the current query. Enter a valid name not to exceed 30 bytes.

Applies to queries

Required/Optional required

Default Q_*n*, where *n* is a number that is unique among query names of the report.

SQL Query Statement

Description The SQL Query Statement property is a SQL SELECT statement that retrieves data from the database for your report. Enter a valid SELECT statement not to exceed 64K. The upper limit may vary between operating systems. All features of the SELECT statement are supported, except the INTO and FOR UPDATE clauses. In building your SELECT statement, you can do the following:

- use the Tables and Columns dialog box
- insert comments
- insert bind and lexical references
- review SQL errors

Applies to queries

Required/Optional required

SQL Query statement example

Following is an example of a SELECT statement:

```
SELECT O.CUSTID, P.DESCRIP, I.ITEMTOT, O.ORDERDATE,
       I.ORDID
FROM ORD O, PRODUCT P, ITEM I
WHERE O.ORDID = I.ORDID AND
      I.PRODID = P.PRODID
ORDER BY O.CUSTID, P.PRODID, O.ORDERDATE
/* For each customer, show the products they bought */
/* (DESCRIP), the amount they spent on that product */
/* (ITEMTOT), the date they ordered the product */
/* (ORDERDATE), and the order id (ITEM.ORDID) of the */
/* order in which they bought the product. */
```

SQL Query statement with bind and lexical references example

Following is an example of a SELECT statement that uses bind and lexical references:

```
SELECT CUSTID, SUM(TOTAL) TOTAL
FROM &FROMGROUP
HAVING SUM(TOTAL) > :MINTOTAL
```

where:

&FROMGROU P Is ORD GROUP BY CUSTID (or some other value that you enter at runtime). Note that, in this case, you must define a parameter named FROMGROUP with an Initial Value specified. If the value of &FROMGROUP is null when the SELECT statement is parsed, you will get an error because the statement has no table name after FROM.

:MINTOTAL Is a column from another query that is used to select customers with a minimum total of orders.

SQL Query statement with lexical reference example

Following is an example of a SELECT statement that uses a lexical reference to a parameter:

```
SELECT ENAME, EMPNO
FROM EMP
WHERE ENAME LIKE UPPER (&NAME)
```

where:

&NAME Refers to a parameter with a default value of 'JO%'. Note that you can also specify a value for the parameter at runtime. For example, from the command line, you could type:
RWRUN60 REP SCOTT/TIGER NAME='SM%'

SQL Query Statement restrictions

Caution: If the width of a column in the database is increased after you have created your report, you need to update both the data model and the layout model to reflect the change. Otherwise, truncation may occur. Database columns are automatically updated when the query is re-parsed (e.g., enter a space in SQL Query Statement and click **OK**). Summaries, formulas, and placeholders whose source columns have increased in width must have their own widths updated manually. Similarly, fields in the layout must be resized or have their sizing settings changed to accommodate the increased size of their source columns.

- The SELECT statement must follow all SQL syntax rules. For details, see the *ORACLE8 Server SQL Language Reference Manual*. Report Builder checks the syntax when you leave SQL Query Statement and when the report is generated or executed.
- No semicolon, slash, or any other character should be used to end the SELECT statement.
- You must have the appropriate privileges on the tables and views referenced in the SELECT statement in order for the SELECT statement to be validated.
- A query that is the child in a link with a SQL Clause of *START WITH* should contain a *CONNECT BY* clause in its SQL Query Statement (without a *START WITH* clause, since that is in the data model link). Similarly, a query that is the child in a link with a SQL Clause of *HAVING* should contain a *GROUP BY* clause in its SQL Query Statement (without a *HAVING* clause, since that is in the data model link).

- If SELECT expressions are added to, renamed in, or deleted from a query, Report Builder automatically creates or deletes the corresponding columns. If the layout has been defined (i.e., layout model contains objects), Report Builder will not update the layout model based upon changes to the SELECT statement. You must update the affected objects in the Layout Model view manually or create a new default layout for the report. The reason Report Builder does not update the layout model for you automatically is that you could inadvertently lose any modifications you have made to the default layout.
- If you make any changes in the database to the datatype or name of columns that are selected by your query, you must manually make whatever changes are necessary to the Report Builder columns and fields that use the changed column as a Source.
- The order of the columns in the SELECT statement determines the initial column order. Subsequent changes to the order of columns in the SELECT statement will not change the order of the columns. If you add new columns to the SELECT statement, they will be added at the end of the lowest child group belonging to the query. The order of the columns is used to order the associated fields in the default layout.

Type

Description The Type property is a read-only field that describes the type of query.

Applies to queries

Database Ref Column properties

Break Order
Comment
Column Type
Datatype
Name
Width

Repeating Frame properties

Additional Hyperlink Attributes
Application Command Line (PDF)
Base Printing On
Bookmark
Column Mode
Comments
Format Trigger
Horiz. Space Between Frames
Horizontal Elasticity
Hyperlink
Hyperlink Destination
Keep With Anchoring Object
Maximum Records Per Page
Minimum Widow Records
Name
Page Break After
Page Break Before
Page Protect
Print Object On
Printer Code After
Printer Code Before
Print Direction
Source
Vert. Space Between Frames
Vertical Elasticity

Repeating Frame restrictions

- A repeating frame can be placed anywhere in the layout model unless its source group is a dimension of a matrix.
- A repeating frame must be behind the objects (e.g, fields and boilerplate) that it contains.

Column Mode

Description The Column Mode property controls how Report Builder fetches and formats data for instances of repeating frames. With Column Mode set to *Yes*, the next instance of a repeating frame can begin formatting before the previous instance is completed.

See the online help for details and illustrations.

With Column Mode set to *No*, the next instance cannot begin formatting before the previous instance is completed.

See the online help for details and illustrations.

Column Mode is used mainly for master repeating frames or repeating frames that contain fields that may expand vertically or horizontally (i.e., elasticity is *Variable* or *Expand*).

Applies to repeating frames

Required/Optional optional

Default No

Column Mode example

See the online help for details and illustrations.

Column Mode restrictions

- You can only specify Column Mode for repeating frames that have either:
 - Print Direction of Across and Horizontal Elasticity of Fixed
 - Print Direction Down and Vertical Elasticity of Fixed
- Repeating frames with a Print Direction of Across/Down or Down/Across cannot have Column Mode specified.

Horiz. Space Between Frames

Description The Horiz. Space Between Frames property is the amount of space (in the report's unit of measurement) that you want horizontally between instances of the repeating frame. Enter a number of zero or more. The maximum amount depends upon the unit of measurement. For inches, the maximum is 512 inches. For centimeters, it is 1,312 centimeters. For picas, it is 36,864 picas. See the online help for details and illustrations.

Applies to repeating frames

Required/Optional required

Default 0

Horiz. Space Between Frames restrictions

- When the Print Direction is *Down*, this setting has no effect.
- When the Print Direction is *Down/Across*, this setting affects space between instances that appear one next to the other.

Maximum Records Per Page

Description The Maximum Records Per Page property is the maximum number of instances of the repeating frame that will be formatted on a logical page. Suppose that you have a repeating frame with many instances. To improve the appearance of your report, you prefer to have at most three instances of the repeating frame on a given logical page. To ensure that you never have more than three instances per logical page, you set Maximum Records Per Page to 3.

Values

a whole number from 1 through 32K	Means that number of instances is the maximum that can be formatted on a logical page.
blank	Means that as many instances of the repeating frame as possible can be formatted on a logical page.

Applies to repeating frames

Required/Optional optional

Default blank

Maximum Records Per Page example

Suppose that you want each instance of a repeating frame to be on a logical page by itself. First, set Maximum Records Per Page to 1 for the repeating frame so that only one instance appears on each logical page. Then, specify Page Break Before and Page Break After, to make sure that the first instance of the repeating frame starts on a new logical page.

Minimum Widow Records

Description The Minimum Widow Records property is the minimum number of instances that should appear on the logical page where the repeating frame starts to print. If the number of instances specified for this property cannot fit on the logical page where the repeating frame is initially triggered to print, then the repeating frame will start formatting on the next page. If set to *1*, there is no minimum and the repeating frame will start formatting on any logical page that can fit one instance. Suppose that to improve the appearance of your report, you want to make sure that you do not have the first instance of the repeating frame printed at the bottom of a logical page, by itself. To make sure that there are at least two instances on the logical page, you set Minimum Widow Records to *2*.

Values

a whole number from 1 through 32K

Means that number of instances is the minimum that can be formatted on a logical page. *1* means that there is no minimum.

blank

Means that there is no minimum.

Applies to repeating frame

Required/Optional required

Default blank

Minimum Widow Records example

Suppose that you want all instances of a repeating frame to appear on one logical page. To do this, you set Minimum Widow Records to a very large number. If not all instances can fit on the first logical page where the repeating frame is triggered to print, none will be formatted on that page. Instead, they are all formatted on the next logical page. Notice, though, that on the next and all subsequent pages, Minimum Widow Records is ignored. Since you are starting with a fresh page, if all instances cannot fit on that logical page, they will not all fit on any one logical page anyway.

Minimum Widow Records restrictions

- Minimum Widow Records applies only to the first logical page on which the object is triggered to be printed. Minimum Widow Records is ignored on subsequent pages.
- If Minimum Widow Records is greater than the total number of instances of the repeating frame, then all instances must be able to fit on the first logical page where the repeating frame is triggered to format. If all instances cannot fit on the first logical page where the repeating frame is triggered to format, the repeating

frame will start formatting on the next page.

Print Direction

Description The Print Direction property is the direction in which successive instances of the repeating frame appear.

Values

- | | |
|-------------|--|
| Across | Means that each instance of the repeating frame subsequent to the first instance is printed to the right of the previous instance across the logical page. |
| Across/Down | Means that each instance of the repeating frame subsequent to the first instance is printed to the right of the previous instance until an entire instance cannot fit between the previous instance and the right margin of the logical page. At that time, Report Builder prints the instance below the leftmost instance on the logical page, provided there is enough vertical space left on the logical page for the instance to print completely. |
| Down | Means that each instance of the repeating frame subsequent to the first instance is printed below the previous instance down the logical page. |
| Down/Across | Means that each instance of the repeating frame subsequent to the first instance is printed below the previous instance until an entire instance cannot fit inside the bottom margin of the logical page. At that time, Report Builder prints the instance to the right of the topmost instance on the logical page, provided there is enough horizontal space left on the logical page for the instance to print completely. |

Applies to repeating frames

Required/Optional required

Default Down

Print Direction example (across/down)

Suppose that you have a repeating frame with Print Direction of *Across/Down*. Instances of the repeating frame print from left to right and then top to bottom. See the online help for details and illustrations.

If, however, there was an object (e.g., a graphic) on a page that prevented two instances of the repeating frame going across the page, the repeating frame instances would only print down the page. Where there is room to print more than one instance across the page, the repeating frame instances begin printing across and then down.

See the online help for details and illustrations.

Print Direction restrictions

- If a nested repeating frame has a Print Direction of *Across* and its enclosing repeating frame has a Print Direction of *Down*, the across repeating frame can cause overflow when it is too wide for the current page. The next instance of the outer, down repeating frame will not print until the previous instances of the across repeating frame completes. **Note:** If you wanted A3 on page 1 to be aligned with the overflow of A3 on page 2, you could use Column Mode. See the online help for details and illustrations.
- Matrix dimension repeating frames whose source groups are in the same "family" hierarchy (i.e., are descendants or ancestors of each other) within a cross-product group must have the same Print Direction. Parent-child relationships within a cross-product group are used to create nesting in the matrix. As a result, the repeating frames associated with such groups must print in the same direction on the page.

Source

Description The Source property is the group that owns the data for the repeating frame. Select any valid group name.

Applies to repeating frames

Required/Optional required

Usage Notes

- Source cannot be a cross-product group.

Vert. Space Between Frames

Description The Vert. Space Between Frames property is the amount of space (in the report's unit of measurement) you want vertically between instances of the repeating frame. Enter a number of zero or more. The maximum amount depends upon the unit of measurement. For inches, the maximum is 512 inches. For centimeters, it is 1,312 centimeters. For picas, it is 36,864 picas.

See the online help for details and illustrations.

Applies to repeating frames

Required/Optional required

Default 0

Vert. Space Between Frames restrictions

- When the Print Direction is *Across*, this setting has no effect.
- When the Print Direction is *Across/Down*, this setting determines the space between instances that appear one below the other.

Report properties

After Form Type
After Form Value
After Page Type
After Page Value
After Report Type
After Report Value
Before Form Type
Before Form Value
Before Page Type
Before Page Value
Before Report Type
Before Report Value
Comments
Design in Character Units
Direction
Disable Host Menu Item
Disable Split Screen Key
Disable Zoom Key
Distribution
Height (Parameter Form Window)
Include Borders
Include Bitmapped Objects
Max. Horizontal Body Pages
Max. Vertical Body Pages
Name
Number of Pages
Page Navigation Control Type
Page Navigation Control Value
Panel Print Order
Previewer Hint Line Text
Previewer Status Line Text
Role Name
Suppress Previewer Title
Start in Zoom
Previewer Title
Unit of Measurement
Use Previewer Hint Line
Use Previewer Status Line

Width (Parameter Form Window)

Unit of Measurement

Description The Unit of Measurement property is the standard of measure used to specify the dimensions of the report and the parameter form.

Values

Centimeter

Inch

Point Is a printer's type size that makes up a pica. Each pica contains twelve points and is equivalent to one-sixth of an inch.

Applies to reports

Required/Optional required

Default Inch

Max. Horizontal Body Pages

Description The Max. Horizontal Body Pages property is the maximum number of body pages in width you want to appear in your report.

Values Any whole number from 1 through 999.

Applies to reports

Required/Optional required

Default 10

Max. Vertical Body Pages

Description The Max. Vertical Body Pages property is the maximum number of body pages in height you want to appear in your report.

Values Any whole number from 1 through 999.

Applies to reports

Required/Optional required

Default 10

Panel Print Order

Description The Panel Print Order is the order in which the physical pages comprising a logical page should be printed. It also determines the order in which panels are numbered within the logical page. The figure illustrates logical pages, physical pages, and panels.

See the online help for details and illustrations.

Values

Across/Down Means the physical pages of the report body will print left-to-right then top-to-bottom.

Down/Across Means the physical pages of the report body will print top-to-bottom and then left-to-right.

Applies to reports

Required/Optional required

Default Across/Down

Direction

Description The Direction property enables you to specify the reading order for the entire report. The value of Direction serves as the default for layout objects in the report (e.g., fields, boilerplate, etc.). It also controls the orientation of the Report Editor (e.g., if Direction is *Right to Left*, the orientation of the Report Editor will be right to left). This means you do not have to move objects manually to simulate a right to left orientation in the Report Editor.

Values

Left to Right
Right to Left

Applies to reports

Required/Optional required

Default The natural writing direction of the language.

Direction restrictions

- Changing the direction of the report or any object only takes effect when running on a bi-directional platform.

Distribution

Description The Distribution property uses the values specified in the Distribution dialog box to define a distribution for the entire report.

Applies to reports

Required/Optional optional

Previewer Title

Description The Previewer Title is the text that appears at the top of the report in the Live Previewer view. To change the default title, type the desired text in the field provided.

Values Enter an alphanumeric string not to exceed 256 bytes. You can restore the default title by deleting whatever title text you entered yourself.

Applies to reports

Role Name

Description The Role Name property specifies the database role to be set for the report at runtime. You can use this option to give end users the ability to run reports that query database tables to which they do not normally have access privileges. Reports Runtime can only set a role that is granted in the database. If Reports Runtime cannot set a role, it will not run the report.

Values Enter a valid database role, not to exceed 30 characters.

Applies to reports

Required/Optional optional

Default The default role(s) specified for the user in the database, which are set when the user connects to the database.

Usage Notes

- You can set the Role Name property in the Property Palette for the Report; to (optionally) set a Role Password, double-click the button in the Role Name value field to display the Set Role dialog.
- If you specify a role using the command line with Reports Runtime (RWRUN60), it overrides the role settings in the report.
- You can set only one role for a report. To set multiple roles, you need to set a role to which the other roles have been granted.

Role Name restrictions

- Setting a role in a PL/SQL trigger can cause conflicts, and is not recommended.

Width (Parameter Form window)

Description The Width property is the width of one parameter form page in the Unit of Measurement.

Values Enter a valid width of zero or more. The maximum width depends upon the Unit of Measurement. For inches, the maximum width is 512 inches. For centimeters, it is 1,312 centimeters. For points, it is 36,864 points.

Applies to reports

Required/Optional required

Default 4 inches or 10 centimeters or 300 points

Width (Parameter Form window) restrictions

- If you change the Unit of Measurement after you have already specified the Width, the Width will change automatically to the approximate width in the new Unit of Measurement.

Height (Parameter Form window)

Description The Height property is the height of one parameter form page in the Unit of Measurement.

Values Enter a valid height of zero or more. The maximum height depends upon the Unit of Measurement. For inches, the maximum height is 512 inches. For centimeters, it is 1,312 centimeters. For points, it is 36,864 points.

Applies to reports

Required/Optional required

Default 4 inches or 10 centimeters or 300 points

Height (Parameter Form window) restrictions

- If you change the Unit of Measurement after you have already specified the Height, the Height will change automatically to the approximate height in the new Unit of Measurement.

Number of Pages

Description The Number of Pages property is the total number of pages you want in your Runtime Parameter Form.

Values Any whole number from 1 through 9,999.

Applies to reports

Required/Optional required

Default 1

Page Navigation Control Type

Description The Page Navigation Control Type property specifies how the scripting will be defined for navigation buttons in HTML page-streamed report output.

Values

- File is any valid text file containing HTML scripting code for page navigation controls.
- Text is any text string containing HTML scripting code for page navigation controls.

Applies to HTML or HTMLCSS page-streamed report output

Required/Optional optional

Default Text

PL/SQL To define this attribute using PL/SQL, use the `SRW.SET_PAGE_NAVIGATION_HTML` procedure.

Page Navigation Control Value

Description The Page Navigation Control Value property specifies the filename or scripting code for the navigation buttons in HTML page-streamed report output.

Values Any valid filename or a text string containing valid HTML commands, depending on what you specified for Page Navigation Control Type. You can click the value field (labeled **V**) to modify the default built-in JavaScript as desired.

Applies to HTML or HTMLCSS page-streamed report output

Required/Optional optional

Default Report Builder built-in JavaScript

Usage Notes

If you choose to provide your own script for navigation controls, you must include two variables in your script:

- `&TotalPages` (total number of pages)
- `&file_name` (the name of the output destination file; e.g., myreport).

Report Builder assigns values to these variables during formatting.

The height of the page navigation control frame is hard coded to 70 points without a scroll bar. The width of the frame is variable. If you want to use icons and images in this frame, keep these dimensions in mind.

PL/SQL To define this attribute using PL/SQL, use the `SRW.SET_PAGE_NAVIGATION_HTML` procedure.

Before Report Type

Description The Before Report Type property specifies the type of header to appear at the beginning of your document when the report is output to an HTML file and viewed in a Web browser.

If you use HTML page streaming, this property applies to the first page of your report output. To specify HTML to apply to the entire report (e.g., background color or images, or any other <body> HTML attributes), you must use the Before Page properties.

Values

Text Is any text string containing valid HTML commands.

File Is any valid text or graphics file.

Applies to reports

Required/Optional Required, if you specified a Before Report Value.

Default Text

PL/SQL To define this attribute using PL/SQL, use the `SRW.SET BEFORE REPORT HTML` procedure.

Before Report Value

Description The Before Report Value property specifies a header to appear at the beginning of your document when the report is output to an HTML file and viewed in a Web browser.

If you use HTML page streaming, this property applies to the first page of your report output. To specify HTML to apply to the entire report (e.g., background color or images, or any other <body> HTML attributes), you must use the Before Page properties.

Values Any valid filename or a text string containing valid HTML commands, depending on what you specified for Before Report Type.

Applies to reports

Required/Optional Required, if you specified a Before Report Type of *File*.

Default The default HTML commands included at the beginning of a report:

```
<html>
<body bgcolor="#ffffff">
```

Usage Notes

- This property is useful for placing a logo or some standard links at the beginning of an HTML document.

PL/SQL To define this attribute using PL/SQL, use the `SRW.SET BEFORE REPORT HTML` procedure.

Before Report Value restrictions

- If you replace the default text, you must ensure that it contains the equivalent HTML commands.

After Report Type

Description The After Report Type property specifies the type of trailer to appear at the end of your document when the report is output to an HTML file and viewed in a Web browser.

If you use HTML page streaming, this property applies to the last page of your report output. To specify HTML to apply to the entire report (e.g., background color or images, or any other <body> HTML attributes), you must use the Before Page properties.

Values

Text Is any text string containing valid HTML commands.

File Is any valid text or graphics file.

Applies to reports

Required/Optional Required, if you specified an After Report Value.

Default Text

PL/SQL To define this attribute using PL/SQL, use the `SRW.SET AFTER REPORT HTML` procedure.

After Report Value

Description The After Report Value property specifies a trailer to appear at the end of your document when the report is output to an HTML file and viewed in a Web browser. If you use HTML page streaming, this property applies to the last page of your report output. To specify HTML to apply to the entire report (e.g., background color or images, or any other <body> HTML attributes), you must use the Before Page properties.

Values Any filename or a text string containing valid HTML commands, depending on what you specified for the After Report Type.

Applies to reports

Required/Optional Required, if you specified an After Report Type of *File*.

Default The default HTML commands included at the end of a report:

```
</body></html>
```

Usage Notes

- This property is useful for placing a logo or some standard links at the end of an HTML document.

PL/SQL To define this attribute using PL/SQL, use the `SRW.SET AFTER REPORT HTML` procedure.

After Report Value restrictions

- If you replace the default text, you must ensure that it contains the equivalent HTML commands.

Before Page Type

Description The Before Page Type property specifies the type of header to appear at the beginning of each page of your document when the report is output to an HTML file and viewed in a Web browser.

If you use HTML page streaming, this property applies to all pages of your report output (e.g., background color or images, or any other <body> HTML attributes). To specify HTML for only the first (header) or last (footer) pages of your report, use the Before Report or After Report properties, respectively.

Values

Text Is any text string containing valid HTML commands.

File Is any valid text or graphics file.

Applies to reports

Required/Optional Required, if you specified a Before Page Value.

Default Text

PL/SQL To define this attribute using PL/SQL, use the `SRW.SET BEFORE PAGE` HTML procedure.

Before Page Value

Description The Before Page Value property specifies a header to appear at the beginning of each page of your document when the report is output to an HTML file and viewed in a Web browser.

If you use HTML page streaming, this property applies to all pages of your report output (e.g., background color or images, or any other <body> HTML attributes). To specify HTML for only the first (header) or last (footer) pages of your report, use the Before Report or After Report properties, respectively.

Values Any filename or a text string containing valid HTML commands, depending on what you specified for the Before Page Type.

Applies to reports

Required/Optional Required, if you specified a Before Page Type of *File*.

Default blank

Usage Notes

- This property is useful for placing a logo or some standard links at the beginning of each page of an HTML document.
- If you want this value to apply only to the current page, you can specify it in a Format Trigger for an object on that page.

PL/SQL To define this attribute using PL/SQL, use the `SRW.SET BEFORE PAGE HTML` procedure.

After Page Type

Description The After Page Type property specifies the type of footer to appear at the end of each page of your document when the report is output to an HTML file and viewed in a Web browser.

If you use HTML page streaming, this property applies to all pages of your report output. To specify HTML for only the first (header) or last (footer) pages of your report, use the Before Report or After Report properties, respectively.

Values

Text	Is any text string containing valid HTML commands.
File	Is any valid text or graphics file.

Applies to reports

Required/Optional Required, if you specified an After Page Value.

Default Text

PL/SQL To define this attribute using PL/SQL, use the `SRW.SET AFTER PAGE` HTML procedure.

After Page Value

Description The After Page Value property specifies a footer to appear at the end of each page of your document when the report is output to an HTML file and viewed in a Web browser.

If you use HTML page streaming, this property applies to all pages of your report output. To specify HTML for only the first (header) or last (footer) pages of your report, use the Before Report or After Report properties, respectively.

Values Any filename or a text string containing valid HTML commands, depending on what you specified for the After Page Type.

Applies to reports

Required/Optional Required, if you specified an After Page Type of *File*.

Default The default HTML commands included at the end of a page:

```
<hr size=5 noshade>
```

Usage Notes

- This property is useful for placing a logo or some standard links at the end of each page of an HTML document.
 - If you want this value to apply only to the current page, you can specify it in a `Format Trigger` for an object on that page.

PL/SQL To define this attribute using PL/SQL, use the `SRW.SET AFTER PAGE HTML` procedure.

After Page Value restrictions

- If you replace the default text, you must ensure that it contains the equivalent HTML commands.

Before Form Type

Description The Before Form Type property specifies the type of header to appear at the beginning of your runtime parameter form when the report is output to an HTML file and viewed in a Web browser.

Values

Text	Is any text string containing valid HTML commands.
File	Is any valid text or graphics file

Applies to reports

Required/Optional Required, if you specified a Before Form Value.

Default Text

PL/SQL To define this attribute using PL/SQL, use the `SRW.SET BEFORE FORM HTML` procedure.

Before Form Value

Description The Before Form Value property specifies a header to appear at the beginning of your runtime parameter form when the report is output to an HTML file and viewed in a Web browser.

Values Any valid filename or a text string containing valid HTML commands, depending on what you specified for Before Form Type.

Applies to reports

Required/Optional Required, if you specified a Before Form Type of *File*.

Default The default HTML commands included at the beginning of a Runtime Parameter Form:

```
<html>
<body bgcolor="#ffffff">
<form method=post>
<center>
<p><hr>
<table border=0 cellspacing=0 cellpadding=0>
<tr>
<td><input type=submit>
<td width=15>
<td><input type=reset>>
</table>
<p><hr><p>
```

Usage Notes

- This property is useful for placing a logo or some standard links at the beginning of a runtime parameter form for an HTML document.

PL/SQL To define this attribute using PL/SQL, use the `SRW.SET BEFORE FORM` HTML procedure.

Before Form Value restrictions

- If you replace the default text, you must ensure that it contains the equivalent HTML commands.

After Form Type

Description The After Form Type property specifies the type of footer to appear at the end of your runtime parameter form when the report is output to an HTML file and viewed in a Web browser.

Values

Text	Is any text string containing valid HTML commands.
File	Is any valid text or graphics file

Applies to reports

Required/Optional Required, if you specified an After Form Value.

Default Text

PL/SQL To define this attribute using PL/SQL, use the `SRW.SET_AFTER_FORM_HTML` procedure.

After Form Value

Description The After Form Value property specifies a footer to appear at the end of your runtime parameter form when the report is output to an HTML file and viewed in a Web browser.

Values Any valid filename or a text string containing valid HTML commands, depending on what you specified for After Form Type.

Applies to reports

Required/Optional Required, if you specified an After Form Type of *File*.

Default The default HTML commands included at the end of a runtime parameter form:

```
</center>  
</body>  
</form>  
</html>
```

Usage Notes

- This property is useful for placing a logo or some standard links at the end of a Runtime Parameter Form for an HTML document.

PL/SQL To define this attribute using PL/SQL, use the `SRW.SET_AFTER_FORM_HTML` procedure.

After Form Value restrictions

- If you replace the default text, you must ensure that it contains the equivalent HTML commands.

Design in Character Units

Description The Design in Character Units property specifies that the Report Editor and its grid use units of character cells. Using this and Include Borders causes the Report Editor to display character-mode borders for objects with line widths greater than zero.

Default No

Use Previewer Hint Line

Description The Previewer Hint Line is the line of the Live Previewer view that is displayed on the second to last line of your screen. To change the default hint line, select *Yes*, and type the desired text in the Previewer Hint Line Text property.

Values If *Yes*, enter the text in the Previewer Hint Line Text property.

Applies to reports

Default No

Usage Notes

- You can restore the default hint line by deleting the hint text you entered in the Previewer Hint Line Text property and setting the Use Previewer Hint Line property to *No*.

Previewer Hint Line Text

Description The Previewer Hint Line is the line of the Live Previewer view that is displayed on the second to last line of your screen.

Values Enter an alphanumeric string not to exceed 80 bytes.

Applies to reports

Use Previewer Status Line

Description The Previewer Status Line is the last line of the Live Previewer view. To change the default status line, select *Yes*, and type the desired text in the Previewer Status Line Text property.

Values If *Yes*, enter the text in the Previewer Status Line Text property.

Applies to reports

Default No

Usage Notes

- You can restore the default status line by deleting the status text you entered in the Previewer Status Line Text property and setting the Use Previewer Status Line property to *No*.

Use Previewer Status Line restrictions

- If Previewer Status Line is set to No, Report Builder does not reserve a line for it. This enables you to see one more line of your report output per screen.
- When you change the Previewer Status Line Text, the report name will not appear in the status line unless you enter it there yourself. For example, you could enter:

```
This report is for Human Resources personnel only  
(Report: deptinfo)
```

Previewer Status Line Text

Description The Previewer Status Line is the last line of the Live Previewer view.

Values Enter an alphanumeric string not to exceed 80 bytes.

Applies to reports

Include Bitmapped Objects

Description The Include Bitmapped Objects property converts all bit-mapped objects (i.e., Graphics Builder displays and boilerplate objects except for text, lines, and rectangles) to boxes when the report is run in character mode. If set to *No*, all bit-mapped objects and any explicit anchors to them will be deleted when the report is run in character mode.

Applies to reports

Default No

Include Bitmapped Objects restrictions

- Lines that are not horizontal or vertical will always be converted to horizontal or vertical lines in character mode, regardless of this setting. Similarly, rotated rectangles will be converted to non-rotated rectangles in character mode, regardless of this setting.

Include Borders

Description The Include Borders Property turns on borders in character mode for any objects that have a line width greater than zero in bit-mapped. If set to *No*, all borders are turned off in character mode.

Applies to reports

Default No

Usage Notes

- Setting both this property and the Design in Character property to *Yes* causes the Report Editor to display character-mode borders for objects with line widths greater than zero.

Disable Host Menu Item

Description The Disable Host Menu Item property disables the Host item in the File menu in both Runtime Parameter Form and Live Previewer views. The Host item appears in the File menu of character-mode Report Builder.

Applies to reports

Default No

Disable Split Screen Key

Description The Disable Split Screen Key disables the [Split Vertical] and [Split Horizontal] function keys.

Applies to reports

Default No

Disable Zoom Key

Description The Disable Zoom Key property disables the [Zoom/Unzoom] function key in the Live Previewer view.

Applies to reports

Default No

Disable Zoom Key restrictions

- If Disable Zoom Key and Start in Zoom are both set to Yes, the Live Previewer view will be displayed with Zoom toggled on, and you will be unable to toggle out of Zoom.

Start in Zoom

Description The Start in Zoom property displays the Live Previewer view with Zoom toggled on, (i.e., the border that normally appears around the Live Previewer view is not visible and all you see is the report output).

Applies to reports

Default No

Start in Zoom restrictions

- (If Disable Zoom Key and Start in Zoom are both set to *Yes*, the Live Previewer view will be displayed with Zoom toggled on, and you will be unable to toggle out of Zoom.

Suppress Previewer Title

Description The Suppress Previewer Title property suppresses the display of the Title in the character mode Previewer.

Applies to reports

Default Yes

Ref cursor query properties

Comment
Maximum Rows to Fetch
Name
PL/SQL Statement
Type

PL/SQL Statement

Description The PL/SQL Statement property is a button that displays the PL/SQL Editor in which you can enter a PL/SQL function that uses a ref cursor to return data.

Applies to queries

Required/Optional Required

Section properties

Distribution
Height
Horizontal Panels per Page
Orientation
Report Height
Report Width
Vertical Panels per Page
Width

Distribution

Description The Distribution property uses the values specified in the Distribution dialog box to define a distribution for the current section.

Applies to sections

Required/Optional optional

Height

Description The Height property is the height of one physical page (including the margin) in the Unit of Measurement (e.g., 11 inches).

Values Enter a valid height of zero or more. The maximum height depends upon the Unit of Measurement. For inches, the maximum height is 512 inches. For centimeters, it is 1,312 centimeters. For points, it is 36,864 points.

Applies to sections

Required/Optional required

Default 11 inches

Height restrictions

- The page must be large enough to contain the section. For example, if a frame in a section expands to a size larger than the Height, the report will not run.
- If you change the Unit of Measurement after you have already specified the Height, the Height will automatically change to the approximate height in the new Unit of Measurement.
- If you change the Width or Height of the physical page, the size of the logical page will change accordingly, so that the dimensions of the margin are preserved wherever possible.

Caution: On some printers the printable area of the physical page is restricted. For example, the sheet of paper a printer takes may be 8.5 x 11 inches, but the printer might only be able to print on an area of 8 x 10.5 inches. If you define a Width and Height that is bigger than the printable area your printer allows, clipping may occur in your report output. To avoid clipping, you can either increase the printable area for the printer (if your operating system allows it) or you can set the Width and Height to be the size of the printable area of the page.

- You can override the dimensions of a page by specifying different values for the PAGESIZE parameter on the command line.
- The default top and bottom margin is half an inch each.

Horizontal Panels per Page

Description The Horizontal Panels per Page property is the width of the logical page (report page) in physical pages (printer pages). For instance, a Horizontal Panels per Page size of 1 means that each logical page is one physical page wide.

Values Any whole number from 1 through 50.

Applies to sections

Required/Optional required

Default 1

Horizontal/Vertical Panels per Page example

Panels are the representations of the physical pages within a logical page. In this example, one logical page is made up of six physical pages. The logical page is three physical pages wide and two physical pages high. Consequently, Horizontal Panels per Page size is 3 and Vertical Panels per Page size is 2. If you wanted the logical page to be two physical pages wide and three physical pages high, you would specify a Horizontal Panels per Page size of 2 and a Vertical Panels per Page size of 3. See the online help for details and illustrations.

Orientation

Description The Orientation property specifies whether the section output is portrait or landscape. This setting overrides orientation specified by the ORIENTATION command line parameter, system parameter, or Page Setup dialog box.

Values

- Default uses the orientation defined by the ORIENTATION command line parameter, system parameter, or Page Setup dialog box.
- Portrait section output in vertical.
- Landscape section output is horizontal.

Applies to sections

Required/Optional optional

Default Default

Report Height

Description The Report Height property is the height of your report page in character units.

Values Any whole number from 1 through 9,999

Applies to sections

Required/Optional required

Default 80 characters

Report Height restrictions

- When displaying a character-mode report in the bit-mapped Live Previewer view, Report Builder uses this setting to choose the font to display. The font may be slightly larger or smaller than the actual character cell size. As a result, you may see some truncation.

Report Width

Description The Report Width property is the width of your report page in character units.

Values Any whole number from 1 through 9,999

Applies to sections

Required/Optional required

Default 66 characters

Report Width restrictions

- When displaying a character mode report in the bit-mapped Live Previewer view, Report Builder uses this setting to choose the font to display. The font may be slightly larger or smaller than the actual character cell size. As a result, you may see some truncation.

Vertical Panels per Page

Description The Vertical Panels per Page property is the height of the logical page (report page) in physical pages (printer pages). For instance, a Vertical Panels per Page size of *1* means that each logical page is one physical page in height.

Values Any whole number from 1 through 50.

Applies to sections

Required/Optional required

Default 1

Width

Description The Width property is the width of one physical page (including the margin) in the Unit of Measurement (e.g., 8.5 inches).

Values Enter a valid width of zero or more. The maximum width depends upon the Unit of Measurement. For inches, the maximum width is 512 inches. For centimeters, it is 1,312 centimeters. For points, it is 36,864 points.

Applies to sections

Required/Optional required

Default 8.5 inches

Width restrictions

- The page must be large enough to contain the section. For example, if a frame in a section expands to a size larger than the Width, the report will not run.
- If you change the Unit of Measurement after you have already specified the Width, the Width will automatically change to the approximate width in the new Unit of Measurement.
- If you change the Width or Height of the physical page, the size of the logical page will change accordingly, so that the dimensions of the margin are preserved wherever possible.

Caution: On some printers the printable area of the physical page is restricted. For example, the sheet of paper a printer takes may be 8.5 x 11 inches, but the printer might only be able to print on an area of 8 x 10.5 inches. If you define a Width and Height that is bigger than the printable area your printer allows, clipping may occur in your report output. To avoid clipping, you can either increase the printable area for the printer (if your operating system allows it) or you can set the Width and Height to be the size of the printable area of the page.

- You can override the dimensions of a page by specifying different values for the PAGESIZE parameter on the command line.
- The default right and left margin size is zero.

Summary Column properties

Break Order
Column Type
Comment
Compute At
Datatype
Function
File Format
Name
Product Order
Read from File
Reset At
Set Break Order
Source
Value If Null
Width

Compute At

Description The Compute At property is the group over which a *% of Total* summary column is computed. Compute At is used only for columns with a Function of *% of Total*. When you calculate a percentage, you divide a value by a total (e.g., SMITH's salary/total department salaries). Compute At defines the total for a percentage calculation. Using this field, Report Builder determines the total of which each source column value is a percentage. For matrix reports, Compute At can be multiple groups.

Values

Page	Is used to calculate the source column as a percentage of the total values on a page.
Report	Is used to calculate the source column as a percentage of the total values in the report.
<names>	Is used to calculate the source column as a percentage of the total values in the group(s).

Applies to summary columns

Required/Optional required, for summary columns with a Function of *% of Total*.

Default The parent group of the summary column's group

Compute At example (group report)

Suppose that you have a report that looks like the one below:

Deptno	10	Dname	ACCOUNTING		
Ename		Sal	%Dept	%Total	
MILLER		1300.00	14.86%	6.62%	
CLARK		2450.00	28.00%	12.48%	
KING		5000.00	57.14%	25.48%	

Deptno	20	Dname	RESEARCH		
Ename		Sal	%Dept	%Total	
SMITH		800.00	7.36%	4.08%	
ADAMS		1100.00	10.11%	5.61%	
JONES		2975.00	27.36%	15.16%	
SCOTT		3000.00	27.59%	15.29%	
FORD		3000.00	27.59%	15.29%	

The parent group, G_DEPT, contains the DEPTNO and DNAME columns. The child group, G_EMP, contains the ENAME and SAL columns.

To get a summary that calculates each employee's salary as a percentage of total department salaries (Pct_Dept), you create a summary column in G_EMP with the following settings:

Function	Reset At	Compute At
<i>% of Total</i>	G_EMP	G_DEPT

In this case, Reset At indicates that the summary should be reset to null for each employee. This is because you want a separate percentage for each individual employee. Compute At indicates that the total of salaries for the department is the whole (100%) across which to compute the percentages. It should also be noted that this summary is placed in the G_EMP group because you want the percentage to appear for each employee.

To get a summary that calculates each employee's salary as a percentage of total salaries for the entire report (Pct_Total), you create a summary column in G_EMP with the following settings:

Function	Reset At	Compute At
<i>% of Total</i>	G_EMP	Report

In this case, Compute At indicates that the total of salaries for the entire report is the whole (100%) across which to compute the percentages. As with the Pct_Dept summary above, Reset At indicates that the summary should be reset to null for each employee and the summary is placed in the G_EMP group because you want the percentage to appear for each employee.

Compute At example (matrix report)

Suppose that you have a matrix report that looks something like the one below and you want to add some percentage summaries to it.

Dept	Analyst	Clerk	Manager	Total
10		1300	2450	3750
20	6000	1900	2975	10875
30		950	2850	3800
Total	6000	4150	8275	18425

Group G_JOB contains the column JOB, G_DEPT contains DEPTNO, G_SAL contains SUMSAL, and G_CROSS is the cross-product group. (G_CROSS contains all of the summaries for the matrix.)

To get a summary that calculates each job category's total salaries per department as a percentage of total salaries for that job in all departments, you would enter the following settings for a summary column in G_CROSS:

Function	Reset At	Compute At	Product Order
<i>% of Total</i>	G_DEPT	G_JOB	G_JOB, G_DEPT

In this case, Reset At indicates that the summary should be reset to null for each department. This is because you want a separate percentage for each individual department. Compute At indicates that the total of salaries for the job category is the whole (100%) across which to compute the percentages. It should also be noted that this summary is placed in G_CROSS because all summaries that apply to the matrix itself must belong to the cross-product group.

Note: Because this is a matrix report, you need to use Product Order for the summary. For further information about Product Order, see "Product Order". (One of the examples uses the same output as this example and more fully explains the use of Product Order.)

See the online help for details and illustrations.

Compute At restrictions

- Compute At is only editable when the column is a summary and the Function is *% of Total*.
- The Compute At group of a summary column may be any group above the summary column's group, including Report and Page, if the summary column's group is not Report.
- Summary columns that are owned by a cross-product group cannot have a Compute At of *Page*.
- If a summary has a Compute At of *Page*, then the sizing of the field or boilerplate object that displays it must be fixed.

Function

Description The Function property is the computation to be performed on the values of the column specified in Source. To create running summaries, you place the column in the appropriate group and use Function in conjunction with Reset At.

Values

Average	Calculates the average of the column's values within the reset group.
Count	Counts the number of records within the reset group.
First	Prints the column's first value fetched for the reset group.
Last	Prints the column's last value fetched for the reset group.
Maximum	Calculates the column's highest value within the reset group.
Minimum	Calculates the column's lowest value within the reset group.
% of Total	Calculates the column's percent of the total within the reset group.
Std. Deviation	Calculates the column's positive square root of the variance for the reset group.
Sum	Calculates the total of the column's values within the reset group.
Variance	Sums the squares of each column value's distance from the mean value of the reset group and divides the total by the number of values minus 1.

Applies to summary columns

Required/Optional required, if column is a summary column.

Default blank

Function restrictions

- Function is only editable when the Type of the column is Summary.
- *Sum*, *Average*, *% of Total*, *Std. Deviation*, and *Variance* can only be applied to fields with a Datatype of *Number*.
- Report Builder sets the Datatype for the summary based on the Datatype of the column being summarized.
- If the column being summarized is changed, the datatype of the summary changes to match.

- A report-level summary cannot have a Function of *% of Total*. Such a column would make no sense because it would always be 100%.
- A column with a Function of *% of Total, Sum, or Variance* cannot be a break column and cannot have Break Order set.
- The *Average* function will ignore null values in a column (i.e., it does not assume a null value equals zero).

Product Order

Description The Product Order property is the order in which groups are evaluated in the cross product for a summary. Product Order also defines the frequency of a summary, formula, or placeholder in a cross product group. That is, the summary, formula, or placeholder has one value for each combination of values of the groups in its Product Order. Product Order is used only for columns owned by cross-product groups. Because a cross product relates multiple groups, the groups in the cross product could be evaluated in any one of many different orders. Therefore, when creating a summary for a cross product, you must use Product Order to specify for Report Builder which group should be evaluated first, which second, and so on. You must also use Product Order to specify the frequency of a summary, formula, or placeholder within the cross product.

Values A valid group combination.

Applies to summary, formula, and placeholder columns

Required/Optional required, if the column is a summary or formula column belonging to a cross product group.

Product Order example (single group)

Suppose that you have a matrix report.

See the online help for details and illustrations.

Group G_JOB contains the column JOB, G_DEPT contains DEPTNO, G_YEAR contains YEAR, G_SAL contains SUMSAL, and G_CROSS is the cross product group. (G_CROSS contains all of the summaries for the matrix.)

To get the summary of salaries by job, you create a summary column in G_CROSS with the following settings:

Function	Reset At	Product Order
<i>% of Total</i>	G_JOB	G_JOB

In this case, you want the summary to calculate a sum for each record of G_JOB (i.e., each job category) so Product Order is G_JOB. (This summary appears along the bottom of the matrix.)

To get the summary of salaries by year, you create a summary column in G_CROSS with the following settings:

Function	Reset At	Product Order
<i>Sum</i>	G_YEAR	G_YEAR

In this case, you want the summary to calculate a sum for each record of G_YEAR (i.e., each year) so Product Order is G_YEAR. (This summary appears along the right edge of the matrix directly underneath a boilerplate line.)

To get the summary of salaries for each department in each year, you create a summary column in G_CROSS with the following settings:

Function	Reset At	Product Order
-----------------	-----------------	----------------------

To get a summary that calculates each department's total salaries per job category as a percentage of total salaries in the report, you would enter the following settings for a summary column in G_CROSS:

Function	Reset At	Compute At	Product Order
% of Total	G_JOB	Report	G_DEPT, G_JOB

In this case, the summary could go across the matrix and then down, or down the matrix and then across because you are computing this summary for the entire report (e.g., Compute At is *Report*). Product Order could be either G_DEPT, G_JOB (across, down) or G_JOB, G_DEPT (down, across). Note, however, that Reset At must be the group that has the highest frequency: G_DEPT if Product Order is G_JOB, G_DEPT; G_JOB if Product Order is G_DEPT, G_JOB.

See the online help for details and illustrations.

Product Order example (formula)

Suppose that you have a matrix report that looks something like the one below and you want to add a formula to it.

Dept	Analyst	Clerk	Manager	President	Salesman
10		1300	2450	5000	
20	6000	1900	2975		
30		950	2850		5600

Group G_JOB contains the column JOB, G_DEPT contains DEPTNO, G_SAL contains SUMSAL, and G_CROSS is the cross product group.

To add a formula that calculates the sum of after-tax salaries for each department, you first create a summary column (DEPTSUM) in G_CROSS with the following settings:

Function	Reset At	Product Order
% of Total	G_DEPT	G_DEPT

Next, you create a formula (AFTERTAX) in G_CROSS with the following settings:

Formula	Product Order
:deptsum * .9	G_DEPT

Because the formula is in G_CROSS, you must give it a Product Order. In this case, the frequency of the formula is the same as the summary column, DEPTSUM. This would calculate the formula as follows:

Dept	Analyst	Clerk	Manager	President	Salesman	Comp
10		1300	2450	5000		7875
20	6000	1900	2975			9788
30		950	2850		5600	8460

Product Order restrictions

- This field is only available if the summary, formula, or placeholder column's group is a cross product.

- A summary which is not a running summary (i.e., its *Reset At* is not *Report*) must have a Product Order that is a subset of its source column's Product Order. The source column of the summary must be at least as frequent as the summary, which means that the summary's Product Order should be some subset of the source column's Product Order.
- A running summary (i.e., its *Reset At* is *Report*) must have a Product Order that is a prefix of the source column's Product Order. For example, if the source column's Product Order were A,B,C, then the summary column's Product Order would have to be either A (by itself) or A,B. The summary's Product Order could not be B,A.

Reset At

Description The Reset At property is the group at which the summary column value resets to zero (if Function is *Count*), null (if Function is not *Count*), or Value If Null (if the column has one). Reset At has a list of values containing valid reset groups. Reset At determines if the summary is a running summary or a periodic (e.g., group-level) summary.

Values

Page	Is used for page-level summaries. The summary is reset between pages.
Report	Is used for summaries that apply to the entire report, such as grand totals. The summary is reset after the report (i.e., it is never reset).
<names>	Is a valid group name. The summary is reset after each record of the group.

Applies to summary columns

Required/Optional required, for columns of Type Summary.

Usage Notes

- The reset group of a summary column may be its group or any group above its group, including Report and Page.
- To ensure that page summaries behave in a predictable way, make sure each record fits on a single page, or if that is not feasible, force each new record to print on a new page.

Reset At example (group)

Suppose that you want to create a group report like the one below:

DEPT	NAME	SAL	RESETSUM	RUNSUM
10	SMITH	1000	1000	1000
	JONES	1000	2000	2000
	KING	1000	3000	3000
20	JOHNSON	1500	1500	4500
	WARD	1000	2500	5500

The parent group, G_DEPT, contains the DEPTNO column. The child group, G_EMP, contains ENAME and SAL. To get the sum of salaries for each department (RESETSUM), you create a summary column in G_EMP with the following settings

Function	Reset At
<i>Sum</i>	G_DEPT

In this case, **Reset At** indicates that the summary should be set to null after each department.

To get the running sum of salaries (RUNSUM), you create a summary column in G_EMP with the following settings:

Function	Reset At
<i>Sum</i>	<i>Report</i>

In this case, **Reset At** indicates that the summary should not be reset to null but should accumulate throughout the report.

Reset At restrictions

- **Reset At** is only editable for summary columns.
- Summary columns that are owned by a cross-product group cannot have a **Reset At** of *Page*.
- If a summary has a **Reset At** of *Page*, then the sizing of the field or boilerplate object that displays it must be fixed.
- If a column has a **Reset At** of *Page*, then any fields that use it as a source must be contained in repeating frames that use the column's group as their source.
- A **Reset At** of *Page* will not work for reports that have multiple repeating frames associated with a group that is involved in the summary. One common case of this is a matrix report. As a result, matrix reports cannot have page-level summaries.
- For multiple summary columns that are in child groups that share the same parent group, a **Reset At** of *Page* may produce incorrect results.
- If your data model contains multiple child groups with a shared parent (e.g., master-detail-detail), summary columns with a **Reset At** of *Page* may give unexpected results.

Source

Description The Source property is the name of the column whose values will be summarized.

Values Any valid column name.

Applies to summary columns

Required/Optional required, if the column is a summary column.

Default blank

Source restrictions

- The source of a summary column may be any column within or below the summary column's group (with a valid Datatype for the function).
- If you change the Source of a summary column, the Datatype of the summary column changes to reflect the Datatype of the new source column, unless the column has Read from File specified.
- Any column can be the Source of multiple summary columns.
- If you specify a source for a summary column and later delete the source column from your report, your summary column no longer has access to data to summarize.

Template properties

Alignment
Align Summaries with Fields
Background Color
Between Field and Labels (Horizontal)
Between Frame and Fields (Horizontal)
Between Frame and Fields (Vertical)
Between Master and Detail (Horizontal)
Between Master and Detail (Vertical)
Between Page and Frames (Horizontal)
Between Page and Frames (Vertical)
Between Sibling Frames (Horizontal)
Between Sibling Frames (Vertical)
Borders
Character Justification
Dash
Date Justification
Edge Background Color
Edge Foreground Color
Edge Pattern
Fields Per Line
Fill Pattern
Font
Foreground Color
Image
Inter-Field (Horizontal)
Inter-Field (Vertical)
Inter-Frame (Horizontal)
Inter-Frame (Vertical)
Justify
Number Justification
Place Labels Above Fields
Position
Style
Text
Text Color
Use Vertical Spacing

Alignment

Description The Alignment property defines how labels and fields on a line are positioned across the page. This property applies only if the Style is *Form*. The number of fields and columns per line is determined by the Fields Per Line property.

Values

Left	Means the left-most label or field is aligned with the left side of the page. See the online help for details and illustrations.
Right	Means the right-most label or field is aligned with the right side of the page. See the online help for details and illustrations.
Center	Means the labels and fields are centered across the page. See the online help for details and illustrations.
Flush	Means the labels and fields are spaced evenly across the page, with the left-most label or field aligned with the left side of the page and the right-most label or field aligned with the right side of the page. See the online help for details and illustrations.
Column	Means labels and fields are aligned in columns spaced evenly across the page, based on the widths of the largest label and largest field. See the online help for details and illustrations.

Applies to templates

Default Left

Align Summaries with Fields

Description The Align Summaries with Fields property places summary fields under their source fields, if set to *Yes*. This property applies only if the Style is *Form*. See the online help for details and illustrations.

Applies to templates

Default Yes

Background Color

Description The Background Color property defines the background color of the Fill Pattern property setting.

Values Choose a valid color from the color palette.

Applies to templates

Default white

Usage Notes

- If the Fill Pattern property is set to *transparent*, then both the Background Color and Foreground Color settings are ignored.
- If the Fill Pattern property is set to *solid*, then the Background Color setting is ignored, and only the Foreground Color setting is visible.

PL/SQL To define this attribute using PL/SQL, use the `SRW.SET BACKGROUND FILL COLOR` procedure.

Between Field and Labels (Horizontal)

Description The Between Field and Labels (Horizontal) property defines the amount of horizontal space between fields and their associated labels in a group. This property applies only if the Style is *Form*, in which labels are to the left of fields. Thus, there is no vertical space counterpart to this property.

Values Enter a value of zero or more. The value will be in the Unit of Measurement for the report (inches, centimeters, or points).

Applies to templates

Default 0

Usage Notes

- If Grid Snap is on, objects will snap to the nearest grid snap points, regardless of this setting. If you want a greater level of spacing control, you can turn off Grid Snap when you create the report. However, this means that objects will no longer snap to the grid snap points.
- If you change the Unit of Measurement for the report, you'll need to change this setting accordingly.

Between Frame and Fields (Horizontal)

Description The Between Frame and Fields (Horizontal) property defines the amount of horizontal space between frames and the fields they enclose.

Values Enter a value of zero or more. The value will be in the Unit of Measurement for the report (inches, centimeters, or points).

Applies to templates

Default 0

Usage Notes

- If Grid Snap is on, objects will snap to the nearest grid snap points, regardless of this setting. If you want a greater level of spacing control, you can turn off Grid Snap when you create the report. However, this means that objects will no longer snap to the grid snap points.
- If you change the Unit of Measurement for the report, you'll need to change this setting accordingly.

Between Frame and Fields (Vertical)

Description The Between Frame and Fields (Vertical) property defines the amount of vertical space between frames and the fields they enclose.

Values Enter a value of zero or more. The value will be in the Unit of Measurement for the report (inches, centimeters, or points).

Applies to templates

Default 0

Usage Notes

- If Grid Snap is on, objects will snap to the nearest grid snap points, regardless of this setting. If you want a greater level of spacing control, you can turn off Grid Snap when you create the report. However, this means that objects will no longer snap to the grid snap points.
- If you change the Unit of Measurement for the report, you'll need to change this setting accordingly.

Between Master and Detail (Horizontal)

Description The Between Master and Detail (Horizontal) property defines the amount of horizontal space between the master group frame and the detail group frame.

Values Enter a value of zero or more. The value will be in the Unit of Measurement for the report (inches, centimeters, or points).

Applies to templates

Default 0

Usage Notes

- If Grid Snap is on, objects will snap to the nearest grid snap points, regardless of this setting. If you want a greater level of spacing control, you can turn off Grid Snap when you create the report. However, this means that objects will no longer snap to the grid snap points.
- If you change the Unit of Measurement for the report, you'll need to change this setting accordingly.

Between Master and Detail (Vertical)

Description The Between Master and Detail (Vertical) property defines the amount of vertical space between the master group frame and the detail group frame.

Values Enter a value of zero or more. The value will be in the Unit of Measurement for the report (inches, centimeters, or points).

Applies to templates

Default 0

Usage Notes

- If Grid Snap is on, objects will snap to the nearest grid snap points, regardless of this setting. If you want a greater level of spacing control, you can turn off Grid Snap when you create the report. However, this means that objects will no longer snap to the grid snap points.
- If you change the Unit of Measurement for the report, you'll need to change this setting accordingly.

Between Page and Frames (Horizontal)

Description The Between Page and Frames (Horizontal) property defines the amount of horizontal space between the edge of the page and the highest-level group frame.

Values Enter a value of zero or more. The value will be in the Unit of Measurement for the report (inches, centimeters, or points).

Applies to templates

Default 0

Usage Notes

- If Grid Snap is on, objects will snap to the nearest grid snap points, regardless of this setting. If you want a greater level of spacing control, you can turn off Grid Snap when you create the report. However, this means that objects will no longer snap to the grid snap points.
- If you change the Unit of Measurement for the report, you'll need to change this setting accordingly.

Between Page and Frames (Vertical)

Description The Between Page and Frames (Vertical) property defines the amount of vertical space between the edge of the page and the highest-level group frame.

Values Enter a value of zero or more. The value will be in the Unit of Measurement for the report (inches, centimeters, or points).

Applies to templates

Default

Usage Notes

- If Grid Snap is on, objects will snap to the nearest grid snap points, regardless of this setting. If you want a greater level of spacing control, you can turn off Grid Snap when you create the report. However, this means that objects will no longer snap to the grid snap points.
- If you change the Unit of Measurement for the report, you'll need to change this setting accordingly.

Between Sibling Frames (Horizontal)

Description The Between Sibling Frames (Horizontal) property defines the amount of horizontal space between the frames of groups at the same level. The groups may be separate, or they may share the same parent group.

Values Enter a value of zero or more. The value will be in the Unit of Measurement for the report (inches, centimeters, or points).

Applies to templates

Default 0

Usage Notes

- If Grid Snap is on, objects will snap to the nearest grid snap points, regardless of this setting. If you want a greater level of spacing control, you can turn off Grid Snap when you create the report. However, this means that objects will no longer snap to the grid snap points.
- If you change the Unit of Measurement for the report, you'll need to change this setting accordingly.

Between Sibling Frames (Vertical)

Description The Between Sibling Frames (Vertical) property defines the amount of vertical space between the frames of groups at the same level. The groups may be separate, or they may share the same parent group.

Values Enter a value of zero or more. The value will be in the Unit of Measurement for the report (inches, centimeters, or points).

Applies to templates

Default 0

Usage Notes

- If Grid Snap is on, objects will snap to the nearest grid snap points, regardless of this setting. If you want a greater level of spacing control, you can turn off Grid Snap when you create the report. However, this means that objects will no longer snap to the grid snap points.
- If you change the Unit of Measurement for the report, you'll need to change this setting accordingly.

Borders

Description The Borders property defines which borders to display for objects.

Values

All	Means display all borders.
Top Only	Means display top borders only.
Bottom Only	Means display bottom borders only.
Left Only	Means display left borders only.
Right Only	Means display right borders only.
Top and Bottom	Means display top and bottom borders only.
Left and Right	Means display left and right borders only.
Top and Right	Means display top and right borders only.
Top and Left	Means display top and left borders only.
Bottom and Right	Means display bottom and right borders only.
Bottom and Left	Means display bottom and left borders only.
All but Top	Means display all except top borders.
All but Bottom	Means display all except bottom borders.
All but Left	Means display all except left borders.
All but Right	Means display all except right borders.
None	Means display no borders.

Applies to templates

Default All

Character Justification

Description The Character Justification property defines the method to use for aligning the text of the field or label, if the datatype of the field's Source is CHARACTER.

Values

Left	Means the text is aligned left.
Center	Means the text is centered.
Right	Means the text is aligned right.
Start	Means the text is aligned left or right depending on where it would normally start for the language. For example, for languages that flow from left to right, Start justifies the text Left. Alternatively, for languages that flow from right to left, Start justifies the text Right.
End	Means the text is aligned left or right depending on where it would normally end for the language. For example, for languages that flow from left to right, End justifies the text Right. Alternatively, for languages that flow from right to left, End justifies the text Left.

Applies to templates

Default Start

PL/SQL To define this attribute using PL/SQL, use the `SRW.SET JUSTIFICATION` procedure.

Dash

Description The Dash property defines the line/dash style for the object.

Values Choose a valid line/dash pattern from the list.

Applies to templates

Default Solid

Date Justification

Description The Date Justification property defines the method to use for aligning the text of the field or label, if the datatype of the field's Source is DATE.

Values

Left	Means the text is aligned left.
Center	Means the text is centered.
Right	Means the text is aligned right.
Start	Means the text is aligned left or right depending on where it would normally start for the language. For example, for languages that flow from left to right, Start justifies the text Left. Alternatively, for languages that flow from right to left, Start justifies the text Right.
End	Means the text is aligned left or right depending on where it would normally end for the language. For example, for languages that flow from left to right, End justifies the text Right. Alternatively, for languages that flow from right to left, End justifies the text Left.

Values Enter

Applies to templates

Default Start

PL/SQL To define this attribute using PL/SQL, use the `SRW.SET JUSTIFICATION` procedure.

Edge Foreground Color

Description The Edge Foreground Color defines the foreground color of the Edge Pattern property setting.

Values Choose a valid color from the color palette.

Applies to templates

Default black

Usage Notes

- If the Edge Pattern property is set to *transparent*, then both the Edge Foreground Color and Edge Background Color settings are ignored.
- If the Edge Pattern property is set to *solid*, then only the Edge Foreground Color setting is visible (the Edge Background Color setting is ignored).

PL/SQL To define this attribute using PL/SQL, use the `SRW.SET FOREGROUND BORDER COLOR` procedure.

Edge Background Color

Description The Edge Background Color property defines the background color of the Edge Pattern property setting.

Values Choose a valid color from the color palette.

Applies to templates

Default white

Usage Notes

- If the Edge Pattern property is set to *transparent*, then both the Edge Background Color and Edge Foreground Color settings are ignored.
- If the Edge Pattern property is set to *solid*, then the Edge Background Color setting is ignored, and only the Edge Foreground Color setting is visible.

PL/SQL To define this attribute using PL/SQL, use the `SRW.SET BACKGROUND BORDER COLOR` procedure.

Edge Pattern

Description The Edge Pattern property defines the pattern to use for the borders of the objects. You can define the background and foreground colors of the edge pattern using the Edge Foreground Color and Edge Background Color properties.

Values Choose a valid pattern from the pattern palette.

Applies to templates

Default transparent

Usage Notes

- If you set the Edge Pattern to *transparent*, the Edge Foreground Color and Edge Background Color property settings are ignored.
- If you set the Edge Pattern to *solid*, then only the Edge Foreground Color property setting is visible.

PL/SQL To define this attribute using PL/SQL, use the `SRW.SET BORDER PATTERN` procedure.

Fields Per Line

Description The Fields Per Line property defines the maximum number of field/label combinations that may appear on a line, space permitting. For example, if there is only space for two field/label combinations, and the value is set to three, then only two field/label combinations will appear. However, if there is space for four field/label combinations and the value is set to three, then only three will appear since three is the maximum allowed. This property applies only if the Style is *Form*.

Values Enter a value of zero or more. Zero means that there is no maximum value, and as many field/label combinations as will fit on a line will appear.

Applies to templates

Default 0

Fill Pattern

Description The Fill Pattern property defines the pattern to use for the space enclosed by the objects. You can define the background and foreground colors of the fill pattern using the Foreground Color and Background Color properties.

Values Choose a valid pattern from the pattern palette.

Applies to templates

Default transparent

Usage Notes

- If you set the Fill Pattern to *transparent*, the Foreground Color and Background Color property settings are ignored.
- If you set the Fill Pattern to *solid*, then only the Foreground Color property setting is visible.

PL/SQL To define this attribute using PL/SQL, use the `SRW.SET FILL PATTERN` procedure.

Font

Description The Font property defines the text font to use for the object(s).

Values Choose a valid font from the list of fonts for your system.

Applies to templates

Default Courier, 10 pt., Regular

PL/SQL To define the font face using PL/SQL, use the `SRW.SET FONT FACE` procedure. To define the font size using PL/SQL, use the `SRW.SET FONT SIZE` procedure. To define the font style using PL/SQL, use the `SRW.SET FONT STYLE` procedure.

Foreground Color

Description The Foreground Color defines the foreground color of the Fill Pattern property setting.

Values Choose a valid color from the color palette.

Applies to templates

Default black

Usage Notes

- If the Fill Pattern property is set to *transparent*, then both the Foreground Color and Background Color settings are ignored.
- If the Fill Pattern property is set to *solid*, then only the Foreground Color setting is visible (the Background Color setting is ignored).

PL/SQL To define this attribute using PL/SQL, use the `SRW.SET FOREGROUND FILL COLOR` procedure.

Image

Description The Image property is the substitution file to use for images referenced in link file (external) boilerplate objects in your templates.

Values Enter the name of a valid text file.

Applies to templates

Default none

Inter-Field (Horizontal)

Description The Inter-Field (Horizontal) property defines the amount of horizontal space between fields in a group. If the Style is *Form*, this property defines the amount of horizontal space between the end of one field/label combination and the beginning of another field/label combination in a group. The distance between the field and label is determined by the Between Field and Labels (Horizontal) property setting.

Values Enter a value of zero or more. The value will be in the Unit of Measurement for the report (inches, centimeters, or points).

Applies to templates

Default .174 inches

Usage Notes

- If Grid Snap is on, objects will snap to the nearest grid snap points, regardless of this setting. If you want a greater level of spacing control, you can turn off Grid Snap when you create the report. However, this means that objects will no longer snap to the grid snap points.
- If you change the Unit of Measurement for the report, you'll need to change this setting accordingly.

Inter-Field (Vertical)

Description The Inter-Field (Vertical) property defines the amount of vertical space between fields in a group.

Values Enter a value of zero or more. The value will be in the Unit of Measurement for the report (inches, centimeters, or points).

Applies to templates

Default 0

Usage Notes

- If Grid Snap is on, objects will snap to the nearest grid snap points, regardless of this setting. If you want a greater level of spacing control, you can turn off Grid Snap when you create the report. However, this means that objects will no longer snap to the grid snap points.
- If you change the Unit of Measurement for the report, you'll need to change this setting accordingly.

Inter-Frame (Horizontal)

Description The Inter-Frame (Horizontal) property defines the amount of horizontal space between frames in a group.

Values Enter a value of zero or more. The value will be in the Unit of Measurement for the report (inches, centimeters, or points).

Applies to templates

Default .0087 inches

Usage Notes

- If Grid Snap is on, objects will snap to the nearest grid snap points, regardless of this setting. If you want a greater level of spacing control, you can turn off Grid Snap when you create the report. However, this means that objects will no longer snap to the grid snap points.
- If you change the Unit of Measurement for the report, you'll need to change this setting accordingly.

Inter-Frame (Vertical)

Description The Inter-Frame (Vertical) property defines the amount of vertical space between frames in a group.

Values Enter a value of zero or more. The value will be in the Unit of Measurement for the report (inches, centimeters, or points).

Applies to templates

Default .0136 inches

Usage Notes

- If Grid Snap is on, objects will snap to the nearest grid snap points, regardless of this setting. If you want a greater level of spacing control, you can turn off Grid Snap when you create the report. However, this means that objects will no longer snap to the grid snap points.
- If you change the Unit of Measurement for the report, you'll need to change this setting accordingly.

Justify

Description The Justify property defines the method to use for aligning the text in the title.

Values

Left	Means the text is aligned left.
Center	Means the text is centered.
Flush	Means the text is aligned both left and right.
Right	Means the text is aligned right.
Start	Means the text is aligned left or right depending on where it would normally start for the language. For example, for languages that flow from left to right, Start justifies the text Left. Alternatively, for languages that flow from right to left, Start justifies the text Right.
End	Means the text is aligned left or right depending on where it would normally end for the language. For example, for languages that flow from left to right, End justifies the text Right. Alternatively, for languages that flow from right to left, End justifies the text Left.

Applies to templates

Default Start

Number Justification

Description The Number Justification property defines the method to use for aligning the text of the field or label, if the datatype of the field's Source is NUMBER.

Values

Left	Means the text is aligned left.
Center	Means the text is centered.
Right	Means the text is aligned right.
Start	Means the text is aligned left or right depending on where it would normally start for the language. For example, for languages that flow from left to right, Start justifies the text Left. Alternatively, for languages that flow from right to left, Start justifies the text Right.
End	Means the text is aligned left or right depending on where it would normally end for the language. For example, for languages that flow from left to right, End justifies the text Right. Alternatively, for languages that flow from right to left, End justifies the text Left.

Applies to templates

Default Start

PL/SQL To define this attribute using PL/SQL, use the `SRW.SET JUSTIFICATION` procedure.

Place Labels Above Fields

Description The Place Labels Above Fields property positions labels above fields, rather than to the left of fields, if set to *Yes*. This property applies only if the Style is *Form*.

Applies to templates

Default No

Position

Description The Position property defines the placement of the parent object in relation to the child object.

Applies to templates

Default The parent object is displayed above the child object.

Style

Description The Style property defines whether the group is given a Tabular or a Form style layout. You can use this property to override the defaults, listed below.

Values

Tabular	Means that labels are above fields, and are outside the repeating frame containing the fields. See the online help for details and illustrations.
Form	Means that labels are to the left of fields, and are inside the repeating frame containing the fields. See the online help for details and illustrations.

Applies to templates

Default The default Style is determined by the layout style for the template, as follows:

Tabular	Tabular
Group Left	Tabular
Group Above	Form, for groups with a child group. Tabular, for groups with no child group.
Form-like	Form
Form Letter	Not applicable (cannot override).
Mailing Label	Not applicable (cannot override).
Matrix	Tabular (cannot override).
Matrix Break	Tabular (cannot override).

Text

Description The Text property is the substitution file to use for boilerplate text objects in templates.

Values Enter the name of a valid text file.

Applies to templates

Default none

Text Color

Description The Text Color property defines the text color to use for the object(s).

Values Choose a valid color from the color palette.

Applies to templates

Default black

PL/SQL To define this attribute using PL/SQL, use the `SRW.SET COLOR TEXT` procedure.

Use Vertical Spacing

Description The Use Vertical Spacing property places objects into unused spaces on previous lines, if set to Yes. For example, if an object on a subsequent line can fit into unused space on a previous line, the object is moved up to occupy the unused space.

See the online help for details and illustrations.
This property applies only if the Style is *Form*.

Applies to templates

Default Yes

XML Properties

XML Tag
XML Tag Attributes
Exclude from XML Output
XML Prolog Type
XML Prolog Value
Outer XML Tag
Outer XML Attributes
Contains XML Tags

XML Tag

Description The XML Tag property contains the element name of the currently selected report, group or column, and may be user-defined. You may choose any name for the element, as the XML tag's purpose is to uniquely identify the object as a particular type or category of data. One reason for generating XML output is to configure objects according to external DTD specifications, which permits integration of your report with a third-party application.

Applies to report, group, or column

Required/Optional Optional

Default Object name (as displayed in the Name property)

Usage Notes

- Can contain any characters in a string of any length.
- The name of the XML element is dependent on the name of the object in the Property Palette. Changing the Name property will change the XML Tag property to match.

If you do not wish to output XML for this object but do wish to output child tags and data, leave this property blank. For more information, see the XML Tag example.

XML Tag Example

Suppose you want to create XML output for the report, group(s), and columns of your query. A query statement such as:

```
SELECT deptno, dname, loc FROM dept
```

in a tabular report this will produce the following XML output:

```
<MODULE1>
<LIST_G_DEPTNO>
  <G_DEPTNO>
    <DEPTNO>10</DEPTNO>
    <DNAME>ACCOUNTING</DNAME>
    <LOC>NEW YORK</LOC>
  </G_DEPTNO>
  <G_DEPTNO>
    <DEPTNO>20</DEPTNO>
    <DNAME>RESEARCH</DNAME>
    <LOC>DALLAS</LOC>
  </G_DEPTNO>
  <G_DEPTNO>
    <DEPTNO>30</DEPTNO>
    <DNAME>SALES</DNAME>
    <LOC>CHICAGO</LOC>
  </G_DEPTNO>
</G_DEPTNO>
```

```

        <DEPTNO>40</DEPTNO>
        <DNAME>OPERATIONS</DNAME>
        <LOC>BOSTON</LOC>
    </G_DEPTNO>
</LIST_G_DEPTNO>
</MODULE1>

```

If you wish to suppress various tags from appearing in the output, simply delete the default name in the XML Tag property field, leaving it blank. This will eliminate tag output, but will not exclude any data associated with child tags. For instance, if you were to delete the G_DEPTNO tag, the XML output would read as follows:

```

<MODULE1>
  <LIST_>
    <DEPTNO>10</DEPTNO>
    <DNAME>ACCOUNTING</DNAME>
    <LOC>NEW YORK</LOC>
    <DEPTNO>20</DEPTNO>
    <DNAME>RESEARCH</DNAME>
    <LOC>DALLAS</LOC>
    <DEPTNO>30</DEPTNO>
    <DNAME>SALES</DNAME>
    <LOC>CHICAGO</LOC>
    <DEPTNO>40</DEPTNO>
    <DNAME>OPERATIONS</DNAME>
    <LOC>BOSTON</LOC>
  </LIST_>
</MODULE1>

```

Note that the tag formerly called <LIST_G_DEPTNO> is now called <LIST_>. This is called an Outer XML Tag and is only associated with group tags. The outer XML tag name is selected from the group XML tag name. If you wish to remove the <LIST_> tag as well, you may leave the Outer XML Tag field blank. See Outer XML Tag Property for more information.

XML Tag Attributes

Description The XML Tag Attributes property enables you to enter XML attributes. The purpose of adding an attribute to an object's tag is to further describe or pinpoint the data value for that tag. Using tag attributes is one method of organizing your XML output. If you use this method, you need to enter a value for this property. The other method is to assign a separate tag for each type (group or column) of data. In this case, you would leave the property blank.

Applies to report, group, or column

Required/Optional Optional

Default Blank

Syntax XML attributes must be entered as space-separated, name-value pairs. For example:

```
XML Tag      [attribute name]="&[report object name]"
Attributes:
```

where the text within square brackets ([]) is chosen by you. The report object name can be any column or user parameter in the report. You may add as many attributes to this property field as you wish by leaving a space between each one.

Usage Notes

- Any data object added to another XML tag as an attribute should itself be excluded from XML output to minimize redundancy. See XML Tag Attributes example (group level) for more information.
- Attributes must refer to columns or variables that occur at the same or lower frequency

XML Tag Attributes Example (group-level)

Suppose your tabular report uses the following query:

```
SELECT * FROM DEPT
```

Your default XML output looks like the following:

```
<MODULE2>
  <LIST_G_DEPTNO>
    <G_DEPTNO>
      <DEPTNO>10</DEPTNO>
      <DNAME>ACCOUNTING</DNAME>
      <LOC>NEW YORK</LOC>
    </G_DEPTNO>
  <G_DEPTNO>
    <DEPTNO>20</DEPTNO>
```

```

    <DNAME>RESEARCH</DNAME>
    <LOC>DALLAS</LOC>
  </G_DEPTNO>

```

You want to generate XML output that folds all data associated with a given department number (DEPTNO) into each department number's <DEPTNO> element. This enables you to reduce output elements from many to few.

From the data model, specify the following properties for the DEPTNO column:

```

XML Tag:          DEPTNO
XML Tag          NAME="&DNAME" LOCATION="&LOC"
Attributes:

```

where &DNAME is a department name and &LOC is the location of that department. Both objects repeat at the same frequency as DEPTNO.

The new XML output resembles the following:

```

<G_DEPTNO>
  <DEPTNO NAME="ACCOUNTING" LOCATION="NEW YORK">10</DEPTNO>
  <DNAME>ACCOUNTING</DNAME>
  <LOC>NEW YORK</LOC>
</G_DEPTNO>
<G_DEPTNO>
  <DEPTNO NAME="RESEARCH" LOCATION="DALLAS">20</DEPTNO>
  <DNAME>RESEARCH</DNAME>
  <LOC>DALLAS</LOC>
</G_DEPTNO>

```

As you can see, values for DNAME and LOC are now output twice: once as attributes and once as XML elements. In order to eliminate them as XML elements, you must also use the Exclude from XML Output property.

From the data model, set the Exclude from XML Output property to Yes for the DNAME and LOC columns:

The resulting XML output will look like this:

```

<G_DEPTNO>
  <DEPTNO NAME="ACCOUNTING" LOCATION="NEW YORK">10</DEPTNO>
</G_DEPTNO>
<G_DEPTNO>
  <DEPTNO NAME="RESEARCH" LOCATION="DALLAS">20</DEPTNO>
</G_DEPTNO>
<G_DEPTNO>
  <DEPTNO NAME="SALES" LOCATION="CHICAGO">30</DEPTNO>
</G_DEPTNO>
<G_DEPTNO>
  <DEPTNO NAME="OPERATIONS" LOCATION="BOSTON">40</DEPTNO>
</G_DEPTNO>

```


XML Tag Attributes Example (report-level)

The following example demonstrates how a report-level attribute might be useful in XML.

Suppose your report uses the following query:

```
select deptno, job, ename, hiredate from emp
```

You have designed your report as group-left, with DEPTNO as the first-level group and JOB as the second-level group. For the purpose of this example, you have selected the DEPTNO and JOB columns in the data model and set the Break Order property to Descending. The arrows to the left of the DEPTNO and JOB columns in the data model should now point down.

Your data model looks like the following:

(See the on-line help to view the data model)

Your XML output begins as follows:

```
<MODULE2>
  <LIST_G_DEPTNO>
    <G_DEPTNO>
      <DEPTNO>30</DEPTNO>
      <LIST_G_JOB>
        <G_JOB>
          <JOB>SALESMAN</JOB>
          <LIST_G_ENAME>
            <G_ENAME>
              <ENAME>ALLEN</ENAME>
              <HIREDATE>20-FEB-81</HIREDATE>
            </G_ENAME>
            <G_ENAME>
              <ENAME>WARD</ENAME>
              <HIREDATE>22-FEB-81</HIREDATE>
            </G_ENAME>
            <G_ENAME>
              <ENAME>MARTIN</ENAME>
              <HIREDATE>28-SEP-81</HIREDATE>
            </G_ENAME>
          </LIST_G_ENAME>
        </G_JOB>
      </LIST_G_JOB>
    </G_DEPTNO>
  </LIST_G_DEPTNO>
</MODULE2>
```

For later reference, note that within department number 30, the first three employee's hiredate ranges from 20-FEB-81 to 28-SEP-81.

Next you add a user parameter to your report that will select or exclude data according to hiredate. Double-click the query (Q_1) in the data model.

In the SQL Query Statement dialog box, add the following text at the end of the existing query:

```
where hiredate > :p_date
```

so that now your query reads:

```
Select deptno, job, ename, hiredate from emp where hiredate > :p_date
```

This SQL statement directs Report Builder to create the parameter named p_date.

However, you must manually change the Datatype property to Date.

1. In the Object Navigator, find user parameters under Data Model. Click the plus sign to the left of User Parameters.
2. Double-click the object P_DATE.
3. In the Property Palette, click the Datatype property field and select Date from the drop-down menu.

Next, you want to assign this user parameter as an attribute to the report element. The parameter will provide the rule for inclusion or exclusion of various elements throughout the report's XML output.

Double-click the report node in the Object Navigator.

In the Property Palette, enter `HiredateLaterThan="&p_date"` in the XML Tag Attributes property field.

Run the report in Live Previewer. A Runtime Parameter Form will appear, asking for a date value (P Date). Enter `1-JUL-81` and click the run button.

Your XML now looks like the following:

```
<MODULE2 HiredateLaterThan="01-JUL-81">
  <LIST_G_DEPTNO>
    <G_DEPTNO>
      <DEPTNO>30</DEPTNO>
      <LIST_G_JOB>
        <G_JOB>
          <JOB>SALESMAN</JOB>
          <LIST_G_ENAME>
            <G_ENAME>
              <ENAME>MARTIN</ENAME>
              <HIREDATE>28-SEP-81</HIREDATE>
            </G_ENAME>
          </LIST_G_ENAME>
        </G_JOB>
      </LIST_G_JOB>
    </G_DEPTNO>
  </LIST_G_DEPTNO>
</MODULE2>
```

Note that instances of the group G_ENAME with HIREDATE values before the date 1-July-81 are no longer available in XML. The attribute which defines this parameter, `HiredateLaterThan="01-JUL-81"`, now appears within the report element, MODULE2.

Exclude from XML Output

Description The Exclude from XML Output property enables you to suppress both the XML tag and associated data output, including child data, for the selected group or column. If you choose to exclude an object, it will not output the XML tag, attributes, data values and child data, e.g., columns within groups. This property is useful for reformatting the XML document to exclude data that may be deemed redundant or unnecessary.

Applies to report, group, and column.

Required/Optional Optional

Default No

Values

Yes

No

Usage Notes

- This property is especially useful when you have added an object as an attribute to another element and no longer wish for that object's data to appear as a separate element in XML output.
- If you set the report-level Exclude from XML Output property to Yes, you will be unable to generate any XML from the report, as all data is a child of the report.

Exclude from XML Output Example (limit data output)

Suppose you want to create XML output that limits the data to a few, specific elements. Currently, your XML output includes Department Number, Department Name, and Location. It looks like the following:

```
<G_DEPTNO>
  <DEPTNO>10</DEPTNO>
  <DNAME>ACCOUNTING</DNAME>
  <LOC>NEW YORK</LOC>
</G_DEPTNO>
<G_DEPTNO>
  <DEPTNO>20</DEPTNO>
  <DNAME>RESEARCH</DNAME>
  <LOC>DALLAS</LOC>
</G_DEPTNO>
<G_DEPTNO>
  <DEPTNO>30</DEPTNO>
  <DNAME>SALES</DNAME>
  <LOC>CHICAGO</LOC>
</G_DEPTNO>
<G_DEPTNO>
  <DEPTNO>40</DEPTNO>
```

```

    <DNAME>OPERATIONS</DNAME>
    <LOC>BOSTON</LOC>
  </G_DEPTNO>

```

You decide to remove the <LOC> tag and its associated data. From the Data Model view, double-click the LOC column and specify the following property in the Property Palette:

Exclude from XML Yes
Output:

The XML output will now look like the following:

```

<LIST_G_DEPTNO>
  <G_DEPTNO>
    <DEPTNO>10</DEPTNO>
    <DNAME>ACCOUNTING</DNAME>
  </G_DEPTNO>
  <G_DEPTNO>
    <DEPTNO>20</DEPTNO>
    <DNAME>RESEARCH</DNAME>
  </G_DEPTNO>
  <G_DEPTNO>
    <DEPTNO>30</DEPTNO>
    <DNAME>SALES</DNAME>
  </G_DEPTNO>
  <G_DEPTNO>
    <DEPTNO>40</DEPTNO>
    <DNAME>OPERATIONS</DNAME>
  </G_DEPTNO>
</LIST_G_DEPTNO>

```

If you wish to remove the <LIST_G_DEPTNO> or <G_DEPTNO> tag without excluding child data (such as DEPTNO or DNAME), select the group G_DEPTNO from the data model and open its Property Palette. Delete the XML tag G_DEPTNO from the XML Tag property field and delete the LIST_G_DEPTNO XML tag from the Outer XML Tag property field. See Outer XML Tag Property for more information.

Exclude from XML Output Example (limit redundancy)

Suppose you have decided to use attributes in your XML output which now looks like this:

```

<G_DEPTNO>
  <DEPTNO NAME="ACCOUNTING" LOCATION="NEW YORK">10</DEPTNO>
  <DNAME>ACCOUNTING</DNAME>
  <LOC>NEW YORK</LOC>
</G_DEPTNO>
<G_DEPTNO>
  <DEPTNO NAME="RESEARCH" LOCATION="DALLAS">20</DEPTNO>
  <DNAME>RESEARCH</DNAME>

```

```
<LOC>DALLAS</LOC>
</G_DEPTNO>
```

In order to eliminate redundancy, you will need to exclude the DNAME and LOC elements from XML output.

From the Data Model view, display the Property Palette for the DNAME and LOC columns and specify the following property:

Exclude from XML output: Yes

The resulting XML output will look like the following:

```
<G_DEPTNO>
  <DEPTNO NAME="ACCOUNTING" LOCATION="NEW YORK">10</DEPTNO>
</G_DEPTNO>
<G_DEPTNO>
  <DEPTNO NAME="RESEARCH" LOCATION="DALLAS">20</DEPTNO>
</G_DEPTNO>
<G_DEPTNO>
  <DEPTNO NAME="SALES" LOCATION="CHICAGO">30</DEPTNO>
</G_DEPTNO>
<G_DEPTNO>
  <DEPTNO NAME="OPERATIONS" LOCATION="BOSTON">40</DEPTNO>
</G_DEPTNO>
```

XML Prolog Type

Description The XML Prolog Type property enables you to choose between an external prolog file and a text prolog generated by Report Builder. The XML prolog can include XML declaration, comments, processing instructions (PI), and a document type declaration (DTD). For information on editing the prolog, see XML Prolog Value.

Applies to report

Required/Optional Required.

Default Text

Values

- | | |
|------|--|
| Text | Enter prolog as text in the XML Prolog Value property |
| File | Navigate to external prolog file using the XML Prolog Value property |

XML Prolog Value

Description The XML Prolog Value property enables you to edit the values of the XML Prolog that should appear at the beginning of all well-formed XML output. The XML prolog should include an XML declaration but can also include comments, processing instructions (such as a stylesheet [XSL]), and a DTD.

Applies to report

Required/Optional An XML declaration is required for well-formed XML output. However, any alpha-numeric character in this property field will allow raw XML output to be generated in Report Builder.

Default `<?xml version="1.0"?>`

Syntax examples

XML	<code><?xml version="1.0" encoding="UTF-8"?></code>
Declaration:	
Comments:	<code><!-- this is how you do comments in XML --></code>
Processing instruction (PI):	<code><?xml:stylesheet type="text/xsl" href="emp.xsl" ?></code>
Document Type Declaration (DTD):	<code><!DOCTYPE EMP_TABULAR [<!ELEMENT EMP_TABULAR (EMP_ITEM)*> <!ELEMENT EMP_ITEM (EMPNO)> <!ELEMENT EMPNO (#PCDATA)> ></code>

XML Prolog Value Example (prolog type: text)

Suppose you want to include the prolog as text at the beginning of your XML output. With a report open in Report Builder, select the report icon in the Object Navigator and display the Property Palette. Set the following property:

XML Prolog Text
Type:

Next, click once in the XML Prolog Value property field to view the ellipsis button. Click this button to view the XML Prolog Value dialog box. Type the prolog text and click OK.

The following is an example of prolog text which contains (in order) an XML declaration, a comment, a processing instruction (PI), and a document type declaration (DTD):

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- This is how you make comments in XML -->
<?xml:stylesheet type="text/xsl" href="emp.xsl" ?>
<!DOCTYPE EMP_TABULAR [
<!ELEMENT EMP_TABULAR (EMP_ITEM)*>
<!ELEMENT EMP_ITEM (EMPNO)>
<!ELEMENT EMPNO (#PCDATA)>
]
```

```
<!ELEMENT EMPNO (#PCDATA)>
]>
```

XML Prolog Value Example (prolog type: file)

Suppose you want to have your prolog in an external file.

With a report open in Report Builder, set the following property at the report level:

XML Prolog File

Type

Next, click once in the XML Prolog Value property field to view the ellipsis button.

Click this button to navigate to an external file containing prolog text.

The following is an example of prolog text which contains (in order) an XML declaration, a comment, a processing instruction (PI), and a document type declaration (DTD):

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- This is how you make comments in XML -->
<?xml:stylesheet type="text/xsl" href="emp.xsl" ?>
<!DOCTYPE EMP_TABULAR [
<!ELEMENT EMP_TABULAR (EMP_ITEM)*>
<!ELEMENT EMP_ITEM (EMPNO)>
<!ELEMENT EMPNO (#PCDATA)>
]>
```

Outer XML Tag

Description The Outer XML Tag property field displays the name of the selected group, with a LIST_ prefix. The purpose of the outer XML tag is to enclose all tags and data from instances of a group object. In a report, you may have repeating instances of a group object (e.g. deptno, which outputs 10, 20, 30, etc.). The outer XML tag simply encapsulates data for all repeating instances.

Applies to group

Required/Optional Optional

Default <LIST_G_[groupname]>

Usage Note

- You may leave this field blank, but this might only be useful if you have only one instance of a group in your report's query.

Outer XML Tag Example

Suppose you have a report which contains the following query:

```
SELECT * from EMP
```

You have decided to view this data as a group-left report with DEPTNO as the first-level group and JOB as the second-level group.

For the purposes of this example, you wish to reverse the order of output for the groups DEPTNO and JOB, from ascending to descending.

1. From the data model, select the column DEPTNO and open the Property Palette.
2. Change the Break Order property from Ascending to Descending.
3. Repeat the first two steps for the column JOB.

Your data model will look like the following:

See the on-line help for a view of the data model.

You now want to view the hierarchical structure of your XML output for groups in your report. Since you have made the group G_JOB a child of group G_DEPTNO, you decide to collapse all elements within the <LIST_G_DEPTNO> outer group in order to expand the structure of data one level at a time.

```
- <MODULE1>
  - <LIST_G_DEPTNO>
    + <G_DEPTNO>
    + <G_DEPTNO>
    + <G_DEPTNO>
  </LIST_G_DEPTNO>
</MODULE1>
```

In the XML example above, you can see that for the group G_DEPTNO (represented by the outer tag <LIST_G_DEPTNO>) there are three unique instances of DEPTNO data. Now you wish to see the next-level group, G_JOB, which is a child of G_DEPTNO. If

you expand the first XML tag `<G_DEPTNO>` you will see the following:

```
- <MODULE1>
  - <LIST_G_DEPTNO>
    - <G_DEPTNO>
      <DEPTNO>30</DEPTNO>
      + <LIST_G_JOB>
        </G_DEPTNO>
      + <G_DEPTNO>
        + <G_DEPTNO>
          <LIST_G_DEPTNO>
        </MODULE1>
```

Just as `<LIST_G_DEPTNO>` contains repeating instances of `<G_DEPTNO>`, `+ <LIST_G_JOB>` contains repeating instances of `<G_JOB>`. If you expand the element `+ <LIST_G_JOB>`, you will see the following XML output in its place:

```
- <LIST_G_JOB>
  + <G_JOB>
  + <G_JOB>
  + <G_JOB>
- </LIST_G_JOB>
```

If you expand the first `G_JOB` element (`+<G_JOB>`), you will see the following XML output in its place:

```
- <G_JOB>
  <JOB>SALESMAN</JOB>
  + <LIST_G_EMPNO>
  </G_JOB>
```

As you can see, `<G_JOB>` contains instances of another repeating group, `<G_EMPNO>`. If you expand `+<LIST_G_EMPNO>`, instances of `<G_EMPNO>` appear. In this case, there will be no further child groups, but there are a number of column values available for the `G_EMPNO` group. The XML output looks like the following:

```
- <LIST_G_EMPNO>
  - <G_EMPNO>
    <EMPNO>7499</EMPNO>
    <ENAME>ALLEN</ENAME>
    <MGR>7698</MGR>
    <HIREDATE>20-FEB-81</HIREDATE>
    <SAL>1600</SAL>
    <COMM>300</COMM>
  </G_EMPNO>
```

`</LIST_G_EMPNO>`

The outer element `<LIST_[groupname]>` is useful as a container for repeating groups.

Outer XML Attributes

Description The Outer XML Attributes property enables you to enter XML attributes for an outer XML tag. The purpose of adding an attribute to an outer tag is to further describe or pinpoint the data value for that tag.

Applies to group

Required/Optional Optional

Default Blank

Syntax Outer XML attributes must be entered as space-separated, name-value pairs (as with HTML attributes). For example:

```
Outer XML Tag   [attribute name]="&[report object name]"
Attributes:
```

where the text within square brackets ([]) is chosen by you. You may add as many attributes to this property as you wish by leaving a space between each one.

Usage Notes

- Keep in mind that any data object added to another XML tag as an attribute should itself be excluded from XML output to minimize redundancy. See Exclude from XML Output for more information.
- Attributes must refer to columns or variables that occur at a lower frequency.

Outer XML Attributes Example

Suppose your report uses the following query:

```
SELECT deptno, job, ename, sal FROM EMP
```

You have decided to view this data as a group-left report with DEPTNO as the first-level group and JOB as the second-level group.

For the purposes of this example, you wish to reverse the order of output for the groups DEPTNO and JOB, from ascending to descending.

1. From the data model, select the column DEPTNO and open the Property Palette.
2. Change the Break Order property from Ascending to Descending.
3. Repeat the first two steps for the column JOB.

Your data model will look like the following:

See the on-line help for a view of the data model.

Currently, your XML output looks like the following:

```
<MODULE1>
  <LIST_G_DEPTNO>
```

```

<G_DEPTNO>
  <DEPTNO>30</DEPTNO>
  <LIST_G_JOB>
  <G_JOB>
    <JOB>SALESMAN</JOB>
  <LIST_G_ENAME>
    <G_ENAME>
      <ENAME>ALLEN</ENAME>
      <SAL>1600</SAL>
    </G_ENAME>
    <G_ENAME>
      <ENAME>WARD</ENAME>
      <SAL>1250</SAL>
    </G_ENAME>
  </LIST_G_ENAME>
</G_JOB>
</LIST_G_JOB>
</G_DEPTNO>

```

where one unique value for <JOB> applies to at least two occurrences of <G_ENAME>. Specifically, at least two employees (ALLEN and WARD) hold the job of SALESMAN. Instead of nesting the job title (<JOB>) in a separate XML tag on a separate line, you want to fold this unique value into the <LIST_G_ENAME> element since the value for <JOB> will occur once per occurrence of <LIST_G_ENAME>.

From the data model view, specify the following properties:

Group Object	Property	Action	Value
G_ENAME	Outer XML Attributes	Type:	JOBTITLE="&JOB"
JOB	Exclude from XML Output:	Select:	Yes
G_JOB	XML Tag	Delete:	G_JOB

Your XML output now resembles the following:

```

<MODULE1>
  <LIST_G_DEPTNO>
  <G_DEPTNO>
    <DEPTNO>30</DEPTNO>
    <LIST_G_JOB>
    <LIST_G_ENAME JOBTITLE="SALESMAN">
      <G_ENAME>
        <ENAME>ALLEN</ENAME>
        <SAL>1600</SAL>
      </G_ENAME>
      <G_ENAME>
        <ENAME>WARD</ENAME>
        <SAL>1250</SAL>
      </G_ENAME>
    </LIST_G_ENAME>
  </G_JOB>
</LIST_G_JOB>
</G_DEPTNO>

```

</G_ENAME>

Contains XML Tags

Description The Contains XML Tags property indicates whether or not a report column contains XML tags. If you set the property to Yes, Report Builder does not convert XML syntax-related characters, such as <, >, and & to <, > and & respectively.

Applies to column with a source datatype of character

Required/Optional Optional

Default No

Contains XML Tags Example

Suppose your database already has XML tags included in the values of a column. In the first instance below, the XML tag DESTINATION represents a user-specified parameter called &p_dest. If the Contains XML Tags property is set to Yes, Report Builder will recognize <DESTINATION> as an XML tag, interpret &p_dest as a changing value according to user specifications, and substitute a unique value in the place of &p_dest in XML output. This is true for any value in a column with the syntax of "&[report object name]".

Variable column value with XML tag:

Original Text String in column:	<DESTINATION>&p_dest<DESTINATION>
Contains XML Tags = Yes:	<DESTINATION>FIJI<DESTINATION>
Contains XML Tags = No:	<DESTINATION>&p_dest<DESTINATION>

Non-variable column value with XML tag:

Original Text String in column:	<JOB>Clerk<JOB>
Contains XML Tags = Yes:	<JOB>Clerk<JOB>
Contains XML Tags = No:	<JOB>Clerk<JOB>

Executables

Executable names

Letters are appended to the executable names listed above to indicate if the GUI mode of the executable is something other than the native GUI. For example, "a" indicates a GUI mode of Apple MacIntosh and "m" a GUI mode of Motif. (No added letter means the executable's GUI mode is the native GUI.) In addition, an "x" is used to indicate if user exits are linked to the executable.

Executable invocation

Executables can be invoked in one of two ways:

- by command line
- by double-clicking the executable's icon

Executable Arguments When you invoke an executable, you can enter executable arguments that govern how the executable behaves. If you invoke the executable via command line, you enter the executable arguments on the command line after the executable name.

Help on command line options

If you are on a command line, you can view an executable's command line options by typing the executable's name, then a space and a question mark, and then pressing Return. For example, `rwrrun60 "?"`.

Note: Some operating systems require quotes around the question mark. For example, to view the command line options for the RWRUN60 executable in the UNIX C shell, you need to enter `rwrrun60 "?"`.

Executable syntax restrictions

- When you enter an executable on the command line, it generally has the form shown below:
`executable_name keyword1=value1 keyword2=value2 ...`
- If your operating system does not have a command line available (e.g., Apple Macintosh or Windows) and you invoke the executable via icon, an intermediary dialog box may appear where you can enter values for the executable arguments or you may need to use the Runtime Values and Runtime Parameters tab on the Preferences dialog box (**Tools**→**Preferences**) for executable arguments.
- Each keyword=value is called an executable argument. Because it would be cumbersome to enter the keywords each time, they may be omitted.
- No spaces should be placed before or after the equal sign of an argument.
- Separate arguments with one or more spaces; do not use commas to separate arguments.
- Values of arguments may be in single or double quotes. The effect of single or double quotes is operating system-specific.
- The keyword= part of all arguments is not case sensitive. The value portion of the SOURCE, DEST, CMDFILE, TERM, DESNAME, DESFORMAT, DNAME, and SNAME arguments may be case sensitive, depending on your operating system. The value portion of all other arguments is not case sensitive.
- Keywords may sometimes be omitted.
- To pass a single quote from the command line, you must enter two quotes (one quote as an escape and one as the actual quote). For example:
`rwrrun60 module=myrep destype=file desname=run.out batch=yes
p_value="Roy's Batch Report"`

To pass a parameter value with a single quote in it using the Forms RUN_PRODUCT packaged procedure, you must enter four single quotes (PL/SQL

collapses two quotes to one and so does Reports). For example, if you wanted to pass the following literal value to an Reports' parameter:

```
where '='
```

you would enter the following string in RUN_PRODUCT:

```
'where ''='' '''
```

- To give an argument an empty string as its value, you enter keyword = or keyword="". If you enter an empty string in this way, the Runtime Parameter Form will display an empty string for the argument instead of its Initial Value.
- Full pathnames are supported for all file references (e.g., `desname=/revenues/q1/nwsales`). If you do not specify the full path, the file path search order is used to find the file. **Note:** To help ensure the portability of your reports, you can use the environment variable `REPORTS30_PATH` for referencing reports, external queries, external PL/SQL libraries, text files, graphics, etc. `REPORTS30_PATH` enables you to access files stored outside of Report Builder or Reports Runtime without hardcoding directory paths into your report.
- Values entered from the Runtime Parameter Form override those entered on the command line. For example, if you specified `RWRUN60` on the command line with `COPIES` equal to 1, but, in the Runtime Parameter Form, specified `COPIES` equal to 2, then two copies of the report would be generated.

Similarly, values entered on the command line override values specified in command files. For example, suppose that you specify `RWRUN60` from the command line with `COPIES` equal to 1 and `CMDFILE` equal to `RUNONE` (a command file). In `RUNONE`, `COPIES` is set to 2 by a `RWRUN60` command. Only one copy of the report would be generated in this case.

You can specify values for `DESTYPE`, `DESNAME`, `DESFORMAT`, `ORIENTATION`, and `COPIES` in a number of different places. The following list shows the order of precedence for the places where you specify these values. The first item in the list takes precedence over all others, the second takes precedence over all but the first item, and so on:

- Print Job dialog box
- Runtime Parameter Form
- Runtime Parameters/Settings tab of Tools Options or Runtime Options dialog box
- keywords on the command line
- values specified in the report definition

- Choose Printer dialog

Keyword usage

The keyword= part of each argument is optional as long as the arguments are entered in the order specified by the syntax. If you skip one of the arguments, you must use keywords after the skipped argument.

If an executable has four arguments and you want to specify only the first, second, and fourth arguments, you can specify the first two without the keywords. The fourth would require a keyword, though, because you are not entering it in the order specified by the syntax.

Examples (Valid):

Below are examples of valid command line sequences:

```
rwrun60 userid=scott/tiger paramform=no destype=mail
       desname=rmiller
rwrun60 scott/tiger no destype=mail desname=rmiller
```

Notice that, in this example, after the keyword DESTYPE is used all subsequent arguments use keywords, too. Once you have entered a keyword, the rest of the arguments on the command line must also use keywords.

Examples (Invalid):

Below are examples of invalid command line sequences:

```
rwrun60 paramform=no scott/tiger destype=mail
       desname=rmiller
```

The sequence above is invalid because scott/tiger has been typed out of order and without the USERID keyword.

```
rwrun60 scott/tiger mail rmiller no
```

The sequence above is invalid because the value NO has been entered out of order and without the PARAMFORM keyword.

Database login

You can connect to a database in the following ways:

- Explicit login
- Automatic login
- Remote login

Explicit login

You explicitly login to the database by using the `USERID` keyword, or by choosing **File**→**Connect**.

USERID

Description USERID is your ORACLE username and password with an optional database name, SQL*Net communication protocol to access a remote database, or ODBC datasource name (if accessing a non-Oracle datasource). If you omit your password, a logon screen is provided. For more information about the optional SQL*Net communication protocol, see "Remote Login". For more information about automatic login, see "Automatic Login".

Values

The logon definition must be in one of the following forms and cannot exceed 512 bytes in length:

`username[/password]`

`username[/password][@database]`

`[user[/password]]@ODBC:datasource[:database] or [user[/password]]@ODBC:*`

Automatic login

The ORACLE automatic login feature allows you to connect to the database without explicitly stating a username and password.

When you connect in this way, ORACLE uses a default logon (e.g., OPS\$) to associate your ORACLE username to your operating system username. To invoke an automatic login procedure, do one of the following:

- Type `rwconv60 /` on the command line.
- Type a slash (/) in the logon screen as your username.
- Omit your username/password on the command line and press the Return or Enter key twice when the logon form appears.

Note: When logging on in this way, you can access a remote login by typing `@database` after the slash.

For more information on default logons, see the *ORACLE8 Server SQL Language Reference Manual*.

Remote login

Through SQL*Net, you can specify a remote database (i.e., one not on your computer). When you use the remote login feature, the remote database is transparent to you. For more information on SQL*Net, refer to your SQL*Net documentation.

Usage Notes

- To perform a remote connect, you must have an account on the remote computer node where the target database resides.

RWBLD60

Description RWBLD60 invokes Report Builder, to create and maintain reports.

Syntax

```
RWBLD60 [ [keyword=]value ] ...
```

where keyword=value is a valid command line argument.

RWBLD60 Command Line Arguments

```
[MODULE | REPORT=] modulename
[USERID=] userid
[PARAMFORM=] {YES | NO}
[CMDFILE=] filename
[ARRAYSIZE=] n
[DESTYPE=] {SCREEN | FILE | PRINTER | PREVIEW | MAIL}
[DESNAME=] desname
[DESFORMAT=] desformat
[COPIES=] n
[CURRENCY=] currency_symbol
[THOUSANDS=] thousands_symbol
[DECIMAL=] decimal_symbol
[READONLY=] {YES | NO}
[BUFFERS=] n
[PAGESIZE=] width x height
[PROFILE=] profiler_file
[RUNDEBUG=] {YES | NO}
[ONSUCCESS=] {COMMIT | ROLLBACK | NOACTION}
[ONFAILURE=] {COMMIT | ROLLBACK | NOACTION}
[ERRFILE=] error_file
[LONGCHUNK=] n
[ACCESS=] {file | database}
[ORIENTATION=] {DEFAULT | LANDSCAPE | PORTRAIT}
[BACKGROUND=] {YES | NO}
[MODE=] {BITMAP | CHARACTER | DEFAULT}
[PRINTJOB] {YES | NO}
[TRACEFILE=] tracefile
[TRACEMODE=] {TRACE_APPEND | TRACE_REPLACE}
[TRACEOPTS=] {TRACE_ERR | TRACE_PRF | TRACE_BRK | TRACE_APP |
TRACE_PLS | TRACE_SQL | TRACE_TMS | TRACE_DST | TRACE_ALL | (opt1,
opt2, ...)}
[AUTOCOMMIT=] {YES | NO}
[NONBLOCKSQL=] {YES | NO}
[ROLE=] rolename / [rolepassword]
[BLANKPAGES=] {YES | NO}
[MAXIMIZE=] {YES | NO}
[DISABLEPRINT=] {YES | NO}
[DISABLEMAIL=] {YES | NO}
[DISABLEFILE=] {YES | NO}
[DISABLENEW=] {YES | NO}
[DISTRIBUTE=] {YES | NO}
[DESTINATION=] filename.DST
```

```
[CACHELOB=]{YES|NO}
[DELIMITER=]value
[CELLWRAPPER=]value
[NUMBERFORMATMASK=]mask
[DATEFORMATMASK=]mask
[PAGESTREAM=]{YES|NO}
EXPRESS_SERVER="server=[server]/domain=[domain]/user=[userid
]/password=[passwd]"
[CUSTOMIZE]=filename.xml | (filename1.xml, filename2.xml, .
. .)
[SAVE_RDF]=filename.rdf
<param>=value
```

MODULE|REPORT (RWBLD60)

Description MODULE|REPORT is a module (report, external query, or external PL/SQL library) that you want to open in Report Builder. (REPORT is allowed for backward compatibility.) If you do not specify a report using MODULE|REPORT, Report Builder will start up with a new report definition open.

Syntax

```
[MODULE|REPORT=]runfile
```

Values

Any valid report, external query, or external PL/SQL library name. Report Builder will look in the location specified with the ACCESS keyword for the module. If ACCESS is not specified, Report Builder will use its file path search order. If the module is not found, Report Builder will start up without opening the specified module.

Usage Notes

- If you open a character mode report via bit-mapped RWBLD60, Report Builder displays a warning, then opens the report using a page size of 8.5" x 11" and a form size of 7" x 6".

PARAMFORM (RWBLD60)

Description PARAMFORM specifies whether to display the Runtime Parameter Form when you execute a report.

Syntax

[PARAMFORM=] {YES|NO}

Values

YES means to display the Form.

NO means to suppress the Form.

Default

YES

CMDFILE (RWBLD60)

Description CMDFILE is a file that contains arguments for the RWBLD60 command. This option enables you to run a report without having to specify a large number of arguments each time you invoke RWBLD60.

Syntax

```
[CMDFILE=]cmdfile
```

Values

Any valid command file.

Usage Notes

- A command file may reference another command file.
- Command file syntax for RWBLD60 arguments is identical to that used on the command line.
- Values entered on the command line override values specified in command files. For example, suppose that you specify RWBLD60 from the command line with COPIES equal to 1 and CMDFILE equal to RUNONE (a command file). In RUNONE, COPIES is set to 2. Only one copy of the report would be generated in this case. For more information, see Executable Syntax Restrictions .
- The argument(s) for this keyword may be operating system-specific.

ARRAYSIZE (RWBLD60)

Description ARRAYSIZE is the size of the array in kilobytes for use with ORACLE array processing.

Syntax

[ARRAYSIZE=]n

Values

A number from 1 through 9,999. This means that Report Builder can use this number of kilobytes of memory per query in your report.

Default

The default arraysize is 10. For details on ORACLE array processing, see the Oracle8 Server Administrator's Guide.

DESTYPE (RWBLD60)

Description DESTYPE is the type of destination device that will receive the report output.

Syntax

[DESTYPE=]{SCREEN|FILE|PRINTER|PREVIEW|MAIL}

Values

SCREEN	Routes the output to the Previewer.
FILE	Saves the output to a file named in DESNAME.
PRINTER	Routes the output to the printer named in DESNAME.
PREVIEW	Routes the output to the Previewer for interactive viewing. However, Preview causes the output to be formatted as PostScript output. The Previewer will use DESNAME to determine which printer's fonts to use to display the output.
MAIL	Sends the output to the mail IDs specified in DESNAME. You can send mail to any mail system that is MAPI-compliant or has the service provider driver installed. The report is sent as an attached file.

Default

Taken from the Initial Value property of the DESTYPE parameter.

Usage Notes

- SYSOUT is **not** a valid option for DESTYPE while using RWBLD60. SYSOUT can only be used with RWRUN60 when BATCH=YES.
- If you set DESTYPE to PRINTER, FILE, or MAIL, the Print Job dialog box will appear after you accept the Runtime Parameter Form, unless it is suppressed. After you accept the Print Job dialog box, the report is executed and the output is sent to the printer (specified in **File→Choose Printer**), file, or mail ID. (If DESTYPE is something other than PRINTER, FILE, or MAIL, then the report is executed after you accept the Runtime Parameter Form.) The appearance of the dialog box varies between operating systems.
- In some cases, this parameter may be overridden by your operating system.
- Some DESTYPE values are not supported on some operating systems.

DESNAME (RWBLD60)

Description DESNAME is the name of the file, printer, or email ID (or distribution list) to which the report output will be sent. To send the report output via email, specify the email ID as you do in your email application (any MAPI-compliant application on Windows or your native mail application on UNIX). You can specify multiple usernames by enclosing the names in parentheses and separating them by commas (e.g., (name, name, . . .name)).

Syntax

```
[DESNAME=]desname
```

Values

Any valid filename, printer name, or email ID not to exceed 1K in length. For printer names, you can optionally specify a port. For example:

```
DESNAME=printer,LPT1:
```

```
DESNAME=printer,FILE:
```

Default

Taken from the Initial Value property of the DESNAME parameter. If DESTYPE=FILE and DESNAME is an empty string, it defaults to `reportname.lis` at runtime.

Usage Notes

- This keyword is ignored if DESTYPE is SCREEN.
- If DESTYPE is PREVIEW, Report Builder uses DESNAME to determine which printer's fonts to use to display the output.
- The argument(s) for this keyword may be case sensitive, depending on your operating system.
- In some cases, this parameter may be overridden by your operating system.

DESFORMAT (RWBLD60)

Description In bit-mapped environments, DESFORMAT specifies the printer driver to be used when DESTYPE is FILE. In character-mode environments, it specifies the characteristics of the printer named in DESNAME.

Syntax

[DESFORMAT=]desformat

Values

Any valid destination format not to exceed 1K in length. Examples of valid values for this keyword are hpl, hplwide, dec, decwide, decland, decl80, dflt, wide, etc. Ask your System Administrator for a list of valid destination formats.

PDF	Means that the report output will be sent to a file that can be read by a PDF viewer. PDF output is based upon the currently configured printer for your system. The drivers for the currently selected printer is used to produce the output; you must have a printer configured for the machine on which you are running the report.
HTML	Means that the report output will be sent to a file that can be read by an HTML 3.0 compliant browser (e.g., Netscape 2.2).
HTMLCSS	Means that the report output sent to a file will include style sheet extensions that can be read by an HTML 3.0 compliant browser that supports cascading style sheets.
HTMLCSSIE	Means that the report output sent to a file will include style sheet extensions that can be read by Microsoft Internet Explorer 3.x.
RTF	Means that the report output will be sent to a file that can be read by standard word processors (such as Microsoft Word). Note that when you open the file in MS Word, you must choose View→Page Layout to view all the graphics and objects in your report.
DELIMITED	Means that the report output will be sent to a file that can be read by standard spreadsheet utilities, such as Microsoft Excel. Note that if you do not choose a delimiter, the default delimiter is a TAB..
XML	Means that the report output will be sent to a file that can be read by any XML-supporting application, including a fifth-generation or higher web browser.

Default

Taken from the Initial Value property of the DESFORMAT parameter. For bit-mapped Report Builder, if DESFORMAT is blank or dflt, then the current printer driver (specified in **File→Choose Printer**) is used. If nothing has been selected in Choose Printer, then Postscript is used by default.

Usage Notes

- This keyword is ignored if DESTYPE is SCREEN.
- The value(s) for this keyword may be case sensitive, depending on your operating system.

COPIES (RWBLD60)

Description COPIES is the number of copies of the report output to print.

Syntax

[COPIES=]n

Values

Any valid integer from 1 through 9,999.

Default

Taken from the Initial Value property of the COPIES parameter.

Usage Notes

- This keyword is ignored if DESTYPE is not Printer.
- If COPIES is left blank on the Runtime Parameter Form, it defaults to one.

CACHELOB (RWBLD60)

Description CACHELOB specifies whether to cache retrieved Oracle8 large object(s) in the temporary file directory (specified by `REPORTS60_TMP`).

Values

YES means to cache the LOB in the temporary file directory.

NO means to not cache the LOB in the temporary file directory.

Default

YES

Usage Notes

- You can only set this option on the command line.
- If the location of the temporary file directory does not have sufficient available disk space, it is preferable to set this value to NO. Setting the value to NO, however, may decrease performance, as the LOB may need to be fetched from the server multiple times.

CURRENCY (RWBLD60)

Description CURRENCY is the currency character to be used in number formats.

Syntax

[CURRENCY=]currency_symbol

Values

Any valid alphanumeric string not to exceed 1K in length.

Default

The default for ORACLE is determined by the ORACLE National Language Support facilities. You can also set a default of up to four characters in the Initial Value property of the CURRENCY parameter.

Usage Notes

- A CURRENCY value entered in Property Palette will override any CURRENCY value entered on the command line.

THOUSANDS (RWBLD60)

Description THOUSANDS is the thousands character to be used in number formats.

Syntax

[THOUSANDS=] thousands_symbol

Values

Any valid alphanumeric character.

Default

The default for ORACLE is determined by the ORACLE National Language Support facilities. You can also set a default of up to four characters in the Initial Value property of the THOUSANDS parameter.

Usage Notes

- A THOUSANDS value entered on the Parameter property sheet will override any THOUSANDS value entered on the command line.
- The alphanumeric character defined as the THOUSANDS value is the actual value that is returned. For example, if you define "," as the THOUSANDS value, "," is returned.

DECIMAL (RWBLD60)

Description DECIMAL is the decimal character to be used in number formats.

Syntax

```
[DECIMAL=]decimal_symbol
```

Values

Any valid alphanumeric character.

Default

The default for ORACLE is determined by the ORACLE National Language Support facilities. You can also set a default in the Initial Value property of the DECIMAL parameter.

Usage Notes

- A DECIMAL value entered on the Parameter property sheet will override any DECIMAL value entered on the command line.
- The alphanumeric character defined as the DECIMAL value is actual value that is returned. For example, if you define “.” as the DECIMAL value, “.” is returned.

READONLY (RWBLD60)

Description READONLY requests read consistency across multiple queries in a report. When accessing data from ORACLE, read consistency is accomplished by a SET TRANSACTION READ ONLY statement (refer to your Oracle8 Server SQL Language Reference Manual for more information on SET TRANSACTION READ ONLY).

Syntax

```
[READONLY=]{YES|NO}
```

Values

YES requests read consistency.

NO means do not provide read consistency.

Default

NO

Usage Notes

- This keyword is only useful for reports using multiple queries, because ORACLE automatically provides read consistency, without locking, for single query reports.

READONLY Restrictions

- In Report trigger order of execution , notice where the SET TRANSACTION READONLY occurs.

BUFFERS (RWBLD60)

Description BUFFERS is the size of the virtual memory cache in kilobytes. You should tune this setting to ensure that you have enough space to run your reports, but not so much that you are using too much of your system's resources.

Syntax

[BUFFERS=]n

Values

A number from 1 through 9,999. For some operating systems, the upper limit may be lower.

Default

640K

Usage Notes

- If this setting is changed in the middle of your session, the change does not take effect until the next time the report is run.

PAGESIZE (RWBLD60)

Description PAGESIZE is the dimensions of the physical page (i.e., the size of the page that the printer outputs). Note that the page must be large enough to contain the report. For example, if a frame in a report expands to a size larger than the page dimensions, the report will not run.

Syntax

```
[PAGESIZE=]width x height
```

Values

Any valid page dimensions of the form: page width x page height, where page width and page height are zero or more. The maximum width/height depends upon the unit of measurement. For inches, the maximum width/height is 512 inches. For centimeters, it is 1312 centimeters. For picas, it is 36,864 picas.

Default

For bitmap, 8.5 x 11 inches. For character mode, 80 x 66 characters. If the report was designed for character mode and is being run or converted on bitmap, the following formula is used to determine page size if none is specified: (default page size * character page size)/default character page size. For example, if the character page size is 80 x 20, the bit-mapped page size would be: $(8.5 * 80)/80 \times (11 * 20)/66 = 8.5 \times 3.33$.

Usage Notes

- On some printers the printable area of the physical page is restricted. For example, the sheet of paper a printer takes may be 8.5 x 11 inches, but the printer might only be able to print on an area of 8 x 10.5 inches. If you define a page width x page height in Report Builder that is bigger than the printable area your printer allows, clipping may occur in your report output. To avoid clipping, you can either increase the printable area for the printer (if your operating system allows it) or you can set the page width x page height to be the size of the printable area of the page.
- If this keyword is used, its value overrides the page dimensions of the report definition.
- A PAGESIZE value entered on the Runtime Parameter Form will override any PAGESIZE value entered on the command line.

PROFILE (RWBLD60)

Description PROFILE is the name of a file in which you want to store performance statistics on report execution. If you specify a filename, Report Builder calculates statistics on the elapsed and CPU time spent running the report. PROFILE calculates the following statistics:

- **TOTAL ELAPSED TIME** is the amount of time that passes between when you issue RWBLD60 and when you leave the designer. TOTAL ELAPSED TIME is the sum of Report Builder Time and ORACLE Time.
- **Time** is the amount of time spent in Report Builder.
- **ORACLE Time** is the amount of time spent in the database and is composed of the following:
 - **UPI** is the amount of time spent to do such things as connect to the database, parse the SQL, and fetch the data.
 - **SQL** is the amount of time spent performing SRW.DO_SQL.
- **TOTAL CPU Time used by process** is the CPU time spent while in the designer.

Note: For some operating systems, the Report Builder time includes the database time because the database is included in Report Builder's process.

Syntax

```
[PROFILE=]profiler_file
```

Values

Any valid filename in the current directory.

RUNDEBUG (RWBLD60)

Description RUNDEBUG is whether you want extra runtime checking for logical errors in reports. RUNDEBUG checks for things that are not errors but might result in undesirable output. RUNDEBUG checks for the following:

- frames or repeating frames that overlap but do not enclose another object. This can lead to objects overwriting other objects in the output.
- layout objects with page-dependent references that do not have fixed sizing. Report Builder will make such objects fixed in size regardless of the Vertical and Horizontal Elasticity properties.
- bind variables referenced at the wrong frequency in PL/SQL.

Syntax

```
[RUNDEBUG=]{YES|NO}
```

Values

YES means perform extra runtime error checking.

NO means do not perform extra runtime error checking.

Default

YES

ONSUCCESS (RWBLD60)

Description ONSUCCESS is whether you want a COMMIT or ROLLBACK performed when a report is finished executing.

Syntax

```
[ONSUCCESS=] {COMMIT|ROLLBACK|NOACTION}
```

Values

COMMIT means perform a COMMIT when a report is done.

ROLLBACK means perform a ROLLBACK when a report is done.

NOACTION means do nothing when a report is done.

Default

COMMIT, if a USERID is provided.

NOACTION, if called from an external source (e.g., Oracle Forms Developer) with no USERID provided.

Usage Notes

- The COMMIT or ROLLBACK for ONSUCCESS is performed after the After Report trigger fires. Other COMMITs and ROLLBACKs can occur prior to this one. For more information, see READONLY.

ONFAILURE (RWBLD60)

Description ONFAILURE is whether you want a COMMIT or ROLLBACK performed if an error occurs and a report fails to complete.

Syntax

```
[ONFAILURE=] {COMMIT|ROLLBACK|NOACTION}
```

Values

COMMIT means perform a COMMIT if a report fails.

ROLLBACK means perform a ROLLBACK if a report fails.

NOACTION means do nothing if a report fails.

Default

ROLLBACK, if a USERID is provided.

NOACTION, if called from an external source (e.g., Oracle Forms Developer) with no USERID provided.

Usage Notes

- The COMMIT or ROLLBACK for ONFAILURE is performed after the report fails. Other COMMITs and ROLLBACKs can occur prior to this one. For more information, see READONLY.

ERRFILE (RWBLD60)

Description ERRFILE is the name of a file in which you want Report Builder to store error messages.

Syntax

```
[ERRFILE=]error_file
```

Values

Any valid filename.

LONGCHUNK (RWBLD60)

Description LONGCHUNK is the size (in kilobytes) of the increments in which Report Builder retrieves a LONG column value. When retrieving a LONG value, you may want to retrieve it in increments rather than all at once because of memory size restrictions. LONGCHUNK applies only to Oracle7 and Oracle8.

Syntax

[LONGCHUNK=]n

Values

A number from 1 through 9,999. For some operating systems, the upper limit may be lower.

Default

10K

ACCESS (RWBLD60)

Description ACCESS is the location from which modules should be opened. Report Builder will search in this location for the module you specify with MODULE. If ACCESS is not specified, Report Builder will use its file path search order to find the file.

Syntax

```
[ACCESS=]{FILE|DATABASE}
```

Values

FILE

DATABASE

Default

FILE

ORIENTATION (RWBLD60)

Description ORIENTATION controls the direction in which the pages of the report will print.

Syntax

```
[ORIENTATION= ] {DEFAULT | LANDSCAPE | PORTRAIT }
```

Values

DEFAULT means use the current printer setting for orientation.

LANDSCAPE means landscape orientation

PORTRAIT means portrait orientation

Default

DEFAULT

Usage Notes

- If ORIENTATION=LANDSCAPE for a character mode report, you must ensure that your printer definition file contains a landscape clause.

BACKGROUND (RWBLD60)

Description BACKGROUND specifies whether the report should be run in the background.

Syntax

[BACKGROUND=] {YES|NO}

Values

YES is not a valid option for RWBLD60.

NO

Default

NO

MODE (RWBLD60)

Description MODE specifies whether to run the report in character mode or bitmap. This enables you to run a character-mode report from bit-mapped Report Builder or vice versa. For example, if you want to send a report to a Postscript printer from a terminal (e.g., a vt220), you could invoke character-mode RWRUN60 and run the report with MODE=BITMAP. On Windows, specifying MODE=CHARACTER means that the Report Builder ASCII driver will be used to produce editable ASCII output.

Syntax

```
[MODE=] { BITMAP | CHARACTER | DEFAULT }
```

Values

BITMAP

DEFAULT means to run the report in the mode of the current executable being used.

CHARACTER

Default

DEFAULT

PRINTJOB (RWBLD60)

Description PRINTJOB specifies whether the Print Job dialog box should be displayed before running a report.

Syntax

```
[PRINTJOB=]{YES|NO}
```

Values

YES or NO

Default

YES

Usage Notes

- When a report is run as a spawned process (i.e., one executable, such as RWRUN60, is called from within another executable, such as RWBLD60), the Print Job dialog box will not appear, regardless of PRINTJOB.
- When DESTYPE=MAIL, the Print Job dialog box will not appear, regardless of PRINTJOB.

TRACEFILE (RWBLD60)

Description TRACEFILE is the name of the file in which Report Builder logs trace information.

Syntax

```
[TRACEFILE=]tracefile
```

Values

Any valid filename.

Usage Notes

- Trace information can only be generated when running a .rdf file. You cannot specify logging when running a .rep file.
- If you specify LOGFILE or ERRFILE as well as TRACEFILE, all of the trace information will be placed in the most recently specified file. For example, in the following case, all of the specified trace information would be placed in err.log because it is the last file specified in the RWRUN60 command.

```
RWRUN60 MODULE=order_entry  
USERID=scott/tiger  
TRACEFILE=trace.log LOGFILE=mylog.log  
ERRFILE=err.log
```

TRACEMODE (RWBLD60)

Description TRACEMODE indicates whether Report Builder should add the trace information to the file or overwrite the entire file.

Syntax

```
[TRACEMODE=] {TRACE_APPEND|TRACE_REPLACE}
```

Values

TRACE_APPEND adds the new information to the end of the file.

TRACE_REPLACE overwrites the file.

Default

TRACE_APPEND

Usage Notes

- Trace information can only be generated when running a .rdf file. You cannot specify logging when running a .rep file.

TRACEOPTS (RWBLD60)

Description TRACEOPTS indicates the tracing information that you want to be logged in the trace file when you run the report.

Syntax

```
[TRACEOPTS=] {TRACE_ERR|TRACE_PRF|TRACE_APP|TRACE_PLS|  
TRACE_SQL|TRACE_TMS|TRACE_DST|TRACE_ALL| (opt1, opt2, ...)}
```

Values

A list of options in parentheses means you want all of the enclosed options to be used. For example, TRACE_OPTS=(TRACE_APP, TRACE_PRF) means you want TRACE_APP and TRACE_PRF applied.

TRACE_ALL means log all possible trace information in the trace file.

TRACE_APP means log trace information on all the report objects in the trace file.

TRACE_BRK means list breakpoints in the trace file.

TRACE_DST means list distribution lists in the trace file. You can use this information to determine which section was sent to which destination. The trace file format is very similar to the .DST file format, so you can cut and past to generate a .DST file from the trace file.

TRACE_ERR means list error messages and warnings in the trace file.

TRACE_PLS means log trace information on all the PL/SQL objects in the trace file.

TRACE_PRF means log performance statistics in the trace file.

TRACE_SQL means log trace information on all the SQL in the trace file.

TRACE_TMS means enter a timestamp for each entry in the trace file.

Default

TRACE_ALL

Usage Notes

- Trace information can only be generated when running a .rdf file. You cannot specify logging when running a .rep file.

AUTOCOMMIT (RWBLD60)

Description Specifies whether database changes (e.g., CREATE) should be automatically committed to the database. Note that some non-ORACLE databases (e.g., SQL Server) require that AUTOCOMMIT=YES.

Syntax

```
[AUTOCOMMIT= ] {YES|NO}
```

Values

YES or NO

Default

NO

NONBLOCKSQL (RWBLD60)

Description Specifies whether to allow other programs to execute while Report Builder is fetching data from the database.

Syntax

[NONBLOCKSQL=] {YES | NO}

Values

YES means that other programs can execute while data is being fetched.

NO means that other programs cannot execute while data is being fetched.

Default

YES

ROLE (RWBLD60)

Description ROLE specifies the database role to be checked for the report at runtime. ROLE is ignored for RWBLD60.

Syntax

```
[ROLE=]{rolename/[rolepassword]}
```

Values

A valid role and (optionally) a role password.

BLANKPAGES (RWBLD60)

Description BLANKPAGES specifies whether to suppress blank pages when you print a report. Use this keyword when there are blank pages in your report output that you do not want to print.

Syntax

```
[BLANKPAGES=] {YES|NO}
```

Values

YES means print all blank pages

NO means do not print blank pages

Default

YES

Usage Notes BLANKPAGES is especially useful if your logical page spans multiple physical pages (or panels), and you wish to suppress the printing of any blank physical pages.

MAXIMIZE (RWBLD60)

Description MAXIMIZE specifies whether to maximize the Reports Runtime window. This keyword has no affect on Report Builder, it should only be used with Reports Runtime.

DISABLEPRINT (RWBLD60)

Description DISABLEPRINT specifies whether to disable **File→Print**, **File→Page Setup**, **File→Choose Printer** (on Motif) and the equivalent toolbar buttons in the Runtime Previewer.

Syntax

[DISABLEPRINT=] {YES | NO}

Values

YES or NO

Default

NO

DISABLEMAIL (RWBLD60)

Description DISABLEMAIL specifies whether to disable the Mail menu and the equivalent toolbar buttons in the Runtime Previewer.

Syntax

[DISABLEMAIL=] {YES|NO}

Values

YES or NO

Default

NO

DISABLEFILE (RWBLD60)

Description DISABLEFILE specifies whether to disable the **File→Generate to File** menu in the Runtime Previewer.

Syntax

[DISABLEFILE=] {YES|NO}

Values

YES or NO

Default

NO

DISABLENEW (RWBLD60)

Description DISABLENEW specifies whether to disable the **View→New Previewer** menu to prevent the ability to display a new instance of the Runtime Previewer.

Syntax

[DISABLENEW=] {YES | NO}

Values

YES or NO

Default

NO

DELIMITER (RWBLD60)

Description DELIMITER specifies the character(s) to use to separate the cells in your report output.

Syntax

[DELIMITER=]value

Values

Any alphanumeric character or string of alphanumeric characters, such as:

- , means a comma separates each cell
- . means a period separates each cell

You can also use any of these four reserved values:

- tab means a tab separates each cell
- space means a space separates each cell
- return means a new line separates each cell
- none means no delimiter is used

You can also use escape sequences based on the ASCII character set, such as:

- \t means a tab separates each cell
- \n means a new line separates each cell

Default

Tab

Usage Notes

- This argument can only be used if you've specified DESFORMAT=DELIMITED.

CELLWRAPPER (RWBLD60)

Description CELLWRAPPER specifies the character(s) that displays around the delimited cells in your report output.

Syntax

[CELLWRAPPER=]value

Value

Any alphanumeric character or string of alphanumeric characters.

- “ means a double quotation mark displays on each side of the cell
- ‘ means a single quotation mark displays on each side of the cell

You can also use any of these four reserved values:

- tab means a tab displays on each side of the cell
- space means a single space displays on each side of the cell
- return means a new line displays on each side of the cell
- none means no cell wrapper is used

You can also use escape sequences based on the ASCII character set, such as:

- \t means a tab displays on each side of the cell
- \n means a new line displays on each side of the cell

Default

None

Usage Notes

- This argument can only be used if you've specified DESFORMAT=DELIMITED.
- The cell wrapper is different from the actual delimiter .

DATEFORMATMASK (RWBLD60)

Description DATEFORMATMASK specifies how date values display in your delimited report output.

Syntax

[DATEFORMATMASK=]mask

Values

Any valid date format mask .

Usage Notes

- This argument can only be used if you've specified DESFORMAT=DELIMITED.

NUMBERFORMATMASK (RWBLD60)

Description NUMBERFORMATMASK specifies how number values display in your delimited report output.

Syntax

[NUMBERFORMATMASK=]mask

Values

Any valid number format mask .

Usage Notes

- This argument can only be used if you've specified DESFORMAT=DELIMITED.

DESTINATION (RWBLD60)

Description The DESTINATION keyword allows you to specify the name of a .DST file that defines the distribution for the current run of the report.

Syntax

`[DESTINATION=]filename.DST`

Values

The name of a .DST file.

Usage Notes

- To enable the DESTINATION keyword, you must specify DISTRIBUTE=YES on the command line.

DISTRIBUTE (RWBLD60)

Description DISTRIBUTE enables or disables distributing the report output to multiple destinations, as specified by the distribution list defined in the Distribution dialog box or a .DST file.

Syntax

```
[DISTRIBUTE=] {YES|NO}
```

Values

YES means to distribute the report to the distribution list.

NO means to ignore the distribution list and output the report as specified by the DESNAME and DESFORMAT parameters. This is fundamentally a debug mode to allow running a report set up for distribution without actually executing the distribution.

Default

NO

Usage Notes

- To enable the DESTINATION keyword, you must specify DISTRIBUTE=YES.

PAGESTREAM (RWBLD60)

Description PAGESTREAM enables or disables page streaming for the report when formatted as HTML or HTMLCSS output, using the navigation controls set by either of the following:

- the Page Navigation Control Type and Page Navigation Control Value properties in the Report Property Palette.
- PL/SQL in a Before Report trigger (SRW.SET_PAGE_NAVIGATION_HTML)

Syntax

```
[PAGESTREAM= ] {YES|NO}
```

Values

YES means to stream the pages.

NO means to output the report without page streaming.

Default

NO

EXPRESS_SERVER (RWBLD60)

EXPRESS_SERVER specifies the Express Server to which you want to connect.

Syntax

```
EXPRESS_SERVER="server=[server]/domain=[domain]/  
user=[userid]/password=[passwd]"
```

Values

A valid connect string enclosed in double quotes (") where

server	is the Express server string (e.g., ncacn_ip_tcp:olap2- pc/sl=x/st=x/ct=x/sv= x/). See below for more details on the server string.
domain	is the Express Server domain.
user	is the userid to log into the Express Server.
password	is the password for the userid.

The server value contains four parameters that correspond to settings that are made in the Oracle Express Connection Editor and stored in connection (.xcf) files. All four parameters are required and can be specified in any order. The following table describes the parameters and their settings.

Parameter	Description	Setting
sl	Server Login	-2: Host (Domain Login) -1: Host (Server Login) 0: No authentication required 1: Host (Domain Login) and Connect security 2: Host (Domain Login) and Call security 3: Host (Domain Login) and Packet security 4: Host (Domain Login) and Integrity security 5: Host (Domain Login) and Privacy security Notes: Windows NT uses all the settings. Unix systems use only the settings 0, -1, and -2. See the Express Connection Editor Help system for information on these settings.
st	Server Type	1: Express Server
ct	Connection Type	0: Express connection

sv Server Version 1: Express 6.2 or greater

Usage Notes

- You can have spaces in the string if necessary (e.g., if the userid is John Smith) because the entire string is inside of quotes.
- If a forward slash (/) is required in the string, you must use another forward slash as an escape character. For example, if the domain were tools/reports, the command line should be as follows:

```
EXPRESS_SERVER="server=ncacn_ip_tcp:olap2-pc/sl=0/  
st=1/ct=0/sv=1/ domain=tools//reports"
```
- You can use single quotes within the string. It is not treated specially because it is enclosed within double quotes.

EXPRESS_SERVER with userid, password, and domain (RWBLD60)

example

```
rwbl60 userid=scott/tiger@nt805  
express_server="server=ncacn_ip_tcp:olap2-pc/sl=1/st=1/ct=0/  
sv=1/user=orbuild/domain=tools/password=buildme/"  
report=exp.rdf destype=file desname=expl.html  
desformat=html batch=yes
```

EXPRESS_SERVER without userid, password, and domain

(RWBLD60) example

```
rwbl60 userid=scott/tiger@nt805  
express_server="server=ncacn_ip_tcp:olap2-pc/sl=0/st=1/ct=0/  
sv=1/" report=exp.rdf destype=file desname=expl.html  
desformat=html batch=yes
```

CUSTOMIZE (RWBLD60)

Description CUSTOMIZE specifies an XML file or list of files that you want to apply to the report selected in the MODULE or REPORT keyword. The XML file contains customizations (e.g., font changes or color changes) that change the report definition in some way.

For more information, refer to the *Publishing Reports* manual.

Syntax

```
[CUSTOMIZE=]filename.xml|(filename1.xml, filename2.xml, ...)
```

Values

The names of files that contain valid XML report definitions, with path information prefixed to the filenames if necessary.

Usage Notes

Typically, the file extension of an XML report definition is .XML, but it does not have to be when it is used with the CUSTOMIZE keyword.

SAVE_RDF (RWBLD60)

Description SAVE_RDF specifies a file to which you want to save the report selected in the MODULE or REPORT keyword combined with the XML customization file(s) selected in the CUSTOMIZE keyword. This argument is most useful when you have an .RDF file to which you are applying an XML file via the CUSTOMIZE keyword and want to save the combination of the two to a new .RDF file.

For more information, refer to the *Publishing Reports* manual.

Syntax

```
[SAVE_RDF=]filename.rdf
```

Values

Any valid filename.

<param> (RWBLD60)

Description <param> is a parameter that is part of the report's definition. The value you specify for the parameter is used for that parameter's value.

Syntax

```
<param>=value
```

Values

Any valid value for that parameter.

Default

Taken from the Initial Value property of the parameter.

Usage Notes

- You can override the default value on the Runtime Parameter Form.
- Values of arguments may be in single or double quotes. The effect of single or double quotes is operating-system specific.
- The PARAM value is not parsed, or validated, because the parameters may refer to those that have not yet been defined.

RWCGI60

Description RWCGI60 is the Reports Web CGI executable that provides a connection between a Web Server and the Reports Server (RWMTS60), allowing you to run reports dynamically from your Web browser. For more information about RWCGI60 , refer to [d2kcfg.htm](#).

RWCLI60

Description RWCLI60 parses and transfers the command line to the specified Reports Server (RWMTS60). It uses a command line very similar to RWRUN60.

Syntax

```
RWCLI60 MODULE|REPORT=runfile USERID=userid  
[ [keyword=]value|(value1, value2, ...) ] SERVER=tnsname
```

where keyword=value is a valid command line argument.

Usage Notes

- All file names and paths specified in the client command line refer to files and directories on the server machine, except for command file.
- If the command line contains CMDFILE=, the command file will be read and appended to the original command line before being sent to the Reports Server. The runtime engine will not re-read the command file.

RWCLI60 Command Line Arguments

```
[MODULE|REPORT=]runfile
[USERID=]userid
[PARAMFORM=]NO
[CMDFILE=]cmdfile
[ARRAYSIZE=]n
[DESTYPE=]{CACHE|LOCALFILE|SCREEN|FILE|PRINTER|PREVIEW|SYSOUT|MAIL}
[DESNAME=]desname
[DESFORMAT=]desformat
[CACHELOB=]{YES|NO}
[COPIES=]n
[CURRENCY=]currency_symbol
[THOUSANDS=]thousands_symbol
[DECIMAL=]decimal_symbol
[READONLY=]{YES|NO}
[LOGFILE=]logfile
[BUFFERS=]n
[BATCH=]YES
[PAGESIZE=]width x height
[PROFILE=]profiler_file
[RUNDEBUG=]{YES|NO}
[ONSUCCESS=]{COMMIT|ROLLBACK|NOACTION}
[ONFAILURE=]{COMMIT|ROLLBACK|NOACTION}
[ERRFILE=]error_file
[LONGCHUNK=]n
[ORIENTATION=]{DEFAULT|LANDSCAPE|PORTRAIT}
[BACKGROUND=]{YES|NO}
[MODE=]{BITMAP|CHARACTER|DEFAULT}
[TRACEFILE=]tracefile
[TRACEMODE=]{TRACE_APPEND|TRACE_REPLACE}
[TRACEOPTS=]{TRACE_ERR|TRACE_PRF|TRACE_APP|TRACE_PLS|TRACE_SQL|TRACE_TMS|TRACE_DST|TRACE_ALL|(opt1, opt2, ...)}
[AUTOCOMMIT]{YES|NO}
[NONBLOCKSQL=]{YES|NO}
[ROLE=]rolename/[rolepassword]|(rolename/[rolepassword],. . .)
.)
[BLANKPAGES=]{YES|NO}
[SERVER=]tnsname
[JOBNAME=]string
[SCHEDULE=]string
[TOLERANCE=]number
[DELIMITER=]value
```

```
[CELLWRAPPER=]value
[DATEFORMATMASK=]mask
[NUMBERFORMATMASK=]mask
[DESTINATION=]filename.DST
[DISTRIBUTE=]{YES|NO}
EXPRESS_SERVER="server=[server]/domain=[domain]/user=[userid
]/password=[passwd]"
[AUTHID]=username/password
[CUSTOMIZE]=filename.xml | (filename1.xml, filename2.xml, .
. .)
[SAVE_RDF]=filename.rdf
<param>=value
```

PARAMFORM (RWCLI60)

Description If PARAMFORM is specified, it must be NO.

Syntax

[PARAMFORM=]NO

DESTYPE (RWCLI60)

Description DESTYPE is the type of device that will receive the report output.

Syntax

[DESTYPE=]{CACHE|LOCALFILE|FILE|PRINTER|SYSOUT|MAIL}

Values

CACHE	Sends the output directly to the Reports Server's cache. Note that DESTYPE=CACHE is not compatible with the DISTRIBUTE keyword. If the server encounters DISTRIBUTE on the command line, it will ignore the DESTYPE=CACHE command line argument.
LOCALFILE	Sends the output to a file on the client machine and forces a synchronous call, regardless of the BACKGROUND value.
FILE	Sends the output to the file on the server machine named in DESNAME.
PRINTER	Sends the output to the printer on the server machine named in DESNAME.
MAIL	Sends the output to the mail users specified in DESNAME. You can send mail to any mail system that is MAPI compliant or has the service provider driver installed. The report is sent as an attached file.
SYSOUT	Sends the output to the client machine's default output device and forces a synchronous call.

Default

Taken from the Initial Value property of the DESTYPE parameter.

Usage Notes

- Screen and Preview cannot be used for DESTYPE with RWCLI60.

BATCH (RWCLI60)

Description If BATCH is specified, it must be YES.

Syntax

[BATCH=] YES

BACKGROUND (RWCLI60)

Description BACKGROUND is whether the call is synchronous (BACKGROUND=NO) or asynchronous (BACKGROUND=YES). A synchronous call means that the client waits for the report to queue, be assigned to a runtime engine, run, and finish. An asynchronous call means that the client simply sends the call without waiting for it to complete. If the client process is killed during a synchronous call, the job is canceled.

Syntax

[BACKGROUND=] {YES|NO}

Values

YES or NO

Default

NO

SERVER (RWCLI60)

Description SERVER is the TNS address of the Reports Server.

Syntax

[SERVER=]tnsname

Values

Any valid TNSNAME.

JOBNAME (RWCLI60)

Description JOBNAME is the name for a job to appear in the Reports Queue Manager. It is treated as a comment and has nothing to do with the running of the job. If it is not specified, the queue manager will show the report name as the job name.

Syntax

`[JOBNAME=]string`

SCHEDULE (RWCLI60)

Description SCHEDULE is a scheduling command. The default is now. To eliminate the need for quoting the scheduling command, use underscore (_) instead of a space. For example:

```
schedule=every_first_fri_of_month_from_15:53_Oct_23,_1999_retry_3_after_1_hour
schedule=last_weekday_before_15_from_15:53_Oct_23,_1999_retry_after_1_hour
```

Note: Earlier forms of the SCHEDULE syntax are supported, but only the current SCHEDULE syntax is documented here.

Syntax

```
[SCHEDULE=]string
```

where the string is:

```
[FREQ from] TIME [retry {n} + after LEN]
```

FREQ	hourly daily weekly monthly {every LEN DAYREPEAT} {last {WEEKDAYS weekday weekend} before {n}+}
LEN	{n}+ {minute[s] hour[s] day[s] week[s] month[s]}
DAYREPEAT	{first second third fourth fifth} WEEKDAYS of month
WEEKDAYS	mon tue wed thu fri sat sun
TIME	now CLOCK [DATE]
CLOCK	h:m h:mm hh:m hh:mm
DATE	today tomorrow {MONTHS {d dd} [,year]}
MONTHS	jan feb mar apr may jun jul aug sep oct nov dec

TOLERANCE (RWCLI60)

Description TOLERANCE is the time tolerance for duplicate job detection in minutes.

Syntax

[TOLERANCE=]number

AUTHID

Description AUTHID is the username and password used to authenticate users to a secured Reports Server. Oracle Reports Developer uses Oracle WebDB to perform the security check. WebDB ensures that users running the reports have the access privileges needed to run the requested report. Refer to Oracle WebDB online help for more information about Oracle Reports Developer Security.

When users successfully log on, their browser is sent an encrypted cookie that authenticates them to the secured Reports Server. By default, the cookie expires after 30 minutes. When a cookie expires, subsequent requests (i.e., ones sent to the secured server) must be re-authenticated.

You can use the REPORTS60_COOKIE_EXPIRE environment variable to change the expiration time of the authentication cookie.

If you want users to authenticate and remain authenticated until the cookie expires, omit the AUTHID command from the report request. If you want users to authenticate every time they make a Web request, then include the Web CGI or Web Cartridge commands SHOWAUTH and AUTHYPE=S in the report URL, or include the %S argument in the key mapping entry in the owscmd.dat (Web Cartridge) or cgicmd.dat (Web CGI) file.

Syntax

[AUTHID=]username/password

Values

Any valid username and password created in Oracle WebDB. See your DBA to create new user profiles in WebDB.

USERID

Description USERID is the ORACLE username or placeholder username (i.e., \$username) and password with an optional database name, Net8 communication protocol to access a remote database, or ODBC datasource name (if accessing a non-Oracle datasource).

For more information about the optional Net8 communication protocol, see "[Remote Logon](#)". For more information about automatic logon, see "[Automatic Logon](#)".

If you want users to log on to the database, omit the USERID command from the report request. If you want users to log on every time they make a Web request, then include the Web CGI or Web Cartridge commands SHOWAUTH and AUTHTYPE=D in the report URL, or include the %D argument in the key mapping entry in the owscmd.dat (Web Cartridge) or cgicmd.dat (Web CGI) file.

Values

The database connection string must be in one of the following forms and cannot exceed 512 bytes in length:

```
username[/password]
username[/password] [@database]
[user[/password]]@ODBC:datasource[:database] or [user[/password]]@ODBC:*
<$username>[/password]
<$username>[/password] [@database]
```

RWCON60

Description RWCON60 enables you to convert one or more report definitions or PL/SQL libraries from one storage format to another.

The following report conversions can be performed using RWCON60:

- convert a report stored in the database into a .rep, .rex, .rdf file, or a template (.tdf file)
- convert a report stored in a .rdf file into a .rep or .rex file, or a database report, or a template (.tdf file)
- convert a report stored in a .rex file into a database report, .rdf file, .rep file, or a template (.tdf file)

The following PL/SQL library conversions can be performed using RWCON60:

- convert a library stored in the database into a .pld or .pll file
- convert a library stored in a .pld file into a database library or a .pll file
- convert a library stored in a .pll file into a database library or a .pld file.

Syntax

```
rwconv60 [ [keyword=]value ] ...
```

where keyword=value is a valid command line argument.

Usage Notes

- In some cases, Convert will automatically compile the report's PL/SQL as part of the conversion process. Provided that your conversion Destination is not a .rex file, PL/SQL is automatically compiled under the following conditions. In all other situations, you must compile the report's PL/SQL yourself (e.g., **Program→Compile→All**).
 - You try to create a .rep file using Convert. If there are compile errors, an error message is displayed and the .rep file is not created.
 - You use a .rex file as the Source report. If there are compile errors, a warning is displayed, but the conversion continues.
 - You use a report that was created on another platform as the Source report. If there are compile errors, a warning is displayed, but the conversion continues.
- Fonts are mapped when a report is opened by Report Builder and Reports Runtime.

Fonts are not mapped during the conversion.

- When converting a report to a template, only objects in the report's header and trailer pages, and the margin area are used in the template. Objects in the body are ignored.
- When converting reports that have attached libraries, you should convert the .pll files attached to the report before converting the .rdf file.

RWCON60 Command Line Arguments

```
[USERID=]userid
[STYPE=]{DATABASE|PLLDB|PLDFILE|PLLFILE|
RDFFILE|REXFILE}
[SOURCE=]{sname|(sname1,sname2,...)}
[DTYPE=]{DATABASE|PLLDB|PLDFILE|PLLFILE|
RDFFILE|REXFILE|REXFILE|TDFFILE}
[DEST=]{dname|(dname1,dname2,...)|pathname}
[CMDFILE=]cmdfile
[LOGFILE=]logfile
[OVERWRITE=]{YES|NO|PROMPT}
[BATCH=]{YES|NO}
[DUNIT=]{CENTIMETER|CHARACTER|INCH|POINT}
[PAGESIZE=]width x height
[FORMSIZE=]width x height
[CUSTOMIZE=]filename.xml |(filename1.xml, filename2.xml, .
. .)
```

STYPE (RWCON60)

Description STYPE is the format of the report(s) or libraries to be converted.

Syntax

[STYPE=] { DATABASE | PLLDB | PLDFILE | PLLFILE | RDFFILE | REXFILE }

Values

DATABASE means that the source report(s) are stored in ORACLE.

PLLDB means that the source PL/SQL libraries are stored in the database.

PLDFILE means that the source PL/SQL libraries are stored in files in ASCII format.

PLLFILE means that the source PL/SQL libraries are stored in files containing source code and P-code.

RDFFILE means that the source report(s) are stored in one or more report definition files (files with an extension of .rdf).

REXFILE means that the source report(s) are stored in one or more text files (files with an extension of .rex).

Default

DATABASE

SOURCE (RWCON60)

Description SOURCE is the report/library or list of reports/libraries to be converted. RWCON60 requires that you specify a source report or library.

Syntax

```
[SOURCE=]{sname|(sname1,sname2,...)}
```

Values

Any valid report/library name or filename (e.g., `qanda`).

A list of valid report/library names or filenames enclosed by parentheses with a comma separating the names (e.g., `(qanda,test,dmast)`).

Usage Notes

- SQL wildcard characters (`%` and `_`) may be used for reports or libraries that are stored in the database. For example, `R%` would fetch all reports stored in the database that begin with `R`. All reports that match will be converted.
- A list of report/library names or filenames must be enclosed in parentheses with commas separating the names. For example:

```
(qanda,test,dmast)
```

```
(qanda, test, dmast)
```
- Wildcard characters are invalid for reports/libraries stored in files (i.e., with extensions of `.rdf`, `.rep`, `.rex`, `.pld`, `.pll`).
- The argument(s) for this keyword may be operating system-specific.
- If you are converting reports/libraries stored in ORACLE and are using system-owned Report Builder tables, you may list reports/libraries of other users, for example:

```
SOURCE=(sday.qanda,dsmith.test,dmast.sal)
```
- If you are using user-owned Report Builder tables, reports/libraries from multiple users must be converted for each user individually.
- You must have created the reports/libraries, or have been granted access to the ones you did not create, in order to convert them. If no `userid` is prefixed to the report/library name, the `userid` is assumed to be the current user.
- If you are converting from database to file and the database report/library name is too long to be a filename, Report Builder prompts you to provide a filename of the correct length for your operating system.

DTYPE (RWCON60)

Description DTYPE is the format to which to convert the reports or libraries .

Syntax

[DTYPE=] {DATABASE | PLLDB | PLDFILE | PLLFILE | RDFFILE | REPFIL | REXFILE | TDFFILE }

Values

DATABASE means that the converted reports will be stored in ORACLE.

PLLDDB means that the converted PL/SQL libraries will be stored in the database.

PLDFILE means that the converted PL/SQL libraries will be stored in files in ASCII format.

PLLFILE means that the converted PL/SQL libraries will be stored in files containing source code and P-code.

RDFFILE means that the converted reports will be stored in one or more report definition files (files with an extension of .rdf).

REPFIL means that the converted reports will be stored in one or more binary runfiles (files with an extension of .rep).

REXFILE means that the converted reports will be stored in one or more text files (files with an extension of .rex).

TDFFILE means that the report will be converted to a template (.tdf file).

Default

REXFILE

Usage Notes

- When you try to create a .rep file using RWCON60, the source report's PL/SQL is automatically compiled. If there are compile errors, an error message is displayed and the .rep file is not created. To avoid this problem, make sure you compile the source report's PL/SQL using **File**→**Compile** before you try to create a .rep file.
- When converting a report to a template, only objects in the report's header and trailer pages, and the margin area are used in the template. Objects in the body are ignored.

DEST (RWCON60)

Description DEST is the name(s) of the converted reports or libraries, if they are stored in the database, or the names of the files.

Syntax

```
[DEST=]{dname|(dname1,dname2,...)|pathname}
```

Values

Any valid report/library name or filename (e.g., `qanda`).

A list of valid report/library names or filenames enclosed by parentheses with a comma separating the names (e.g., `(qanda,test,dmast)`).

Default

If the DEST keyword is not specified, RWCON60 uses the following default names:

- If DTYPE is DATABASE or PLLDB, then DEST is SOURCE.
- If DTYPE is PLDFILE, then DEST is SOURCE with the `.pld` extension.
- If DTYPE is PLLFILE, then DEST is SOURCE with the `.pll` file extension.
- If DTYPE is RDIFFILE, then DEST is SOURCE with the `.rdf` extension.
- If DTYPE is REPFIL, the DEST is SOURCE with the `.rep` extension.
- If DTYPE is REXFILE, then DEST is `expdat.rex`.

Usage Notes

- A list of report/library names or filenames must be enclosed in parentheses with commas separating the names. For example:

```
(qanda,test,dmast)
```

```
(qanda, test, dmast)
```
- If you have more destination names than there are source names, the extra destination names are ignored. If you have fewer destination names than there are source names, default names will be used after the destination names run out.
- The argument(s) for this keyword may be operating system-specific.

CMDFILE (RWCON60)

Description CMDFILE is a file that contains arguments for the RWCON60 command. This option enables you to convert a report/library without having to specify a large number of arguments each time you invoke RWCON60.

Syntax

```
[CMDFILE=]cmdfile
```

Values

Any valid command file.

Usage Notes

- A command file may reference another command file.
- Command file syntax for RWCON60 arguments is identical to that used on the command line.
- Values entered on the command line override values specified in command files. For example, suppose that you specify RWRUN60 from the command line with COPIES equal to 1 and CMDFILE equal to RUNONE (a command file). In RUNONE, COPIES is set to 2. Only one copy of the report would be generated in this case. For more information, see "Command Line Syntax Rules" .
- The argument(s) for this keyword may be operating system-specific.

LOGFILE (RWCON60)

Description LOGFILE is the name of the file to which status and error output is sent.
`[LOGFILE=]logfile`

Values

Any valid filename.

Default

dfltrep.log in the current directory.

OVERWRITE (RWCON60)

Description OVERWRITE specifies whether to overwrite existing files or database objects with the converted files or objects.

Syntax

[OVERWRITE=] {YES|NO|PROMPT}

Values

YES means automatically overwrite any existing files or database objects of the same name.

NO means not to convert reports if there are existing files or database objects of the same name and display a warning message.

PROMPT means to prompt you before overwriting any existing or database objects.

Default

NO

BATCH (RWCON60)

Description BATCH suppresses all terminal input and output, in order to convert reports/libraries without user intervention.

Syntax

[BATCH=] { YES | NO }

Values

YES suppresses all terminal input and output.

NO allows special terminal input and output. The Convert dialog box is displayed and, when you accept the dialog box, the conversion is performed.

Default

NO

DUNIT (RWCON60)

Description DUNIT is the destination unit of measurement to which the report should be converted. If specified, DUNIT must differ from the SOURCE report's unit of measurement. If left blank, the SOURCE report's unit of measurement is used as the DEST report's unit of measurement.

Syntax

[DUNIT=] {CENTIMETER | CHARACTER | INCH | POINT}

Values

CENTIMETER means that the converted reports will initially use centimeters as the unit of measurement.

CHARACTER means that the converted reports will initially use characters as the unit of measurement.

INCH means that the converted reports will initially use inches as the unit of measurement.

POINT means that the converted reports will initially use points as the unit of measurement.

Default

Blank

PAGESIZE (RWCON60)

Description PAGESIZE is the size of a logical page for the converted reports in terms of destination unit of measurement (specified using the DUNIT keyword).

Syntax

[PAGESIZE=]width x height

Values

Any valid value in the unit of measurement and of the form width x height.

Default

For bitmap, 8.5 x 11 inches. For character mode, 80 x 66 characters. If the report was designed for character mode and is being run or converted on bitmap, the following formula is used to determine page size if none is specified: (default page size * character page size)/default character page size. For example, if the character page size is 80 x 20, the bit-mapped page size would be: $(8.5 * 80)/80 \times (11 * 20)/66 = 8.5 \times 3.33$.

Usage Notes

- For non-character DUNITs, you can use a decimal to specify fractions (e.g., 8.5 x 11).

FORMSIZE (RWCON60)

Description FORMSIZE is the size of the Runtime Parameter Form for the converted report in terms of destination unit of measurement (specified using the DUNIT keyword).

Syntax

```
[FORMSIZE=]width x height
```

Values

Any valid value in the unit of measurement and of the form width x height.

Usage Notes

- For non-character DUNITs, you can use a decimal to specify fractions (e.g., 8.5 x 11).

CUSTOMIZE (RWCON60)

Description CUSTOMIZE specifies an XML file or list of files that you want to apply to the reports listed in the SOURCE keyword. The XML file contains customizations (e.g., font changes or color changes) that change the report definition in some way. For more information, refer to the *Publishing Reports* manual.

Syntax

```
[CUSTOMIZE=]filename.xml | (filename1.xml, filename2.xml, . . .)
```

Values

A filename or list of filenames that contain a valid XML report definition, with path information prefixed to the filename(s) if necessary.

RWMTS60

Description RWMTS60 is the Reports Multi-tier Server (hereafter the Reports Server) executable. To implement a multi-tiered architecture for running your reports, the Reports Server enables you to run reports on a remote application server. To start RWMTS60 from the command line, use RWCLI60.

When used in conjunction with the Reports Web Cartridge or Reports Web CGI, the Reports Server also enables you to run reports from a Web browser using standard URL syntax. The server can be installed on Windows NT, Windows 95, or UNIX. It handles client requests to run reports by entering all requests into a job queue. When one of the server's runtime engines becomes available, the next job in the queue is dispatched to run. As the number of jobs in the queue increases, the server can start more runtime engines until it reaches the maximum limit specified when the server process was started. Similarly, idle engines are shut down after having been idle for longer than a specified period of time.

You can also use the server to run reports automatically. The scheduling command enables you to specify a time and frequency for the report. The server keeps track of a predefined maximum number of past jobs. Information on when the jobs are queued, started, and finished is kept, as well as the final status of the report. This information can be retrieved and reviewed on Windows from the Reports Queue Manager (RWRQM60) or via the API. The Reports Queue Manager may reside on the same machine as the Reports Server or on a client machine. On UNIX, you can use the Reports Queue Viewer (RWRQV60) to view the Reports Server's queue.

RWRQM60

Description RWRQM60 is the Reports Queue Manager executable (on the Windows platform). When you use the Reports Server executable (RWMTS60) to schedule running reports on a remote server, the Reports Queue Manager enables you to review when the jobs are queued, started, and finished, as well as the final status of the report. You can also manipulate the jobs and the server. The Reports Queue Manager may reside on the same machine as the Reports Server or on a client machine. On UNIX, you can use the Reports Queue Viewer (RWRQV60) to view the Reports Server's queue.

RWRQV60

Description RWRQV60 is the Reports Queue Viewer executable (on the UNIX platform). When you use the Reports Server executable (RWMTS60) to schedule running reports on a remote server, the Reports Queue Viewer enables you to view the Reports Server's queue. On the command line, type RWRQV60 "?" for keyword help. On Windows, you can use the Reports Queue Manager (RWRQM60) to view and manipulate the Reports Server's queue.

RWOWS60

Description RWOWS60 is the Reports Web Cartridge that provides an easy connection between the Oracle Web Application Server and the Reports Server (RWMTS60). This connection enables a Web client to run reports on the Reports Server and see the report output on the client Web browser. Like other Web Cartridges, RWOWS60 is a dynamically-linked, or shared, library that is managed by a Web Request Broker, which is associated with a particular Oracle Web Application Server Listener. RWOWS60 is an alternative to the Reports Web CGI (RWCGI60) and has the same functionality, but it also has native integration with the Oracle Web Application Server.

RWRBE60

Description RWRBE60 is the Reports Background Engine. The Reports Background Engine waits for any command lines to run reports. When it receives a command line, it places that report in its queue. Typically, the Reports Background Engine receives command lines from a form or graphic using the RUN_PRODUCT packaged procedure, or from a report using SRW.RUN_REPORT.

Syntax:

`rwrbe60`

Usage Notes

- ■ RWRBE60 was called R25SRV in earlier versions.
- ■ You can also start the RWRBE60 executable by double-clicking it.
 - On Windows, you can use RWISV60 to submit a report to the Reports Background Engine for execution. This is a more efficient method because it incurs less performance overhead than Reports Runtime. The command line arguments for RWISV60 are the same those for RWRUN60.
- ■ RWRBE60 is only used to run reports locally. Use RWCLI60 to send reports to the Reports Server (RWMTS60).

RWRUN60

Description RWRUN60 runs a report (i.e., .rdf or .rep file) interactively or in batch.

Syntax:

```
RWRUN60 [ [keyword=]value|(value1, value2, ...) ]
```

where keyword=value is a valid command line argument.

Usage Notes

- If you are running an .rep file, the PL/SQL is already compiled and will not be recompiled. If you are running an .rdf file, the PL/SQL is automatically recompiled if necessary (i.e., you didn't compile and save the report from Report Builder, or the platform or version on which you are running the report is incompatible with the platform on which it was last compiled and saved).
- There are two versions of RWRUN60. It is important to note that with either executable, you can run character-mode or bit-mapped reports. The only limiting factor is the output device (e.g., you could not run a bit-mapped report to the screen of a vt220).
 - The character-mode executable is for users who are running reports from character-mode machines (e.g., VT220).
 - The GUI executable is for users who are running reports from bit-mapped machines (e.g., a PC running Windows).

RWRUN60 Command Line Arguments

```
[MODULE|REPORT=]runfile
[USERID=]userid
[PARAMFORM=]{YES|NO}
[CMDFILE=]cmdfile
[TERM=]termfile
[ARRAYSIZE=]n
[DESTYPE=]{SCREEN|FILE|PRINTER|PREVIEW|SYSOUT|MAIL}
[DESNAME=]desname
[DESFORMAT=]desformat
[CACHELOB=]{YES|NO}
[COPIES=]n
[CURRENCY=]currency_symbol
[THOUSANDS=]thousands_symbol
[DECIMAL=]decimal_symbol
[READONLY=]{YES|NO}
[LOGFILE=]logfile
[BUFFERS=]n
[BATCH=]{YES|NO}
[PAGESIZE=]width x height
[PROFILE=]profiler_file
[RUNDEBUG=]{YES|NO}
[ONSUCCESS=]{COMMIT|ROLLBACK|NOACTION}
[ONFAILURE=]{COMMIT|ROLLBACK|NOACTION}
[KEYIN=]keyin_file
[KEYOUT=]keyout_file
[ERRFILE=]error_file
[LONGCHUNK=]n
[ORIENTATION=]{DEFAULT|LANDSCAPE|PORTRAIT}
[BACKGROUND=]{YES|NO}
[MODE=]{BITMAP|CHARACTER|DEFAULT}
[PRINTJOB]{YES|NO}
[TRACEFILE=]tracefile
[TRACEMODE=]{TRACE_APPEND|TRACE_REPLACE}
[TRACEOPTS=]{TRACE_ERR|TRACE_PRF|TRACE_APP|TRACE_PLS|
TRACE_SQL|TRACE_TMS|TRACE_DST|TRACE_ALL|(opt1, opt2, ...)}
[AUTOCOMMIT=]{YES|NO}
[NONBLOCKSQL=]{YES|NO}
[ROLE=]rolename/[rolepassword]
[BLANKPAGES=]{YES|NO}
[DISABLEPRINT=]{YES|NO}
[DISABLEMAIL=]{YES|NO}
[DISABLEFILE=]{YES|NO}
```

```
[DISABLENEW=]{YES|NO}
[DELIMITER=]value
[CELLWRAPPER=]value
[DATEFORMATMASK=]mask
[NUMBERFORMATMASK=]mask
[DESTINATION=]filename.DST
[DISTRIBUTE=]{YES|NO}
[PAGESTREAM=]{YES|NO}
EXPRESS_SERVER="server=[server]/domain=[domain]/user=[userid
]/password=[passwd]"
[CUSTOMIZE]=filename.xml | (filename1.xml, filename2.xml, .
. .)
[SAVE_RDF]=filename.rdf
<param>=value
```

MODULE|REPORT (RWRUN60)

Description MODULE|REPORT is the name of the report to run. (REPORT is allowed for backward compatibility.)

Syntax

[MODULE|REPORT=]runfile

Values

Any valid runfile (i.e., a file with an extension of .rdf or .rep). If you do not enter a file extension, Reports Runtime searches first for a file with extension .rep, then extension .rdf, and then no extension. Reports Runtime will use its file path search order to find the file.

MODULE|REPORT Restrictions

- If you specify a runfile using MODULE|REPORT, the RWRUN60 window is not raised. (This behavior provides seamless entry into RWRUN60 from other applications.) If you do not specify a runfile using MODULE/REPORT and you specify BATCH=NO, the RWRUN60 window is opened, which enables you to run reports interactively.
- If you run a character-mode report via bit-mapped RWRUN60, Reports Runtime displays a warning, then runs the report using a page size of 8.5" x 11" and a form size of 7" x 6".

PARAMFORM (RWRUN60)

Description PARAMFORM specifies whether to display the Runtime Parameter Form when you execute a report.

Syntax

[PARAMFORM=] {YES|NO}

Values

YES means to display the Form.

NO means to suppress the Form.

Default

YES

Usage Notes

PARAMFORM=YES is incompatible with BATCH=YES because it is not meaningful to have the Runtime Parameter Form appear in batch mode.

CMDFILE (RWRUN60)

Description CMDFILE is a file that contains arguments for the RWRUN60 command. This option enables you to run a report without having to specify a large number of arguments each time you invoke RWRUN60.

Syntax

`[CMDFILE=]cmdfile`

Values

Any valid command file.

CMDFILE Restrictions

- A command file may reference another command file.
- Command file syntax for RWRUN60 arguments is identical to that used on the command line.
- Values entered on the command line override values specified in command files. For example, suppose that you specify RWRUN60 from the command line with COPIES equal to 1 and CMDFILE equal to RUNONE (a command file). In RUNONE, COPIES is set to 2. Only one copy of the report would be generated in this case.
- The argument(s) for this keyword may be operating system-specific.

TERM (RWRUN60)

Description TERM is the type of terminal on which you are using RWRUN60. TERM is useful for the Runtime Parameter Form and Runtime Previewer only. This keyword is only used in character mode.

Syntax

[TERM=] *termtype*

Values

Any valid terminal type.

Default

Installation dependent. (See your System Administrator for a compatible definition.)

Usage Notes

- The argument(s) for this keyword may be case sensitive, depending on your operating system.

ARRAYSIZE (RWRUN60)

Description ARRAYSIZE is the size (in kilobytes) for use with ORACLE array processing. Generally, the larger the array size, the faster the report will run.

Syntax

[ARRAYSIZE=]n

Values

A number from 1 through 9,999. This means that Reports Runtime can use this number of kilobytes of memory per query in your report.

Default

The default array size is 10K. For details about the ORACLE array processing, see the *Oracle8 Server Administrator's Guide*.

DESTYPE (RWRUN60)

Description DESTYPE is the type of device that will receive the report output.

Syntax

[DESTYPE=]{ SCREEN | FILE | PRINTER | PREVIEW | SYSOUT | MAIL }

Values

SCREEN	Routes the output to the Previewer for interactive viewing. This value is valid only when the BATCH=NO. Font aliasing is not performed for Screen.
FILE	Saves the output to a file named in DESNAME (the next keyword discussed).
PRINTER	Routes the output to the printer named in DESNAME (the next keyword discussed).
PREVIEW	Routes the output to the Previewer for interactive viewing. However, Preview causes the output to be formatted as PostScript output. The Previewer will use DESNAME to determine which printer's fonts to use to display the output. Font aliasing is performed for Preview.
SYSOUT	Sends the output to your operating system's default output device. This value is valid only when BATCH=YES.
MAIL	Sends the output to the mail users specified in DESNAME. You can send mail to any mail system that is MAPI compliant or has the service provider driver installed. The report is sent as an attached file.

Default

Taken from the Initial Value property of the DESTYPE parameter.

Usage Notes

- If you set DESTYPE to PRINTER, FILE, or MAIL, the Print Job dialog box will appear after you accept the Runtime Parameter Form, unless it is suppressed. After you accept the Print Job dialog box, the report is executed and the output is sent to the printer (specified in **File→Choose Printer**), file, or mail ID. (If DESTYPE is something other than PRINTER, FILE, or MAIL, then the report is executed after you accept the Runtime Parameter Form.) The appearance of the dialog box varies between operating systems.

- In some cases, this parameter may be overridden by your operating system.
- Some DESTYPE values are not supported on some operating systems.

DESNAME (RWRUN60)

Description DESNAME is the name of the file, printer, or email ID (or distribution list) to which the report output will be sent. To send the report output via email, specify the email ID as you do in your email application (any MAPI-compliant application on Windows or your native mail application on UNIX). You can specify multiple usernames by enclosing the names in parentheses and separating them by commas (e.g., (name, name, . . .name)).

Syntax

```
[DESNAME=]desname
```

Values

Any valid filename, printer name, or email ID not to exceed 1K in length. For printer names, you can optionally specify a port. For example:

```
DESNAME=printer,LPT1:
```

```
DESNAME=printer,FILE:
```

Default

Taken from the Initial Value property of the DESNAME parameter. If DESTYPE=FILE and DESNAME is an empty string, it defaults to `reportname.lis` at runtime.

Usage Notes

- This keyword is ignored if DESTYPE is SCREEN or SYSOUT.
- If DESTYPE is PREVIEW, Reports Runtime uses DESNAME to determine which printer's fonts to use to display the output.
- The argument(s) for this keyword may be case sensitive, depending on your operating system.
- In some cases, this parameter may be overridden by your operating system.

DESFORMAT (RWRUN60)

Description In bit-mapped environments, DESFORMAT specifies the printer driver to be used when DESTYPE is FILE. In character-mode environments, it specifies the characteristics of the printer named in DESNAME.

Syntax

```
[DESFORMAT=]desformat
```

Values

Any valid destination format not to exceed 1K in length. Examples of valid values for this keyword are hpl, hplwide, dec, decwide, decland, decl80, dflt, wide, etc. Ask your System Administrator for a list of valid destination formats.. In addition, Report Builder supports the following destination formats:

PDF	Means that the report output will be sent to a file that can be read by a PDF viewer. PDF output is based upon the currently configured printer for your system. The drivers for the currently selected printer is used to produce the output; you must have a printer configured for the machine on which you are running the report.
HTML	Means that the report output will be sent to a file that can be read by an HTML 3.0 compliant browser (e.g., Netscape 2.2).
HTMLCSS	Means that the report output sent to a file will include style sheet extensions that can be read by an HTML 3.0 compliant browser that supports cascading style sheets.
HTMLCSSIE	Means that the report output sent to a file will include style sheet extensions that can be read by Microsoft Internet Explorer 3.x.
RTF	Means that the report output will be sent to a file that can be read by standard word processors (such as Microsoft Word). Note that when you open the file in MS Word, you must choose View → Page Layout to view all the graphics and objects in your report.
DELIMITED	Means that the report output will be sent to a file that can be read by standard spreadsheet utilities, such as Microsoft Excel. Note that you must also specify a DELIMITER.
XML	Means that the report output will be an XML document, saved as a separate file with the .XML extension. This report can be opened and read in an XML-supporting browser, or your choice of XML viewing application.

Default

Taken from the Initial Value property of the DESFORMAT parameter. For bit-mapped Reports Runtime, if DESFORMAT is blank or dflt, then the current printer (specified in **File→Choose Printer**) is used. If nothing has been selected in Choose Printer, then Postscript is used by default.

Usage Notes

- This keyword is ignored if DESTYPE is SCREEN or SYSOUT.
- The value(s) for this keyword may be case sensitive, depending on your operating system.

COPIES (RWRUN60)

Description COPIES specifies the number of copies of the report output to print.

Syntax

[COPIES=]n

Values

Any valid integer from 1 through 9,999.

Default

Taken from the Initial Value property of the COPIES parameter.

Usage Notes

- This keyword is ignored if DESTYPE is not Printer.
- If COPIES is left blank on the Runtime Parameter Form, it defaults to one.

CACHELOB (RWRUN60)

Description CACHELOB specifies whether to cache retrieved Oracle8 large object(s) in the temporary file directory (specified by REPORTS60_TMP).

Values

YES means to cache the LOB in the temporary file directory.

NO means to not cache the LOB in the temporary file directory.

Default

YES

Usage Notes

- You can only set this option on the command line.
- If the location of the temporary file directory does not have sufficient available disk space, it is preferable to set this value to NO. Setting the value to NO, however, may decrease performance, as the LOB may need to be fetched from the server multiple times.

CURRENCY (RWRUN60)

Description CURRENCY is the currency character to be used in number formats.

Syntax

[CURRENCY=]currency_symbol

Values

Any valid alphanumeric string not to exceed 1K in length.

Default

The default for ORACLE is determined by the ORACLE National Language Support facilities. You can also set a default of up to four characters in the Initial Value field of the Parameter property sheet for the CURRENCY parameter.

Usage Notes

- A CURRENCY value entered on the Runtime Parameter Form will override any CURRENCY value entered on the command line.

THOUSANDS (RWRUN60)

Description THOUSANDS is the thousands character to be used in number formats.

Syntax

[THOUSANDS=] thousands_symbol

Values

Any valid alphanumeric character.

Default

The default for ORACLE is determined by the ORACLE National Language Support facilities. You can also set a default in the Initial Value property of the THOUSANDS parameter.

Usage Notes

- A THOUSANDS value entered on the Runtime Parameter Form will override any THOUSANDS value entered on the command line.
- The alphanumeric character defined as the THOUSANDS value is the actual value that is returned. For example, if you define "," as the THOUSANDS value, "," is returned.

DECIMAL (RWRUN60)

Description DECIMAL is the decimal character to be used in number formats.

Syntax

```
[DECIMAL=]decimal_symbol
```

Values

Any valid alphanumeric character.

Default

The default for ORACLE is determined by the ORACLE National Language Support facilities. You can also set a default in the Initial Value property for the DECIMAL parameter.

Usage Notes

- A DECIMAL value entered on the Runtime Parameter Form will override any DECIMAL value entered on the command line.
- The alphanumeric character defined as the DECIMAL value is actual value that is returned. For example, if you define “.” as the DECIMAL value, “.” is returned.

READONLY (RWRUN60)

Description READONLY requests read consistency across multiple queries in a report. When accessing data from ORACLE, read consistency is accomplished by a SET TRANSACTION READ ONLY statement (refer to your *Oracle8 Server SQL Language Reference Manual* for more information on SET TRANSACTION READ ONLY).

Syntax

```
[READONLY=]{YES|NO}
```

Values

YES requests read consistency.

NO means do not provide read consistency.

Default

NO

Usage Notes

- READONLY is only useful for reports using multiple queries, because ORACLE automatically provides read consistency, without locking, for single query reports.

LOGFILE (RWRUN60)

Description LOGFILE is the name of the file to which **File**→**Print Screen** output is sent. If the specified file already exists, output will be appended to it. This keyword is only used in character mode.

Syntax

```
[LOGFILE=]logfile
```

Values

Any valid filename.

Default

dfltrep.log in the current directory.

BUFFERS (RWRUN60)

Description BUFFERS is the size of the virtual memory cache in kilobytes. You should tune this setting to ensure that you have enough space to run your reports, but not so much that you are using too much of your system's resources.

Syntax

[BUFFERS=]n

Values

A number from 1 through 9,999. For some operating systems, the upper limit may be lower.

Default

640K

Usage Notes

- If this setting is changed in the middle of your session, the change does not take effect until the next time the report is run.

BATCH (RWRUN60)

Description BATCH suppresses all terminal input and output, in order to run reports without user intervention. BATCH=YES enables you to run the report without bringing up the RWRUN60 interface. To run reports in the background, see BACKGROUND.

Syntax

```
[ BATCH= ] { YES | NO }
```

Values

YES suppresses all terminal input and output.

NO allows special terminal input and output.

Default

NO

Usage Notes

- If BATCH=YES, error messages are sent to SYSOUT. For more information on SYSOUT, see DESNAME.
- If BATCH=YES, PARAMFORM=YES is invalid because it is not meaningful to have the Runtime Parameter Form appear in batch mode.
- When BATCH=NO and you are using the character-mode RWRUN60, bit-mapped reports cannot be run. (It is not possible to run a bit-mapped report to the screen of a character-mode device.)
- When BATCH=YES, MODE=BITMAP, and you are using the character-mode RWRUN60, you cannot run a character-mode report.

PAGESIZE (RWRUN60)

Description PAGESIZE is the dimensions of the physical page (i.e., the size of the page that the printer outputs). Note that the page must be large enough to contain the report. For example, if a frame in a report expands to a size larger than the page dimensions, the report will not run.

Syntax

```
[PAGESIZE=]width x height
```

Values

Any valid page dimensions of the form: page width x page height, where page width and page height are zero or more. The maximum width/height depends upon the unit of measurement. For inches, the maximum width/height is 512 inches. For centimeters, it is 1312 centimeters. For picas, it is 36,864 picas.

Default

For bitmap, 8.5 x 11 inches. For character mode, 80 x 66 characters. If the report was designed for character mode and is being run or converted on bitmap, the following formula is used to determine page size if none is specified: (default page size * character page size)/default character page size. For example, if the character page size is 80 x 20, the bit-mapped page size would be: $(8.5 * 80)/80 \times (11 * 20)/66 = 8.5 \times 3.33$.

Usage Notes

- On some printers the printable area of the physical page is restricted. For example, the sheet of paper a printer takes may be 8.5 x 11 inches, but the printer might only be able to print on an area of 8 x 10.5 inches. If you define a page width x page height in Reports Runtime that is bigger than the printable area your printer allows, clipping may occur in your report output. To avoid clipping, you can either increase the printable area for the printer (if your operating system allows it) or you can set the page width x page height to be the size of the printable area of the page.
- If this keyword is used, its value overrides the page dimensions of the report definition.
- A PAGESIZE value entered on the Runtime Parameter Form will override any PAGESIZE value entered on the command line.

PROFILE (RWRUN60)

Description PROFILE is the name of a file in which you want to store performance statistics on report execution. If you specify a filename, Reports Runtime calculates statistics on the elapsed and CPU time spent running the report. PROFILE calculates the following statistics:

- **TOTAL ELAPSED TIME** is the amount of time that passes between when you issue RWRUN60 and when it finishes running the report. TOTAL ELAPSED TIME is the sum of Reports Runtime Time and ORACLE Time.
- **Time** is the amount of time spent in Reports Runtime.
- **ORACLE Time** is the amount of time spent in the database and is composed of the following:
 - **UPI** is the amount of time spent to do such things as connect to the database, parse the SQL, and fetch the data.
 - **SQL** is the amount of time spent performing SRW.DO_SQL.
- **TOTAL CPU Time used by process** is the CPU time spent running the report.

Note: For some operating systems, the Reports Runtime time includes the database time because the database is included in Reports Runtime's process.

Syntax

```
[PROFILE=]profiler_file
```

Values

Any valid filename in the current directory.

RUNDEBUG (RWRUN60)

Description RUNDEBUG is whether you want extra runtime checking for logical errors in the report. RUNDEBUG checks for things that are not errors but might result in undesirable output. RUNDEBUG checks for the following:

- frames or repeating frames that overlap but do not enclose another object. This can lead to objects overwriting other objects in the output.
- layout objects with page-dependent references that do not have fixed sizing. Reports Runtime will make such objects fixed in size regardless of the Vertical Elasticity and Horizontal Elasticity property settings.
- bind variables referenced at the wrong frequency in PL/SQL.
- the report is character mode or bitmap and the environment is the opposite.

Syntax

[RUNDEBUG=]{YES|NO}

Values

YES means perform extra runtime error checking.

NO means do not perform extra runtime error checking.

Default

NO

ONSUCCESS (RWRUN60)

Description ONSUCCESS is whether you want a COMMIT or ROLLBACK performed when the report is finished executing.

Syntax

```
[ONSUCCESS=] {COMMIT|ROLLBACK|NOACTION}
```

Values

COMMIT means perform a COMMIT when the report is done.

ROLLBACK means perform a ROLLBACK when the report is done.

NOACTION means do nothing when the report is done.

Default

COMMIT, if a USERID is provided.

NOACTION, if called from an external source (e.g., Oracle Forms Developer) with no USERID provided.

Usage Notes

- The COMMIT or ROLLBACK for ONSUCCESS is performed after the After Report trigger fires. Other COMMITs and ROLLBACKs can occur prior to this one. For more information, see READONLY.

ONFAILURE (RWRUN60)

Description ONFAILURE is whether you want a COMMIT or ROLLBACK performed if an error occurs and the report fails to complete.

Syntax

```
[ONFAILURE=] {COMMIT|ROLLBACK|NOACTION}
```

Values

COMMIT means perform a COMMIT if the report fails.

ROLLBACK means perform a ROLLBACK if the report fails.

NOACTION means do nothing if the report fails.

Default

ROLLBACK, if a USERID is provided.

NOACTION, if called from an external source (e.g., Form Builder) with no USERID provided.

Usage Notes

- The COMMIT or ROLLBACK for ONFAILURE is performed after the report fails. Other COMMITs and ROLLBACKs can occur prior to this one. For more information, see READONLY.

KEYIN (RWRUN60)

Description KEYIN is the name of a keystroke file that you want to execute at runtime. KEYIN is used to run the keystroke files created with KEYOUT. Since KEYIN is used to execute a keystroke file, it is only relevant when running in a character-mode environment.

Syntax

```
[KEYIN=]keyin_file
```

Values

Any valid key filename in the current directory.

KEYOUT (RWRUN60)

Description KEYOUT is the name of a keystroke file in which you want Reports Runtime to record all of your keystrokes. You can then use KEYIN to execute the keystroke file. KEYOUT and KEYIN are useful when you have certain keystrokes that you want to do each time you run a report. They are also useful for debugging purposes. Since KEYOUT is used to create a keystroke file, it is only relevant when running reports in a character-mode environment.

Syntax

```
[KEYOUT=]keyout_file
```

Values

Any valid filename.

ERRFILE (RWRUN60)

Description ERRFILE is the name of a file in which you want Reports Runtime to store all error messages that are issued during the execution of your report.

Syntax

```
[ERRFILE=]error_file
```

Values

Any valid filename.

LONGCHUNK (RWRUN60)

Description LONGCHUNK is the size (in kilobytes) of the increments in which Reports Runtime retrieves a LONG column value. When retrieving a LONG value, you may want to retrieve it in increments rather than all at once because of memory size restrictions. LONGCHUNK applies only to Oracle7 and Oracle8.

Syntax

[LONGCHUNK=]n

Values

A number from 1 through 9,999. For some operating systems, the upper limit may be lower.

Default

10K

ORIENTATION (RWRUN60)

Description ORIENTATION controls the direction in which the pages of the report will print.

Syntax

```
[ORIENTATION= ] {DEFAULT | LANDSCAPE | PORTRAIT }
```

Values

DEFAULT means use the current printer setting for orientation.

LANDSCAPE

PORTRAIT

Default

DEFAULT

Usage Notes

- If ORIENTATION=LANDSCAPE for a character-mode report, you must ensure that your printer definition file contains a landscape clause.

BACKGROUND (RWRUN60)

Description BACKGROUND specifies whether the report should be run in the background. When BACKGROUND is YES, another process is spawned. This option is useful when you want to work on something else in the foreground while the report runs. On Windows, specifying BACKGROUND=YES means that the report will go into the Reports Background Engine's queue.

Syntax

```
[BACKGROUND= ] {YES|NO}
```

Values

YES or NO

Default

NO

Usage Notes

- On Windows, you can use RWRBE60 to submit a report to the Reports Background Engine for execution. This is a more efficient method because it incurs less performance overhead than Reports Runtime. The command line arguments for RWRBE60 are the same as the ones for RWRUN60.
- On the command line, BACKGROUND=YES is not supported for BATCH=YES, DESTYPE=SCREEN or PREVIEW, or RWBLD60. If you are using SRW.RUN_REPORT, though, BACKGROUND=YES with BATCH=YES is supported.
- If you run a report from Reports Runtime (i.e., not the command line or SRW.RUN_REPORT), you should commit database changes you make in the Before Form, After Form, and Validation triggers before the report runs. When running in this way, these triggers will share the parent process' database connection. When the report is actually executed, however, it will establish its own database connection.

MODE (RWRUN60)

Description MODE specifies whether to run the report in character mode or bitmap. This enables you to run a character-mode report from bit-mapped Reports Runtime or vice versa. For example, if you want to send a report to a Postscript printer from a terminal (e.g., a vt220), you could invoke character-mode RWRUN60 and run the report with MODE=BITMAP. On Windows, specifying MODE=CHARACTER means that the Reports Runtime ASCII driver will be used to produce editable ASCII output.

Syntax

```
[MODE=] { BITMAP | CHARACTER | DEFAULT }
```

Values

BITMAP

DEFAULT means to run the report in the mode of the current executable being used.

CHARACTER

Default

DEFAULT

PRINTJOB (RWRUN60)

Description PRINTJOB specifies whether the Print Job dialog box should be displayed before running a report.

Syntax

[PRINTJOB=]{YES|NO}

Values

YES or NO

Default

YES

Usage Notes

- When a report is run as a spawned process (i.e., one executable, such as RWRUN60, is called from within another executable, such as RWBLD60), the Print Job dialog box will not appear, regardless of PRINTJOB.
- When DESTYPE=MAIL, the Print Job dialog box will not appear, regardless of PRINTJOB.

TRACEFILE (RWRUN60)

Description TRACEFILE is the name of the file in which Reports Runtime logs trace information.

Syntax

```
[TRACEFILE=]tracefile
```

Values

Any valid filename.

Default

trace.dat

Usage Notes

- Trace information can only be generated when running a .rdf file. You cannot specify logging when running a .rep file.
- If you specify LOGFILE or ERRFILE as well as TRACEFILE, all of the trace information will be placed in the most recently specified file. For example, in the following case, all of the specified trace information would be placed in err.log because it is the last file specified in the RWRUN60 command.

```
rwrun60 MODULE=order_entry USERID=scott/tiger  
TRACEFILE=trace.log LOGFILE=mylog.log  
ERRFILE=err.log
```

TRACEMODE (RWRUN60)

Description TRACEMODE indicates whether Reports Runtime should add the trace information to the file or overwrite the entire file.

Syntax

```
[TRACEMODE=] {TRACE_APPEND|TRACE_REPLACE}
```

Values

TRACE_APPEND adds the new information to the end of the file.

TRACE_REPLACE overwrites the file.

Default

TRACE_APPEND

Usage Notes

- Trace information can only be generated when running a .rdf file. You cannot specify logging when running a .rep file.

TRACEOPTS (RWRUN60)

Description TRACEOPTS indicates the tracing information that you want to be logged in the trace file.

Syntax

```
[TRACEOPTS=] {TRACE_ERR|TRACE_PRF|TRACE_APP|TRACE_PLS|  
TRACE_SQL|TRACE_TMS|TRACE_DST|TRACE_ALL} (opt1, opt2, ...)
```

Values

A list of options in parentheses means you want all of the enclosed options to be used.

For example, TRACE_OPTS=(TRACE_APP, TRACE_PRF) means you want TRACE_APP and TRACE_PRF applied.

TRACE_ALL means log all possible trace information in the trace file.

TRACE_APP means log trace information on all the report objects in the trace file.

TRACE_DST means list distribution lists in the trace file. You can use this information to determine which section was sent to which destination.

TRACE_ERR means list error messages and warnings in the trace file.

TRACE_PLS means log trace information on all the PL/SQL objects in the trace file.

TRACE_PRF means log performance statistics in the trace file.

TRACE_SQL means log trace information on all the SQL in the trace file.

TRACE_TMS means enter a timestamp for each entry in the trace file.

Default

TRACE_ALL

Usage Notes

- Trace information can only be generated when running a .rdf file. You cannot specify logging when running a .rep file.

AUTOCOMMIT (RWRUN60)

Description AUTOCOMMIT specifies whether database changes (e.g., CREATE) should be automatically committed to the database. Note that some non-ORACLE databases (e.g., SQL Server) require that AUTOCOMMIT=YES.

Syntax

```
[AUTOCOMMIT=] {YES|NO}
```

Values

YES or NO

Default

NO

NONBLOCKSQL (RWRUN60)

Description NONBLOCKSQL specifies whether to allow other programs to execute while Reports Runtime is fetching data from the database.

Syntax

[NONBLOCKSQL=] {YES|NO}

Values

YES means that other programs can execute while data is being fetched.

NO means that other programs cannot execute while data is being fetched.

Default

YES

ROLE (RWRUN60)

Description ROLE specifies the database role to be checked for the report at runtime. ROLE is useful for giving you the ability to run reports that query database tables to which you would not normally have access privileges.

Syntax

`[ROLE=]rolename/[rolepassword]`

Values

A valid role and (optionally) a role password.

BLANKPAGES (RWRUN60)

Description BLANKPAGES specifies whether to suppress blank pages when you print a report. Use this keyword when there are blank pages in your report output that you do not want to print.

Syntax

[BLANKPAGES=] {YES|NO}

Values

YES means print all blank pages

NO means do not print blank pages

Default

YES (RWRUN60)

Usage Notes BLANKPAGES is especially useful if your logical page spans multiple physical pages (or panels), and you wish to suppress the printing of any blank physical pages.

DISABLEPRINT (RWRUN60)

Description DISABLEPRINT specifies whether to disable **File→Print**, **File→Page Setup**, **File→Choose Printer** (on Motif) and the equivalent toolbar buttons in the Runtime Previewer.

Syntax

[DISABLEPRINT=] {YES|NO}

Values

YES or NO

Default

NO

DISABLEMAIL (RWRUN60)

Description DISABLEMAIL specifies whether to disable the Mail menu and the equivalent toolbar buttons in the Runtime Previewer.

Syntax

[DISABLEMAIL=] {YES|NO}

Values

YES or NO

Default

NO

DISABLEFILE (RWRUN60)

Description DISABLEFILE specifies whether to disable the **File**→**Generate to File** menu in the Runtime Previewer.

Syntax

[DISABLEFILE=] {YES|NO}

Values

YES or NO

Default

NO

DISABLENEW (RWRUN60)

Description DISABLENEW specifies whether to disable the **View→New Previewer** menu to prevent the ability to display a new instance of the Runtime Previewer.

Syntax

[DISABLENEW=] {YES|NO}

Values

YES or NO

Default

NO

DESTINATION (RWRUN60)

Description The DESTINATION keyword allows you to specify the name of a .DST file that defines the distribution for the current run of the report.

Syntax

[DESTINATION=]filename.DST

Values

The name of a .DST file that defines a report or report section distribution.

Usage Notes

- To enable the DESTINATION keyword, you must specify DISTRIBUTE=YES on the command line.

DISTRIBUTE (RWRUN60)

Description DISTRIBUTE enables or disables distributing the report output to multiple destinations, as specified by the distribution list defined in the report distribution definition or a .DST file.

Syntax

```
[DISTRIBUTE=] {YES|NO}
```

Values

YES means to distribute the report to the distribution list.

NO means to ignore the distribution list and output the report as specified by the DESNAME and DESFORMAT parameters. This is fundamentally a debug mode to allow running a report set up for distribution without actually executing the distribution.

Default

NO

Usage Notes

- To enable the DESTINATION keyword, you must specify DISTRIBUTE=YES.

DELIMITER (RWRUN60)

Description DELIMITER specifies the character(s) to use to separate the cells in your report output.

Syntax

[DELIMITER=]value

Values

Any alphanumeric character or string of alphanumeric characters, such as:

- , means a comma separates each cell
- .

You can also use any of these four reserved values:

- tab means a tab separates each cell
- space means a space separates each cell
- return means a new line separates each cell
- none means no delimiter is used

You can also use escape sequences based on the ASCII character set, such as:

- \t means a tab separates each cell
- \n means a new line separates each cell

Default

Tab

Usage Notes

This argument can only be used if you've specified DESFORMAT=DELIMITED.

CELLWRAPPER (RWRUN60)

Description CELLWRAPPER specifies the character(s) that displays around the delimited cells in your report output.

Syntax

[CELLWRAPPER=]value

Values

Any alphanumeric character or string of alphanumeric characters.

- “ means a double quotation mark displays on each side of the cell
- ‘ means a single quotation mark displays on each side of the cell

You can also use any of these four reserved values:

- tab means a tab displays on each side of the cell
- space means a single space displays on each side of the cell
- return means a new line displays on each side of the cell
- none means no cell wrapper is used

You can also use escape sequences based on the ASCII character set, such as:

- \t means a tab displays on each side of the cell
- \n means a new line displays on each side of the cell

Default

None

Usage Notes

- This argument can only be used if you've specified DESFORMAT=DELIMITED.
- The cell wrapper is different from the actual delimiter .

DATEFORMATMASK (RWRUN60)

Description DATEFORMATMASK specifies how date values display in your delimited report output.

Syntax

[DATEFORMATMASK=]mask

Values

Any valid date format mask

Usage Notes

- This argument can only be used if you've specified DESFORMAT=DELIMITED.

NUMBERFORMATMASK (RWRUN60)

Description NUMBERFORMATMASK specifies how number values display in your delimited report output.

Syntax

[NUMBERFORMATMASK=]mask

Values

Any valid number format mask

Usage Notes

- This argument can only be used if you've specified DESFORMAT=DELIMITED.

PAGESTREAM (RWRUN60)

Description PAGESTREAM enables or disables page streaming for the report when formatted as HTML or HTMLCSS output, using the navigation controls set by the report developer using either of the following:

- the Page Navigation Control Type and Page Navigation Control Value properties in the Report Property Palette.
- PL/SQL in a Before Report trigger (SRW.SET_PAGE_NAVIGATION_HTML).

Syntax

```
[PAGESTREAM= ] {YES|NO}
```

Values

YES means to stream the pages.

NO means to output the report without page streaming.

Default

NO

EXPRESS_SERVER (RWRUN60)

EXPRESS_SERVER specifies the Express Server to which you want to connect.

Syntax

```
EXPRESS_SERVER="server=[server]/domain=[domain]/  
user=[userid]/password=[passwd]"
```

Values

A valid connect string enclosed in double quotes (") where

server	is the Express server string (e.g., ncacn_ip_tcp:olap2- pc/sl=x/st=x/ct=x/sv= x/).
Domain	is the Express Server domain.
User	is the userid to log into the Express Server.
Password	is the password for the userid.

The server value contains four parameters that correspond to settings that are made in the Oracle Express Connection Editor and stored in connection (.xcf) files. All four parameters are required and can be specified in any order. The following table describes the parameters and their settings.

Parameter	Description	Setting
sl	Server Login	-2: Host (Domain Login) -1: Host (Server Login) 0: No authentication required 1: Host (Domain Login) and Connect security 2: Host (Domain Login) and Call security 3: Host (Domain Login) and Packet security 4: Host (Domain Login) and Integrity security 5: Host (Domain Login) and Privacy security Notes: Windows NT uses all the settings. Unix systems use only the settings 0, -1, and -2. See the Express Connection Editor Help system for information on these settings.

st	Server Type	1: Express Server
ct	Connection Type	0: Express connection
sv	Server Version	1: Express 6.2 or greater

Usage Notes

- You can have spaces in the string if necessary (e.g., if the userid is John Smith) because the entire string is inside of quotes.
- If a forward slash (/) is required in the string, you must use another forward slash as an escape character. For example, if the domain were tools/reports, the command line should be as follows:

```
EXPRESS_SERVER="server=ncacn_ip_tcp:olap2-pc/sl=0/st=1/ct=0/sv=1/ domain=tools//reports"
```
- You can use single quotes within the string. It is not treated specially because it is enclosed within double quotes.

EXPRESS_SERVER with userid, password, and domain

(RWRUN60) example

```
rwr60 userid=scott/tiger@nt805
express_server="server=ncacn_ip_tcp:olap2-pc/sl=1/st=1/ct=0/
sv=1/ user=orbuild/domain=tools/password=buildme/"
report=exp.rdf destype=file desname=expl.html
desformat=html batch=yes
```

EXPRESS_SERVER without userid, password, and domain

(RWRUN60) example

```
rwr60 userid=scott/tiger@nt805
express_server="server=ncacn_ip_tcp:olap2-pc/sl=0/st=1/ct=0/
sv=1/" report=exp.rdf destype=file desname=expl.html
desformat=html batch=yes
```

CUSTOMIZE (RWRUN60)

Description CUSTOMIZE specifies an XML file or list of files that you want to apply to the report selected in the MODULE or REPORT keyword. The XML file contains customizations (e.g., font changes or color changes) that change the report definition in some way.

For more information, refer to the *Publishing Reports* manual.

Syntax

```
[CUSTOMIZE=]filename.xml|(filename1.xml, filename2.xml, ...)
```

Values

The names of files that contain valid XML report definitions, with path information prefixed to the filenames if necessary.

Usage Notes

Typically, the file extension of an XML report definition is .XML, but it does not have to be when it is used with the CUSTOMIZE keyword.

SAVE_RDF (RWRUN60)

Description SAVE_RDF specifies a file to which you want to save the report selected in the MODULE or REPORT keyword combined with the XML customization file(s) selected in the CUSTOMIZE keyword. This argument is most useful when you have an .RDF file to which you are applying an XML file via the CUSTOMIZE keyword and want to save the combination of the two to a new .RDF file.

For more information, refer to the *Publishing Reports* manual.

Syntax

```
[SAVE_RDF=]filename.rdf
```

Values

Any valid filename.

<param> (RWRUN60)

Description <param> is a parameter that is part of the report's definition. The value you specify is used for that parameter's value.

Syntax

<param>=value

Values

Any valid value for that parameter.

Default

Taken from the Initial Value property of the parameter.

Usage Notes

- You can override the default value on the Runtime Parameter Form.
- Values of arguments may be in single or double quotes. The effect of single or double quotes is operating-system specific.
- If you specify runfile(s) using the REPORT keyword, the PARAM values will be validated using the settings specified for each parameter (e.g., Input Mask, Validation Trigger) in the report(s).
- If you do not specify runfile(s) using the REPORT keyword, the PARAM value is not parsed, or validated, because the parameters may refer to those that have not yet been defined.

Index

- Add property, 298
- Additional Attributes (HTML) property, 299
- Additional Hyperlink Attributes property, 253
- After Form Type property, 373
- After Form Value property, 374
- After Page Type property, 369
- After Page Value property, 370
- After Parameter Form trigger, 152
- After Report trigger, 153
- After Report Type property, 365
- After Report Value property, 366
- Align Summaries with Fields property, 413
- Alignment property, 412
- Application Command Line (PDF) property, 249
- Background Color property, 414
- Base Printing On property, 235
- Before Form Type property, 371
- Before Form Value property, 372
- Before Page Type property, 367
- Before Page Value property, 368
- Before Parameter Form trigger, 154
- Before Report trigger, 155
- Before Report Type property, 363
- Before Report Value property, 364
- Between Field and Labels (Horizontal) property, 415
- Between Frame and Fields (Horizontal) property, 416
- Between Frame and Fields (Vertical) property, 417
- Between Master and Detail (Horizontal) property, 418
- Between Master and Detail (Vertical) property, 419
- Between Page and Frames (Horizontal) property, 420
- Between Page and Frames (Vertical) property, 421
- Between Pages trigger, 156
- Between Sibling Frames (Horizontal) property, 422
- Between Sibling Frames (Vertical) property, 423
- Bookmark property, 248
- Borders property, 424
- Break Order property, 208
- built-ins
 - Report Builder, 97
 - Report Builder, 45, 46, 49, 51, 102, 103, 104, 107, 109, 113, 114, 123, 130
- Character Justification property, 425
- Chart Column property, 205
- Chart Filename property, 198
- Chart Hyperlink property, 199
- Chart Parameter property, 201
- Chart Query property, 203
- Child Column property, 283
- Child Edge Percent property, 167
- Child Edge Type property, 168
- Child Object Name property, 169
- Child Query property, 284
- closing
 - Property Palette, 162
- closing:, 162
- Collapse Horizontally property, 170
- Collapse Vertically property, 171
- column
 - summary column properties, 398
- Column Mode property, 339
- column:, 398
- command line
 - RWBLD60, 481, 482
 - RWCLI60, 536, 537
 - RWCON60, 549, 551
 - RWRUN60, 569, 570
- command line:, 482, 537, 551, 570
- Comment property
 - query, 329
- Comment property:, 329

Comments property, 224
 Compute At property, 399
 Condition property, 285
 Conditional Formatting property, 250
 Contains HTML Tags property, 177
 Contains XML Tags property, 467
 Cross Product Group property, 290
 Dash property, 426
 Datatype property (parameter form field), 317
 Date Justification property, 427
 Design in Character Units property, 375
 Direction property, 354
 Disable Host Menu Item property, 382
 Disable Split Screen Key property, 383
 Disable Zoom Key property, 384
 Display Name property, 251
 Distribution property, 355, 390
 Edge Background Color property, 429
 Edge Foreground Color property, 428
 Edge Pattern property, 430
 End of Layout Section property, 252
 Exclude from XML Output property, 455
 executables
 arguments, 471
 keyword usage, 475
 executables:, 475
 External Query Source File property, 330
 Fields Per Line property, 431
 File Format property, 214
 Fill Pattern property, 432
 filter
 group, 138
 Font property, 433
 Foreground Color property, 434
 Format Mask property, 258
 frame
 properties, 273
 Frame properties, 273
 frame:, 273
 Function property, 402
 Height property, 391
 Hide First Column property, 302
 Horiz. Space Between Frames property, 340
 Horizontal Elasticity property, 225
 Horizontal Panels per Page property, 392
 Horizontal Repeating Frame property, 292
 Hyperlink Destination property, 247
 Hyperlink property, 245
 Icon Name property, 189
 Image property, 435
 Include Bitmapped Objects property, 380
 Include Borders property, 381
 Initial Value property, 303
 Input Mask property, 304
 Inter-Field (Horizontal) property, 436
 Inter-Field (Vertical) property, 437
 Inter-Frame (Horizontal) property, 438
 Inter-Frame (Vertical) property, 439
 Justify property, 440
 Keep With Anchoring Object property, 227
 Label Type property, 187
 Line Stretch with Frame property, 183
 List of Values property, 312
 Max. Horizontal Body Pages property, 351
 Max. Vertical Body Pages property, 352
 Maximum Records Per Page property, 341
 Maximum Rows to Fetch property, 331
 Minimum Widow Records property, 342
 Multimedia Column property, 193
 Multimedia Column Type property, 194
 Multimedia File property, 191
 Multimedia File Type property, 192
 multiple selections
 Property Palette, 161
 Name property, 229, 305
 Name property (Parameter Form boilerplate), 314
 Name property (parameter form field), 318
 Number Justification property, 441
 Number of Pages property, 360
 Number of Records property, 279
 object
 comparing properties, 164
 object:, 164
 OLE2 object
 properties, 295
 OLE2 object:, 295
 OLE2 properties, 295
 Oracle8
 restrictions, 158
 usage notes, 158

- Oracle8:, 158
- Orientation property, 393
- Outer XML Attributes property, 464
- Outer XML Tag property, 461
- Page Break After property, 231
- Page Break Before property, 232
- Page Navigation Control Type property, 361
- Page Navigation Control Value property, 362
- Page Numbering property, 267
- Page Protect property, 233
- Panel Print Order property, 353
- parameter
 - properties, 296
 - system, 297
- Parameter Form
 - boilerplate properties, 313
 - field properties, 316
- Parameter Form:, 313, 316
- Parameter properties, 296
- parameter:, 296, 297
- Parent Column property, 286
- Parent Edge Percent property, 173
- Parent Edge Type property, 174
- Parent Group property, 287
- Parent Object Name property, 175
- PL/SQL
 - group filter, 138
- PL/SQL Formula property, 322
- PL/SQL Trigger property, 195
- Place Labels Above Fields property, 442
- Position property, 443
- Previewer Hint Line Text property, 377
- Previewer Status Line Text property, 379
- Previewer Title property, 356
- Print Direction property, 344
- Print Object On property, 236
- Printer Code After property, 240
- Printer Code Before property, 239
- Product Order property, 404
- properties
 - chart, 196
 - comparing, 164
 - database column, 254
 - frame, 273
 - OLE2 object, 295
 - parameter, 296
 - Parameter Form boilerplate, 313
 - Parameter Form field, 316
 - query, 328
 - ref cursor query, 387
 - report, 348
 - section, 389
 - setting, 163
 - summary column, 398
 - template, 411
- properties:, 163, 164, 273, 295, 296, 313, 316, 328, 348, 387, 389, 398, 411
- Property Palette
 - about, 160
 - closing, 162
 - displaying, 162
 - setting properties, 163
- Property Palette:, 160, 162, 163
- query
 - comment, 329
 - properties, 328
 - ref cursor, 149
- Query properties, 328
- query:, 328, 329
- queueing
 - using Reports Queue Manager (RWRQM60 on Windows), 565
 - using Reports Queue Viewer (rwrqv60 on Unix), 566
- queueing:, 565, 566
- Read from File property, 216
- ref cursor
 - properties, 387
 - query, 149
- ref cursor:, 387
- Remove property, 306
- report
 - properties, 348
- Report Column (for Chart Column) property, 206
- Report Column (for Chart Parameter) property, 202
- Report Group property, 204
- Report Height property, 394
- Report properties, 348
- Report Width property, 395

- report:, 348
- Reports Queue Manager (RWRQM60), 565
- Reports Queue Viewer (rwrqv60), 566
- Reports Web Cartridge (RWCGI60), 535
- Reports Web Cartridge (RWOWS60), 567
- Reset At property, 408
- Restrict List to Predetermined Values
 - property, 307
- Role Name property, 357
- RWBLD60
 - keywords, 478
- RWBLD60:, 478
- RWCGI60, 535
- RWCLI60
 - keywords, 539, 540, 541, 542, 543, 544, 545, 546
- RWCLI60:, 539, 540, 541, 542, 543, 544, 545, 546
- RWCON60
 - keywords, 478, 552, 553, 554, 555, 556, 557, 558, 559, 560, 561, 562
- RWCON60:, 478, 552, 553, 554, 555, 556, 557, 558, 559, 560, 561, 562
- RWOWS60, 567
- RWRBE60, 568
- RWRQM60 (Reports Queue Manager), 565
- rwrqv60 (Reports Queue Viewer), 566
- RWRUN60
 - command line arguments, 570
 - keywords, 478
- RWRUN60:, 478, 570
- section
 - properties, 389
- Section properties, 389
- section:, 389
- SELECT Statement/Static Values property, 308
- Set Break Order property, 221
- Source Datatype property, 257
- Source File Format property, 179
- Source Filename property, 180
- Source property (repeating frame), 346
- SQL Clause property, 288
- SQL Query Statement property, 333
- SRW.ADD_DEFINITION, 42
- SRW.APPLY_DEFINITION, 43
- SRW.BREAK, 44
- SRW.CONTEXT_FAILURE, 45
- SRW.DO_SQL, 46
- SRW.DO_SQL_FAILURE, 49
- SRW.GET_PAGE_NUM, 52
- SRW.GETERR_RUN, 51
- SRW.INTEGER_ERROR, 102
- SRW.MAXROW_INERR, 103
- SRW.MAXROW_UNSET, 104
- SRW.NULL_ARGUMENTS, 107
- SRW.PROGRAM_ABORT, 108
- SRW.REFERENCE, 109
- SRW.RUN_REPORT, 110, 111
- SRW.RUN_REPORT_BATCHNO, 113
- SRW.RUN_REPORT_FAILURE, 114
- SRW.SET_ATTR, 115
- SRW.SET_FIELD, 76
- SRW.SET_FIELD_CHAR, 78
- SRW.SET_FIELD_DATE, 79
- SRW.SET_FIELD_NUM, 80
- SRW.SET_MAXROW, 95, 96
- SRW.SET_PRINTER_TRAY, 99
- SRW.TRACE_END, 121
- SRW.TRACE_START, 123
- SRW.TRUNCATED_VALUE, 125
- SRW.UNKNOWN_QUERY, 126
- SRW.UNKNOWN_USER_EXIT, 127
- SRW.USER_EXIT, 128
- SRW.USER_EXIT_FAILURE, 131
- SRW.USER_EXIT20, 130
- Start in Zoom property, 385
- Style property, 444
- summary column
 - properties, 398
- summary column properties, 398
- summary column:, 398
- Suppress Previewer Title property, 386
- system parameter
 - about, 297
- system parameter:, 297
- template
 - properties, 411
- template:, 411
- Text Color property, 446
- toolbar commands
 - Property Palette, 162

- trigger
 - action, 148
 - format, 144
 - formula, 140
 - report, 133, 135, 152, 153, 154, 155, 156
 - validation, 142
- Type property (boilerplate), 182
- Type property (Parameter Form boilerplate), 315
- Type property (query), 336
- Unit of Measurement property, 350
- Use Previewer Hint Line property, 376
- Use Previewer Status Line property, 378
- Use Vertical Spacing property, 447
- Validation Trigger property, 309
- Value If Null property, 218
- Value property, 310
- Vert. Space Between Frames property, 347
- Vertical Elasticity property, 241
- Vertical Panels per Page property, 396
- Vertical Repeating Frame property, 293
- Visible property, 265
- Web report
 - managing queue (on Windows), 565
 - running dynamically, 535, 567
 - viewing queue (on Unix), 566
- Web report:, 535, 565, 566, 567
- XML Prolog Type property, 458
- XML Prolog Value property, 459
- XML Tag Attributes property, 451
- XML Tag property, 449

