



Reports

Authoring

Visual

Environment

Visual Report Designer

MANUAL
for Reference & Learning

This manual and all material accompanying it is
Copyright © 1995-2008, Nevrona Designs, All Rights Reserved
Rave revision 7.0.5C

Rave Reports

Copyright © 1995-2008, Nevrona Designs, All Rights Reserved

Single User License Agreement

This is a legal Agreement between you, as the end user, and Nevrona Designs. By opening the enclosed sealed disk package, or by using the disk, you are agreeing to be bound by the terms of this Agreement. If you do not agree with the terms of this Agreement, promptly return the unopened disk package and accompanying items, (including written materials), to the place you obtained them for a full refund.

1. Grant of License:

Nevrona Designs grants to you the right to use one copy of the enclosed Nevrona Designs program, (the Software), on a single terminal connected to a single computer (i.e. CPU). You may make one copy of the Software for back-up purposes for use on your own computer. You must reproduce and include the copyright notice on the back-up copy. You may not network the Software or use it on more than a single computer or computer terminal at any time, unless a copy is purchased for each computer or terminal on the network that will use the Software. You may transfer this Software from one computer to another, provided that the Software is used on only one computer at a time. You may not rent or lease the Software, but you may transfer the Software and accompanying written material and this license to another person on a permanent basis provided you retain no copies and the other person agrees to accept the terms and conditions of this Agreement. THIS SOFTWARE MAY NOT BE DISTRIBUTED, IN MODIFIED OR UNMODIFIED FORM, AS PART OF ANY APPLICATION PROGRAM OR OTHER SOFTWARE THAT IS A LIBRARY-TYPE PRODUCT, DEVELOPMENT TOOL OR OPERATING SYSTEM, OR THAT MAY BE COMPETITIVE WITH, OR USED IN LIEU OF, THE PROGRAM PRODUCT, WITHOUT THE EXPRESS WRITTEN PERMISSION OF NEVRONA DESIGNS. This license does include the right to distribute applications using the enclosed software provided the above requirements are met.

2.Term:

This Agreement is effective until you terminate it by destroying the Software, together with all copies. It will also terminate if you fail to follow this agreement. You agree upon termination to destroy the Software, together with all copies thereof.

3. Copyright:

The software is owned by Nevrona Designs and is protected by United States laws and international treaty provisions. Therefore, you must treat the Software like any other copyrighted material (e.g. a book or musical recording) EXCEPT that you may either (a) make one copy of the Software solely for back-up or archival purposes, or (b) transfer the Software to a single hard disk provided you keep the original solely for back-up or archival purposes. You may not copy the written materials accompanying the Software.

All rights reserved. No parts of this work may be reproduced in any form or by any means - graphic, electronic, or mechanical, including photocopying, recording, taping, or information storage and retrieval systems - without the written permission of Nevrona Designs.

Products that are referred to in this document may be either trademarks and/or registered trademarks of the respective owners. The publisher and the author make no claim to these trademarks.

While every precaution has been taken in the preparation of this document, the publisher and the author assume no responsibility for errors or omissions, or for damages resulting from the use of information contained in this document or from the use of programs and source code that may accompany it. In no event shall the publisher and the author be liable for any loss of profit or any other commercial damage caused or alleged to have been caused directly or indirectly by this document.

Prepared: December 2008 in Arizona

Rave Reports

Copyright © 1995-2008, Nevrona Designs, All Rights Reserved

Limited Warranty

1. Limited Warranty:

Nevrona Designs warrants that the disks on which the Software is furnished to be free from defects in material and workmanship, under normal use, for a period of 90 days after the date of the original purchase. If, during this 90-day period, a defect in the disk should occur, the disk may be returned with proof of purchase to Nevrona Designs, which will replace the disk without charge. Nevrona Designs warrants that the Software will perform substantially in accordance with the accompanying written materials. Nevrona Designs does not warrant that the functions contained in the Software will meet your requirements, or any operation of the Software will be uninterrupted or error-free. However, Nevrona Designs will, after being notified of significant errors during the 90-day period, correct demonstrable and significant Software or documentation errors within a reasonable period of time, or refund all or a fair portion of the price you have paid for the Software at Nevrona Designs' option.

2. Disclaimer of Warranties:

Nevrona Designs disclaims all other warranties, either expressed or implied, including but not limited to implied warranties of merchantability of fitness from particular purpose, with respect to the Software and accompanying written materials. This limited warranty gives you specific legal rights, you may have others, varying from state to state. Nevrona Designs will have no consequential damages. In no event, shall Nevrona Designs or its suppliers be liable for damages whatsoever, (including without limitation, damages for loss of business profits, business interruption, loss of business information, or any pecuniary loss), arising out of the use or the inability to this Nevrona Designs product, even if Nevrona Designs has been advised of the possibility of such damages. Some states do not allow the exclusion of limitation of liability for consequential or incidental damages, and this limitation may not apply to you.

3. Sole Remedy:

Nevrona Designs' entire liability in your inclusive remedy shall be, at Nevrona Designs' option, either: (1) The return of the purchase price paid; or (2) Repair or replacement of the Software that does not meet Nevrona Designs' limited warranty, which is returned to Nevrona Designs with a copy of your receipt.

4. Governing Law:

This Agreement will be construed and governed in accordance with laws of the State of Arizona.

5. U.S. Government Restricted Rights:

This Software and documentation are provided with restrictive rights. Use, duplication or disclosure by the Government is subject to restrictions set forth in Section c(1)(ii) of the Rights and Technical Data in Computer Software clause at 52.227-7013.

Nevrona Support

Technical Support

Technical support is provided to registered users for questions or problems with Rave. For fastest service contact us by e-mail or fax. Please include both the Rave version and product serial number (found on the Help About screen) along with any information related to the problem.

Internet: tech@nevrona.com

Web page: <http://www.nevrona.com>

Fax Phone: 602.296-0189

Mailing Address: Nevrona Designs
5301 S Superstition Mountain Dr Ste 104-345
Gold Canyon AZ 85218-1917

News Groups

Several newsgroups are provided free of charge to assist you in getting help with our products. When you visit our newsgroups you will be connected to other users with similar interests. If you have a question, just post it to the newsgroups and others reading the newsgroups will see the message and be able to respond to it. You'll also see questions and solutions from other users as they are posted.

The Nevrona Designs newsgroups are available at <news://news.nevrona.com>.

To access the Nevrona Designs newsgroups, create a new server entry in your newsgroup reader with the Host address of [news.nevrona.com](news://news.nevrona.com) and open that server. You should then see several newsgroups that you can subscribe to.

Sales Support

Internet: sales@nevrona.com

Phone: 480 . 491 - 5492

Table of Contents

Part I Rave	2
1 Introduction	2
RAVE	2
First Glance	3
Navigation Area	4
The Page (Foundation of Rave)	4
Project Tree Panel	5
Property Panel	6
2 Editors	7
Anchor Editor	7
Band Style Editor	8
Color Palette	9
DataText Editor	11
Font Editor	13
3 Main Screen	13
Main Screen	13
Property Panel	14
Project Tree Panel	14
Report Library	15
Global Page Catalog	16
DataView Dictionary	16
Naming Components	17
4 Generating Output	17
Executing Reports	17
Preferences Dialog	18
Report Preview	18
Sending to Printer	22
NDR and PRN files	23
HTML	24
PDF	25
5 Preferences	26
Environment Preferences	26
Designer Preferences	27
Default Preferences	28
Printing Preferences	29
Shortcuts Preferences	31
6 Wizards	31
Simple Wizard	32
Master Detail Wizard	33
Report Expert	35
7 Event Scripting	36
Event Editor	37
Syntax	38
Functions	39
Events	40
Classes	41
Example GreenBar Effect	41
Example Parameters	42
Example OnBeforePrint	42
Example OnGetText	43
Example Section OnPrint	43
Part II Rave Components	48

1	Components	48
2	Toolbar	48
3	DataField Types	49
4	Toolbars	50
	Alignment Toolbar	50
	Colors Toolbar	51
	Designer Toolbar	52
	Fonts Toolbar	52
	Lines Toolbar	52
	Project Toolbar	53
	Zoom Toolbar	53
5	BarCode Components	54
	BarCode Components	56
	PostNetBarCode	57
	I2of5BarCode	58
	Code39BarCode	59
	Code128BarCode	60
	UPCBarCode	61
	EANBarCode	62
6	Drawing Components	63
	Drawing Components	63
	Circle Component	64
	Ellipse Component	64
	Rectangle Component	64
	Square Component	65
7	Project Components	66
	DataView Component	66
	Page Component	66
	ProjectManager Component	67
	Report Component	67
8	Report Components	68
	Report Components	68
	Band Component	68
	CalcController Component	69
	CalcOp Component	70
	CalcText Component	71
	CalcTotal Component	72
	DataBand Component	73
	DataCycle Component	74
	DataMemo Component	75
	DataMirrorSection Component	76
	DataContext Component	78
	Region Component	79
9	Standard Components	80
	Standard Components	80
	Bitmap Component	81
	FontMaster Component	81
	Memo Component	82
	MetaFile Component	84
	PageNumInit Component	84
	Section Component	84
	Text Component	86

Part III Adaptable Reports 90

1	Anchors	90
2	Waste Fit	91

Part IV Batch and Chain Reporting 94

1 Batch Pages	94
2 Calling Pages	95
Calling Pages Lesson	95
3 Chain Pages	97
Chain Pages Lesson	97
4 Different First Page format	97
Different First Page Lesson	98
5 Different Odd/Even Page format	98
Different Odd/Even Page Lesson	98

Part V Property Descriptions 100

1 AllowSplit property	100
2 AlwaysGenerate property	100
3 Anchor property	100
4 AutoSize property	101
5 BandStyle property	101
6 BarCodeJustify property	101
7 BarCodeRotation property	102
8 BarHeight property	102
9 BarTop property	102
10 BarWidth property	102
11 Bin property	103
12 BinCustom property	103
13 BorderColor property	103
14 BorderStyle property	104
15 BorderWidth property	104
16 BorderWidthType property	104
17 Bottom property	105
18 CalcType property	105
19 CalcVar property	105
20 Categories property	106
21 Category property	106
22 Center property	106
23 CodePage property	106
24 Collate property	107
25 Color property	107
26 Columns property	107
27 ColumnSpacing property	108
28 ConnectionName property	108
29 ContainsRTF property	108
30 Controller property	108
31 ControllerBand property	109
32 Copies property	109
33 CountBlanks property	109
34 CountNulls property	110

35	CountValue property	110
36	Cursor property	110
37	DataField property	111
38	DataView property	111
39	Description property	111
40	DesignerHide property	112
41	DestParam property	112
42	DestPIVar property	112
43	DetailKey property	113
44	DevLocked property	113
45	DisplayFormat property	113
46	DisplayOn property	114
47	DisplayType property	114
48	Duplex property	114
49	ExpandParent property	115
50	Extended property	115
51	FieldName property	115
52	FileLink property	116
53	FillColor property	116
54	FillStyle property	116
55	FinishNewPage property	117
56	FirstPage property	117
57	Font property	117
58	FontJustify property	117
59	FontMirror property	118
60	FullName property	118
61	GotoMode property	118
62	GotoPage property	119
63	GridLines property	119
64	GridSpacing property	119
65	GroupDataView property	120
66	GroupKey property	120
67	Height property	120
68	HRadius property	121
69	Image property	121
70	InitCalcVar property	121
71	InitDataField property	122
72	InitDataView property	122
73	Initializer property	122
74	InitToFirst property	123
75	InitValue property	123
76	KeepBodyTogether property	123
77	KeepRowTogether property	124
78	Left property	124
79	LineStyle property	124

80	LineWidth property	125
81	LineWidthType property	125
82	Locked property	125
83	LookupDataView property	126
84	LookupDisplay property	126
85	LookupField property	126
86	LookupInvalid property	126
87	MailMergeItems property	127
88	MasterDataView property	127
89	MasterKey property	127
90	MatchSide property	128
91	MaxPages property	128
92	MaxRows property	128
93	MinHeightLeft property	129
94	Mirror property	129
95	Name property	129
96	NullText property	130
97	Operator property	130
98	Orientation property	130
99	OrphanRows property	131
100	PageHeight property	131
101	PageList property	131
102	PageWidth property	131
103	PaperSize property	132
104	Parameters property	132
105	PIVars property	132
106	PositionMode property	133
107	PositionValue property	133
108	PrintChecksum property	133
109	Printer property	134
110	PrintReadable property	134
111	PrintTop property	134
112	ReprintLocs property	135
113	Resolution property	135
114	ResultFunction property	135
115	Right property	136
116	Rotation property	136
117	RunningTotal property	137
118	Size property	137
119	SortKey property	137
120	Src1CalcVar property	138
121	Src1DataField property	138
122	Src1DataView property	138
123	Src1Function property	139
124	Src1Value property	139

125	StartNewPage property	139
126	Tag property	140
127	Text property	140
128	TextFalse property	140
129	TextJustify property	141
130	TextTrue property	141
131	Top property	141
132	Truncate property	142
133	Units property	142
134	UnitsFactor property	142
135	UseChecksum property	143
136	Visible property	143
137	VRadius property	143
138	WasteFit property	144
139	WideFactor property	144
140	WidowRows property	144
141	Width property	145

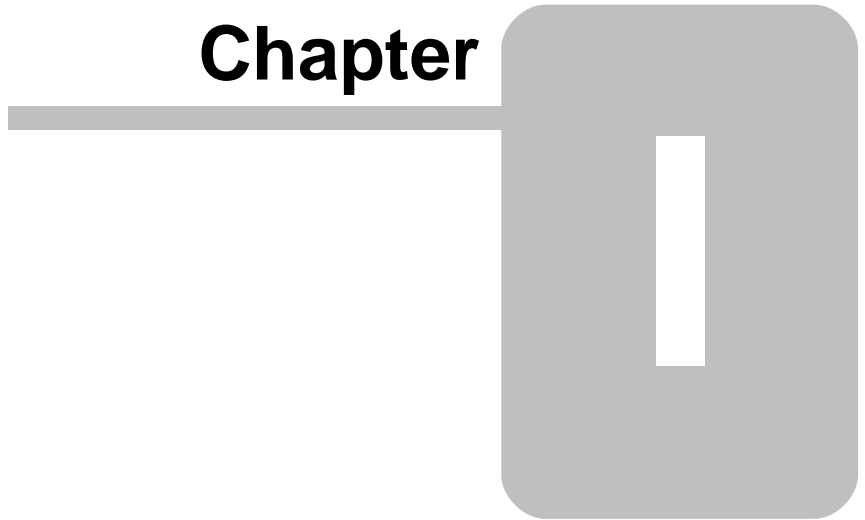
Part VI Appendix 148

1	Lessons	148
	Quick Start	148
	CalcOp	149
	Calculated Fields	153
	Custom Data Connection	154
	Data Connections	158
	Font Master	159
2	FAQ	165
	Design - Footers	165
	Design - Header on ALL pages	165
	Design - Page Numbers	166
	Error - Duplicate License	166
	Events - Getting & Setting Parameters	167
	Events - Hiding Bands or Components	167
	Font - Dynamically Changing Fonts	168
	Font - Missing Text	168
	Font - Overlapping Words	169
	Font - Unicode Support	169
3	Format Codes	169
	Alphanumeric Items	170
	Date Time Items	173
4	Shortcuts	176
5	Variables	177

Index 179

Rave

Chapter



1 Rave

1.1 Introduction

What's All the RAVE About?



Reporting can be one of the most complex, yet most important tasks for anyone dealing with a database. Reports are the primary visual means to express information retrieved from a body of data. To solve the problems associated with presenting a visual report of data in a meaningful and informative manner, traditional visual reporting tools have offered banded layout tools geared towards table-style listings of data. Today, however, much more complex reporting requirements exist and are not easily handled by banded layout tools.

Welcome to the next level in visual reporting! The Rave visual designer offers many unique features that help to make the reporting process simpler, quicker and more efficient. Rave is an intuitive page based visual design environment that can easily handle a wide variety of report formats, much more than a purely banded style tool. Rave also includes mirroring and other technologies to encourage you to reuse the contents of your reports for quicker changes and easier maintenance. In general though, Rave has been designed to offer the most flexibility and functionality in an easy to learn format.

Where do you begin? Since Rave is page-based designer, many of its features should be easy to use with only a little practice. There are a lot of options and some might not be obvious when just starting. Remember that many of these options can be ignored in the beginning, but as your needs and knowledge increases these options are readily available. In fact, Wizards generate "standard" reports without having to know behind-the-scene details. However, we do recommend that you take some time and do a quick read of this manual.

Included with Rave is a project called "RaveDemo" that contains several report samples. To see common designs, start the RaveDemo.exe and open the project to access the different report types. Exploring the RaveDemo project and other samples are excellent ways to learn Rave.

1.1.1 RAVE

A report can be described as data presented in a visual manner, whether it is on paper or displayed electronically. Typically, there are sets of database tables that provide data to create a report. For example, suppose there are: a Customer table, a Products table, and an Items Sold table. These tables could be combined to produce form letters, invoices or customer lists.



Let's begin with a quick overview of Rave. When you start the Rave program, the first thing seen will be a sheet representing a page on the screen as well as two windows on the side of the page and toolbars across the top of the page. There appears to be many items in the Rave designer, so where do you start? Let's start with what is shown when Rave is running. First, understand that there are two groups of toolbars displayed in the Rave designer; components and tools.

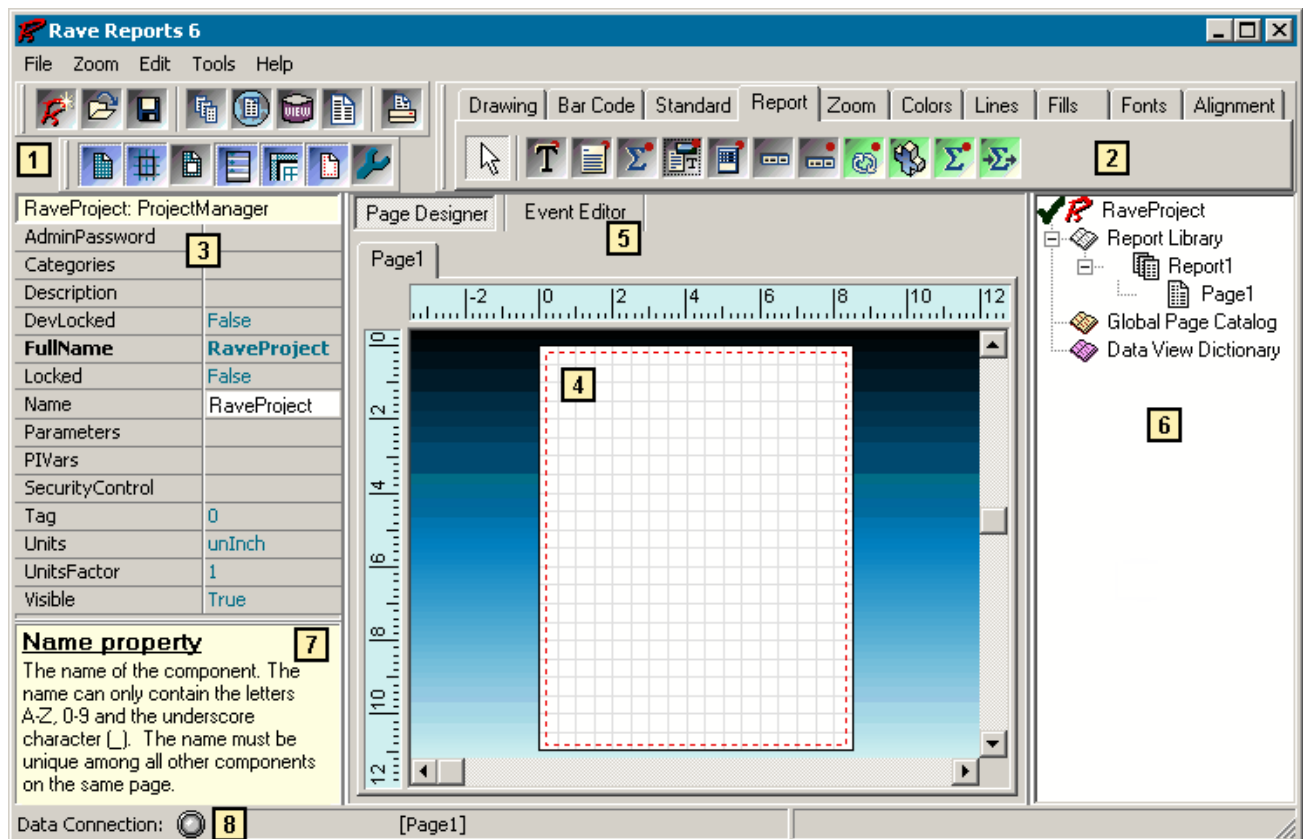
Components are items that are "dropped" on the page editor. These might be bands, bar codes, lines, shapes, etc. Don't worry about what these are yet. If the object can be seen on the page layout, it is a component.

Toolbars have change or modify components, thus distinguishing them as tools. There are several toolbars: alignment, color palette, font editor, etc. If there is a box on the page and it needs to be filled with a background color, first select that item (the box) by clicking on it. Then, use the color tool to change the fill color to any desired color.

That was a fast overview. There are many settings and properties that control the behavior of almost every part of the visual designer. This manual has two main parts, a description of the Rave system including the components and tools, then a reference section that list details about each property. First, go through the description section, then browse the reference section to get a better understanding on just how much control can be had over various design features.

1.1.2 First Glance

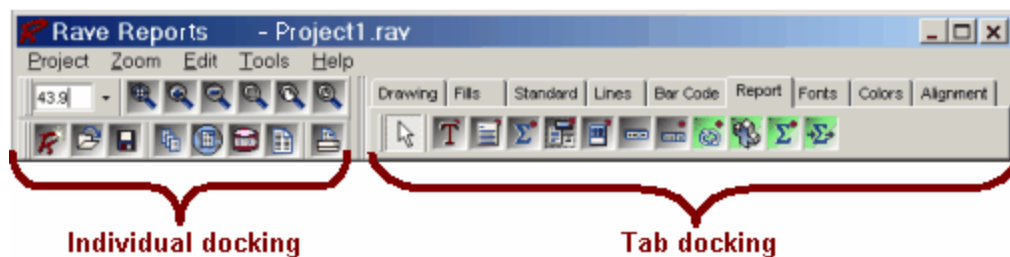
When Rave first starts, it will open a screen that will look similar to the image below. We are going to first explore each part of this screen to give a general tour of the Rave Designer.



- 1) Individually docked toolbars
- 2) Tab docked toolbars
- 3) Property Panel
- 4) Page Layout
- 5) Event Editor tab
- 6) Project Tree Panel
- 7) Short description of selected property
- 8) Data connection status LED

1.1.3 Navigation Area

The top of the visual designer is the Rave Navigation area. This is where toolbars can be docked individually, and can also be tab docked. Individually docking the toolbars will allow the user to see all the buttons on the toolbar and to see more than one toolbar at a time. Tab docking will allow easy access to each toolbar, but only one toolbar can be seen at a time.



The toolbars contain both tools and components. A tool is a feature that will be used to modify an object already on the page, like the Font Editor or the Color Palette. A component is an object that will be placed on the page like a band, line, text, region, or section.

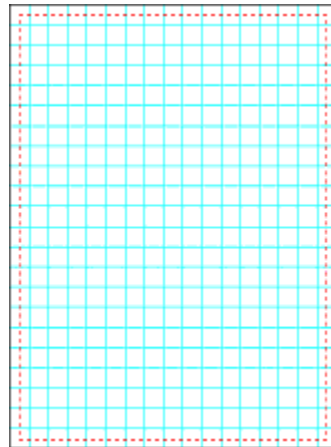


To find out the name of the button control on a toolbar, simply move the mouse pointer over a button, and a popup window will appear with the name of the tool or component. Also, notice the raising of the button when it is about to be selected. It will become 'depressed' when the button is clicked on.

1.1.4 The Page (Foundation of Rave)

The starting point with the Rave Visual Designer is the Page. The page is the foundation and is where all the designing action is done. The Page is represented with the grid pattern, which will look something like the image shown below. The look and feel of the Page can be changed with the preference settings, which will be covered in Preferences section.

One important thing to know and remember is that the Page has properties, such as height and width. To see or change the Page properties, go to the "Project Tree" Panel and expand the Report node by clicking on the "+" sign, then click on the report name (default is "Report1") and this will show a line that will be defaulted to "Page1". Click once on the Page reference and a green check mark will appear next to the Page name. The green check mark means that the Page has been selected and now the Page properties can be seen in the Property Panel.

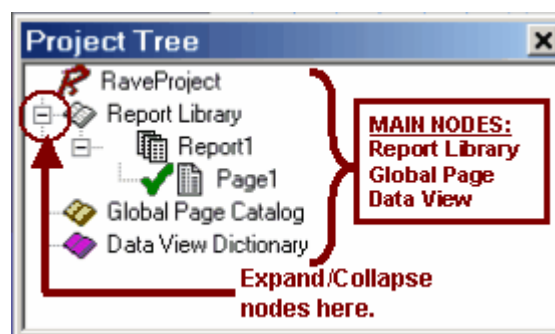


Page1: Page	
Bin	
BinCustom	
Description	
DevLocked	False
FullName	Page1
GotoMode	gmGotoDone
GotoPage	
GridLines	5
GridSpacing	0.1
Locked	False
Name	Page1
Orientation	poDefault
PageHeight	11
PageWidth	8.5
PaperSize	Custom
Parameters	
PIVars	
Tag	0
Visible	True
WasteFit	False

1.1.5 Project Tree Panel

The Project Tree panel is a very informative part of the Rave designer and it also provides an easy way to navigate the reporting project structure.

For now just the parts of the Project Tree will be examined. More details are covered in the "Main Screen" "Project Tree Panel" section. There are three main nodes in the Project Tree: Report Library, Global Page Catalog, and Data View Dictionary. Each of these nodes (and any sub nodes) can be expanded or collapsed by clicking on the plus/minus symbol. Sub-nodes can be created and added, by selecting a desired option (New Report, New Global Page, and New Data Object) from the Project Menu.



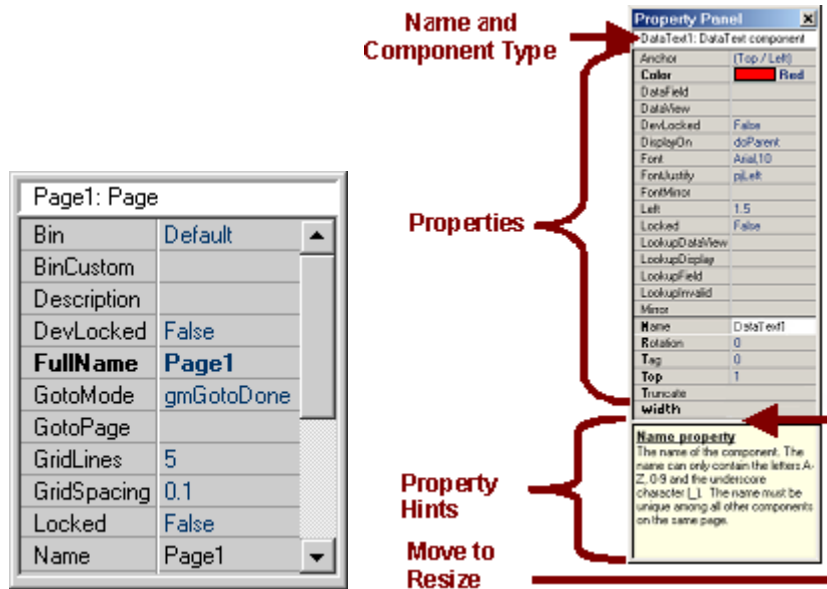
The Report Library node is where all of the reports within the project are contained. Each report will have one or more pages. Each of those pages will normally have one or more components within them.

The Global Page Catalog node is where reporting templates are managed. The reporting templates can contain one or more components. These reporting templates can then be reused via Rave's unique mirroring technology. This could include items such as letter headings and footers, pre-printed forms, watermark designs or complete page definitions that could be the foundation for other reports.

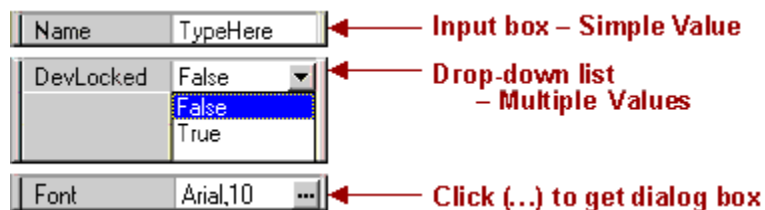
The Data View Dictionary node is where all the data connections for reports are defined. A data view retrieves data from the application through data connections installed within that application.

1.1.6 Property Panel

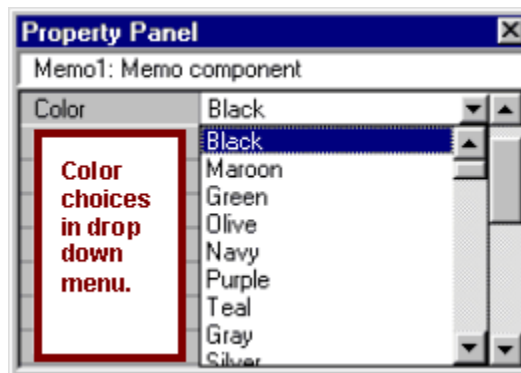
The Property Panel helps to customize the way components appear or behave. When a component is selected on the page, the Property Panel will reflect the selection by displaying the different properties associated with the selected component.



Changing the properties values is done by using various drop-down menus and edit boxes. If no component is selected, then the property panel will appear blank with no options to choose from.



Another way to change a property is select a value from a list of possible choices. For example, the Color property has a down arrow button. Clicking on the down arrow button will display a list of colors that can be selected. Any property that has a list of choices can also be double-clicked (instead of clicking on the down arrow button) to advance to the next item in the list.

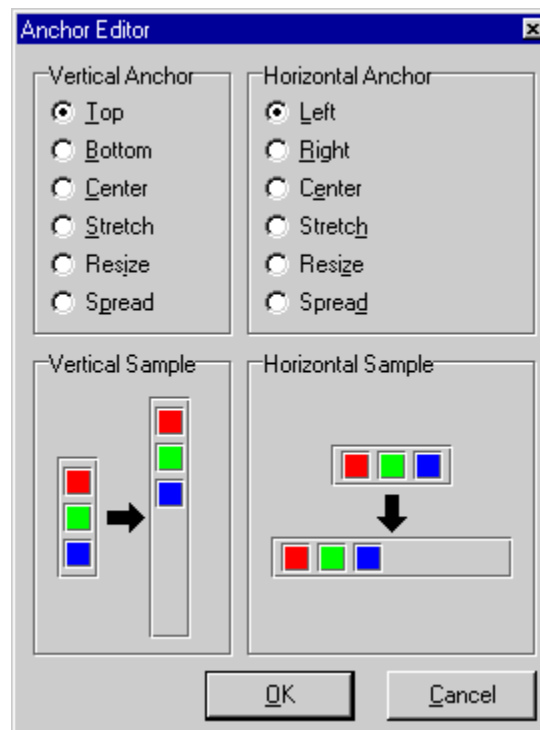


1.2 Editors

1.2.1 Anchor Editor

Normally, report bands / columns / paragraphs are justified to the Left and Top of a design area. This will be fine for the majority of reporting needs. However, have you ever wanted to control the starting or ending position of your report components dynamically? This could be a two column section, where the left column is a memo component which changes in height and the right column is a text component which is always a one line item. How do you make both components "float" and align to the bottom of the design block? In Rave that is done with the *Anchor* property.

To change the anchor style you go to the *Anchor* property of the desired component and click on the ellipse symbol, it will open the Anchor editor like the one shown above. This provides a method to select the anchor style you want for that component by using the appropriate vertical and horizontal radio button. Note that a representation appears below each anchor selection to give you a visual indication of what that setting is designed to accomplish. The last three settings, Stretch, Resize, Spread are a little difficult to explain but if you watch the sample you will get a picture showing the differences between them.



An important point about the *Anchor* property settings is that they are relative to that components parent. So if the parent is a page then the settings are relative to the page margins. If the parent is a section then the settings are relative to the section borders. One way to visually see the parentage is to examine the component(s) on the Report Node in question in the Project Tree Panel. Go up one level and that is the parent.

You certainly are not restricted to the following combinations, but normally the anchor settings will be paired as follows:

- Left/Top justified
- Right/Bottom justified
- both Center justified
- both Stretch
- both Resize
- both Spread

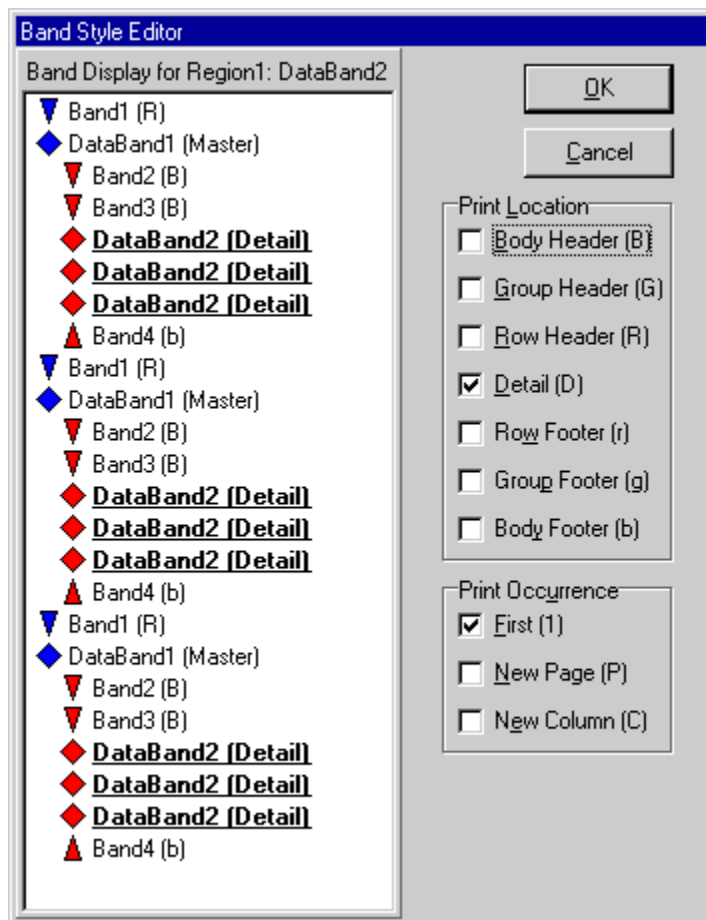
[ExpandParent](#), [Waste Fit](#)

Editors:

[Band Style](#), [DataText](#), [Fonts](#), [Line](#)

1.2.2 Band Style Editor

Select the BandStyle property on a DataBand. Click on the ellipse symbol it will open the Band Style Editor like the one shown below. This provides a simple method to select the features you want for that band by using the check boxes to activate or deactivate them. Note that a band can have several different features active at a time. This means that it is even possible for a band to be both a header and footer at the same time.



The display area in the Band Style editor has been designed to give you a representation of the flow of your report in pseudo layout style. The DataBand(s) are duplicated 3 times on purpose to show that *this is a repeating data area* (usually detail area). The active band for the editor will be displayed with both Bold and Underline formatting.

Both symbols and letters are used on the Band Style editor display area and one the bands in the page layout area and are designed to inform you about each bands behavior. The major difference between these two representations is that the band style editor display will arrange the bands in a pseudo flow according to the definitions of each band. The region display of the bands will be arranged in the order that you placed them during design. The order of operation is controlled in some cases by this order. Headers (capital letters) will print first, then the DataBand, then the footers (lower case letters) for each level. However, if you have more than one header defined for a particular level then each header band will process in the order that you have them in the region. So it technically possible to put all the headers at the top, all the DataBand's in middle and all the footers at the bottom of a region for all levels of a master-detail. Or you could "group" each level, with the appropriate headers, footers and DataBands all together for each level. RAVE allows you to use the region style that makes the most sense to you. Just remember the order of precedence of like bands at the same level is controlled by their order within the region.

There are several symbols that are designed to show the parent - child / master - detail relations of the various bands. The triangle symbol (up / down arrows) indicates that band is controlled by a master band with the same color (level) and can be found in the direction of the arrow. The Diamond symbol represents a Master or controlling band. These symbols are both color coded and indented to represent the level of Master - Detail flow. Remember that we could have Master - Detail - Detail where both details are both controlled by the same master or one of the details could be controlled by the other detail.

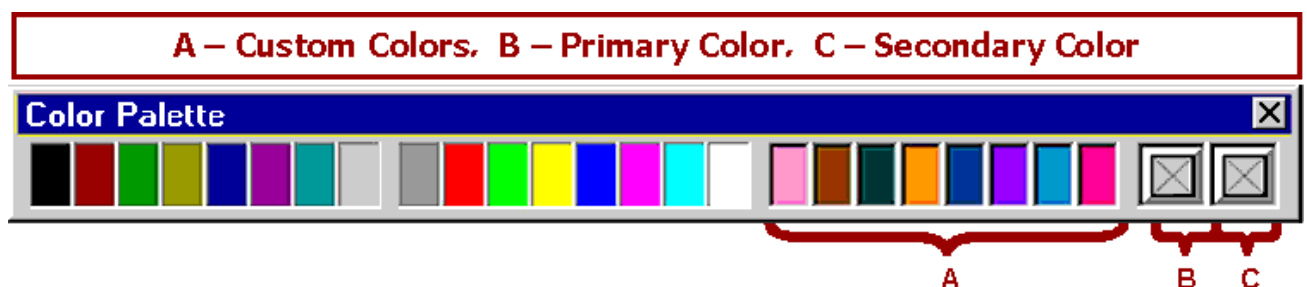
The title bar of each band contains information about that band. On the left side of the band is a name that indicates the region it is in - "RegionName:BandName". The right side of Band uses several letters to remind you of the band style settings for that band. The order of these letters on a master band is "MASTER 1PC". The order of these letters on a controlled band is "BGRDrgb1PC". If the letter is subdued (gray) then that setting is inactive (off). If the letter is bold then the setting is active (ON). The following table shows the various letters and what they mean.

	Header		Footer
Body	B		b
Group	G		g
Row	R		r
Detail		D	
First Page		1	
New Page		P	
New Column		C	

Editors: [Anchor](#), [DataText](#), [Fonts](#), [Line](#)

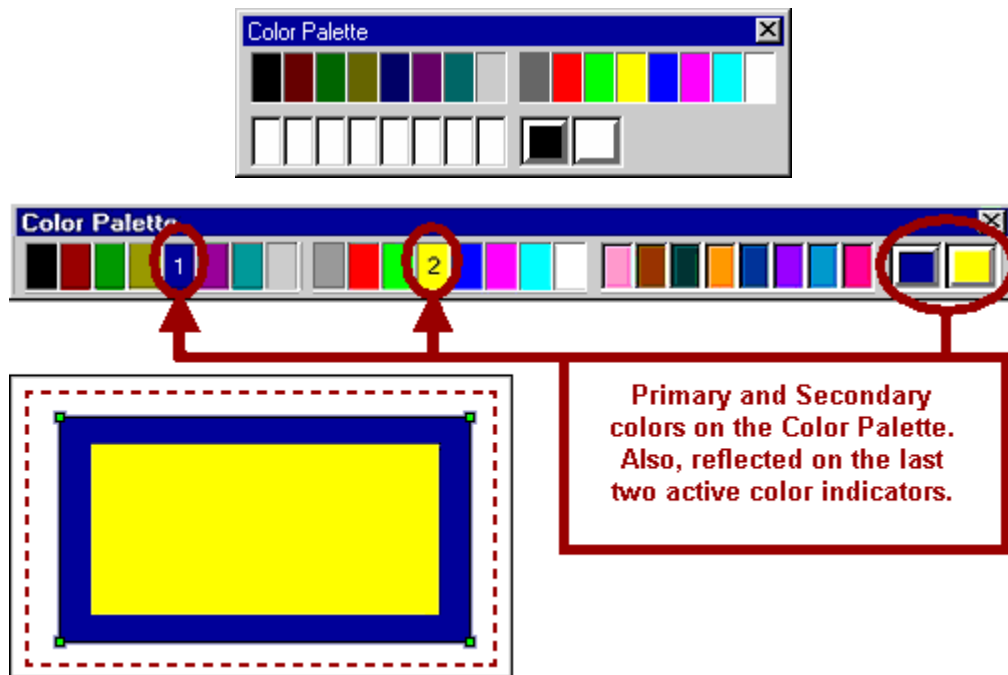
1.2.3 Color Palette

The Color Palette is the tool used to change a color of a selected object such as a line, box text, or the fill area in one of the shapes (circle, ellipse, rectangle or square).



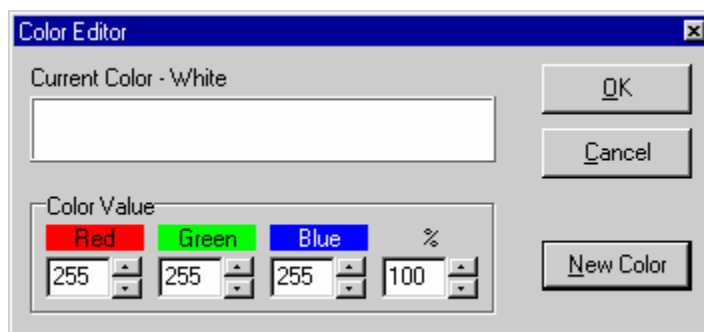
In the figure above, the section labeled A displays any custom colors created by the user. Section B displays the Primary color currently selected. Section C displays the currently selected secondary color. Any of the other color squares serve as the actual color palette from which A, B, and C may be created or selected.

To change the color of an object, select the object and either left or right click on the Color Palette. A left-click selects the foreground or primary color while the right-click selects the background or secondary color. When a primary color is selected, a "1" will be placed on the corresponding color on the palette. Similarly, a "2" will be placed indicating the secondary color. This is in addition to sections B and C explained previously.

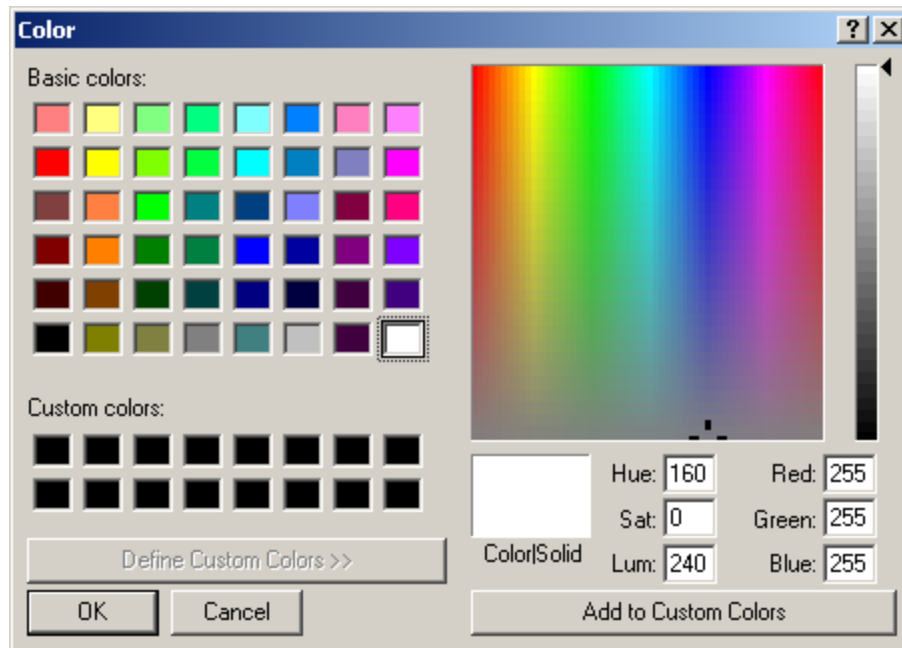


In the figure above, to change the border of the rectangle, first select it and then click on the desired color with the left mouse button (in this case Navy Blue). To change the fill color, click with the right mouse button on the color Yellow.

Double-clicking on the user defined Custom Colors (section A) or on the foreground and background color boxes (section B and C) will open a more detailed color control window called the Color Editor. From the Color Editor, custom colors can be created by adjusting the red, green and blue values, as well as the saturation of the color by increasing or decreasing the color percentage.



To create and save a new color, use one of the Custom Colors to enter into the Color Editor, then click on the New Color button and choose the desired color from the Color dialog box. By entering the Color Editor from one of the Custom Color boxes, the new color will be saved. If the Color Editor is entered from either the primary or secondary color boxes, the color will only be saved in those boxes until a new color is chosen (either by clicking on the right or left mouse button or by the creation of another color).



It is important to note that the actual exhibited colors are dependent on the display settings of the computer. If there are issues with colored objects when transferring reports from one system to another, check the display color settings. A 256-color setting only uses 256 colors for display, whereas a True Color setting uses millions of colors. Thus, for example, if a report is created on a system set at True Color, and then viewed on another system set at 256 Colors, some objects may appear grainy or to have the wrong color shadings.

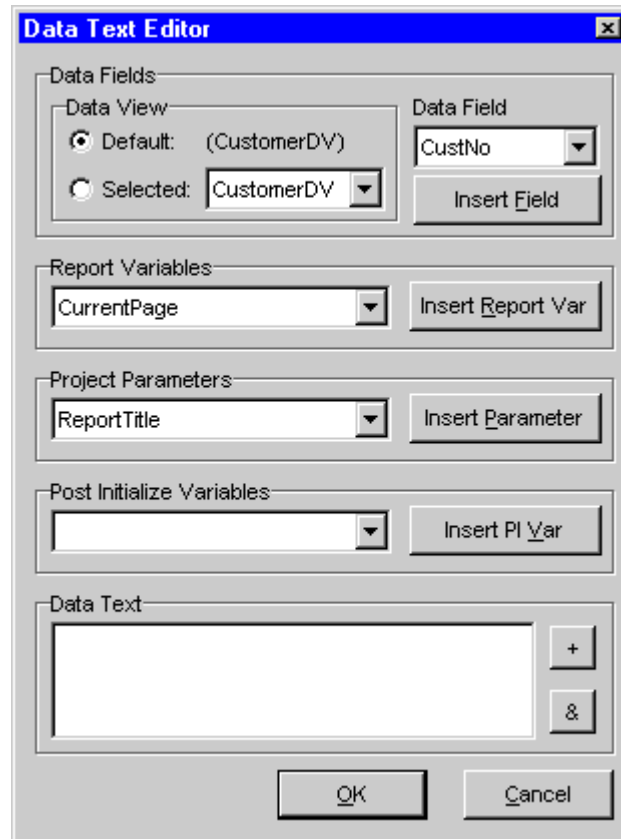
1.2.4 DataText Editor

You have two options for entering data in a DataField property. You can either select a single field using the drop list option. This is fine for normal database reporting needs where you only want a single data field for each data text item. However, there are reporting requirements where you want to combine various fields together. Two common examples are City State and Zip Code or Firstname Lastname combinations. In code this would be accomplished using a statement like the following:

City + ',_' + State + '___' + Zip or FirstName + "___" + LastName

NOTE: the underscore character represents a space.

The DataField property has a Data Text Editor which assists you in building complex composite fields. To do this you click on the ellipsis and open the Data Text Editor. This editor will give the power to concatenate fields, parameters, or variables together to build a very complex data aware text field simply by dropping the different list boxes and selecting the item you want. There is a lot of combinations in this editor, we will cover them quickly here, but try the different combinations your self and it should help the learning curve.



Note that the dialog box is divided into several groups, Data Fields, Report Variables, Project Parameters, Post Initialize Variables and Data Text. Data Text is the result window. So watch this window as you insert different items. The two buttons on the right side of this window are + or &. The "plus" sign will add the two items together with no spaces while the "&" will concatenate them with a single space. So your first step is decide on doing a + or &, then selecting the text from one of the three groups above the Data Text window.

If you want to add the field "OrderNo" to the "CustNo", then click once on the "+" sign, go up to the Data Fields group, drop the Data Field list box, select "OrderNo". Then click once on the "Insert Field" button and that will be added for you in the Data Text window. You could add more even more data fields. Notice the "Selected" item in the Data View group. If you have more than one data view active, then you could select another Data View, and then pick a field from that Data View. However, do not restrict yourself to thinking of combining only data fields. You can combine "Report Variables" or "Project Parameters". Go to the "Report Variables" group and drop the list box for variables and notice the ones that are already available.

Another item available to you is project parameters. This could be a "UserName", "ReportTitle" or "UserOption" parameter from the code base side of Rave. To create the list of "Project Parameters", select the "RaveProject" in the project tree panel (very top item). Down in the "Properties" panel will be a "Parameters" property. Click on the ellipsis and you will get a typical strings editor where you can enter the different parameters that you will pass to Rave from your application, like "UserName" etc.

CAUTION

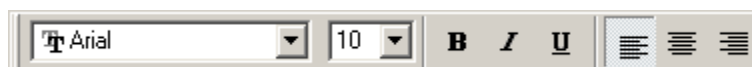
Remember to use a "+" or "&" between each item that you are combining in the Data Text window. You can type in the Data Text window, so you can correct errors by highlighting, deleting or replacing erroneous entry's made in the data text stream.

See Also:

[Anchor](#), [Anchor Editor](#), [Band Style Editor](#), [Font Editor](#), [Line Toolbar](#), [Report Variables](#)

1.2.5 Font Editor

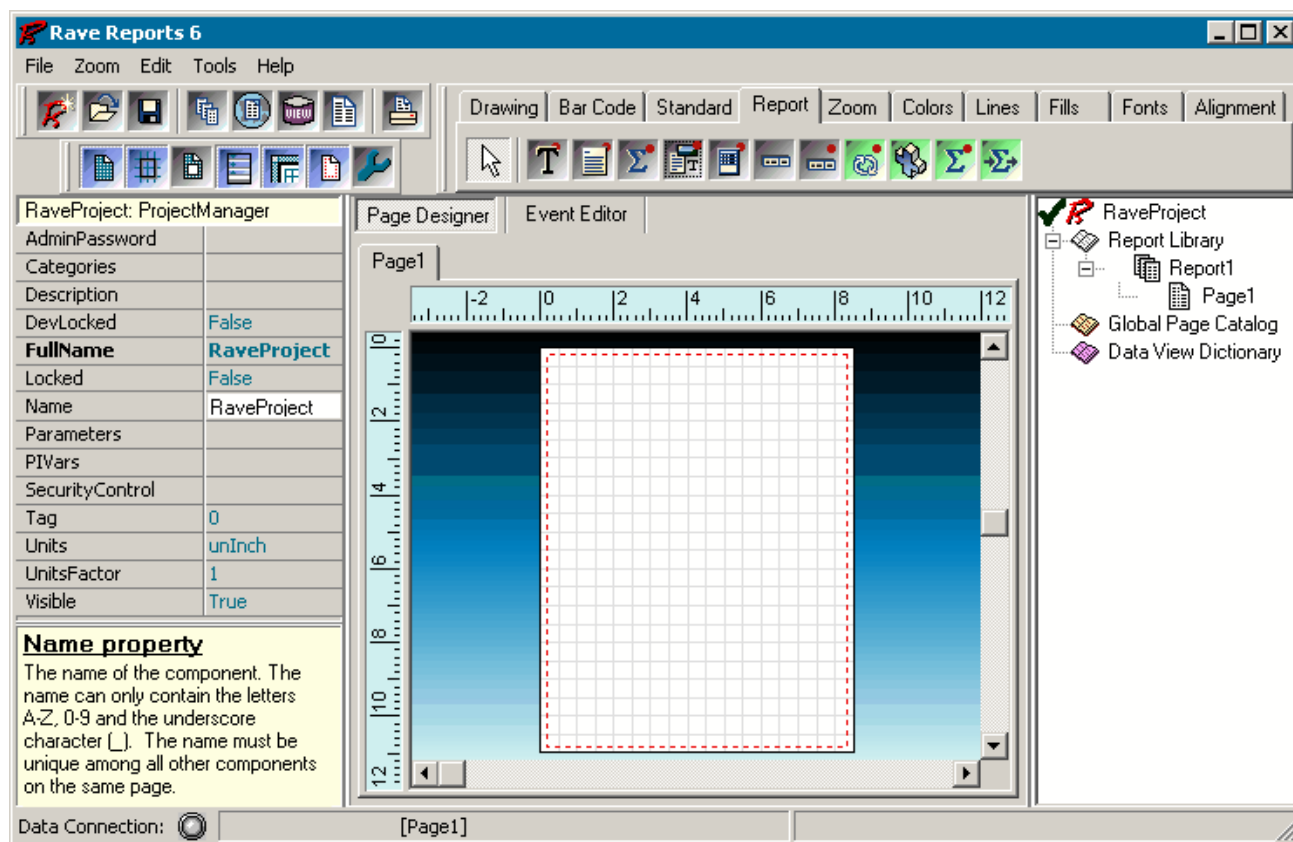
If the selected item is text, then you can use the Font Editor to change the font type, size, attributes or position of the text in the text box. This toolbar is similar to those you would find in a word processor. It will allow you to change the font name and the attributes for the selected item. For instance, you can set the size of the font, whether it is bold, italic or underlined.

**See Also:**

[Anchor](#), [Band Style Editor](#), [DataText Editor](#), [Line Toolbar](#)

1.3 Main Screen**1.3.1 Main Screen**

When you first start RAVE, it will open a screen that looks something like the following. Since RAVE remembers what you did your last session, your screen may look a little different. But this is the fundamental pieces with all menus closed. What we are going to do first is explore each part of this screen. After that we will start putting the pieces together to build some reports.



[COMPONENTS](#), [Standard](#), [Drawing](#), [Report](#), [Bar Code](#)
[TOOLS](#), [Project](#), [Designer](#), [Zoom](#), [Alignment](#), [Color Palette](#), [Line Editor](#), [Font Editor](#)

[Keyboard / Mouse](#), [Report Variables](#), [Shortcuts](#)

PREFERENCES [Default](#), [Designer](#), [Environment](#), [Packages](#), [Printing](#), [Shortcuts](#)

1.3.2 Property Panel

The property panel will allow you to customize the way a component(s) appear or behave. If no component is selected then the property panel will be blank. You select a component on the page layout panel by clicking on it.

There is a special property for the components call "*locked*". If the "*locked*" property is true then you can select that component but will not be able to change it's properties. If you have selected a component(s) that are locked then the pips and the property name for those items will be red. The locked property state will show either a True / False indicating the lock status or it will show the parent component name that is controlling its lock state.

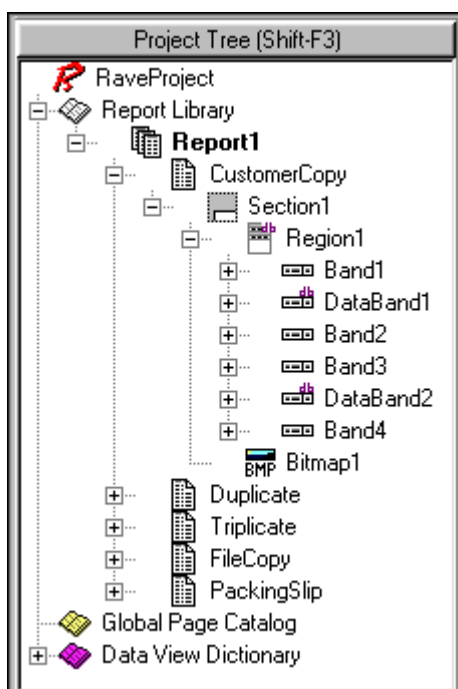
When you select a component, it gets colored pips around it. Remember "*locked*" components always have red pips. The remaining discussion is about component(s) that are NOT locked. If you only have one item selected the pips will be light green in color. If you have multiple items selected the pips will be gray. If it is a mirrored component then the pips will be yellow. You can change the common properties by clicking on the property you wish to change as long as it is not a locked component.

Many of the properties have standard values like colors, True or False or positions. For properties like True or False, you can click on the word to change the value to its opposite. Other properties have associated property editors to set more complicated sets of properties. For example, the Font property has an ellipsis. If you click on the ellipsis, then the Font Editor will appear where you can make your changes. The color property has a down arrow. If you click on the down arrow you will get a list of colors that you can select.

When you selected several different items on a page, the property panel will change to reflect only the common properties. But you could then make one change to a property and it would change for all the selected items. You can change the width of the property panel to your desired size.

1.3.3 Project Tree Panel

The Project Tree panel provides an easy way to navigate your design structure. This tree style view shows your reports and their design structures in an overview or outline type fashion. It is a very informative part of the RAVE designer and once you have designed some reports simple or complex, remember to visit it and you will see what we mean.



Note that the three main areas in the project tree (Report Library, Global Page Catalog and Data View Dictionary) can contain multiple definitions within each category. Each node (and any sub nodes) can be expanded or collapsed by clicking on the plus/minus symbol. You can add items to any of these nodes from the "Project" option on the main menu. There you will see an option for each of the nodes "New Report", "New Global Page" and "New Data View". If you wish to add a "New Page" to the Report Library node that is done through the "Report" option from the main menu. You can change the width of the tree panel to your desired size.

1.3.3.1 Report Library

The Report Library node of the Project Tree is where the design structure including all the components you used in each of your report(s) will be displayed. You probably will get use to selecting a component on the page layout panel by simply clicking on that component. However, you can use the report library tree to select a component(s) by simply clicking on the component name. A green check mark will appear next to the selected item(s). If you want to select multiple components then hold the shift key down while selecting them.

So you can have a complex reporting project with many report definitions. Each report you create would have it's own design structure. You will be able to select / switch to a different report simply by clicking on the on that report name in the report library tree structure.

That brings us back to a point that helps even in simple designs, use the "Name" property. The "Name" property will default to something for you automatically like Report1, Report2 and so on. However, wouldn't it be easier to maintain a reporting system if the reports had understandable name. They can with the "Name" property. You can be creative but remember no spaces or specials characters are allowed in the "Name" property. So if you had a project that dealt with several reports dealing with customers, invoicing etc., wouldn't "CustomerList", "CustomerLabels", "CustomerDue", "Invoice", "PO", "ProductsOnHand", "ProductsOnOrder" be clearer than "Report1" through "Report7".

1.3.3.2 Global Page Catalog

The Global Page Catalog node of the Project Tree is where you would keep page definitions that are to be masters and could be mirrored. This includes things like Letter Heads, Forms, watermark designs, and other page definitions that could be the foundation for several reports. This can even be a complete report design. An example would be mirroring a global page where you wanted to print the same contents (link an "Invoice"), but you wanted to put a different caption at the bottom of the pages like "Original", "File Copy", "Shipping".

1.3.3.3 DataView Dictionary





RAVE has been designed to work with any data connection component that is useable with Delphi or C++Builder. Whether you are using the standard BDE or a BDE replacement, you would use the same techniques that you would without RAVE to establish your data definitions with those products. This usually means you either have a BDE or BDE replacement component(s) on your application form or you have a uses statement referring to the data module that contains your data definitions. The RAVE data connection simply links to those definitions. This insulates RAVE system from the data definition internals of any BDE / BDE replacement product you may use.

The Data View Dictionary node of the Project Tree in the RAVE designer is where all the data connections for your reports would be listed. To add a Data View to this list, you would select a "New Data View" (from the "Project" menu). This will give you a "Data Connections" dialog window. This window will show all the RAVE type data connections that are "active". You can then select any of these and this will add to the list that will be available for your report definition(s).

The following steps show how to add a RAVE connection to your application. There is a more detailed tutorial in the code base manual covering these steps. But the condensed version is,

- 1) Open the form or data module where you have defined the tTable connections.
- 2) On every tTable that is to be visible to the RAVE system, drop on a RPDDataConnection on that form
- 3) Give it a meaningful name and point it to the proper tTable component.
- 4) Compile your program, that's it as far as your application is concerned.
- 5) Launch your application (this makes the RAVE connection active) RAVE only has knowledge of those applications that are running.
- 6) Start RAVE. Now when you do a "New Data Connection", the list of active data connections should list your connection as well as any others that are running.

Also, remember to check the LED color on the status bar. It will reflect the status of current data connection.

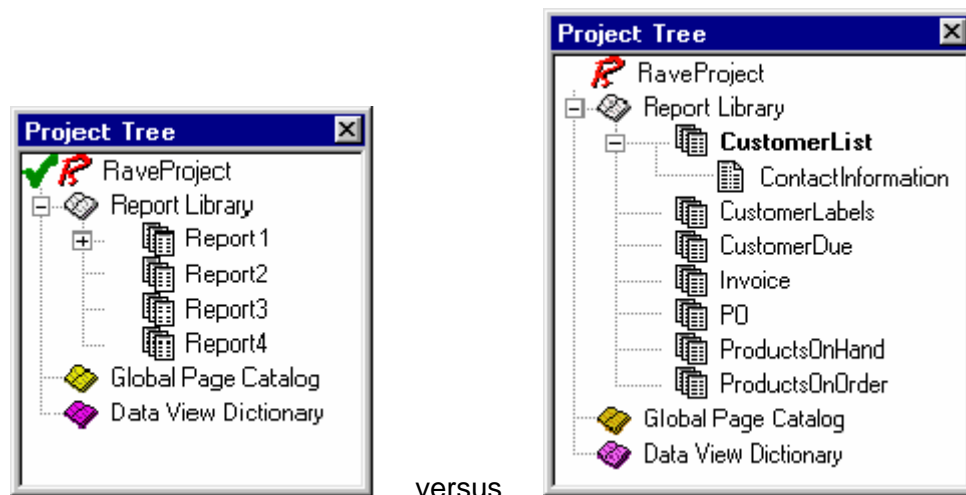
	GREY	Data Connection(s) are inactive
	YELLOW	Data Connection active but waiting for response
	GREEN	Data Connection active and busy
	RED	Data Connection active but has exceeded time-out delay

WARNING

The naming of your data connection is important. It is strongly recommended that your names for each data connection be unique. This would include making it project unique as well. The issue is that every active RPDDataConnection is visible to RAVE not just your application(s). So if you have another application running with a data connection name of "Master1" and you have also named your data connection as "Master1", then RAVE will see both and not be able to distinguish which one you really want to use.

1.3.3.4 Naming Components

Since projects can become very complex, it is suggested that the "Name" property be used to make items in the Project Tree easier to understand. By default, the "Name" property will automatically default to something like "Report1", "Report2", and so on. However, it would be easier to maintain a project system if the reports had useful names. This can be done through the "Name" property. When developing a naming convention, be creative, but remember no spaces or special characters are allowed in the "Name" property.



In the Project Tree Panel image, the project has many reports listed. Instead of dealing with reports listed as "Report1" through "Report7", the many reports were labeled accordingly: "CustomerList", "CustomerLabels", "CustomerDue", "Invoice", "PO", "ProductsOnHand", and "ProductsOnOrder". It is important to understand that the Project Tree Panel is there to make overseeing a project easier. But, without an informative naming system, the Project Tree Panel will not live up to its full potential.

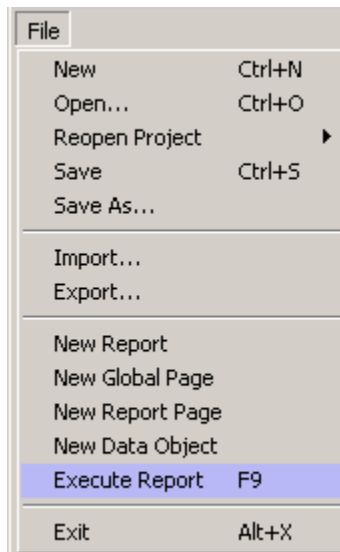
1.4 Generating Output

After a report is created, displaying and making the report electronically mobile will become the next necessary step. There are several ways to get reports onto paper or into electronic formats. This section will cover the different ways to display report output and will also go through simple printing.

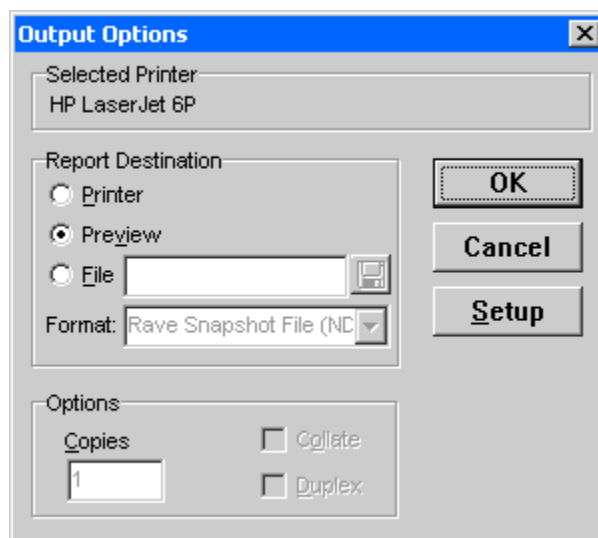
1.4.1 Executing Reports

One of the most common features used in Rave is the actual report execution. Executing a report allows the user to see the output of the report design. Execution is very similar to what many developers know as compiling and actually does involve a certain amount of internal "compiling".

There are various ways that a report can be executed. It is done either via the Project menu, using the F9 function key, or clicking on the Execute Report icon on the Project toolbar. Using the F9 function key becomes very handy since execution becomes a very repetitive and common task performed during all phases of design.



When using any of the ways to execute a report, the user is presented with various options. The Output Options dialog shows the different options available at the time of execution.



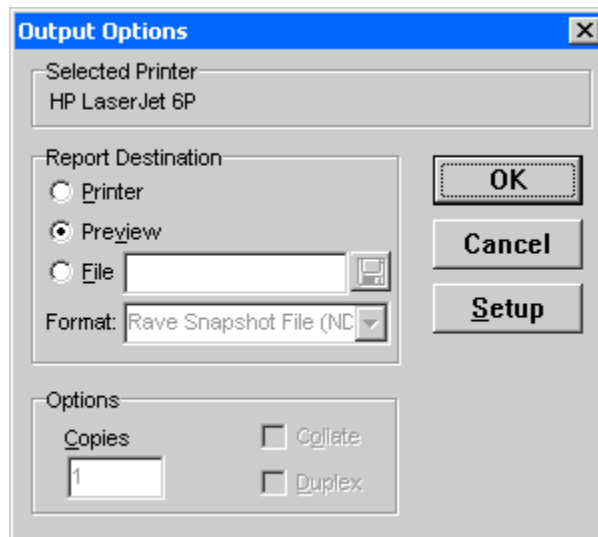
In later parts of this section, each option will be explained in further detail.

1.4.2 Preferences Dialog

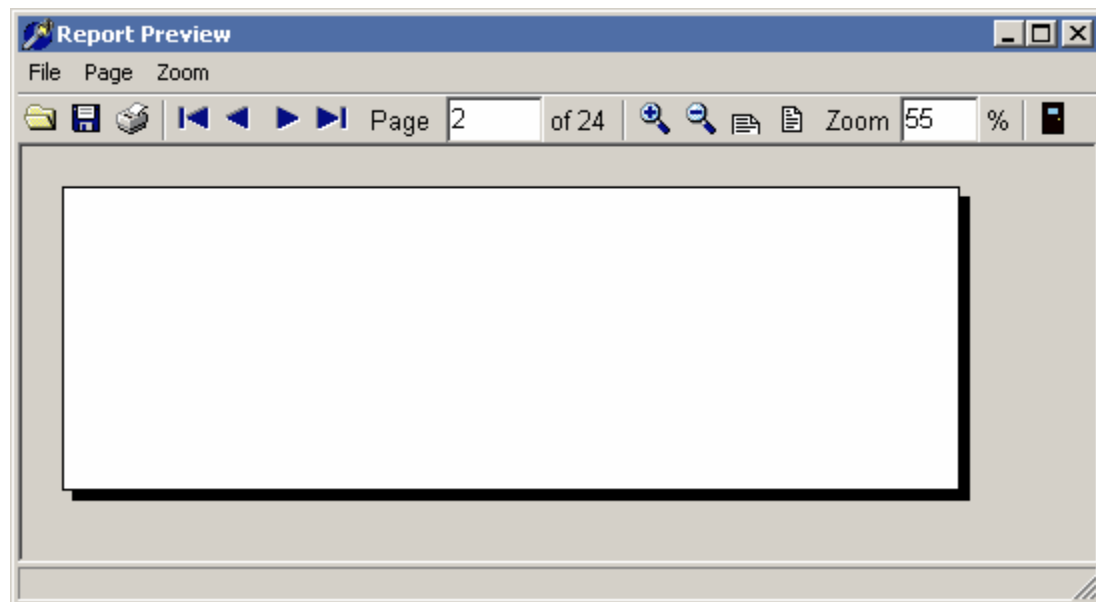
Be sure to check the settings of the Printing Preferences screen. These preference settings will determine the default behavior of several items including print destination, output dialog and preview screen options.

1.4.3 Report Preview

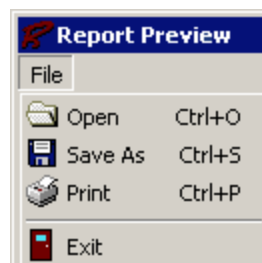
Although the Preview is primarily used for previewing a report, the Preview option has many more capabilities than what is intuitively understood from its name. Begin by executing a report. When the Output Options dialog appears, the Preview radio button can be selected. Clicking OK will continue with the Preview process.



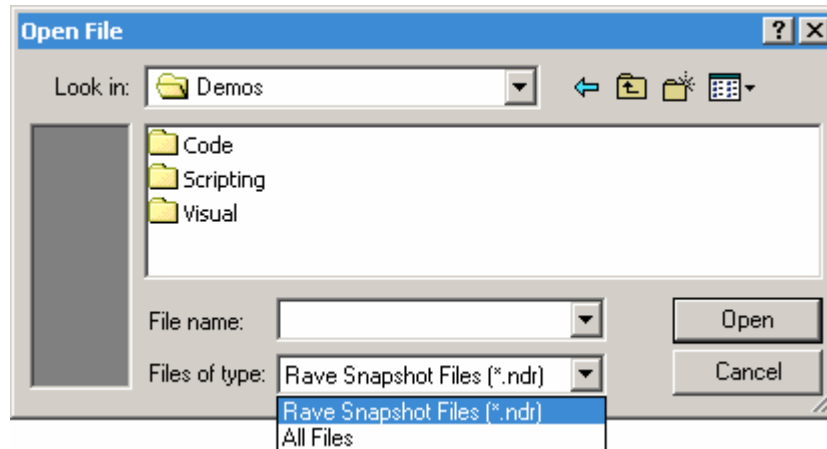
The Preview screen is used to see the output of the designed report. The actual appearance on the screen resembles that of the printed report. The screen is based on a page-by-page output. There is a toolbar at the top of the Preview screen that allows navigation as well as other functions, which are explained in the following paragraphs.



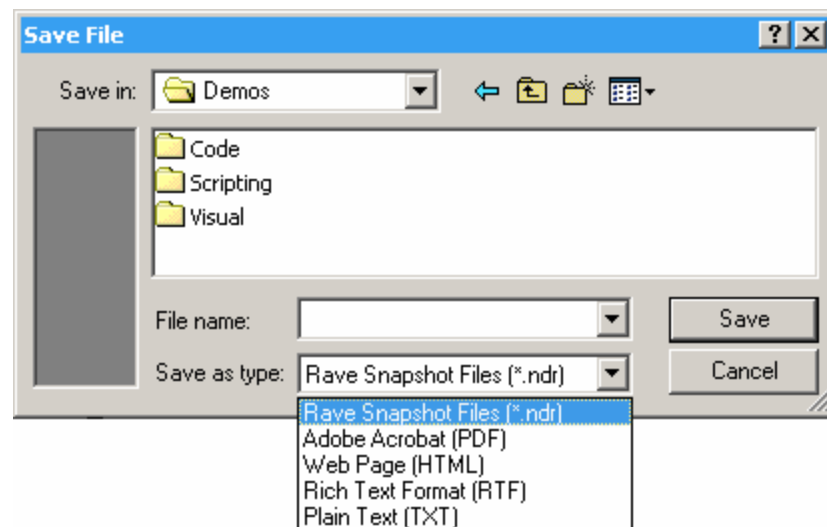
Most of the functions permitted in the Preview screen can either be performed using the toolbar or through the Preview menu options. We will go through and explain the menu options in the rest of this section.



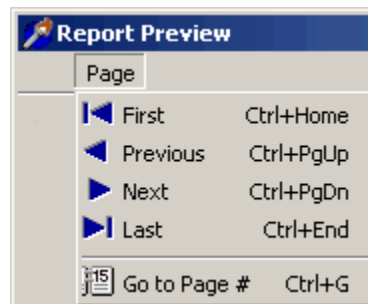
The first group of icons (separated by a vertical line on the toolbar), can be found in the File menu. They are used for opening, saving, printing, and exiting the Preview of a report.



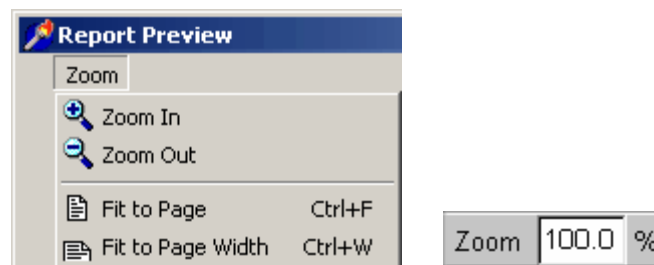
Although the Preview screen displays the current report in the designer, it can also be used to load a previously saved report (with an NDR extension). This can be done by clicking on Open in the File menu. A dialog box appears prompting for the desired report file. When the NDR file is opened, the report that was previously in the Preview is discarded. This does not mean that the report can not be regenerated. To get the previous report, simply re-execute the report and choose Preview. Only the report that was Previewed upon execution can be regenerated. Other reports can only be revisited if they were previously saved or if their projects are re-opened and re-executed.



The Save As is used to save the report that is currently being previewed. Click on "File" then "Save As" and the Save File dialog appears. Use this dialog to save the current report to various formats, including NDR, HTML, PDF, RTF or TXT. The NDR report format is unique Rave file format that can be used to save the report and then load it again if necessary. Reports can be printed directly by choosing Execute and then Printer, the Preview screen also provides the Printer option, as it is often desirable to check the preview of the report before actually printing it.

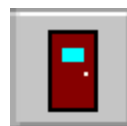


The remaining functions directly affect the viewing of the report on the Preview screen. The second section on the toolbar and the items in the Page menu are used for navigation between pages. The buttons used to designate the options will look familiar to database users, since they follow the same convention. The first button is used to move to the first page of the report. The second button moves to the previous page. The third button is used to move to the next page, and the last button can be used to move to the end of the report. On the toolbar, there is a page indicator, which at all times reflects the current page and the number of total pages. This page indicator can also be used to go to a page directly by typing the page in the edit box. Or use the Go to Page in the Page menu. For example, entering 10 in the edit box will display the 10th page of the report (providing there are 10 or more pages available).



Zooming is also available in Preview. The magnifying glass containing a plus sign will zoom in on the page, while the magnifying glass with a minus sign will zoom out. Similar to the page edit box, there is a zoom box that is used both for displaying the actual zoom and allowing the user to enter a specific percentage value. Just type in a number and the Preview will be zoomed in or out to that zoom percentage.

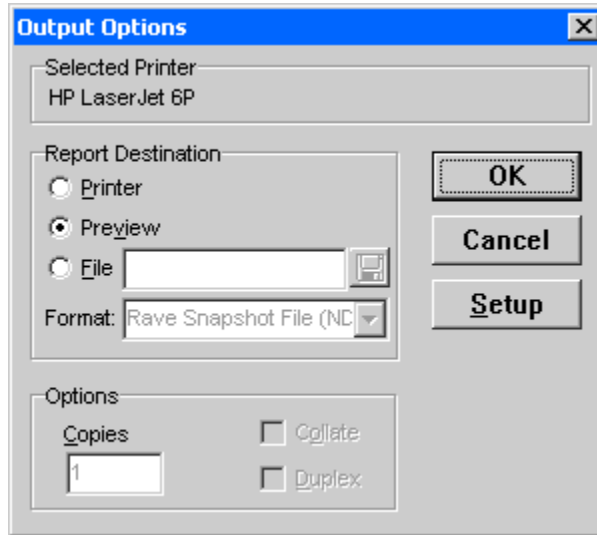
There are also two preset zoom values for fast navigation. The first one is the Fit to Page Width option. By clicking on this, the report can be adjusted so that the width of the page takes up the entire region of the preview screen. The second preset is Fit to Page. It can be used to view the entire page on the screen. The side effect of this is that although the report can be viewed as a whole, the actual contents might be difficult to read. Normally this can be used to get a general overview of the report layout.



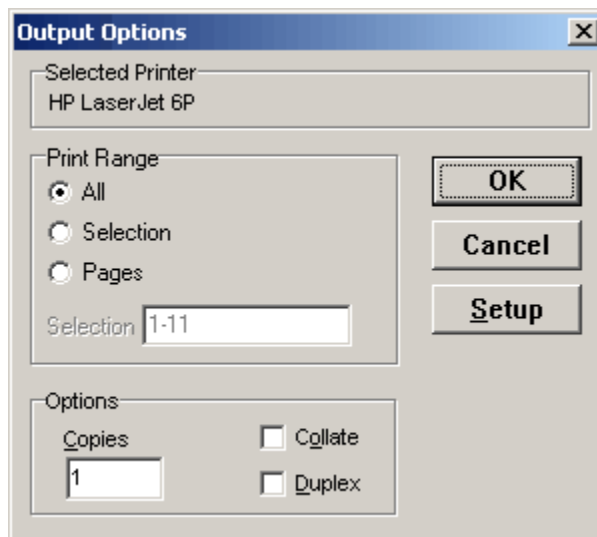
The preview window can be exited by either clicking on the Door icon located on the far right of the toolbar or by clicking on the "X" button on the window (like any other window).

1.4.4 Sending to Printer

Printing can be done via the Preview screen or directly when executing the Report and choosing Printer as the Report Destination. The Output Options dialogs are a little different when selected from the Rave Designer and from the Print Preview.



The Output Options dialog that appears from the Rave Designer has more options than just printing, as shown in the image. In the Output Options Dialog there is a Setup option for Printing. The Setup button is underneath the Cancel button and can be used to adjust the printer settings prior to the actual printing. This includes options such as whether the report should be printed in landscape or portrait, the resolution of the printer, the color depth, paper size, etc.



The dialog box also includes several Options such as the number of copies to be printed, whether the copies should be collated, or duplex printed. The Print Range has three different options.

ALL

This is the "normal" range option and prints ALL pages for the report just generated.

PAGES

The "Pages" range option will open "From-To" boxes. This should be used if you have a

contiguous range of pages to be printed. There are several reasons you might want to use this option. A paper jam occurred and you want to resume printing at the last successfully printed page. Or you may have found several pages that printed "light" and you have now changed the ink cartridge and want to reprint only those pages.

SELECTION

The "Selection" range option will open a text box where you can enter the pages that you want printed. These pages can be split into many different patterns, including **non-contiguous** pages. Selection defines the valid pages in a print job and can contain separate page ranges, separated by commas with ranges defined as First-Last. You also are allowed to select even, odd or reverse order page output by including one of the following:

Type the page numbers with commas between them.

Type the range of pages with a hyphen between the starting and ending numbers in the range.

"A" for "all"

"E" for "even" pages

"O" for "odd" pages

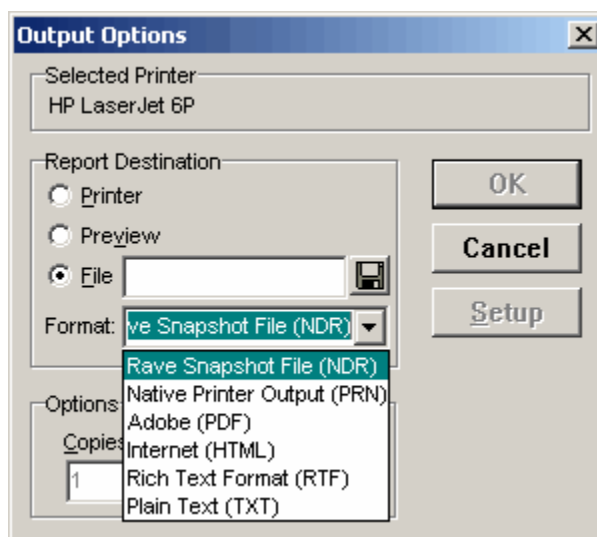
"R" for "reverse order" pages

EXAMPLES:

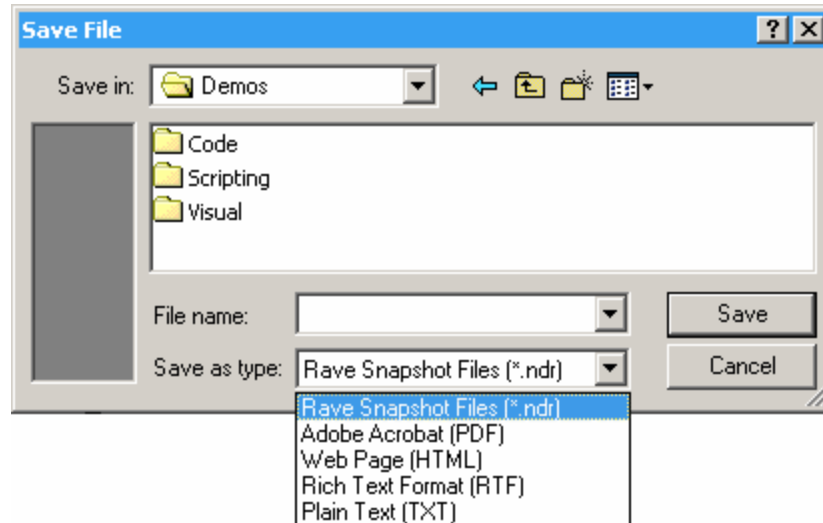
Selection	Will Print Pages
'1-11'	1 thru 11
'5-8,25'	5 thru 8 and page 25
'1,3,6-'	1, 3 and 6 to end of job
'1,e,9-11'	all even pages and page 1, 9 through 11
'o'	Print all odd pages

1.4.5 NDR and PRN files

Printing to a file can be done several ways. We will discuss two that can be done directly from the Output Options dialog. The Output Options dialog is obtained by executing the report (use the F9 key or execute through the Project menu).



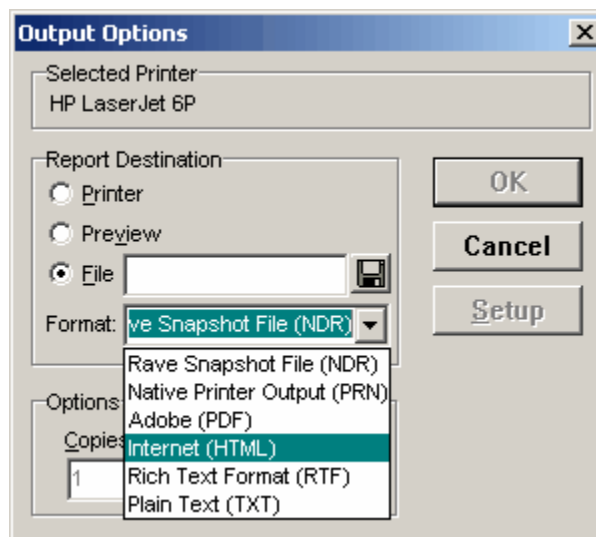
The NDR, or the Rave Format, is a Rave snapshot of the report. An NDR file can be seen and opened from the Preview window, but it cannot be changed or edited in the Rave Designer. The PRN file is the Native Printer Output type file. When this option is selected, the information that is usually sent to the printer to print a report is instead sent to a file. This file is saved based on the file name the user gives. To create either type of file, simply select desired format and then fill the empty File edit box with a name for the file.



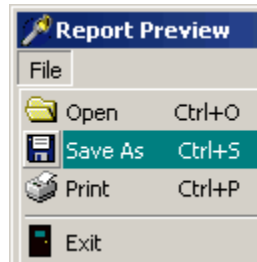
The NDR can also be made using the Save As option in the File menu in the Report Preview window. Using the Save as type drop-down menu and entering a File name will create the NDR file. There are other ways to print to a file. These are only two formats. We will cover PDF and HTML file in the next sections.

1.4.6 HTML .

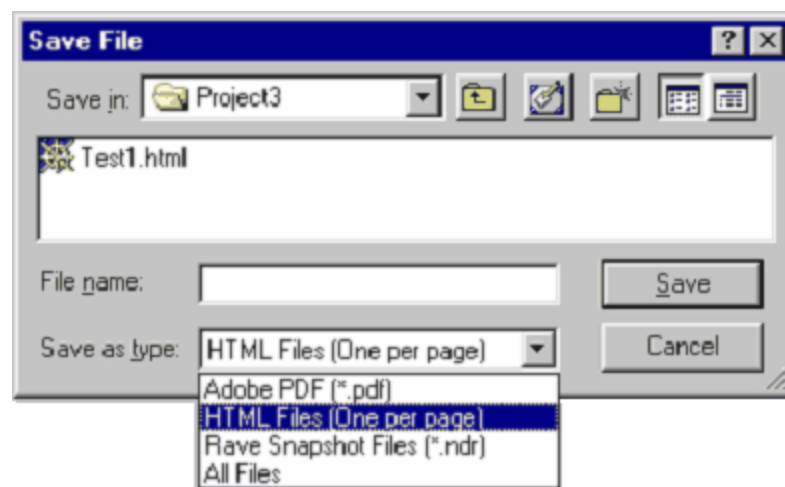
Rave has the capability to save reports as HTML files. This makes it easy to have reports readily available over the Internet or through a company intranet. HTML files can be created in the Visual Designer and in the Report Preview window.



To save a report as HTML from the Visual Designer, first get to the Output Options Dialog. Do this by using the Execute Report from the Project menu, or by using the F9 key. In the Output Options dialog box, choose the File option from the Report Destination area. Using the Format drop down box to get to the HTML option will set the format to HTML. Clicking on the disk, will allow the user to set the path of where to save the file to and to name the HTML file.



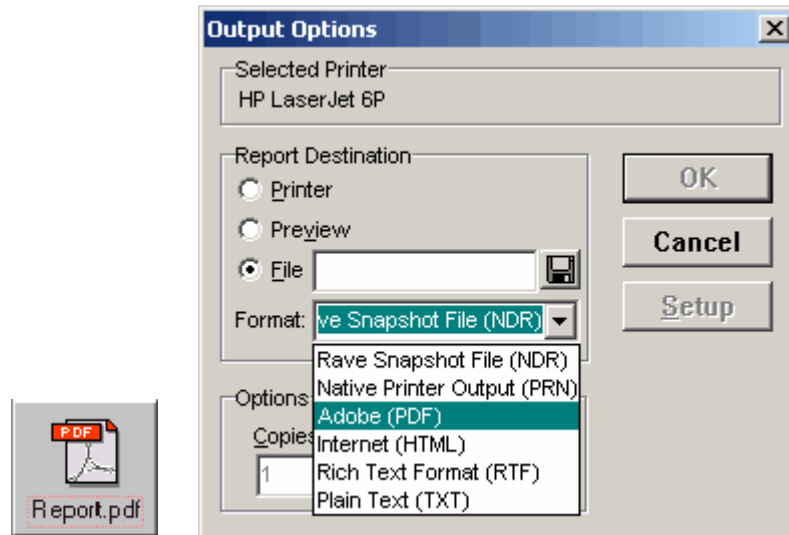
Once a report is saved and properly formatted it may be helpful to first Preview the report. Preview the report by executing the report and choosing Preview from the Output Options dialog. After looking the report over in the Report Preview, choose the Save As option in the File menu. This will bring up the Save File dialog, where a report can be saved into HTML.



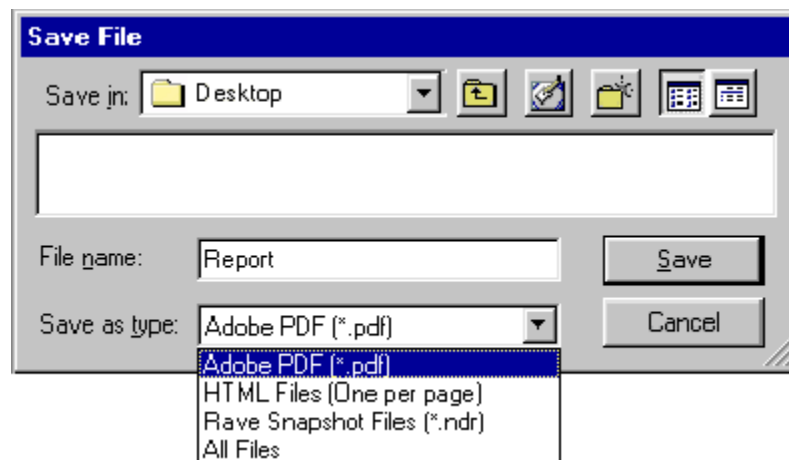
To save the HTML format, in the Save as type drop-down menu, select HTML files. Then type in the name of the new HTML file in the File name edit box. When done, click Save. Each page of the report will be saved as an individual HTML file. For example, a three-page report would produce three pages of HTML. On each page, navigations links to the previous and next pages are provided. Once saved, the HTML files can be viewed with any Internet browser.

1.4.7 PDF .

A useful report is one that is portable over many different platforms. The PDF feature of Rave allows a report to be saved in a PDF format. It can be viewed with the free Adobe Acrobat Reader. Anyone with a PDF viewer can open, view, and print the report.



This option is available through the Execute Report (also F9) option in the Visual Designer File menu or through Save As in the Report Preview File menu. Using the Report Preview allows the user to see the layout of the report before saving it into PDF format. After selecting Execute the Output Options dialog will appear from which the format can be chosen. Also, the file name of the PDF can be set. In the Print Preview, using Save As will bring up the Save File dialog and from there you can save the report as a PDF type, as well as choose the file name.

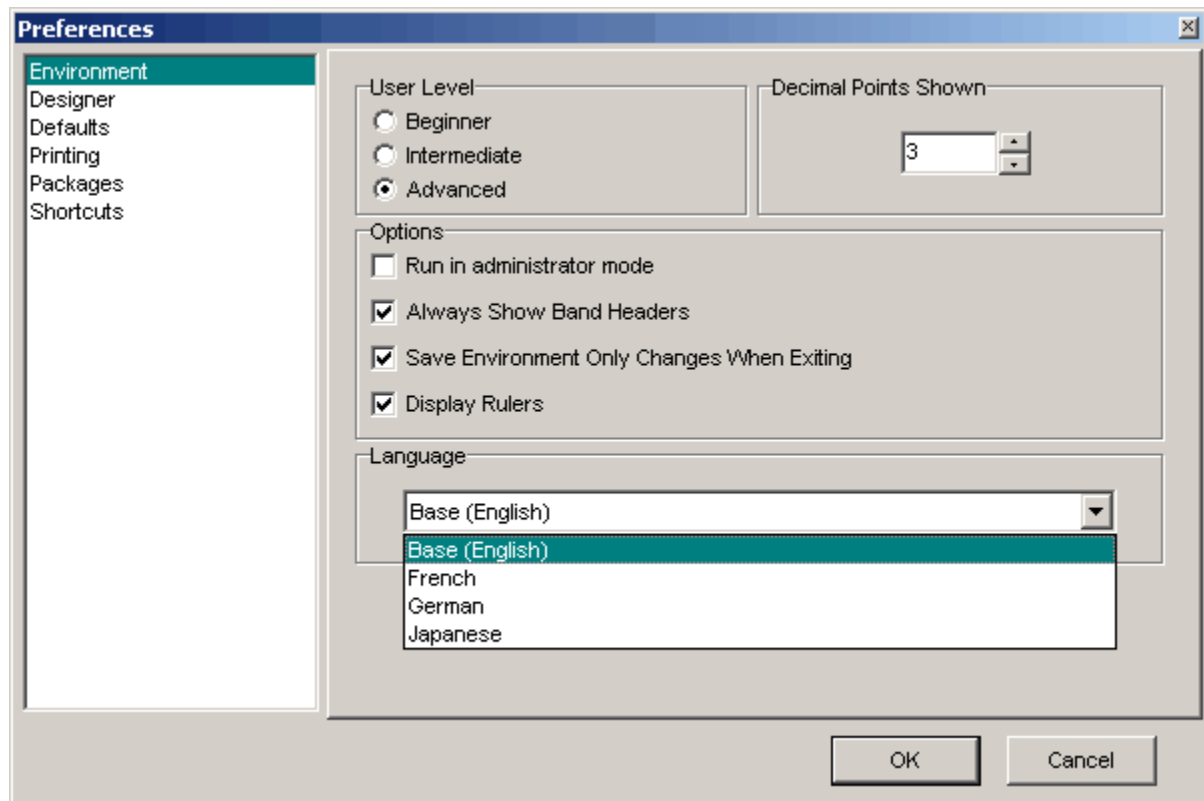


To save a PDF, in the Save as type drop-down menu, select Adobe PDF. Then, type in the name of the new PDF in the File name edit box. When done, click Save.

1.5 Preferences

1.5.1 Environment Preferences

Under the "Environment" tab, you will see that you can change the user level setting. This setting controls the amount of information to be displayed to the user mainly with the various properties window that we will learn about later in this manual. If you don't want to be distracted with all the properties, then try a lower User Level setting. However, if you use a lower setting remember that you may miss some control since not all properties will be displayed.

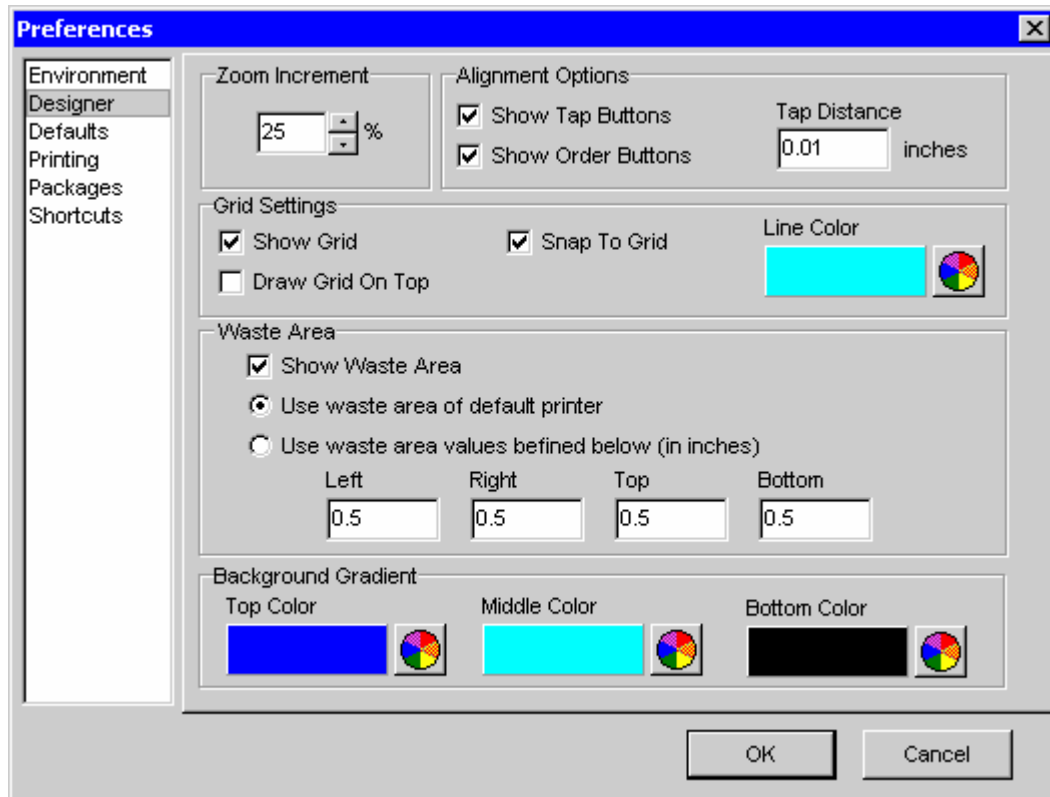


A special option is available called "Auto-remove Component Toolbars". If this option is selected then the expanded toolbars will disappear as soon as you select an item from that toolbar. This might be useful if you have a "small" desktop area and do not want to clutter it with toolbars that are no longer needed.

See Also: Preferences >> [Defaults](#), [Designer](#), [Printing](#), [Shortcuts](#)

1.5.2 Designer Preferences

The "Designer" tab allows you to control the grid system, alignment options, units of measurement and zooming.



Note that the grid system has many options to tailor the grid pattern to your personnel preferences. This includes the color of the grid lines, the grid spacing. You can control options like "Snap To Grid" and keep grid on top. The "Draw Grid On Top" option allows you to still see the grid with components that have a solid background color. If you don't want the grid at all, then make sure that "Grid Active" is not checked.

You can also set the measurement units and the number of decimal points to be displayed in the property displays. The Zoom increment controls the amount of change with the Zoom in / out controls on the Zoom Toolbar.

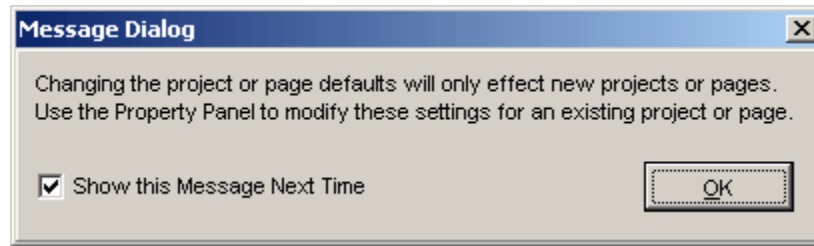


Several of the preferences have been made controllable from the Designer Toolbar. All of the designer toolbar items are toggles. Click it and it will change the state of that item. The first one is a view of a page and it controls whether the grid lines are visible or not on your page layout. The second one controls whether "snap to grid" is active. The third button determines whether the grid lines are always on top or not. The fourth button allows you to cause the toolbars to automatically "disappear" as soon as you make a selection from them. The fifth button controls the displaying of band headers. And the last button will bring up the Preferences dialog windows so that you can any of the preference settings.

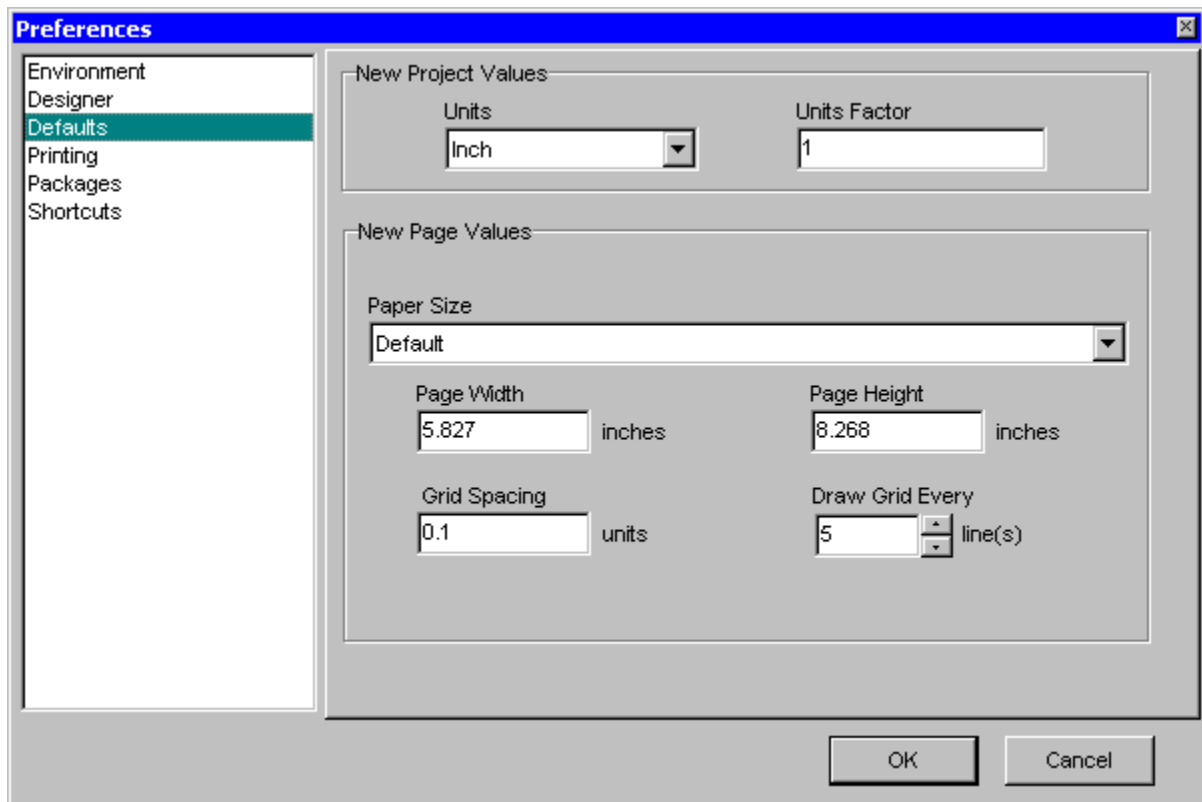
See Also: Preferences >> [Defaults](#), [Environment](#), [Printing](#), [Shortcuts](#)

1.5.3 Default Preferences

The "Defaults" tab provides a place to set many of the RAVE visual designer project and report settings.



These will be the setting used when a new project or new report are created.

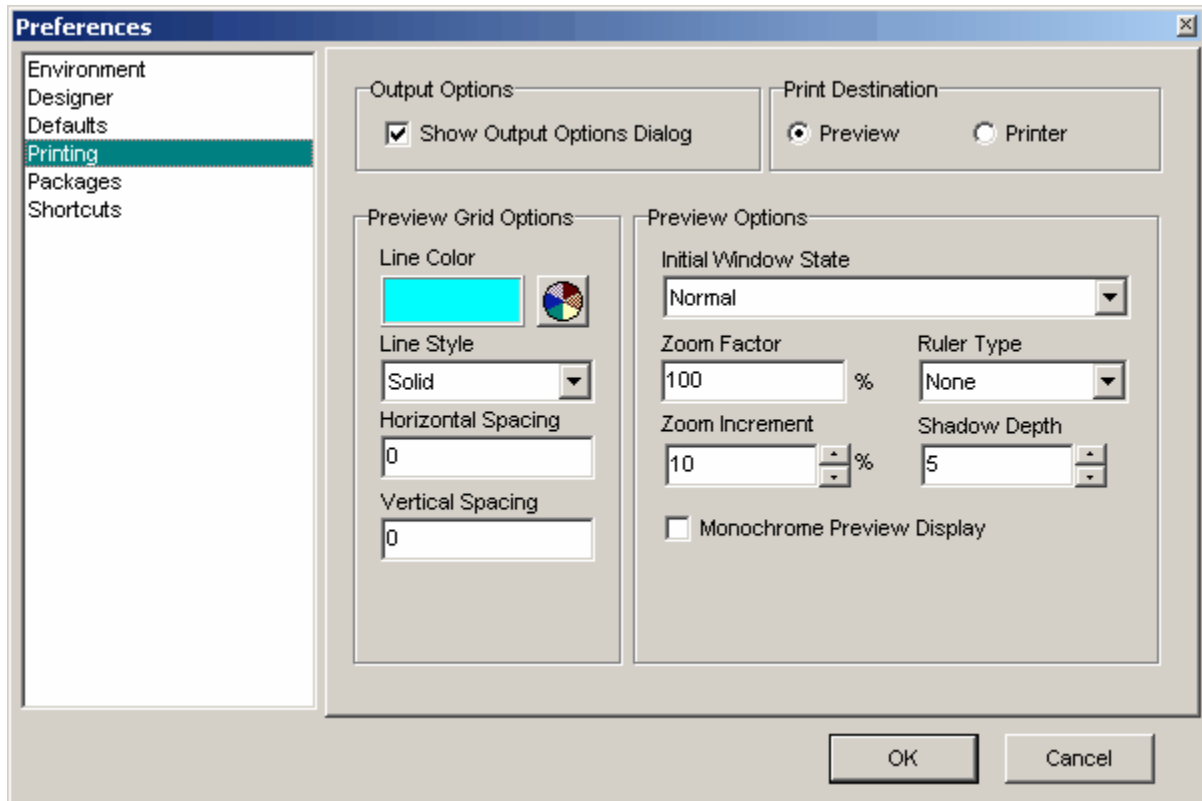


After a project has been started the project properties can be changed at the project level and then they will be saved for that project. The same applies to the reports. For example, this would allow you to have a default grid of 0.25 inches, but on a special report you could set the grid spacing to 0.1 inches.

See Also: Preferences >> [Designer](#), [Environment](#), [Printing](#), [Shortcuts](#)

1.5.4 Printing Preferences

Under the "Printing" tab, you will see that you can change the preview and destination options. Most of these control the behavior of your report output. The preview options are easy to use and can be changed quickly to see their exact impact on the preview screen.



However, the real purpose of the Monochrome option might not be so obvious. The real purpose of the Monochrome setting is to reduce the size of file created by Rave during the report generation mode. If your display setting is at one of the higher color settings, like "True Color", then the output file can be quite large. The time to produce and the size of the output file will be reduced by selecting Monochrome as an output setting. This is especially true if the system has low amounts of RAM (4-8 MB) and the display setting is at one of the higher color settings, like "True Color".

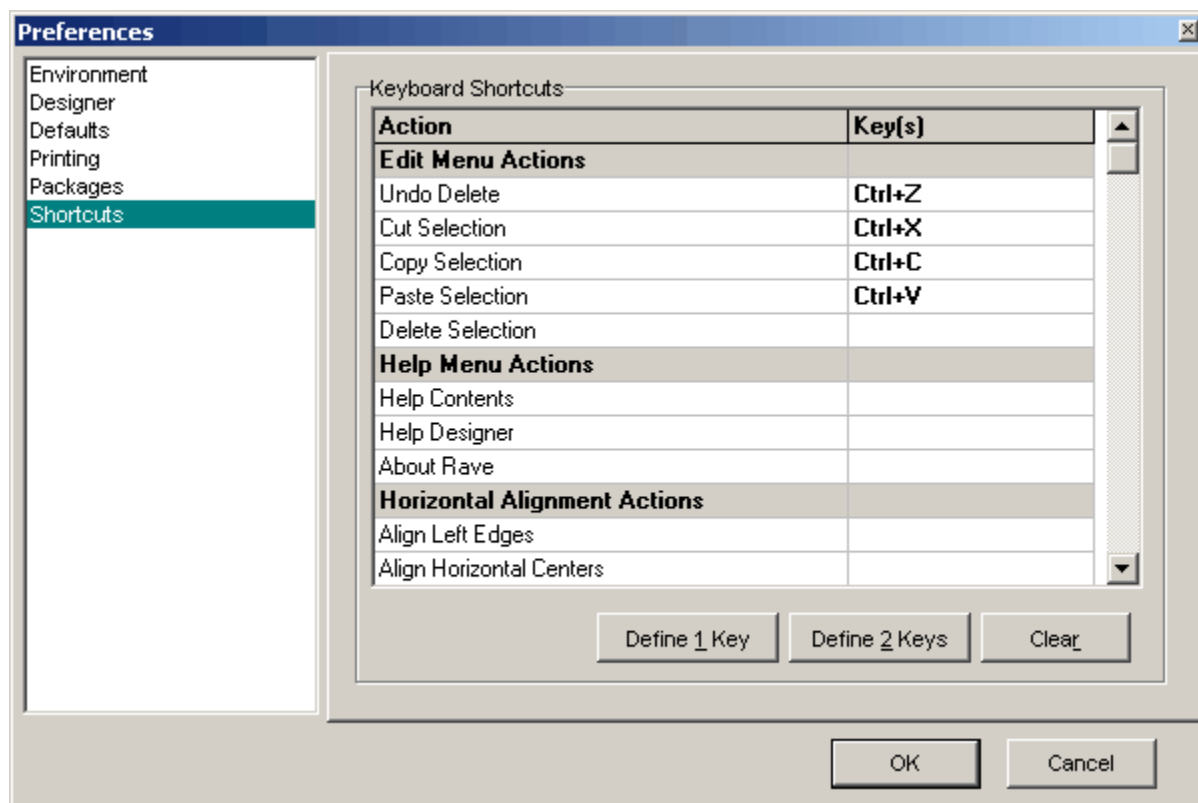
In Output Options area of Printing Preferences, the Show Setup Dialog indicates whether the Output Options dialog (as shown here) is displayed when executing a report. It is sometimes convenient to disable the Output Options dialog, especially if it is during design and testing time. The drawback to this is that some of the parameters that are contained in the dialog box would have to be set before hiding this dialog. Also, any changes desired would have to be done either by re-activating the dialog box or specifying the changes in the Preferences dialog.

The default report print destination is determined by the settings under Print Destination in the Printing Preferences, and is reflected in the Output Options dialog box. Changing the value in the Print Destination area of the Printing Preferences will directly effect the default selection in the Output Options dialog box. There are two options that can be chosen: Preview and Printer. Selecting either one, by choosing the appropriate radio button, will make that option the default value and when the Output Options dialog appears the default value will be selected.

See Also: Preferences >> [Defaults](#), [Designer](#), [Environment](#), [Shortcuts](#)

1.5.5 Shortcuts Preferences

You can define keyboard shortcuts for a wide variety of project actions. If you find that you are frequently repeating a particular task in Rave, then check the list of "Actions" and assign a keystroke combination that you will remember for that action. Rave gives you the ability to set your own shortcuts to many different functions within this program. Open the "Shortcuts" tab in the Preferences dialog to set you own keyboard shortcuts.



NOTE: The shortcuts dialog will not allow you to enter duplicate keyboard assignments. So assign your frequently used items to keystrokes that make sense to you.

See Also:

Preferences >> [Defaults](#), [Designer](#), [Environment](#), [Printing](#), [List of Shortcuts](#)

1.6 Wizards

Wizards are a feature in Rave that allow certain types of reports to be created by answering a series of questions. They can be found under the "Tool" menu option. The Wizards can be added and tailored to each user's needs. It is a perfect way to minimize the end-users interface with reporting tasks.

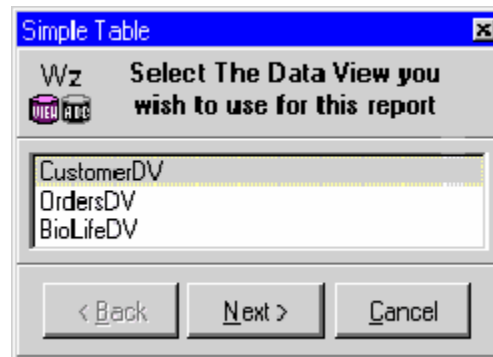
There are two Wizards, a "Simple Table" and a "Master-Detail". Wizards can be designed to ask for the data connection and allow the user to choose the fields that are needed on a report. It is important to note that an active DataView should be available prior to running the wizards, whether it is a Direct Data View or an SQL Data View. Simple Reports are generally used for listings. Common uses include Client reports, Telephone lists, etc. The Master-Detail Wizard is used when more complex reporting is required, such as invoices, product order lists, etc.

Independent of the one that is executed, there are several steps that are common to both. These include the DataView selection, field selection, etc and are covered in greater detail in the exercises included at the end of this section.

It is very important that the user understands that these are general purpose reporting wizards and as such, some aspects are not treated with great detail (amount of text that would fit on a page, layout, etc). They can be used as the building blocks for a more complex report or can be adjusted to suite a particular layout required. The wizards are not intended to build complex reports. But a Report Wizard can be used to create the initial design and then you can load that initial design with the Rave IDE and make as many changes as needed for your final design.

1.6.1 Simple Wizard

The first step when using the simple wizard is to select the desired DataView that will provide the data to the report.

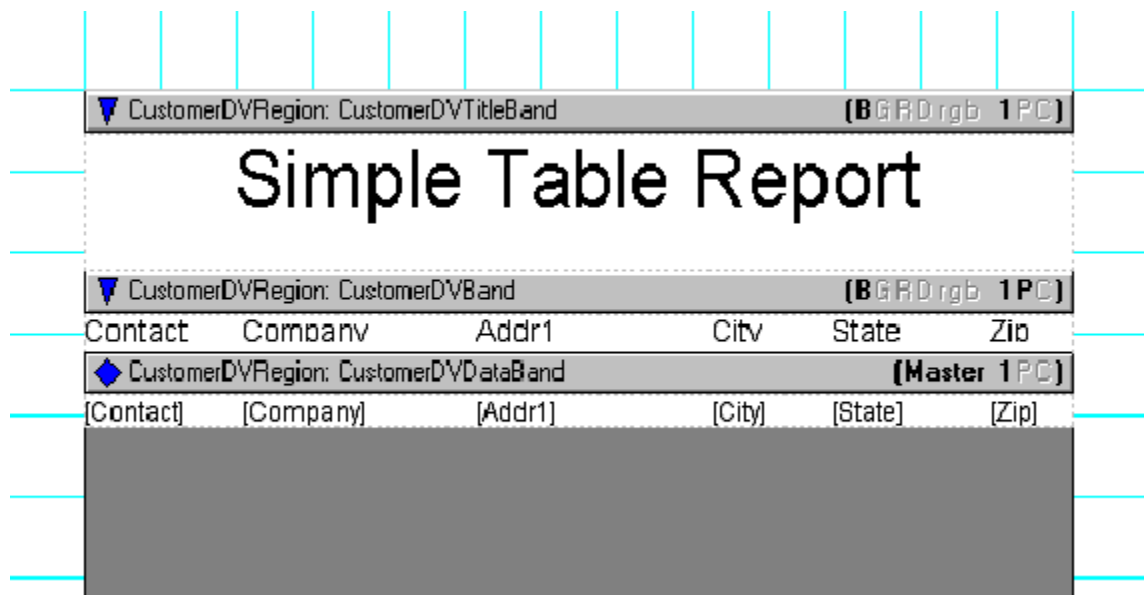


Once the DataView has been selected, a list of fields that should appear on the report can be chosen. Fields are selected by clicking on the box on the left. All fields can be selected by clicking on the All button at the top. Note however that because this is a simple wizard, choosing more fields that would fit on the page will make them overlap. Manual intervention is required after the Wizard's initial design is complete to correct this problem.

If more than one field was selected in the previous step, the Wizard will ask for the ordering of the fields. Moving a field up will place it on the left side of the page. Move it to the lowest position will move the field to the right margin.



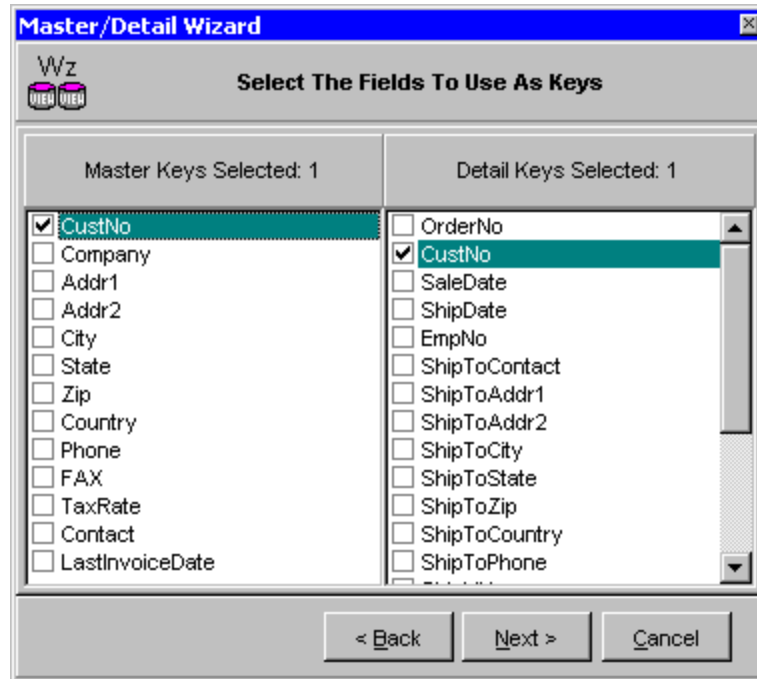
Next you can set the Report layout. Values include the title and the page margins. The last step of the Simple Report Wizard is to choose the fonts that are going to be used in the report. It gives the possibility of changing three fonts: title (of the report), caption (headers of the field names) and the body (actual field values). When you are done, you will have a designer screen that looks like the following. At this point, you can make changes as desired, then save the report design.



1.6.2 Master Detail Wizard

The Master-Detail Report Wizard has a few additional steps more than the Simple Wizard Report. This is due to the fact that more than one table is going to be taking part in the report. In particular, there will be a master (for example customer information) and a detail (items ordered).

1. Similar to the Simple Report Wizard, the first step is to choose the DataView. However, in this case the DataView chosen corresponds to the Master table.
2. In step two, the Detail table can be selected. Note how the DataView selected in step 1 is no longer available to avoid errors.



Similar to the Simple Wizard, the next steps are for selecting the fields and ordering of both the master and the detail table. A new step is determining the key fields. These are field that relate one table to the other. After this, the remaining steps are identical to that of the Simple Report Wizard. Once everything is complete, clicking on Generate will produce a simple master-detail report that, again, can later be adjusted and "touched up" to ones particular needs. The figure below shows a sample output of the Wizard.

CustomerDVRegion: CustomerDVTitleBand (BGRDrGb 1PC)				
Master/Detail Report				
CustomerDVRegion: CustomerDVBand (BGRDrGb 1PC)				
CustNo	Company	City	State	Phone
CustomerDVRegion: CustomerDVDataBand (Master 1PC)				
[CustN]	[Company][City][State][Phone
CustomerDVRegion: OrdersDVBand (BGRDrGb 1PC)				
OrderNo	CustNo	SaleDate	ShipDate	ItemsTotal
CustomerDVRegion: OrdersDVDataBand (BGRDrGb 1PC)				
[OrderNo][CustNo][SaleDate][ShipDate][ItemsTotal

1.6.3 Report Expert

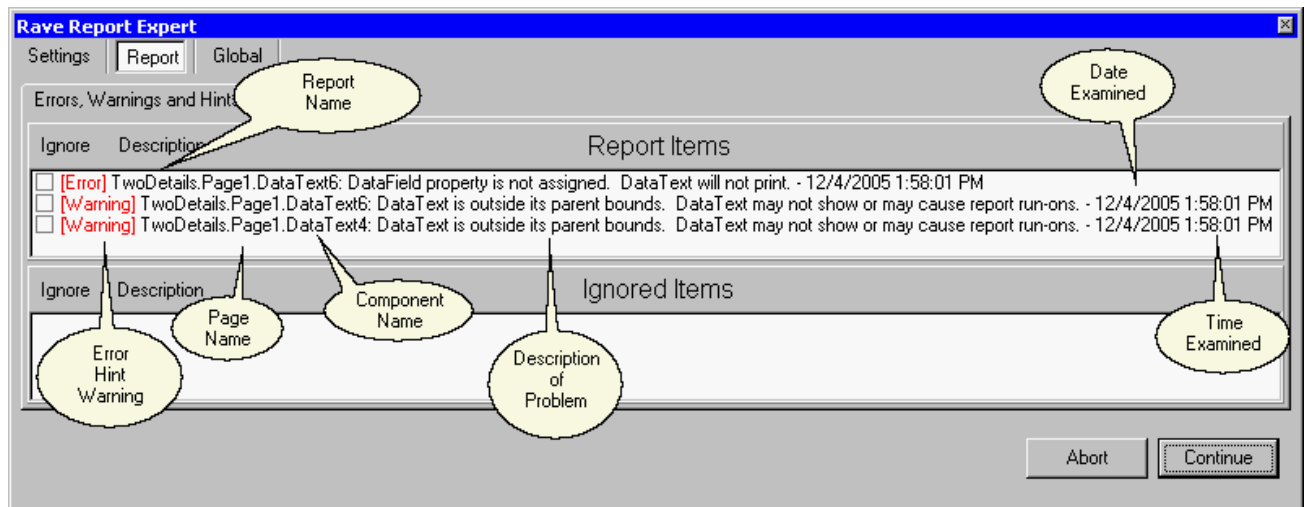
You can use the "Report Expert" to analyze your report designs. It will review each of your report designs and list any "Hints", "Warnings" or "Errors" that it finds.

Settings Tab

You can use the checkbox in the "Settings" tab of the "Report Expert" to control whether it is active or not. When the "Report Expert" is enabled, it will scan your visually designed reports for potential problems. This scan will occur each time you run a report as long as the "Report Expert" is enabled. Once you have analyzed your reports and are satisfied with your report designs, you can turn the "Report Expert" off by clearing the "Enabled" checkbox found on the "Settings" tab of the "Report Expert" .



Report Tab

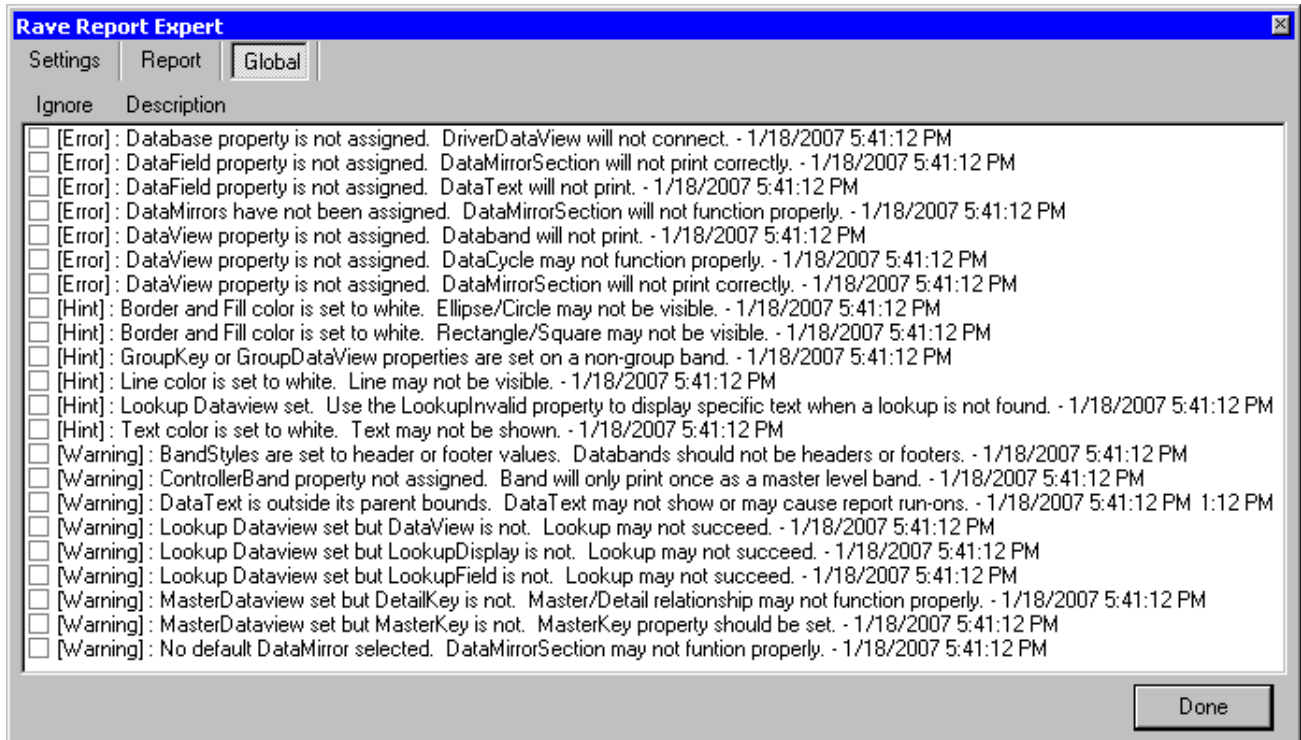


The "Report" tab contains any discrepancies found on the report that you just ran. The "Report Expert" dialog will not be shown if there are "no" discrepancies found. If there are discrepancies, then the "Report" tab will be shown with a list of each item found. The Report Items found that need attention are listed in the message pane that contains the following:

- Type of Discrepancy - Error, Warning or Hint
- Report Name
- Page Name
- Component Name
- Description of Problem
- Date Examined
- Time Examined

The checkbox on the left side of the "Report Items" can be used to "ignore" an item on any future scans of [this](#) report.

Global Tab



The "Global" tab contains all the tests that are available for a report scan. If there are some checks that you do not want done on any report, then make sure that the "Ignore Checkbox" is marked for that item.

1.7 Event Scripting

The Event Editor in Rave allows for the definition of several types of events components within the Rave visual report environment. This can be used to extend the existing functionality of the visual components with custom logic for much more flexible reports. This document describes the basics of the Rave scripting language and the event editor. For examples of several reports that use scripting, view the reporting project normally located in the C:\Rave?\Demo\Scripting folder.

1.7.1 Event Editor



Example of OnGetText Event

1. Event Editor Tab – Select this tab to view the event editor within the Rave report designer.
2. Available Events drop down list – This control shows the current list of available events for the selected component. When an event is selected, it is then created and moved to the Defined Events list. This control is disabled when the Event Editor is in page view mode.
3. Defined Events list – This list shows all of the defined events for the currently selected component (if in selected mode) or for all of the components on the currently selected page (if in page view mode). Selecting different events on this list will change which event is visible in the lower portion of the Event Editor.
4. Event Description – This section will display helpful information on the parameters and usage of the currently selected event.
5. Compile button – This will perform a compile of all events in the currently loaded reporting project. It is not necessary to hit this button each time a change is made as Rave will automatically compile if necessary before a report is run within the designer.
6. Show Events for Selected button – When this button is selected the events displayed in the Defined Events list will be those for the currently selected component (use Project Tree to change selection).
7. Show All Events on Page button – When this button is selected the events displayed in the Defined Events list will be those for all components on the currently selected page (use the Project Tree to change selection). When in this mode, the Available Events drop down list will be disabled.
8. Delete Current Event button – This button can be used to delete an event that is no longer needed.
9. Compile Status – This label will display the results of the most recent compile.
10. Event declaration text – This read-only text will show the actual declaration for the current event.
11. Event header memo – This memo will allow for the definition of variables and other non-code items for the current event.

12. Event body memo – This is where code statements for the event should be placed.

1.7.2 Syntax

The Break, Continue and Exit function exist (similar to Delphi). Variables may be declared (similar to Delphi). The following types are supported:

Boolean, Byte (0 to 255), Char, Currency, Double (8 byte float), Extended, Integer (32 bit signed integer), Shortint (-128 to 127), Single (4 byte float), SmallInt (-32768 to 32767), String, Word (0 to 65535)

AsBlob, AsBoolean, AsCurrency, AsDate, AsDateTime, AsFloat, AsInteger, AsString, AsTime

If statement

```
if condition then
  statement(s);
elseif condition then
  statement(s);
else
  statement(s);
end;
```

For loop

```
for index := start to finish do
  statement(s);
end;

for index := start downto finish do
  statement(s);
end;
```

While loop

```
while condition do
  statement(s);
end;
```

Repeat loop

```
repeat
  statement(s);
until condition;
```

With statement

```
with object do
  statement(s);
end;
```

1.7.3 Functions

```
function Abs(X: integer): integer;
function AbsExt(X: extended): extended;
function ArcTan(X: extended): extended;
function Beep;
function BreakupDate(Date: TDateTime; var Year, Month, Day: word);
function BreakupTime(Time: TDateTime; var Hour, Min, Sec, Msec: word);
function Chr(X: byte): char;
function Compare(S1, S2: string): Integer;
function CompareCase(S1,S2: string): Integer;
function Copy(S1: string; Index,Count: integer): string;
function Cos(X: extended): extended;
function CreateDate(Year, Month, Day: word): TDateTime;
function CreateTime(Hour, Min, Sec, Msec: word): TDateTime;
function CurrToStr(Value: currency): string;
function Date: TDateTime;
function DateTimeToStr(DateTime: TDateTime): string;
function DateToStr(Date: TDateTime): string;
function DayOfWeek(Date: TDateTime): integer;
function Delete(var S1: string; Index, Count: Integer);
function Exp(X: extended): extended;
function FloatToStr(E1: extended): string;
function FormatFloat(Format: string; value: extended): string;
function Frac(X: extended): extended;
function Insert(Source: string; var S1: string; Index: integer);
function IntToStr(I1: integer): string;
function Length(S1: string): Integer;
function Ln(X: extended): extended;
function LowerCase(S1: string): string;
function MakeStr(Ch: char; Count: integer): string;
function Now: TDateTime;
function Odd(X: integer): Boolean;
function Pi: extended;
function Pos(Substr: string; S1: string): Integer;
function QuotedStr(Value: string; Quote: char): string;
function Random(Range: integer): integer;
function Randomize;
function Round(X: extended): integer;
function ShowMessage(S1: string);
function Sin(X: extended): extended;
function Sqr(X: extended): extended;
function Sqrt(X: extended): extended;
function StrToCurr(S1: string): currency;
function StrToDate(S1: string): TDateTime;
function StrToDateTime(S1: string): TDateTime;
function StrToFloat(S1: string): Extended;
function StrToInt(S1: string): Integer;
function StrToIntDef(S1: string; Default: Integer): Integer;
function StrToTime(S1: string): TDateTime;
function TimeToStr(Time: TDateTime): string;
function Trim(S1: string): string;
function TrimLeft(S1: string): string;
function TrimRight(S1: string): string;
function Trunc(X: extended): integer;
function UpperCase(S1: string): string;
```

1.7.4 Events

The "Event Editor" allows you to add code or modify code in any of the events of your reporting components with the Rave scripting language. It is important to note that the "Event Editor" will only see the active component (last one selected). If you wish to write an event for one of your reporting components, remember that you must first select that component in the Page Designer view before opening the "Event Editor" tab. The following table shows which events are available for each Rave component.

RAVE components	On After Print	On After Report	On Before Print	On Before Report	On Calc Value	On Get Group	On Get Text	On Get Value	On Mirror Value	On Print	Uses
All Project Level	X	X	X	X							
DataField	X	X	X	X							RvData
DataRow	X	X	X	X							RvData
Page	X	X	X	X							RvClass
ProjectManager	X	X	X	X							RvProj
Report	X	X	X	X							RvProj
All BAR CODE components	X	X	X	X							RvCsBars
All DRAWING components	X	X	X	X							RvCsDraw
REPORT components											RvCsRpt
Band	X	X	X	X		X					
CalcController	X	X	X	X							
CalcOp	X	X	X	X				X			
CalcText	X	X	X	X	X		X				
CalcTotal	X	X	X	X	X			X			
DataBand	X	X	X	X		X					
DataCycle	X	X	X	X							
DataMemo	X	X	X	X							
DataMirror	X	X	X	X					X	X	
DataText	X	X	X	X			X				
Region	X	X	X	X							
STANDARD components											RvCsStd
Bitmap	X	X	X	X							
FontMaster	X	X	X	X							
Memo	X	X	X	X							
MetaFile	X	X	X	X							
PageNumInit	X	X	X	X							
Section	X	X	X	X						**X**	
Text	X	X	X	X							

You can add your own logic using the Rave [functions](#) to one of the component events to control how a component behaves. You must exercise some care when selecting which component and event you select for your scripting logic. Understanding how the report is being generated will be helpful in determining where to place your logic. The best way to do this is examine *both* the [Tree View](#) panel of the Rave Visual Designer *and* the [Band Style Editor](#). The Band Style Editor shows how the bands are operating. The Tree View shows the "print" flow of the components in your

report. So something at the top of tree "prints" (fires its events) before something lower in the tree view. You can use the [Alignment Toolbar](#) to move components up or down in the tree view.

The order of components in the tree view is not important when you are NOT using scripting and NOT using any of the calc components. However, if you have added some scripting to any of your components OR are using any of the calc components then the order of the components becomes very important. For example, if you write a script that examines data text "ABC" component and use its "value" to set the color of data text "XYZ" component, then the order of those components are important. In this case, data text "ABC" must be before (higher) than data text "XYZ" in the tree view.

Of course, selecting which event you use is also important. If you placed some logic in the "On After Report" event, then that is very late in the printing process and probably would not be a good selection if you were checking the values within a data band. In many cases, you would put your scripting code in an event of the component that you wish to control. However, if that does not achieve the results you want, then try putting that code in another component that "prints" before the one you are trying to control.

Section OnPrint

***For users that have upgraded their Rave Reports, the "section" component has an OnPrint event that provides a "doorway" to many of the methods and properties normally available only to code base users. This allows users of the upgraded versions of Rave Reports to write regular code base statements in their visual report projects. ([see OnPrint example](#))*

1.7.5 Classes

For those users that have source code (Rave BEX), please refer to the *.RVS files in your RaveSource folder for which properties and methods of the Rave classes are available. These source files are written in Rave script and are compiled with the RaveCC compiler to produce the .RVC files that are used directly by Rave. RvBase, RvClass, RvData, RvProj, SysFunc, RvCsBars, RvCsData, RvCsDraw, RvCsRpt, RvCsStd

1.7.6 Example GreenBar Effect

Using events to get a greenbar effect (alternating background line colors).

- Select the band for the desired greenbar effect
- Drop a Rectangle component in that band
- Set rectangle size to match band size, for example, Left=0, Top=0, Height=0.20 and Width=8.0
- Use the Alignment tab to move the rectangle component to be the first item listed in tree view panel for that band
- GoTo the Event Editor (with rectangle selected)
- Create an OnBeforePrint event
- Put the following code in the OnBeforePrint event

```
{ Event for Rectangle.OnBeforePrint }
function Rectangle1_OnBeforePrint(Self: TRaveRectangle);
begin
  if self.FillColor = clLime then
    self.FillColor := clWhite;
  else
    self.FillColor := clLime;
  end if;
end OnBeforePrint;
```

1.7.7 Example Parameters

Sample event that GET's a parameter named "LineCount", "increments" it, and then SET's (saves) it. This could be used if you wanted to have row counts with a Master-Detail-Detail relationship with 2 or more details groups.

Remember that parameters must be a string value.

```
{ Event for DataTextCount1.OnGetText }
function DataTextCount1_OnGetText(Self: TRaveDataText; var Value: string);
var
    iCount: Integer;
    sCount: String;
begin
    iCount := StrToInt(RaveProject.GetParam('LineCount'));
    sCount := IntToStr(iCount + 1);
    Value := sCount;
    RaveProject.SetParam('LineCount', sCount);
end OnGetText;
```

1.7.8 Example OnBeforePrint

You can use events to control the visibility of components. You do this by putting a script that sets the "visible" property in a **parent** of the component you want to control (or in another component). Do **not** put a script that changes visibility in the same component you want to control as it will run until it is "false" but will not run after that as the event "is no longer visible". The first example shows a header band event that controls the visibility of another band.

```
{ Event for HeaderBand.OnBeforePrint }
function HeaderBand_OnBeforePrint(Self: TRaveBand);
begin
    if Frac(DvInventoryLastEdit.AsFloat) < 0.25 then
        BandRowFooter.visible := True;
    elseif Frac(DvInventoryLastEdit.AsFloat) > 0.75 then
        BandRowFooter.visible := True;
    else
        BandRowFooter.visible := False;
    end if;
end OnBeforePrint;
```

This band event changes the visibility of components within the band.

```
{ Event for DetailBand.OnBeforePrint }
function DetailBand_OnBeforePrint(Self: TRaveDataBand);
begin
    if UpperCase(BioLifeDvCategory.AsString) = 'SHARK' then
        dtCM.visible := False;
        BitmapFish.visible := False;
    else
        dtCM.visible := True;
        BitmapFish.visible := True;
    end if;
end OnBeforePrint;
```

1.7.9 Example OnGetText

How to use an event to change colors based upon fractional part of a DateTime field.

```
{ Event for dtLastEdit.OnGetText }
function dtLastEdit_OnGetText(Self: TRaveDataText; var Value: string);
var
  nTime: Double;
  iHour: Integer;
  iMinute: Integer;
begin
  nTime := Frac(StrToFloat(Value));
  if nTime < 0.25 then
    self.Color := clRed;
    Value := 'Before Start';
    self.Left := self.Left + 0.1;
  elseif nTime > 0.75 then
    self.Color := clYellow;
    self.Left := self.Left - 0.1;
    Value := 'After Hours';
  else
    self.Color := clBlack;
    Value := '';
  end if;
  nTime := nTime * 24;
  iHour := Trunc(nTime);
  iMinute := Trunc(Frac(nTime) * 60);
  If iMinute < 10 then
    Value := IntToStr(iHour)+':0'+IntToStr(iMinute);
  else
    Value := IntToStr(iHour)+':' +IntToStr(iMinute);
  end if;
end OnGetText;
```

1.7.10 Example Section OnPrint

The OnPrint event of the Section component provides a way for users of the **upgraded versions** of Rave Reports to access many of the methods and properties normally only available to the code base side of Rave. *The "Section OnPrint" event is **NOT available to the BE (Bundled Edition)** versions of Rave Reports.*

The first example shows how to do a complete report in this OnPrint event that only uses one DataText and one Section component. You can write very complex reports using this technique. The DataText component IS required to notify the visual system so that it can establish a link to your DataView. You **ONLY** need to set the DataView property, please do NOT set any other properties.

- 1) Create a new report (blank)
- 2) Drop a DataText component on the page.
Set the DataView property to your desired DataView (CustomerDV)
- 3) Drop a Section component on the page
set all of the edges to match the page edges (left =0, top=0 etc)
- 4) Select the Section component
- 5) Go to the "Event Editor" tab
- 6) Drop the Events list and select the OnPrint event
- 6) After entering the code into the OnPrint event, click on the "Compile" button
you should get a "Compile finished" message
- 7: Save this project - you **MUST** do a save

```

{ Event for Section1.OnPrint }
function Section1_OnPrint(Self: TRaveSection; Report: TBaseReport);
Var
  sCityState, sCustNo: String;
  IDoHeader: Boolean;
begin
  IDoHeader := True;
  With Report Do
    CustomerDV.First;    // REQUIRED to establish link to your data
    SetPen(clBlack, psSolid, 12, pmCopy);
    While Not CustomerDV.EOF Do
      If IDoHeader Then
        SetFont('Arial',14);
        Bold := True;
        PrintLeft('Number',1.5);
        PrintLeft('Company',3.0);
        PrintLeft('Phone',6.0);
        Bold := False;
        MoveTo(SectionLeft,LineBottom);
        LineTo(SectionRight,LineBottom);
        SetFont('Arial',10);
        NewLine;
        NewLine;
        IDoHeader := False;
      End;          // End of Header
      sCustNo := CustomerDVCustNo.AsString;
      PrintLeft(sCustNo, 1.5);
      PrintLeft(CustomerDVCompany.AsString, 3.0);
      PrintLeft(CustomerDVPhone.AsString, 6.0);
      NewLine;
      PrintLeft(CustomerDVAddr1.AsString, 3.0);
      NewLine;
      If Length(CustomerDVAddr2.AsString) > 0 Then
        PrintLeft(CustomerDVAddr2.AsString, 3.0);
        NewLine;
      End;          // If Address Line 2
      sCityState := CustomerDVCity.AsString+' '+
                  CustomerDVState.AsString+' '+
                  CustomerDVZip.AsString;
      PrintLeft(sCityState, 3.0);
      NewLine;
      NewLine;
      CustomerDV.Next;
      If LinesLeft < 3 Then
        IDoHeader := True;
        NewPage;    // see note below about NewPage
      End;          // LinesLeft
    End;          // While Not EOF
  End;          // With Report
end OnPrint;

```

Technically, NewPage issues a "pause" to this script before doing the page break. During this pause, control is passed to other visual components (like another section component). When they are done, control is returned to this script and the page break is issued.

The next example is a simple watermark which could print the word "DEMO". This one uses several code base methods and properties including GoToXY, PageWidth and PageHeight.

```
{ Event for Section1.OnPrint }
function Section1_OnPrint(Self: TRaveSection; Report: TBaseReport);
begin
With Report Do
  SetFont('Arial', 72);
  FontColor := clYellow; // clYellow clSilver
  FontRotation := 50;
  GoToXY(PageWidth/4,PageHeight/3);
  Bold := True;
  Print('WaterMark');
  Bold := False;
  FontColor := clBlack;
  FontRotation := 0;
End;
end OnPrint;
```

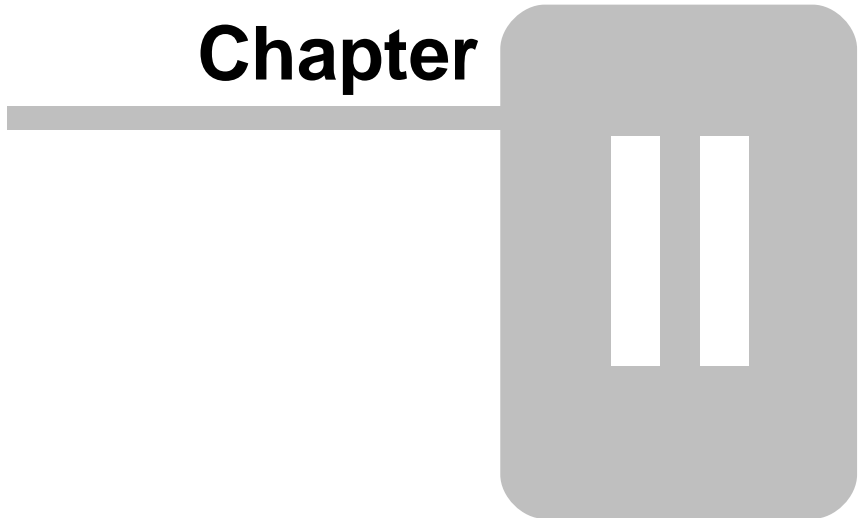
The last example shows how to generate some graphic shapes.

```
{ Event for Section1.OnPrint }
function Section1_OnPrint(Self: TRaveSection; Report: TBaseReport);
begin
  With Report Do
    // Rectangle(X1, Y1, X2, Y2);
    // outside box done first
    SetPen(clBlack, psClear, 12, pmCopy);
    SetBrush(clYellow, bsSolid);
    Rectangle(2.0, 1.0, 5.5, 4.0);
    // inside box done last
    SetPen(clBlack, psClear, 2, pmCopy);
    SetBrush(clSilver, bsSolid);
    Rectangle(3.0, 2.0, 4.5, 3.0);

    // create yellow ellipse
    // Ellipse(X1,Y1, X2,Y2);
    // outside Ellipse done first
    SetBrush(clSilver, bsSolid);
    SetPen(clBlack, psSolid, 12, pmCopy);
    Ellipse(0.0,6.0,7.5,8.0);
    // inside Ellipse done last
    SetBrush(clYellow, bsSolid);
    SetPen(clRed, psSolid, 4, pmCopy);
    Ellipse(1.0,6.5,6.5,7.5);
  End;
end OnPrint;
```


Rave Components

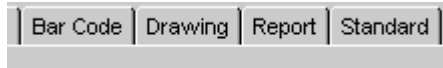
Chapter



2 Rave Components

2.1 Components

There are four component toolbars selectable from the main menu tabs.



They are the "Bar Code", "Drawing", "Report" and "Standard" components tabs. When you click on one of these tabs, it will open the toolbar for that particular set of components. A component is an item that is dropped on the page layout. Most of these will be visible at that point and can be moved or modified to achieved to the report layout that you want. A few of these components are not visible but are place on the page like "[DataCycle](#)", "[Font Master](#)" and many Calc items. These non-visible components can be seen and selected using the treeview panel.

Each of the components are described in more detail in the links below, but a common features to all of them will be the property panel. The properties of each component is where you would set the features for that particular component to what you desire. For example, line width and line color are two of the many properties for a line. Don't let the number of properties bother you. They are there to let you adjust that component and in many cases the default setting is adequate. There are a couple of properties that apply to all components, so we will discuss them here.

The "*Name*" property is used to assign a new name to that control or to find out what the name of the control is. By default, RAVE assigns sequential names based on the type of the component, such as 'Section1', 'Section2', and so on. You can change these (*and are encouraged to do so*) to more meaningful names that make both the Project Tree & code more readable. The "*Name*" property must not contain any spaces or special characters. This is the name that will be used in the Project Tree Panel.

Another common property is the "*Mirror*" property. The ability to mirror or duplicate a section, component etc is a very powerful feature of RAVE. It simple terms mirroring allows you to have a master form, letter, design. Items that are mirrored can not be changed, but more can be added. So if you had a letterhead with company logo saved, then you could mirror that design and then add the additional components for the desired output.

See Also:

[BarCode](#), [Drawing](#), [Report](#), [Standard](#), [Toolbars](#)

2.2 Toolbar

We need to learn something about toolbars. First of all a toolbar is a collection of icons that usually are related in their function(s). RAVE has several toolbars. To give you the most flexibility, these toolbars have been designed to stay out of the way unless you want them visible. This section will describe each item of the visual designer and describe how it works including any settings or properties that influence or control that item.



The main toolbar selection tabs are shown above. The menu item "Tools - Toolbars" has options to control the visibility of each toolbar. Each item acts like a toggle, click it on (checked) and that selected toolbar will be visible. Click it off (unchecked) and that same toolbar tab will not be visible. If a toolbar tab is visible then you can select any of those components by "clicking" on the icon desired. Also, notice that as the mouse moves over a component icon, that you will get pop-up help identifying what that particular component is for. The toolbars (Windows OS) are designed to remember their last position. So you can close it, do some design work and when you bring it back it will be displayed at its last position.



Docked Toolbar (only on Windows OS)



Floating Toolbar (only on Windows OS)

When a toolbar is visible (in the Windows OS) it can be in one of two states, docked or floating. If there is a ribbed section (like the one above) then that toolbar is docked. If it has a title bar on top (like the right one above) with an "X" to close the bar in the top right corner then it is a floating toolbar. This handle is the ribbed section if it is docked style or the title bar if it is floating style. To move a toolbar, place the mouse cursor somewhere on the "toolbar handle", press and hold the left mouse button while dragging the toolbar to its new location. If you move it to an area that it can be docked, the outline shape will change indicating that it is in a docking area and it will indeed dock there if you release the mouse at that point. The best way to get familiar with this is to try it yourself. It doesn't matter which toolbar you use. Activate one, move it around, make it dock on the sides of the page layout area.

Toolbar Tabs



OK now that you are familiar with toolbars in general let's go back and examine those Toolbar tabs more closely. The last 4 tabs (on the far right) are special toolbars (known as "Components Toolbar"). The tabs shown to the left of those "Components Toolbar" are known as tools. A component is usually a visible item that you place on the page, like a region, shape, line or bar code etc. There are a few non-visible components that are placed on a page, like the "Font Master" and some of the "Calc" components. A "tool" is something used to modify an item already on the page or is used to change your preferences on how Rave works..

See Also:

[Components](#)

2.3 DataField Types

[DataField](#) property of the [DataView](#) component. DataField is the base component for the following Rave DataView field types.

Properties	B C D	B L O B	B O O L E A N	C U R R E N C Y	D A T E	D A T E T I M E	F L O A T	G R A P H I C	I N T E G E R	M E M O	S T R I N G	T I M E
<i>Calculated</i>	X	X	X	X	X	X	X	X	X	X	X	X
Description	X	X	X	X	X	X	X	X	X	X	X	X
DevLocked	X	X	X	X	X	X	X	X	X	X	X	X
DisplayFormat	X	-	-	X	X	X	X	-	X	-	-	X
FieldName	X	X	X	X	X	X	X	X	X	X	X	X
FullName	X	X	X	X	X	X	X	X	X	X	X	X
Locked	X	X	X	X	X	X	X	X	X	X	X	X
Name	X	X	X	X	X	X	X	X	X	X	X	X
NullText	X	X	X	X	X	X	X	X	X	X	X	X
Size	X	X	X	X	X	X	X	X	X	X	X	X
Tag	X	X	X	X	X	X	X	X	X	X	X	X
Visible	X	X	X	X	X	X	X	X	X	X	X	X

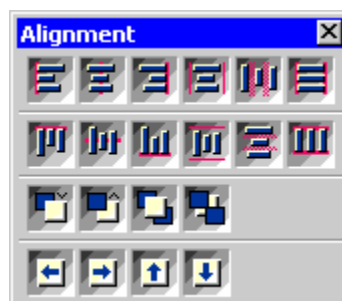
Typical Field Property Panel

CustomerDVCustNo: FloatField	
Calculated	False
Description	
DevLocked	False
DisplayFormat	
FieldName	CustNo
FullName	CustNo
Locked	False
Name	CustomerDVCustNo
NullText	
Size	8
Tag	0
Visible	True

2.4 Toolbars

2.4.1 Alignment Toolbar

The "Alignment" tab selects the toolbar that you would use to align a group of components (items) on your page.



The first step is to select the components that you want to align by holding the shift key down and clicking the mouse on each object that will need to be aligned. If you watch carefully, you will see standard pips appear around each item and these pips will be gray in color. Notice that the alignment toolbar is divided into four areas. The first section (6 tools) control horizontal alignment options. The next section (6 tools) control vertical alignment options. The third set of 4 tools move component(s) to the foreground or background. While the last set of 4 tools will tap (move) components in the direction shown. The visibility of the all tools shown on the bottom row are controlled by settings in the Alignment section of the Designer Tab in the Preferences area.

HINT

The tap tools have a speed key assigned to them. It is the Ctrl key plus the arrow key of the direction you wish the selected component(s) to move.

See Also:

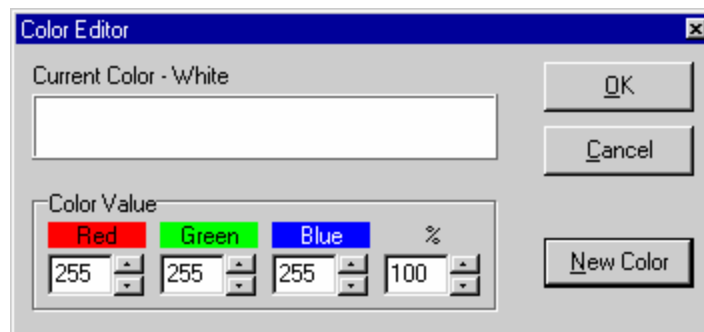
[Bar Code](#), [Components](#), [Designer](#), [Drawing](#), [Fills](#), [Fonts](#), [Lines](#), [Project](#), [Report](#), [Standard](#), [Toolbars](#), [Zoom](#)

2.4.2 Colors Toolbar

The "Colors" tab selects the toolbar that you would use to change a color of a selected object such as a line or box or the fill area in one of the shapes, circle, ellipse, rectangle or square.



This color palette operates similar to many windows color palettes. A left click selects the foreground color while the right click selects the background color.



Double clicking on the user defined colors (bottom row of above depiction) or on the foreground and background colors will open a more detail color control window where you can create custom colors. This panel allows you to adjust and save to one of the eight user defined slots some colors that you like to use. Remember that actual displayed colors are dependent of the video card settings. So if you are set to True Color setting for your video card then the representation should be consistent with other systems with a True Color setting. However, if the other system is set for 256 colors, then that video card will adjust the color to what it perceives is the closest matching color. So it is recommended that you set your system to the same settings as your target systems when defining custom colors. After your definitions have been completed, then you can return your video settings back to where you prefer to keep them.

See Also:

[Alignment](#), [Bar Code](#), [Components](#), [Designer](#), [Drawing](#), [Fills](#), [Fonts](#), [Lines](#), [Project](#), [Report](#), [Standard](#), [Toolbars](#), [Zoom](#)

2.4.3 Designer Toolbar

The "Designer" tab selects the toolbar that allows you to change some of the more common designer preferences.



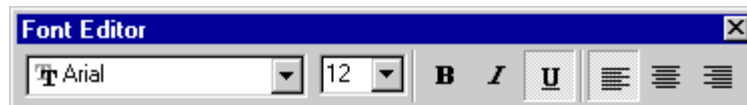
All of the designer toolbar items are toggles. Click it and it will change the state of that item. The first one is a view of a page and it controls whether the grid lines are visible or not on your page layout. The second one controls whether "snap to grid" is active. The third button determines whether the grid lines are always on top or not. The fourth button allows you to cause the toolbars to automatically "disappear" as soon as you make a selection from them. The fifth button controls the displaying of band headers. And the last button will bring up the Preferences dialog windows so that you can any of the preference settings.

See Also:

[Alignment](#), [Bar Code](#), [Colors](#), [Components](#), [Drawing](#), [Fills](#), [Fonts](#), [Lines](#), [Project](#), [Report](#), [Standard](#), [Toolbars](#), [Zoom](#)

2.4.4 Fonts Toolbar

The "Fonts" tab selects the toolbar that you would use to change or modify a group of "Text" components (items) on your page.



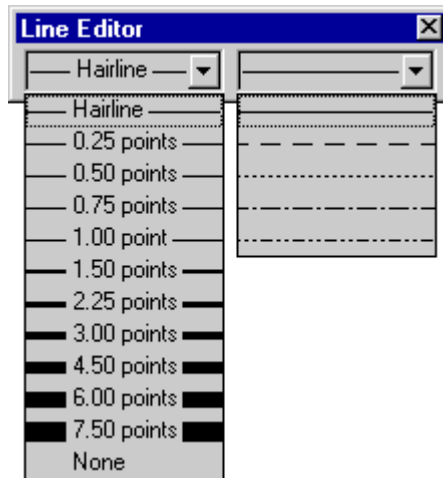
If the selected item(s) are text, then you can use the Font Editor to change the font type, size, attributes or position of the text in the text box. This toolbar is similar to those you would find in a word processor. It will allow you to change the font name and the attributes for the selected item. For instance, you can set the size of the font, whether it is bold, italic or underlined.

See Also:

[Alignment](#), [Bar Code](#), [Colors](#), [Components](#), [Designer](#), [Drawing](#), [Fills](#), [Lines](#), [Project](#), [Report](#), [Standard](#), [Toolbars](#), [Zoom](#)

2.4.5 Lines Toolbar

The "Lines" tab selects the toolbar that you would use to change or modify the style of a group of line components (items) on your page.



If the selected item is a line or a shape, then you can use the Line Editor to modify the style and thickness of the line or the border of the shape. The line editor toolbar allows you to select from several line styles, solid or combinations of dots, dashed. Click the drop down arrow and you will see a list of your options.

NOTE:

The line thickness only applies if the line style is solid. If you have selected another line and give it a thickness other than hairline, then the line style will be changed to solid. This restriction is imposed by the Windows operating system.

See Also:

[Alignment](#), [Anchor Editor](#), [Band Style Editor](#), [Bar Code](#), [Colors](#), [Components](#), [DataText Editor](#), [Designer](#), [Drawing](#), [Fills](#), [Font Editor](#), [Fonts](#), [Project](#), [Report](#), [Standard](#), [Toolbars](#), [Zoom](#)

2.4.6 Project Toolbar

The "Project" tab makes the common project level item(s) visible so that you can select the task desired.



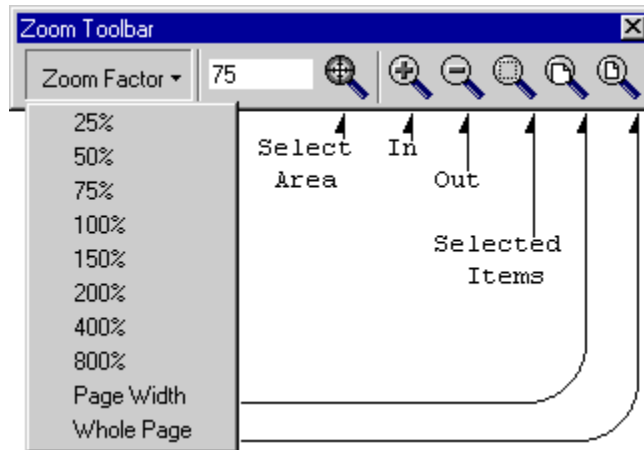
This provides a way to do the common functions that you would use with creating and saving a project. The left group of three icons in the Project Toolbar include "New Project", "Open Project" and "Save Project" functions. The middle group includes "New Report", "New Global Page" and "New Data View" options. The right group provides a way to "New Report Page" and "Execute Report".

See Also:

[Alignment](#), [Bar Code](#), [Colors](#), [Designer](#), [Drawing](#), [Fills](#), [Fonts](#), [Lines](#), [Report](#), [Standard](#), [Zoom](#)

2.4.7 Zoom Toolbar

The "Zoom" tab selects the toolbar that you would use to control viewing size of your page.



It is recommended that you first select an item that you want in the "new" view scale before changing the "Zoom Factor".

The Zoom Toolbar allows you to view your page layout as desired. You can zoom out to full page and get an overall view of all items or you can be zoomed in to get a more detailed view of a portion of the page. The Zoom Factor has a drop box menu to will allow you quickly go to full page, page width or one of several of fixed settings. If you want to zoom in then you can use the magnifying glass with a "+" symbol. Each click would go in by the incremental amount that is set in your preference settings. The "-" symbol does the same thing except it will zoom out with each click. If you want to see the details about item(s) that you have selected then use the glass with doted lines. If you want to look a particular area but it may or may not have selected items, use the glass with both the "+/-" in it and draw a box around the area that you wish to examine. The right two most magnifying glass are for zooming into "page width" or "full page" views. Drop a few components on a page and try the different zoom tools.

See Also:

[Alignment](#), [Bar Code](#), [Colors](#), [Components](#), [Designer](#), [Drawing](#), [Fills](#), [Fonts](#), [Lines](#), [Project](#), [Report](#), [Standard](#), [Toolbars](#)

2.5 BarCode Components

Bar Code components can be used to create many different kinds of Bar Codes in a Report.



Bar Codes are for users who know exactly what they need, as it requires background knowledge about Bar Codes and how they are used. It is not expected that the beginning user would have the background to use these components. To place a Bar Code, click on the needed Bar Code component button and click on the Page.



EANBarCode1: EANBarCode	
Anchor	(Top / Left)
AutoSize	True
BarCodeJustify	pjLeft
BarCodeRotation	Rot0
BarHeight	0.5
BarTop	0.6
BarWidth	0.01
Bottom	1.1
Center	2.875
DataField	
DataRow	
DevLocked	False
DisplayOn	doParent
Font	Arial, 11
Height	0.5
Left	2.4
Locked	False
Mirror	
Name	EANBarCode1
PrintChecksum	False
PrintReadable	True
PrintTop	False
Right	3.35
Tag	0
Text	000000000001
TextJustify	pjLeft
Top	0.6
UseChecksum	True
Visible	True
WideFactor	0
Width	0.95

To define the Bar Code value, go to the Property Panel and type the value into the Text property box. For Bar Codes containing check characters, please do not enter the check character, as it will be calculated automatically.

2.5.1 BarCode Components

The Bar Code tab controls the visibility of the Bar Code components toolbar (shown below).



Once the Bar Code components toolbar is active, you can select one of the bar code types ([2of5](#), [Code39](#), [Code128](#), [EAN](#), [PostNet](#) or [UPC](#)) and drop a box on the page that will contain that bar code style.

PostNetBarCode1: PostNetBarCode	
Anchor	(Top / Left)
AutoSize	True
BarCodeJustify	pjCenter
BarCodeRotation	Rot0
BarTop	1
Bottom	1.125
Center	4.918
DataField	
DataSource	
DevLocked	False
DisplayOn	doParent
Height	0.125
Left	3.75
Locked	False
Mirror	
Name	PostNetBarCode1
Right	6.085
Tag	0
Text	123451234
Top	1
Visible	True
Width	2.335

Once the bar code box is selected you will see the property window for that bar code and you can change the properties to the settings desired if the defaults are not correct. When you can see the selected bar code property window there are many settings. The names of these properties have been created in an attempt to minimize confusion about their purpose. Remember that the Reference section in this manual describes each of these properties. The property window shown below is for the Code 39 bar code. We will do a quick tour of the different bar code properties. Only two of all the bar code properties are unique to a single bar code type, the rest are common to all of the bar code types. The "Extended" property only applies to Code 39 bar code type. While the "CodePage" (not shown) applies only to Code 128 bar code.

"AutoSize" controls whether the size of the box will automatically adjust to fit the contents of the "Text" property. The "BarCodeJustify" determines where the bar code is printed relative to the box that you have drawn. "BarCodeRotation" allows the bar code to be rotated to 4 different orientations. "BarHeight" is the height of the tallest bar and "BarWidth" is the width of the narrow bar. "Bottom", "Center", "Left", "Right" and "Top" are the locations of those position of the box that you drew. "Font" is the name of the font to be printed above or below the bar code. "PrintChecksum" indicates if the readable text includes the checksum character or not. "PrintReadable" determines if you want the bar code text to be printed in a readable form using "Font". "PrintTop" determines if this readable text is to be printed on top or bottom of the barcode. "

Text" is the contents to be displayed as a bar code. The *TextJustify*" controls where the readable text is printed relative to the bar code patterns. *UseChecksum*" specifies whether a checksum character should be included with the bar code or not. *WideFactor*" is the ratio of the wide bar to narrow bar widths. *Width*" is the difference between *Left*" and *Right*" properties.

See Also:

[Alignment](#), [Colors](#), [Components](#), [Designer](#), [Drawing](#), [Fills](#), [Fonts](#), [Lines](#), [Project](#), [Report](#), [Standard](#), [Toolbars](#), [Zoom](#)

2.5.2 PostNetBarCode

PostNet bar code is different from other bar code types in that its shape and size is fixed by rules from the USPS (United States Postal Service). Therefore, all properties dealing with the shape or size should not be changed. This includes the properties that define the Checksum or its behavior.

PostNetBarCode uses the POSTNET (POSTal Numeric Encoding Technique) bar code and is specifically used by the Postal Service. It encodes ZIP Code information on letter mail for rapid and reliable sorting by bar code sorters. The PostNetBarCode can represent a five-digit ZIP code (32 bars), a nine-digit ZIP+4 code (52 bars), or an eleven-digit delivery point code (62 bars). Be aware that for the Post Office to recognize the bar code as being valid strict adherence to the guidelines must be met. Current standards require that the five digit zip plus four be used plus the two digit carrier route, which is most often obtained from the first two digits of the street address. It is recommended that the user check with the Post Office to obtain a copy of their current guidelines.

PostNetBarCode1: PostNetBarCode	
Anchor	(Top / Left)
AutoSize	True
BarCodeJustify	pjCenter
BarCodeRotation	Rot0
BarTop	1
Bottom	1.125
Center	4.918
DataField	
DataRow	
DevLocked	False
DisplayOn	doParent
Height	0.125
Left	3.75
Locked	False
Mirror	
Name	PostNetBarCode1
Right	6.085
Tag	0
Text	123451234
Top	1
Visible	True
Width	2.335



Example of PostNetBarCode: 85210304119



Properties of PostNetBarCode

[Anchor](#), [AutoSize](#), [BarCodeJustify](#), [BarCodeRotation](#), [BarTop](#), [Bottom](#), [Center](#), [DataField](#), [DataView](#), [DevLocked](#), [DisplayOn](#), [Height](#)(Read ONLY), [Left](#), [Locked](#), [Mirror](#), [Name](#), [Right](#), [Tag](#), [Text](#), [Top](#), [Visible](#), [Width](#) (active only if AutoSize is set to False)

2.5.3 I2of5BarCode

2of5Bar Code (interleaved 2 of 5) is a numbers-only bar code; in the Property Panel it is labeled I2of5BarCode. The symbol can be as long as necessary to store the encoded data. The code is a high-density code that can hold up to 18 digits per inch when printed using a 7.5 mil X dimension. "Interleaved" comes from the fact that the digit is encoded in the bars and the next digit is encoded in the spaces. There are five bars, two of which are wide and five spaces, two of which are wide.

I2of5BarCode1: I2of5BarCode	
Anchor	(Top / Left)
AutoSize	True
BarCodeJustify	pjLeft
BarCodeRotation	Rot0
BarHeight	0.25
BarTop	1
BarWidth	0.01
Bottom	1.25
Center	2.025
DataField	
DataView	
DevLocked	False
DisplayOn	doParent
Font	Arial,8
Height	0.25
Left	1.8
Locked	False
Mirror	
Name	I2of5BarCode1
PrintChecksum	False
PrintReadable	True
PrintTop	False
Right	2.25
Tag	0
Text	251
TextJustify	pjLeft
Top	1
UseChecksum	False
Visible	True
WideFactor	3
Width	0.45



2632534


Properties of I2of5BarCode

[Anchor](#), [AutoSize](#), [BarCodeJustify](#), [BarCodeRotation](#), [BarHeight](#), [BarTop](#), [BarWidth](#), [Bottom](#), [Center](#), [DataField](#), [DataView](#), [DevLocked](#), [DisplayOn](#), [Font](#), [Height](#), [Left](#), [Locked](#), [Mirror](#), [Name](#), [PrintChecksum](#), [PrintReadable](#), [PrintTop](#), [Right](#), [Tag](#), [Text](#), [TextJustify](#), [Top](#), [UseChecksum](#), [Visible](#), [WideFactor](#), [Width](#)

2.5.4 Code39BarCode

Code39BarCode is an alphanumeric bar code that can encode decimal numbers, the uppercase alphabet, and the following special symbols "_", ".", "*", "\$", "/", "%", and "+". The characters are constructed using nine elements, five bars and four spaces. Of these nine elements, two of the bars and one of the spaces are wider than the rest. Wide elements represent binary ones (1), and narrow elements represent binary zeros (0).

Code39BarCode1: Code39BarCode	
Anchor	(Top / Left)
AutoSize	True
BarCodeJustify	pjLeft
BarCodeRotation	Rot0
BarHeight	0.25
BarTop	1.8
BarWidth	0.01
Bottom	2.05
Center	3.035
DataField	
DataRowView	
DevLocked	False
DisplayOn	doParent
Extended	False
Font	Arial,8
Height	0.25
Left	2.4
Locked	False
Mirror	
Name	Code39BarCode1
PrintChecksum	False
PrintReadable	True
PrintTop	False
Right	3.67
Tag	0
Text	C39BC1
TextJustify	pjLeft
Top	1.8
UseChecksum	False
Visible	True
WideFactor	3
Width	1.27

CODE 39

See Also:

[Anchor](#), [AutoSize](#), [Bar Code Components](#), [BarCodeJustify](#), [BarCodeRotation](#), [BarHeight](#), [BarTop](#), [BarWidth](#), [Bottom](#), [Center](#), [DataField](#), [DataRowView](#), [DevLocked](#), [DisplayOn](#), [Extended](#), [Font](#), [Height](#), [Left](#), [Locked](#), [Mirror](#), [Name](#), [PrintChecksum](#), [PrintReadable](#), [PrintTop](#), [Right](#), [Tag](#), [Text](#), [TextJustify](#), [Top](#), [UseChecksum](#), [Visible](#), [WideFactor](#), [Width](#)

2.5.5 Code128BarCode

Code 128 bar code is a very high-density alphanumeric Bar Code. The symbol will be as long as necessary to store the encoded data. It is designed to encode all 128 ASCII characters and will use the least amount of space for data of 6 characters or more of any 1-D symbology. Each data character is made up of 11 black or white modules. The stop character is made up of 13 modules. Three bars and three spaces are formed out of these 11 modules. Bar and spaces vary between 1 and 4 modules. The following are the ASCII representations of special codes that may be needed: FNC1 = #241, FNC2 = #242, FNC3 = #243, FNC4 = #244, CodeA = #245, CodeB = #246, CodeC = #247, Shift = #248.



Code128BarCode1: Code128BarCode	
Anchor	(Top / Left)
AutoSize	True
BarCodeJustify	pjLeft
BarCodeRotation	Rot0
BarHeight	0.25
BarTop	1
BarWidth	0.01
Bottom	1.25
Center	2.86
CodePage	cpCodeA
DataField	
DataRow	
DevLocked	False
DisplayOn	doParent
Font	Arial,8
Height	0.25
Left	2.3
Locked	False
Mirror	
Name	Code128BarCode1
Optimize	True
PrintChecksum	False
PrintReadable	True
PrintTop	False
Right	3.42
Tag	0
Text	Code128
TextJustify	pjLeft
Top	1
Visible	True
WideFactor	0
Width	1.12

Example Code 128:

UCC/EAN-128 barcode:

UCC (Uniform Code Council) / EAN (European Article Number) -128 barcode is not a new or unique symbology, but is a standard or definition of how Code128 data is to be formatted. The UCC/EAN encodes your data using standard Code128 symbology. The UCC/EAN-128 standard structure is:

- A code 128 start character (StartA, StartB or StartC)
- A code 128 FNC1 character
- Application Identifier (AI)
- Data to be encoded
- Check Character
- Stop Character
- Termination bar

See Also:

[Anchor](#), [AutoSize](#), [Bar Code Components](#), [BarCodeJustify](#), [BarCodeRotation](#), [BarHeight](#), [BarTop](#), [BarWidth](#), [Bottom](#), [Center](#), [CodePage](#), [DataField](#), [DataView](#), [DevLocked](#), [DisplayOn](#), [Font](#), [Height](#), [Left](#), [Locked](#), [Mirror](#), [Name](#), [PrintChecksum](#), [PrintReadable](#), [PrintTop](#), [Right](#), [Tag](#), [Text](#), [TextJustify](#), [Top](#), [UseChecksum](#), [Visible](#), [WideFactor](#), [Width](#)

2.5.6 UPCBarCode

UPC (Universal Product Code) bar code has a fixed length of 12-digits and can only encode numbers. UPC was designed for coding products. The format allows the symbol to be scanned with any package orientation. The check digit is calculated so there is no need to enter it when typing the value into the Text property.



UPCBarCode1: UPCBarCode	
Anchor	(Top / Left)
AutoSize	True
BarCodeJustify	pjLeft
BarCodeRotation	Rot0
BarHeight	0.5
BarTop	1.5
BarWidth	0.01
Bottom	2
Center	2.875
DataField	
DataView	
DevLocked	False
DisplayOn	doParent
Font	Arial, 11
Height	0.5
Left	2.4
Locked	False
Mirror	
Name	UPCBarCode1
PrintChecksum	False
PrintReadable	True
PrintTop	False
Right	3.35
Tag	0
Text	000000000001
TextJustify	pjLeft
Top	1.5
UseChecksum	True
Visible	True
WideFactor	0
Width	0.95

Example 712345678935

Properties of UPCBarCode

[Anchor](#), [AutoSize](#), [BarCodeJustify](#), [BarCodeRotation](#), [BarHeight](#), [BarTop](#), [BarWidth](#), [Bottom](#), [Center](#), [DataField](#), [DataView](#), [DevLocked](#), [DisplayOn](#), [Extended](#), [Font](#), [Height](#), [Left](#), [Locked](#), [Mirror](#), [Name](#), [PrintChecksum](#), [PrintReadable](#), [PrintTop](#), [Right](#), [Tag](#), [Text](#), [TextJustify](#), [Top](#), [UseChecksum](#), [Visible](#), [WideFactor](#), [Width](#)

2.5.7 EANBarCode

EAN (European Article Numbering System) bar code is identical to the UPC, except for the number of digits. EAN has a length of 13 digits – 10 numeric characters, and 2 "flag" characters that are usually country codes, and a check digit. This Bar Code is typically used for Non-U.S coding. The check digit is calculated for you so you do not need to enter it when typing the value into the Text property.

EANBarCode1: EANBarCode	
Anchor	(Top / Left)
AutoSize	True
BarCodeJustify	pjLeft
BarCodeRotation	Rot0
BarHeight	0.5
BarTop	0.6
BarWidth	0.01
Bottom	1.1
Center	2.875
DataField	
DataView	
DevLocked	False
DisplayOn	doParent
Font	Arial,11
Height	0.5
Left	2.4
Locked	False
Mirror	
Name	EANBarCode1
PrintChecksum	False
PrintReadable	True
PrintTop	False
Right	3.35
Tag	0
Text	000000000001
TextJustify	pjLeft
Top	0.6
UseChecksum	True
Visible	True
WideFactor	0
Width	0.95



Example 3847348484584

See Also:

[Anchor](#), [AutoSize](#), [BarCodeJustify](#), [BarCodeRotation](#), [BarHeight](#), [BarTop](#), [BarWidth](#), [Bottom](#), [Center](#), [DataField](#), [DataView](#), [DevLocked](#), [DisplayOn](#), [Font](#), [Height](#), [Left](#), [Locked](#), [Mirror](#), [Name](#), [PrintChecksum](#), [PrintReadable](#), [PrintTop](#), [Right](#), [Tag](#), [Text](#), [TextJustify](#), [Top](#), [UseChecksum](#), [Visible](#), [WideFactor](#), [Width](#)

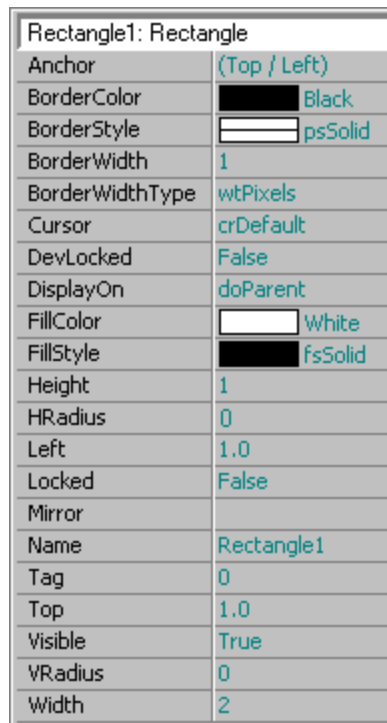
2.6 Drawing Components

2.6.1 Drawing Components

The "Drawing" tab controls the visibility of the Drawing components toolbar (shown below).



Once the Drawing components toolbar is active, you can select one of the shape types (line, horizontal line, vertical line, [rectangle](#), [square](#), [ellipse](#), [circle](#)) and drop a box on the page layout that will contain that shape style.







If the shape is selected you will see the property window for that shape and you can change the properties to the settings desired if the defaults are not correct. As you can see with the property window there are many settings. The names of these properties have been created in an attempt to convey their purpose. Remember that the Reference section in this manual describes each of these properties. The property window shown below is for the Rectangle shape. We will do a quick tour of the different drawing shape properties.

When you select an item, you will get a set of pips around that item. In some respects these pips behave in a normal manner. However, the pips also will give you some additional information about the selected item(s). If the pips are GREEN then you have selected a single item and you can "drag" any of the pips to resize that component. However; if you select more than one item, then the pips will become GRAY and you can move them as a group but can not resize them. If you notice that the pips are RED, then that item is mirrored. You can change the starting location of a mirrored item (left and top properties), but you can not resize a mirrored item. So GREEN pips mean a normal single items has been selected and dragging the different pips is allowed. GRAY or RED pips indicate that the selected item(s) have a restricted behavior.

See Also:

[Circle](#), [Ellipse](#), [HLine](#), [Line](#), [Rectangle](#), [Square](#), [VLine](#),
[Alignment](#), [Bar Code](#), [Colors](#), [Components](#), [Designer](#), [Fills](#), [Fonts](#), [Lines](#), [Project](#), [Report](#), [Standard](#),
[Toolbars](#), [Zoom](#)

2.6.2 Circle Component[Description](#)

Circle1: Circle	
Anchor	(Top / Left)
BorderColor	 Black
BorderStyle	 psSolid
BorderWidth	1
BorderWidthType	wtPixels
Cursor	crDefault
DevLocked	False
DisplayOn	doParent
FillColor	 White
FillStyle	 fsSolid
Height	1
Left	1
Locked	False
Mirror	
Name	Circle1
Tag	0
Top	1
Visible	True
Width	1

Properties of Circle





[Anchor](#), [BorderColor](#), [BorderStyle](#), [BorderWidth](#), [BorderWidthType](#), [Cursor](#), [DevLocked](#), [DisplayOn](#),
[FillColor](#), [Height](#), [Left](#), [Locked](#), [Mirror](#), [Name](#), [Tag](#), [Top](#), [Visible](#), [Width](#)

2.6.3 Ellipse Component**See Also:**

[Anchor](#), [BorderColor](#), [BorderStyle](#), [BorderWidth](#), [BorderWidthType](#), [Cursor](#), [DevLocked](#), [DisplayOn](#),
[FillColor](#), [Height](#), [Left](#), [Locked](#), [Mirror](#), [Name](#), [Tag](#), [Top](#), [Visible](#), [Width](#)

2.6.4 Rectangle Component[Description](#)







Rectangle1: Rectangle	
Anchor	(Top / Left)
BorderColor	 Black
BorderStyle	 psSolid
BorderWidth	1
BorderWidthType	wtPixels
Cursor	crDefault
DevLocked	False
DisplayOn	doParent
FillColor	 White
FillStyle	 fsSolid
Height	1
HRadius	0
Left	1.0
Locked	False
Mirror	
Name	Rectangle1
Tag	0
Top	1.0
Visible	True
VRadius	0
Width	2

Properties of Rectangle

[Anchor](#), [BorderColor](#), [BorderStyle](#), [BorderWidth](#), [BorderWidthType](#), [Cursor](#), [DevLocked](#), [DisplayOn](#), [FillColor](#), [Height](#), [HRadius](#), [Left](#), [Locked](#), [Mirror](#), [Name](#), [Tag](#), [Top](#), [Visible](#), [VRadius](#), [Width](#)

2.6.5 Square Component

Square1: Square	
Anchor	(Top / Left)
BorderColor	 Black
BorderStyle	 psSolid
BorderWidth	1
BorderWidthType	wtPixels
Cursor	crDefault
DevLocked	False
DisplayOn	doParent
FillColor	 White
FillStyle	 fsSolid
Height	1
HRadius	0
Left	2.4
Locked	False
Mirror	
Name	Square1
Tag	0
Top	1.7
Visible	True
VRadius	0
Width	1

Properties of Square

[Anchor](#), [BorderColor](#), [BorderStyle](#), [BorderWidth](#), [BorderWidthType](#), [Cursor](#), [DevLocked](#), [DisplayOn](#), [FillColor](#), [Height](#), [HRadius](#), [Left](#), [Locked](#), [Mirror](#), [Name](#), [Tag](#), [Top](#), [Visible](#), [VRadius](#), [Width](#)

2.7 Project Components**2.7.1 DataView Component**

[DataView](#) property

[DataView](#) dictionary



Property Panel	
CustomerDV: TRaveDataView	
ConnectionName	
Description	
DevLocked	False
FullName	
Locked	False
Name	
Tag	0

See Also:

[ConnectionName](#), [Description](#), [DevLocked](#), [FullName](#), [Locked](#), [Name](#), [Tag](#), [Visible](#)

2.7.2 Page Component

Description



Page1: Page	
Bin	
BinCustom	
Description	
DevLocked	False
FullName	Page1
GotoMode	gmGotoDone
GotoPage	
GridLines	5
GridSpacing	0.1
Locked	False
Name	Page1
Orientation	poDefault
PageHeight	11
PageWidth	8.5
PaperSize	Custom
Parameters	
PIVars	
Tag	0
Visible	True
WasteFit	False

Properties of Page

[Bin](#), [BinCustom](#), [Description](#), [DevLocked](#), [FullName](#), [GotoMode](#), [GotoPage](#), [GridLines](#), [GridSpacing](#), [Locked](#), [Module](#), [Name](#), [Orientation](#), [PageHeight](#), [PageWidth](#), [PaperSize](#), [Parameters](#), [PIVars](#), [Tag](#), [Visible](#), [WasteFit](#)

2.7.3 ProjectManager Component

RaveProject: ProjectManager	
AdminPassword	
Categories	
Description	
DevLocked	False
FullName	RaveProject
Locked	False
Name	RaveProject
Parameters	
PIVars	
SecurityControl	
Tag	0
Units	unInch
UnitsFactor	1
Visible	True

Properties of ProjectManager

[AdminPassword](#), [Categories](#), [Description](#), [DevLocked](#), [FullName](#), [Locked](#), [Name](#), [Parameters](#), [PIVars](#), [SecurityControl](#), [Tag](#), [Units](#), [UnitsFactor](#), [Visible](#)

2.7.4 Report Component

[Click HERE](#)

for description of this component



BioLife: Report	
AlwaysGenerate	False
Category	
Collate	pcDefault
Copies	0
Description	(Memo)
DevLocked	False
Duplex	pdDefault
FirstPage	Page1
FullName	BioLife
Locked	False
MaxPages	0
Name	BioLife
PageList	
Parameters	
PIVars	
Printer	
Resolution	prDefault
SecurityControl	
Tag	0
Visible	True

Properties of Report:

[AlwaysGenerate](#), [Category](#), [Collate](#), [Copies](#), [Description](#), [DevLocked](#), [Duplex](#), [FirstPage](#), [FullName](#), [Locked](#), [MaxPages](#), [Name](#), [PageList](#), [Parameters](#), [PIVars](#), [Printer](#), [Resolution](#), [SecurityControl](#), [Tag](#), [Visible](#)

2.8 Report Components

2.8.1 Report Components

The "Report" tab controls the visibility of the toolbar that allows you to select a "Report" component and place it on your page.



There are several report components. They are the [Region](#) component and many band components ([Band](#), [CalcController](#), [CalcOp](#), [CalcText](#), [CalcTotal](#), [DataBand](#), [DataCycle](#), [DataMemo](#), [DataMirrorSection](#) and [DataText](#)). The first thing to understand is that a region's sole purpose is to hold band(s). You are not limited on the number of bands within one region. Nor are you limited to a single region. You can have many regions on a page or a region within a region or a region within a band. And you can have many bands within each region. Each band type has its own set of properties to control its behavior. This is covered in more detail in the manual.

This is where the flexibility of RAVE starts to appear. You may need to sit back and visualize your report requirements a bit. Get it down to an outline concept and then go to it. These regions within a band could be thought of giving you the ability to do composite or sub-report(s). It could be a master product list with the exploded parts listed below each product item.

See Also:

[Band](#), [Components](#), [DataBand](#), [DataMemo](#), [DataMirrorSection](#), [DataText](#), [Region](#), [Toolbars](#)

Toolbars:

[Alignment](#), [Colors](#), [Designer](#), [Fills](#), [Fonts](#), [Lines](#), [Project](#), [Zoom](#)

Components:

[Bar Code](#), [Drawing](#), [Standard](#)

Non-Visible Components (visible ONLY in the Project Tree view)

[CalcController](#), [CalcOp](#), [CalcText](#), [CalcTotal](#), [DataCycle](#)

2.8.2 Band Component

A band component is placed in a region and is for items that are "fixed" and do not change on the page. In general, the Band component will contain "Text" and "[CalcText](#)" components. The primary examples of a band component would be headers and footers. Remember the Band component can contain data-aware components so you can have a table field in the band. So a Group footer might have a '<TdbTable1.CustomerName> Totals' on this band.



An important property for the band component is the "[ControllerBand](#)". This property determines which DataBand is in control. When the controlling band has been set you will notice that the graphic symbol on the band will point in the direction of that controlling band and that the color of the symbols will match.

There is a preference setting called "Always Show Band Headers" available on the "[Designer Tab](#)" or the "Edit - Preferences menu". This setting will change the appearance of the bands while you are in the design mode. Having this setting "off" will give an appearance closer to the actual output, but will not show you the band headers with their symbols and codes. When you first start using Rave it might be beneficial to try it with this setting "on" and take advantage of the visual clues provided by the band headers. After you are comfortable with the use of bands, then change this setting to fit your needs. The letter codes shown on the right of the band are explain in this section on "Band Style Editor" section. Basically, they are informing you about that bands behavior. The bold letters are ON or active while the subdued letters are off or inactive.

See Also:

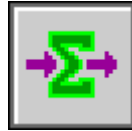
[BandStyle Editor](#), [DataBand](#), [Region](#), [Report Components](#)

Properties:

[AllowSplit](#), [BandStyle](#), [ControllerBand](#), [DesignerHide](#), [DevLocked](#), [FinishNewPage](#), [GroupDataView](#), [GroupKey](#), [Height](#), [Locked](#), [MinHeightLeft](#), [Name](#), [PositionMode](#), [PositionValue](#), [ReprintLocs](#), [StartNewPage](#), [Tag](#), [Visible](#)

2.8.3 CalcController Component

"CalcController" is a non-visual component that acts as a controller, along with DataBands, for CalcText and CalcTotal components through their Controller properties.



CalcController1: CalcController	
DevLocked	False
InitCalcVar	
InitDataField	
InitValue	0
Locked	False
Name	CalcController1
Tag	0
Visible	True

When the controller component is printed, it signals all calculation components that it controls to perform their summation operation. This allows you to perform totals on group bands, detail bands or whole pages depending upon the location of the CalcController component. Another feature of the CalcController component is its ability to initialize a CalcText or CalcTotal component to a specific value (through the InitCalcVar, InitDataField and InitValue properties). A CalcController component will only initialize values if it is used in the Initializer property of CalcText or CalcTotal property.

See Also:

[CalcText](#), [CalcTotal](#), [DevLocked](#), [InitCalcVar](#), [InitDataField](#), [InitValue](#), [Locked](#), [Name](#), [Tag](#), [Visible](#)

2.8.4 CalcOp Component

The "CalcOp" component is a non-visual component that allows you to take the values of two different sources (Src#CalcVar, Src#DataField or Src#Value) and perform an operation on them (defined by the Operator property).



CalcOp1: CalcOp	
DestParam	
DestPIVar	
DevLocked	False
DisplayFormat	
DisplayType	dtNumericFormat
Locked	False
Name	CalcOp1
Operator	coAdd
ResultFunction	cfNone
Src1CalcVar	
Src1DataField	
Src1DataView	
Src1Function	cfNone
Src1Value	0
Src2CalcVar	
Src2DataField	
Src2DataView	
Src2Function	cfNone
Src2Value	0
Tag	0
Visible	True

The result can then be saved in a project parameter like CalcTotal (through the DestParam and DisplayFormat properties). You can also chain CalcOp components together for more complex expressions using the Src#CalcVar properties which can be set to other CalcOp or CalcTotal components. Lastly, each source value has a function that can be applied to it to do things such as rounding or other math functions and the result of the operation can also have a function applied to it through the ResultFunction property.

The CalcOp component can put its result in a project parameter that will most likely be printed by a DataText component. *It is very important to set the print order of your CalcOp components to print before any DataText components that will be printing out the same project parameter. Use the Project Tree and the Alignment Palette to change the print order of your components.*

WARNING

If you use parameters as one or both of the sources for the CalcOp, then you need to be aware that the format of the parameters is important. Negative numbers that are formatted with a minus sign ("-number") with no spaces are OK and will work. However, formatting negative numbers with parenthesis "()" will not work. All parameters are string values and the string to number conversion process will not work with non-numeric characters like parenthesis.

See Also:

[DestParam](#), [DestPIVar](#), [DevLocked](#), [DisplayFormat](#), [DisplayType](#), [Locked](#), [Name](#), [Operator](#), [Report Components](#), [ResultFunction](#), [Src#CalcVar](#), [Src#DataField](#), [Src#DataView](#), [Src#Function](#), [Src#Value](#), [Tag](#), [Visible](#)

2.8.5 CalcText Component

The "CalcText" component is data aware. It is very similar to the dbEdit component in Delphi. This means you can use it to display a field from a dataset just about anywhere on the page layout. The main difference between "DataText" and "CalcText" is that "CalcText" is specially designed to do some form of calculation and placing the results of that calculation somewhere on the page layout.



CalcText1: CalcText	
Anchor	(Top / Left)
CalcType	ctSum
Color	 Black
Controller	
CountBlanks	True
CountNulls	False
CountValue	
DataField	
DataRow	
DevLocked	False
DisplayFormat	
DisplayOn	doParent
DisplayType	dtNumericFormat
Font	Arial,10
FontJustify	plLeft
FontMirror	
Initializer	
Left	0.6
Locked	False
Mirror	
Name	CalcText1
Rotation	0
RunningTotal	False
Tag	0
Top	0.5
Truncate	False
Visible	True
Width	1

The *CalcType* property determines the type of calculation being performed and includes Average, Count, Maximum, Minimum, and Sum. For example, this could be used to print the Totals of an invoice at the top of each page of an invoice.

If your field might contain blanks, then be sure to set the *CountBlanks* property to get the behavior you desire. The *CountBlanks* property determines whether blank field values are included in the Average and Count calculation methods.

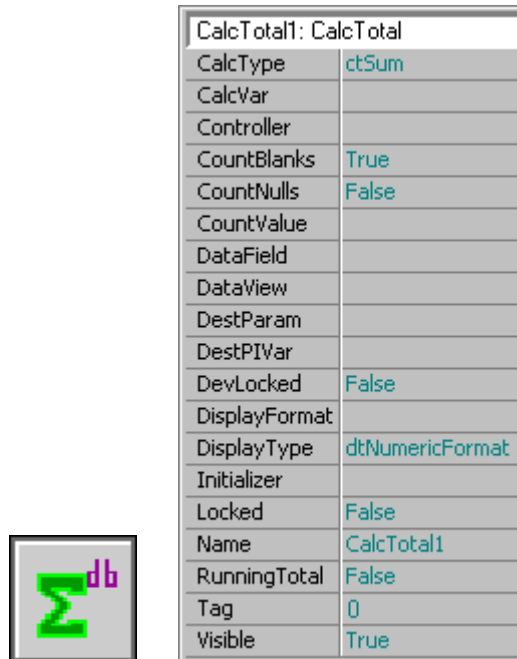
If *RunningTotal* is true then the calculation will not be reset to 0 each time it is printed.

See Also:

[Anchor](#), [CalcType](#), [Color](#), [Controller](#), [CountBlanks](#), [CountNulls](#), [CountValue](#), [DataField](#), [DataRow](#), [DevLocked](#), [DisplayFormat](#), [DisplayOn](#), [DisplayType](#), [Font](#), [FontJustify](#), [FontMirror](#), [Initializer](#), [Left](#), [Locked](#), [Mirror](#), [Name](#), [Report Components](#), [Rotation](#), [RunningTotal](#), [Tag](#), [Top](#), [Truncate](#), [Visible](#), [Width](#)

2.8.6 CalcTotal Component

The "CalcTotal" is a non-visual version of the CalcText component.



When this component is "printed", its value is typically stored in a project parameter (defined by the DestParam property) and formatted according to the DisplayFormat property. This can be useful if you want to perform totaling calculations that will be used with other calculations before being printed. You can also leave DestParam blank if the value of CalcTotal will only be used by other calculation components such as CalcOp.

See Also:

[CalcType](#), [CalcVar](#), [Controller](#), [CountBlanks](#), [CountNulls](#), [CountValue](#), [DataField](#), [DataRow](#), [DestParam](#), [DestPIVar](#), [DevLocked](#), [DisplayFormat](#), [DisplayType](#), [Initializer](#), [Locked](#), [Name](#), [Report Components](#), [RunningTotal](#), [Tag](#), [Visible](#)

2.8.7 DataBand Component

The "DataBand" is a data-aware band (placed in a region) that you would use if you want to display iterating information from a table.

In general, a DataBand will contain several DataText components. An example of that would be an aging report that loops through a data table printing only those rows (records) that are more than 60 days old. Both master and detail records of a master-detail report would be DataBand types. So if you had a typical invoice system with Customer and Items tables. The customer table would be the master and its DataBand would be set as a master band. The items table would be the detail records and its DataBand would be set as controlled by the customer table.



DataBand1: DataBand	
AllowSplit	False
BandStyle	(Band Styles)
Columns	1
ColumnSpacing	0
ControllerBand	
DataView	
DesignerHide	False
DetailKey	
DevLocked	False
FinishNewPage	False
GroupDataView	
GroupKey	
Height	0.2
InitToFirst	True
KeepBodyTogether	False
KeepRowTogether	False
Locked	False
MasterDataView	
MasterKey	
MaxRows	0
MinHeightLeft	0
Name	DataBand1
OrphanRows	0
PositionMode	pmOffset
PositionValue	0
ReprintLocs	(All)
SortKey	
StartNewPage	False
Tag	0
Visible	True
WidowRows	0

See Also:

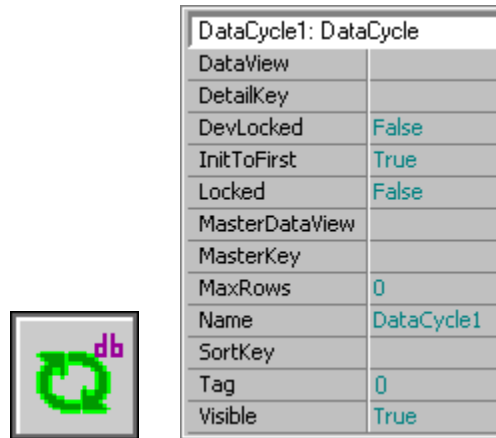
[Band](#), [BandStyle Editor](#), [ControllerBand](#), [Region](#), [Report Components](#)

Properties:

[AllowSplit](#), [BandStyle](#), [Columns](#), [ColumnSpacing](#), [ControllerBand](#), [DataView](#), [DesignerHide](#), [DetailKey](#), [DevLocked](#), [FinishNewPage](#), [GroupDataView](#), [GroupKey](#), [Height](#), [InitToFirst](#), [KeepBodyTogether](#), [KeepRowTogether](#), [Locked](#), [MasterDataView](#), [MasterKey](#), [MaxRows](#), [Name](#), [OrphanRows](#), [PositionMode](#), [PositionValue](#), [ReprintLocs](#), [SortKey](#), [StartNewPage](#), [Tag](#), [Visible](#), [WidowRows](#)

2.8.8 DataCycle Component

The "DataCycle" component is a non-visual data-aware component that you would use if you want to control iterating information from a data view. In general, a DataCycle component would be used to "Loop" or "Cycle" through the detail records until a change occurs at the master record level. This will normally be used with form based reports where one master record needs to control the printing of detail record(s) on one or more pages. The DataCycle component would be used instead of Region / DataBand combination to allow proper printing of DataText components that are placed directly on the page.



A DataCycle component is used to integrate through the data on a "form style" report that has no bands. You can think of DataCycle as a "fetch" your data and then do your printing command. Be sure that the DataCycle component is the first component on the list shown in the Tree View.

See Also:

[DataView](#), [DetailKey](#), [DevLocked](#), [InitToFirst](#), [Locked](#), [MasterDataView](#), [MasterKey](#), [MaxRows](#), [Name](#), [Report Components](#), [SortKey](#), [Tag](#), [Visible](#)

2.8.9 DataMemo Component

The "DataMirrorSection" component is a section that will "Mirror" other "Sections" based on the contents of a "DataField". This component is very flexible since it is mirroring sections. Remember that sections can contain graphics, regions, text, etc.

An example using this "DataMirrorSection" component is included in the RaveDemo project file. This example shows how to have a single report produce different envelope formats when sending to International or US addresses. The template for the international customers includes the country line and is be centered in the envelope while the US format does not have the country line and is offset to the right of center and lower on the envelope.



DataMemo1: DataMemo	
Anchor	(Top / Left)
Color	Black Black
ContainsRTF	False
DataField	
DataSource	
DevLocked	False
DisplayOn	doParent
ExpandParent	True
Font	Arial, 10
FontJustify	pjLeft
FontMirror	
Height	2
Left	0.6
Locked	False
MailMergeItems	
Mirror	
Name	DataMemo1
Tag	0
Top	0.4
Truncate	False
Visible	True
Width	3

This example shows several techniques on how to use sections and mirrors. On page 2 there are three sections. Section 3 has the common address lines used by both templates. Section 1 and 2 contain mirrors to Section 3. The International section has the additional country line added in its template. Note, that "remarks" have been added to the page two design and they are not part of any of the sections. The "US Template" and "International Template" are just plain text remark that assist you in the purpose of each section shown.

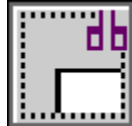
Once you understand how the master templates were done, go to Page 1, and select the "DataMirrorSection". Notice the "DataSource" and "DataField" properties have been set to the field that will be used to select which format template to mirror. To examine that logic, go to the "DataMirrors" property and click on the ellipsis to open the Data Mirror Editor. Select each "DataMirror" item and note the settings in the bottom portion of the dialog box. This example has two "templates", but you can add more to fit more complex reporting needs.

WARNING: Be sure to define one of the settings as a default. If a default is NOT defined and the field value does not match any of the other settings, then the format used will be that set in the mirror property of the DataMirrorSection. You could get unpredictable results since this is usually blank.

[ExpandParent](#)

2.8.10 DataMirrorSection Component

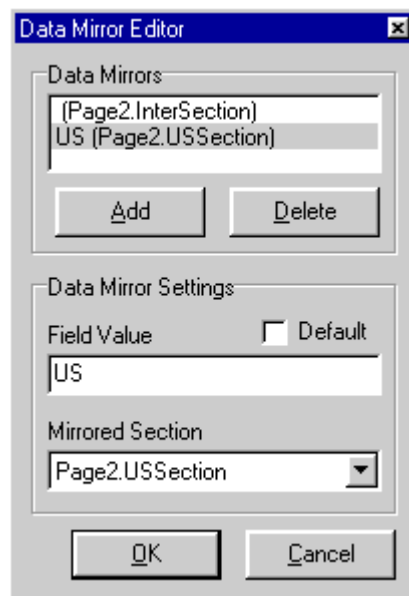
The "DataMirrorSection" component is a section that will "Mirror" other "Sections" based on the contents of a "DataField".



DataMirrorSection1: DataMirrorSection	
Anchor	(Top / Left)
CompareCase	False
DataField	
DataMirrors	
DataSource	
DevLocked	False
DisplayOn	doParent
Height	3
Left	1.1
Locked	False
Mirror	
Name	DataMirrorSection1
Tag	0
Top	0.3
Visible	True
WasteFit	False
Width	3

This component is very flexible since it is mirroring sections. Remember that sections can contain graphics, regions, text, etc.

An example using this "DataMirrorSection" component is included in the RaveDemo project file. This example shows how to have a single report produce different envelope formats when sending to International or US addresses. The template for the international customers includes the country line and is centered in the envelope while the US format does not have the country line and is offset to the right of center and lower on the envelope.



Data Mirror Editor

Data Mirrors

(Page2.InterSection)
US (Page2.USSection)

Add Delete

Data Mirror Settings

Field Value ☐ Default

US

Mirrored Section

Page2.USSection

OK Cancel

This example shows several techniques on how to use sections and mirrors. On page 2 there are three sections. Section 3 has the common address lines used by both templates. Section 1 and 2 contain mirrors to Section 3. The International section has the additional country line added in its template. Note, that "remarks" have been added to the page two design and they are not part of any of the sections. The "US Template" and "International Template" are just plain text remark that assist you in the purpose of each section shown.

Once you understand how the master templates were done, go to Page 1, and select the "DataMirrorSection". Notice the "DataView" and "DataField" properties have been set to the field that will be used to select which format template to mirror. To examine that logic, go to the "DataMirrors" property and click on the ellipsis to open the Data Mirror Editor. Select each "DataMirror" item and note the settings in the bottom portion of the dialog box. This example has two "templates", but you can add more to fit more complex reporting needs.


WARNING

Be sure to define one of the settings as a default. If a default is NOT defined and the field value does not match any of the other settings, then the format used will be that set in the mirror property of the DataMirrorSection. You could get unpredictable results or even strange errors with the RAV file since this is usually blank.

2.8.11 DataText Component

The "DataText" component is data aware. It is very similar to the dbEdit component in Delphi.



DataText1: DataText	
Anchor	(Top / Left)
Color	 Black
DataField	
DataView	
DevLocked	False
DisplayOn	doParent
Font	Arial,10
FontJustify	pjLeft
FontMirror	
Left	1
Locked	False
LookupDataView	
LookupDisplay	
LookupField	
LookupInvalid	
Mirror	
Name	DataText1
Rotation	0
Tag	0
Top	0.4
Truncate	True
Visible	True
Width	1

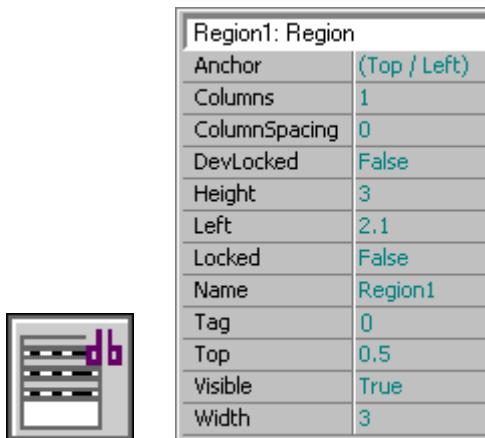
This means you can use it to display a field from a dataset just about anywhere on the page layout. The main difference between "DataText" and "DataBand" is that "DataText" is for single, non-repeating occurrence(s) on the page. For example, this could be used to print the Customer information at the top of each page of an invoice.

See Also

[Anchor](#), [Color](#), [DataField](#), [DataView](#), [DevLocked](#), [DisplayOn](#), [Font](#), [FontJustify](#), [FontMirror](#), [Left](#), [Locked](#), [LookupDataView](#), [LookupDisplay](#), [LookupField](#), [LookupInvalid](#), [Mirror](#), [Name](#), [Report Components](#), [Rotation](#), [Tag](#), [Text](#), [TextJustify](#), [Top](#), [Truncate](#), [Visible](#), [Width](#)

2.8.12 Region Component

So what is a region? A "Region" is a container for bands. In its simplest form the region could be the whole page. This would probably be true for reports that are a list type. Many master-detail reports could be made to fit a single region design. However, do not limit yourself to thinking of regions as the whole page. It may be much simpler to do some difficult formatting by using a multiple region design or a region within a band within a region design.



The properties for a region (shown above) basically deal with its size and location on the page. Creative use of regions will give you more flexibility when trying to design complex reports. You can place multiple regions on a single page. They could be side by side, one above the other or stagger about the page. Do not confuse a region with a section. "Regions" contain bands and only bands. A "[Section](#)" can contain any group of components, including "Regions".

The simple rule is that a band must be in a region. That's it. Note that we did not limit the number of regions on a page. Nor did we limit the number of bands within a region. And if you are still following this, we did not say you couldn't have a region within a band. That's right, you can have a band within a region which is within another band which is in a region and so on. So if you can visualize it, we are sure some combination of the regions and bands will solve many difficult reporting needs.

There are two band type, the Band and DataBand. The DataBand is for iterating needs like the details lines of a Master-Detail report. The Band is for non-iterating needs, like a header, footer or the master portion of a Master-Detail.

See Also:

[Band](#), [BandStyle Editor](#), [DataBand](#), [Report Components](#), [Section](#)

Properties of Region

[Anchor](#), [Columns](#), [ColumnSpacing](#), [DevLocked](#), [Height](#), [Left](#), [Locked](#), [Name](#), [Tag](#), [Top](#), [Visible](#), [Width](#)

2.9 Standard Components

2.9.1 Standard Components

The "Standard" tab controls the visibility of the "Standard" components (shown below) that can then be placed on your page design.



Once the Standard components toolbar is active, you can select and place a [Bitmap](#), [FontMaster](#), [MetaFile](#), [Memo](#), [PageNumInit](#), [Section](#), [Text](#) component on the page layout panel. You set the features you want with the property window for that style component. The names of these properties have been created in an attempt to convey their purpose. Remember that the Reference section in the manual describes each of these properties. However, we will do a quick tour here.

Text1: Text	
Anchor	(Top / Left)
Color	 Black
DevLocked	False
DisplayOn	doParent
Font	Arial,10
FontJustify	pjLeft
FontMirror	
Left	0.7
Locked	False
Mirror	
Name	Text1
Rotation	0
Tag	0
Text	Text1
Top	0.5
Truncate	False
Visible	True
Width	1

Some common Standard component properties are "*Height*" and "*Width*" which reflect the height and width of the drawn box. "*Left*" and "*Top*" are the locations of those position of the box that you drew. "*Height*" and "*Width*" is the height and width of the text box. The "*DisplayOn*" property controls the visibility of the component during preview or print output.

The bitmap box has the following unique properties. The "*Image*" property has an ellipsis that will launch the bitmap editor. You can use this to load a bitmap image from your system. The "*Center*" property determines if the image will be centered in the drawn box. Both the "*MatchHeight*" and "*MatchWidth*" properties control what happens as you stretch the box either horizontally or vertically.

The text box has the following properties. "*AutoSize*" controls the size of the box to fit the contents of the "*Text*" property. The "*Color*" property is the color of the characters printed in the text box. "*Font*" is the name of the font to be used to print the contents of the "*Text*" property. "*FontJustify*" determines the method of justification used on the text contents within the text box. "*Text*" is the contents to be displayed within the text box.

The FontMaster component allows you to define standard fonts for different parts of your reports, like header, body and footer. If you place a FontMaster on a global page, then you can change its name to reflect its purpose like "FontTitles", "FontBody" or "FontTotals" and through the font editor you can set these to any font, font size you want. Then you would use the "FontMirror" property of the Text components to point to appropriate FontMaster definition. The power of this becomes apparent if you need to change the titles to a "new" font. If you have used FontMaster, then you would go to the FontTitles definitions, change it once and it would be changed on every report / page definition that used that FontMaster definition.

See Also:

[Components](#), [Toolbars](#), [Alignment](#), [Bar Code](#), [Bitmap](#), [Colors](#), [Designer](#), [Drawing](#), [Fills](#), [FontMaster](#), [Fonts](#), [Lines](#), [Memo](#), [MetaFile](#), [PageNumInit](#), [Project](#), [Report](#), [Section](#), [Text](#), [Zoom](#)

2.9.2 Bitmap Component



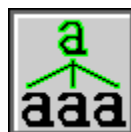
Bitmap1: Bitmap	
Anchor	(Top / Left)
DataField	
DataView	
DevLocked	False
DisplayOn	doParent
FileLink	
Height	0.7
Image	(Bitmap)
Left	0.3
Locked	False
MatchSide	msWidth
Mirror	
Name	Bitmap1
Tag	0
Top	0.3
Visible	True
Width	1.4

Properties:

[Anchor](#), [DataField](#), [DataView](#), [DevLocked](#), [DisplayOn](#), [FileLink](#), [Height](#), [Image](#), [Left](#), [Locked](#), [MatchSide](#), [Mirror](#), [Name](#), [Tag](#), [Top](#), [Visible](#), [Width](#)

2.9.3 FontMaster Component

The FontMaster is a *non-visible* component that allows you to define standard fonts for different parts of your reports, like header, body and footer.



FontMaster1: FontMaster	
DevLocked	False
Font	Arial, 10
Locked	False
Name	FontMaster1
Tag	0
Visible	True

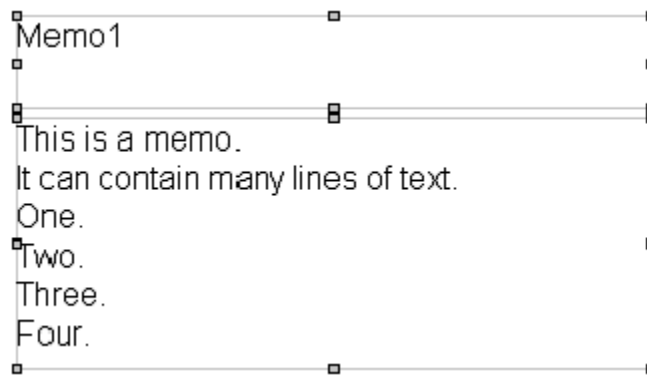
If you place a FontMaster on a global page, then you can change its name to reflect its purpose like "FontTitles", "FontBody" or "FontTotals" and through the font editor you can set these to any font, font size you want. Then you would use the "FontMirror" property of the Text components to point to appropriate FontMaster definition. The power of this becomes apparent if you need to change the titles to a "new" font. If you have used FontMaster, then you would go to the FontTitles definitions, change it once and it would be changed on every report / page definition that used that FontMaster definition.

See Also:

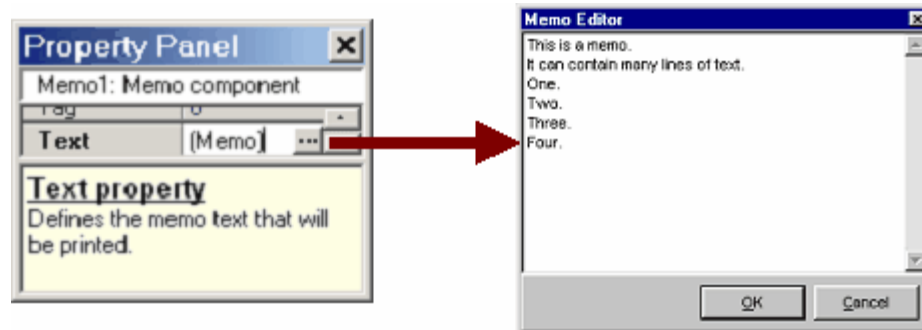
[DevLocked](#), [Font](#), [Locked](#), [Name](#), [Tag](#), [Visible](#)

2.9.4 Memo Component

Memo components are similar to text components. However the most noticeable difference is that Memos can contain multiple lines of text.



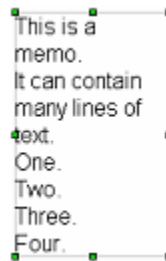
Again, similar to the text component, the border delimiting the Memo can only be seen when the component is selected. When setting the Font of the Memo, all the lines of text in the Memo will have the same font. It is not possible to set different fonts for certain parts of the text.



To change the text in the memo, go to the Property Panel and use the Text property. The text property will show "(Memo)" and three dots. Click on the dots. This will open up the Memo Editor. Text can be entered into the Memo Editor.



One of the main problems with using multi-line text components is that the text might overlap in height on to other components on the report. To prevent this, setting the ExpandParent property to True will expand the parent of the memo component to properly accommodate the memo component.



Once the text is entered, the memo box can be resized. The text within the memo box will be repositioned accordingly. If there appears to be text missing from the memo box, after it has been entered into the Memo Editor, resize the box to allow all text to be seen.

Memos can be used on forms for areas of explanation, and for titles or comments that are longer than one line.

Properties:

[Anchor](#), [Color](#), [DevLocked](#), [DisplayOn](#), [ExpandParent](#), [Font](#), [FontJustify](#), [FontMirror](#), [Height](#), [Left](#), [Locked](#), [MailMergeItems](#), [Mirror](#), [Name](#), [Tag](#), [Text](#), [Top](#), [Truncate](#), [Visible](#), [Width](#)

See Also:

[Standard Components](#)

2.9.5 MetaFile Component



MetaFile1: MetaFile	
Anchor	(Top / Left)
DataField	
DataView	
DevLocked	False
DisplayOn	doParent
FileLink	
Height	3
Image	
Left	1.2
Locked	False
MatchSide	msWidth
Mirror	
Name	MetaFile1
Tag	0
Top	0.4
Visible	True
Width	3

Properties of MetaFile

[Anchor](#), [DataField](#), [DataView](#), [DevLocked](#), [DisplayOn](#), [FileLink](#), [Height](#), [Image](#), [Left](#), [Locked](#), [MatchSide](#), [Mirror](#), [Name](#), [Tag](#), [Top](#), [Visible](#), [Width](#)

2.9.6 PageNumInit Component

The "PageNumInit" component allows you to restart page numbering within the report.



PageNumInit1: PageNumInit	
DevLocked	False
InitDataField	
InitDataView	
InitValue	1
Locked	False
Name	PageNumInit1
Tag	0
Visible	True

This is a non-visual control that sets the relative page number to a given value (either InitDataField or InitValue) when it is printed. To output the relative page number, you need to use the RelativePage report variable instead of CurrentPage.

Properties of PageNumInit

[DevLocked](#), [InitDataField](#), [InitDataView](#), [InitValue](#), [Locked](#), [Name](#), [Tag](#), [Visible](#)

2.9.7 Section Component

A "Section" is a container area that holds a group of components. "[Regions](#)" are a container for Bands and DataBands. Remember that Bands must be placed in a "Region".

Section - Same Page

For example, say you had a logo and address information in a box at the top of a page and wanted to repeat this logo and box on the bottom on the page. The bottom part could be a tear off portion that was to be returned, like a meeting reservation. So for this case, you would first place a section on the top part of the page. Make sure that the section is large enough to hold the logo and address information. Now drop a bitmap component (for the logo) and some text components in that section. To make this realistic, change the properties of the text components to reflect a typical address that you would use. Now come the fun part, duplicating that for the bottom portion. It is only two steps, drop another section component at the desired location on the bottom of the page and set it's mirror property to point at the first section. That's it. Now you should have two sections that have identical contents. To prove it to yourself, go to the first section and make a change and note that the second section changes with the first. You could change the font size or color property. Or drag the bitmap component around a bit and watch the bitmap in section 2 follow the movements you are making in section 1.

Regions vs Sections

We need to emphasis that you should NOT mirror items that have a page state condition.

The short explanation is that you can mirror components or a section definition from another page to your current page. However, we do NOT recommend or encourage that you mirror any sections, which contain region(s), band(s) or memo(s) as they each have page state condition(s). Why? Each region / band / memo must know its page state. For example, a region contains bands and these bands can be in a variety of conditions on each page break. The region maintains its status and knows where it is at with regard to wide variety of items including grouping, master-detail relations, totaling as the report progresses through the table.

You can mirror a section (from another Page or Global Page) but you must not mirror a section on that same page. We are aware that some users have been successful in mirroring regions and bands. However, we do NOT recommend that practice as you are gambling that you will not have a page state conflict or problem. If you do this and have a problem, we will request that you remove the mirror of a section, which contains a region and bands.

That's the secret - mirror sections that contain components, but do NOT mirror sections with regions, bands or memos.



Section1: Section	
Anchor	(Top / Left)
DevLocked	False
DisplayOn	doParent
Height	3
Left	0.9
Locked	False
Mirror	
Name	Section1
Tag	0
Top	0.5
Visible	True
WasteFit	False
Width	3

[Section OnPrint](#)

The "section" component for users that have upgraded their Rave Reports (Rave 7 BEX or later)

has an OnPrint event that provides a "doorway" to many of the methods and properties normally available only to code base users. This allows users of the upgraded versions of Rave Reports to write regular code base statements in their visual report projects. When using the section OnPrint event the borders of the section component will be treated like page borders. In general, that means you should set the section borders to match the page or close to it. So that means set the "Left" and "Top" properties to 0 and the "Height" and "Width" properties to match your page size. ([see OnPrint example](#))

Properties of Section

[Anchor](#), [DevLocked](#), [DisplayOn](#), [Height](#), [Left](#), [Locked](#), [Mirror](#), [Name](#), [Tag](#), [Top](#), [Visible](#), [WasteFit](#), [Width](#)

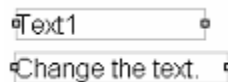
[Regions](#)

2.9.8 Text Component

The "Text" component is useful for displaying a single line of text on the report.



It acts basically like a label that can contain simple text (not data). When placed on the report, the Text box is originally surrounded with a box that indicates its boundaries. This box can only be seen while the text component is selected (represented by the pips).



The component can be used for labels, figure or graphical titles, floating text, form titles and in general just about anything that can be denoted by a single line of text. When the component is selected, the length of the text can be adjusted by resizing it using the pips. Note however that the height of the text is self-adjusting and does not require further intervention by the user.

Like any other text-based component, there is a font property that can be used to change the font type, size and style. The color can be set using the Color property. The actual text of the component is denoted by the Text property. There is an additional property called Rotation, which can be used to rotate the text by a specified number of degrees. The rotation effect can only be seen at runtime.

See Also:

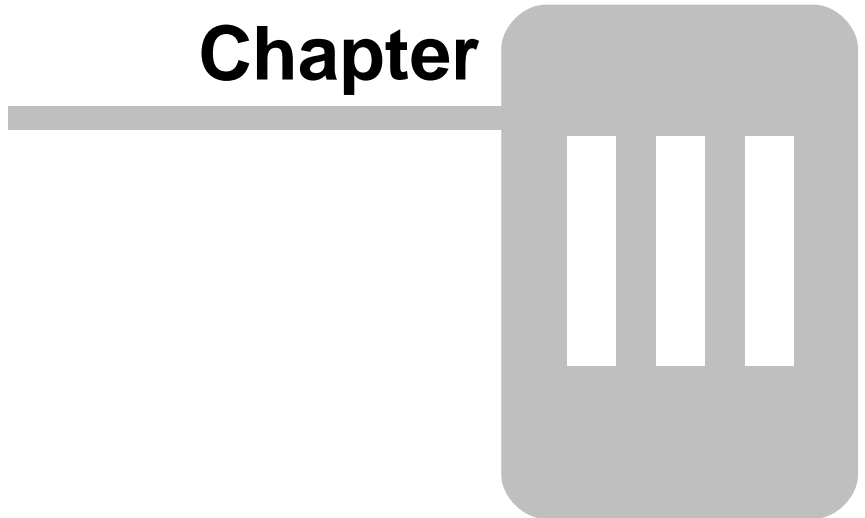
[Standard Components](#)

Properties:

[Anchor](#), [Color](#), [DevLocked](#), [DisplayOn](#), [Font](#), [FontJustify](#), [FontMirror](#), [Left](#), [Locked](#), [Mirror](#), [Name](#), [Rotation](#), [Tag](#), [Text](#), [Top](#), [Truncate](#), [Visible](#), [Width](#)

Adaptable Reports

Chapter



3 Adaptable Reports

One of the biggest problems that designers face when creating reports is adaptability. What is adaptability exactly? First, when a report is created, the process normally takes place on a specific computer, which has one or more printers connected to it. When that report is executed in the field, however, factors such as paper size, orientation and waste areas all come into play. Secondly, reports should allow for a certain amount of flexibility in the overall design and appearance of the layout of the report.

Rave helps overcome these problems by giving the designer access to particular properties of the report. In this section, these properties will be explained in detail and will allow the resulting reports to have a standard look over various platforms. This way, the designer can focus more on creating reports than worrying about deployment difficulties. The reports will also be able to adjust layouts based on external parameters passed in during execution.

3.1 Anchors

Most visual components have a common property called Anchors. This powerful feature defines how the object will move when its Parent is resized. There are two Anchor values that are set, one for the horizontal and one for the vertical. The default Anchor is set to the top left corner of the Parent control. What this means is that when the Parent (e.g. a Section component) is resized, the component will stay the same distance from the left and top corner of the Parent. Two other Anchor values, Bottom and Right, are very similar in function (they will Anchor to the bottom right corner of the Parent).

The Anchor setting of Center will Anchor the Child component to the Parent's center. Stretch will actually resize the Child component so that it's sides stay the same distance from the Parent. Stretch is most often used when you want a Child component to always match the width or height of the Parent. Two other special Anchor settings, Resize and Spread are useful for groups of components. Resize will proportionally resize the components and the spaces between them as the Parent is changed. Spread will proportionally resize the spaces only (the components will stay the same size) as the Parent changes. Drop down a Section component and place a few rectangles or Text components inside and change the values of those components to see how the Anchor property can affect things.

Anchors can be used to create adaptable Reports when combined with other Reporting features. Imagine that a Report needs to be defined that can be printed in either landscape or portrait orientation or that the Report may be run on different size papers. Setting the Anchors properly will allow one Report to adjust to these changing conditions. A typical table listing Report will be composed of a Region component; it's Band components and the Text and DataText components inside each band. If the Region component is set to Anchor Stretch on both vertical and horizontal and the Text and DataText components are all set to a horizontal Anchor of Resize, the Report will adjust to any of these changes.

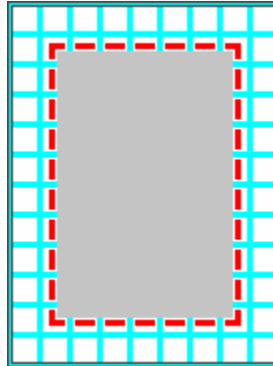
Other changes, such as the printer's individual waste area, can also be solved with Anchors and will be discussed in the next section.

See Also:

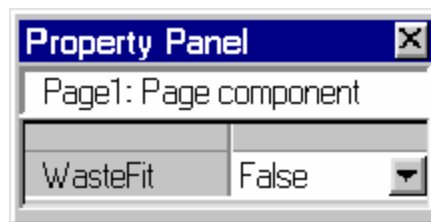
[Anchor Editor](#), [Anchor property](#), [Waste Fit more](#), [Waste Fit property](#)

3.2 Waste Fit

Most printers have a Region on the paper that is called the waste area. What this represents is the portion of the Page where no printing can be done because of restrictions of the actual printing hardware (usually because the printer uses that region of the page to "grab" a hold of the paper when it is feeding it through the rollers). These values may range from 0 (common for dot-matrix printers) to 1.5 inches. Ink jet printers typically have large bottom margins (and sometimes a 0 top margin), while most ink jet and laser printers have about a 0.25 inch left and right waste area.



The red-dotted lines that run on the border of the Page represent where the waste-area of the currently selected printer (or specific values if the preferences are set that way). Anything in between the edge of the Page and the red line is the actual waste-area of that printer. You should avoid placing any components within that area. One problem when designing reports is avoiding the actual waste area of the destination printer. If the Report is designed with large margins then you will be probably be inside the waste area of most printers. However, that would mean a lot of unnecessary blank space will be on each Page. The "WasteFit" and "Anchor" properties are Rave's answer to designing a report that changes its margins based upon the printers waste area.



The Page has a property called WasteFit, which can be either true or false. By default the value is False. Setting the property to true will make the Page adjust the components it contains so that they fit within the waste-area of the destination printer. The components contained will be resized so that they all fit in and adjust accordingly.

However, it is not sufficient with setting the WasteFit property to True for all this to happen automatically. Child components should have their anchor property set accordingly so that they too can adjust accordingly to the changes of the Page. When used properly, this property is a very powerful feature that allows the reports developed to automatically adapt to different destination printers.

See Also:

[Anchor Editor](#), [Anchor property](#), [Waste Fit property](#)

Batch and Chain Reporting

Chapter



IV

4 Batch and Chain Reporting

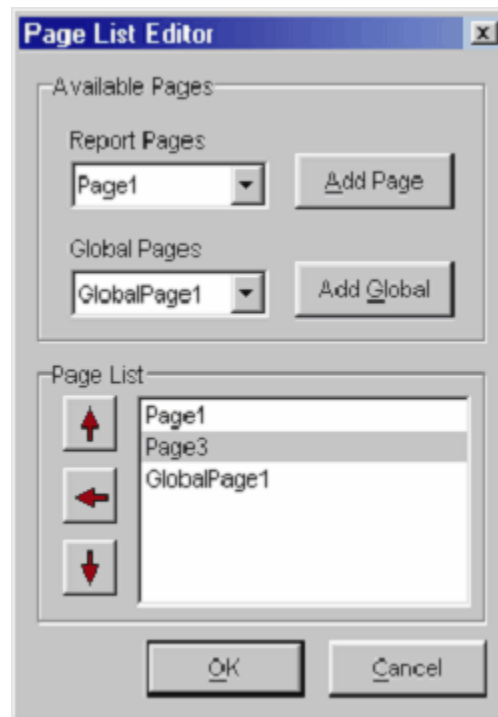
Once the Report Design is complete and working, the day-to-day routines get settled. At this point, it would not be unusual to find out there often is a need to print several Reports in some kind of sequence. This could be a group of Reports that are always generated at the end of each month, quarter, etc. Another example might be a series of executive summaries that are needed on a demand basis by upper level management. Of course, it is possible to run each required Report individually for these repetitive cases. Then when they are all done, group them together and give them to the requesting office. This often means that there is a checklist that details what repetitive Reports are required, when and for whom.

This problem of producing a repetitive sequence of Reports is solved with Rave by linking the Reports to each other. Rave has the ability to link the Report Page definitions in a wide variety of ways by setting different combinations of three properties; `Page.GotoMode`, `Page.GotoPage` and `Report.PageList`. It is important to note that the setting of the `GotoMode` property determines the behavior of the `GotoPage` property. The `PageList` property is at Report Node level and is designed to provide a way to initiate several different `GotoPage` property chains in a defined sequence.

Although we will give examples of how we envision achieving various Batch/Chain sequences of Pages, it will not be possible to include all of the possibilities. The secret is that one should understand and be flexible on how to “mix and match” the various parts of Rave. In particular, pay special attention to the Global Page catalog and the `GotoMode`, `GotoPage`, `Mirror` and `PageList` properties. Different combinations of these provide a wide range of output options. There is a lot of power here, so the best advice is to start with simple Reports. Then add some of these powerful extras and as they make sense.

4.1 Batch Pages

Probably the most common linking of Pages would be a batch processing sequence that defines a list of independent Page definitions. The first Page in the list runs to completion, then it calls the second, which runs to completion, and so on until the last Page of the defined sequence has been completed. The main thing to remember with batch Pages is that each Page definition is independent of the others and runs to completion before the other Page starts. Remember that the requirement to do batch processing is to simplify the administrative tasks when running a group of Page definitions on a recurring basis.



Define the sequence of Pages to print through the PageList property, which is available at the Report node level. When this property is selected (click on the ellipsis button), it will open a dialog window that will build the list from the Page definitions within the selected Report node. The advantage of this method is that the PageList is independent of the individual Page definitions. This means that the PageList would run the whole sequence of Pages (Reports), but individual Reports can still be selected individually and run by itself without invoking the batch link. Remember, Report Pages are not visible to other Reports, so if flexibility of calling separate Page definitions is needed, define the reusable Pages as global Pages instead of Report Pages.

WARNING

Another method to do Batch Pages is to set the first Page definition GotoPage property to the second Page definition name. Now, when the first Page is completed, it will automatically start the second Page definition. The problem with this technique is that anytime the first Page is run it will ALWAYS start the second Page as they are linked together at the Page definition level.

4.2 Calling Pages

(Page Node - GotoMode property - gmCallEach setting)

Another type of linking would be Reports where there must be a particular flow of Pages. The easiest example of this would be a Report where each record ALWAYS produces three Pages of output. This could be a "form Report" where the first page has a patient's demographic information, the second page has medical information and the last page has insurance information. So, when going through the database, each patient's record would produce 3 Pages of output. See the lesson for more step-by-step instructions on how Call Pages.

4.2.1 Calling Pages Lesson

One example of "Calling Pages", would be a Report where each record ALWAYS produces three Pages of output. This could be a "form Report" where Page 1 has a patient's demographic information, Page 2 has medical information and Page 3 has insurance information. So, when going through the database, each patient's record would produce 3 Pages of output.

The way to accomplish this would be:

- Complete all normal definitions for the Page 1.
- Complete all normal definitions for the Page 2.
- Complete all normal definitions for the Page 3.
- Set the GotoPage property of the first Page to point to the second Page.
- Set the GotoMode property of the first Page to gmCallEach setting.
- Set the GotoPage property of the second Page to point to the third Page.
- Set the GotoMode property of the second Page to gmGotoDone setting.
- Insure that the GotoPage property of the last Page definition is blank.

The gmCallEach setting of the Page 1 GotoMode property will activate the GotoPage property after the first physical Page has printed. So, at the end of physical Page 1, Page 2 definition will be started. Because the Page 2 definition has a gmGotoDone setting, it will go to the Page 3 definition when it has finished the printing the second Page. The Page 3 definition will print that Page's definition and then return control to the Page 1 definition because it's GotoPage property is blank. You can join any number of Pages together in this manner. The key is that a gmCallEach setting will save off the calling Page so that whenever a blank GotoPage property in the Page chain is encountered, it will continue with that calling Page.

4.3 Chain Pages

(Page Node-GotoMode property-gmGotoDone setting)

A chain of Pages is similar to a Batch. The Batch technique discussed earlier used the PageList property and cautioned you about using the GotoPage property. However, if you plan ahead, then the use of the GotoPage property with the Pages from the Global Page Catalog is very powerful. The exercise at the end of this section does a Multi-Page definition that includes an invoice, file copy, shipping document and packing slip.

4.3.1 Chain Pages Lesson

For this example, we are going to do a Multi-Page definition that includes an invoice, file copy, shipping document and packing slip.

- Create a section on a Global Page that will be mirrored later. Complete all of your normal definitions for the Invoice Page within this section.
- Create a section on a Global Page that will be mirrored later. Complete all of your normal definitions for the File Copy Page within this section.
- Create a section on a Global Page that will be mirrored later. Complete all of your normal definitions for the Shipping Page within this section.
- Create a section on a Global Page that will be mirrored later. Complete all of your normal definitions for the Packing Slip Page within this section.
- Create a "New Report" that will hold the definitions for these Pages.
- Create a new Page definition, "Invoice" drop a section on this Page definition and set the Left and Top properties mirror the section to the "Global" Invoice Page section drop a text component on the bottom of the Page "INVOICE COPY"
- Create a new Page definition, "FileCopy1" drop a section on this Page definition and set the Left and Top properties mirror the section to the "Global" FileCopy Page section drop a text component on the bottom of the Page "FILE COPY 1"
- Create a new Page definition, "FileCopy2" drop a section on this Page definition and set the Left and Top properties mirror the section to the "Global" FileCopy Page section drop a text component on the bottom of the Page "FILE COPY 2"
- Repeat those steps, for the "Shipping Document" and "Packing Slip"
- Now go back to the "Invoice" Page definition (NOT the global definition) Set the PageMode setting to gmGotoDone Set the PageGoto to point to the "File Copy 1" Page definition.
- Repeat these steps for each of the copies, remember to leave the last one blank

The mirroring of a Global Page offers a lot of "reuse" of a Master Design. This example showed that the "FileCopy" could be mirrored twice and "label" each Page definition differently.

4.4 Different First Page format

(Page Node-GotoMode property-gmGotoNotDone setting)

Another type of linking would be Reports where the first Page format is different than the remaining Pages. This could be a Report that has a title layout maybe even with a company logo on Page 1, but the remaining Pages are all the same design layout. What you need to do for this is to create the Page 1 definition (first Page) that points to Page 2. Then create the Page 2 definition (all remaining Pages) that points to blank. The Exercise at the end of the section goes through the detail steps to complete this process.

4.4.1 Different First Page Lesson

The way to accomplish this would be:

- Complete all normal definitions for Page 1.
- Create a "New Report Page" for the second Page definition.
- Set the GotoPage property of the first Page to point to the second Page.
- Set the GotoMode property of the first Page to gmGotoNotDone setting.
- Complete all normal definitions for Page 2.
- Insure that the GotoPage property of the second Page is blank.

The gmGotoNotDone setting of the GotoMode property will activate the GotoPage property after the first physical Page has printed but only when the current Page definition has not finished (for example, EOF - End Of File). So, at the end of physical Page 1, Page 2 definition will be started. Because the Page 2 definition does NOT have anything in the GotoPage property, the Page 2 definition will remain in effect for all remaining physical Pages until the Report is done.

4.5 Different Odd/Even Page format

(Page Node-GotoMode property-gmGotoNotDone setting)

Another type of linking would be Reports with a format based on an odd/even Page definition. This could be Reports that are going to be printed on both sides of the paper (duplex style) and have holes punched in one side, so that final Report could be put in a binder. This would mean that the inside margin (say 1 inch) would be larger than the outside margin (½ inch). To create a loop where the Page 1 definition (odd Page) points to Page 2 and will call it if the Report is not done when it gets to the end of the Page AND the Page 2 definition (even Page) points to Page 1 and will call it if the Report is not done at the end of that Page. This loop will need to continue until Report is completely printed. See the Exercise at the end of this section to get detailed steps to complete this task.

4.5.1 Different Odd/Even Page Lesson

The way to accomplish this would be:

- Make a Section on the first Page definition that is set for the "Odd" Page margins. For example, set the properties Left = 1.0, Width = 7.0, Top = 0.5 and Height = 10.0.
- Complete all of your normal definitions for Page 1 within this section.
- Create a "New Report Page" for the second Page definition
- Drop a Section on the Page 2 and set the top and left margins for the even Page settings. For example, set the Left property = 0.5 and Top property = 0.5.
- Set the Mirror property of Page 2 Section to point to the Section on Page 1.
- Set the GotoPage property of the second Page to point to the first Page.
- Set the GotoPage property of the first Page to point to the second Page.
- Set the GotoMode property of BOTH Pages to gmGotoNotDone setting.

The gmGotoNotDone setting of the GotoMode property will activate the GotoPage property after each physical Page has printed but only when the current Page definition has not finished (for example, EOF - End Of File). So, at the end of physical Page 1, Page 2 definition will be called. At the end of physical Page 2, Page 1 definition will be called. This loop will continue until one of the Pages is completed. If control would need to be passed to another Page at this point, use the Report components PageList property to select the next Page.

Property Descriptions

Chapter



V

5 Property Descriptions

The following is an alphabetical listing of all properties that make up the RAVE system. Properties are defined by their data type, category, components they are members of, a short description and any relationships they have with other properties. The default values are added where applicable.

5.1 AllowSplit property

Default:

False

Component/Class:

[Band](#), [DataBand](#)

Description:

When set to True, the band to be split across pages. If set to False, the entire band will move to the next page if there is not enough room on the current page.

Note:

If the band is too large to fit on a page it will be split no matter the setting of this property.

5.2 AlwaysGenerate property

Default:

False

Component/Class:

[Report](#)

Description:

When set to true, generation of the complete report is forced before sending it to the output device. This is important for insuring that the report variables like TotalPages are known before the first page is printed.

See Also:

[Variables](#)

5.3 Anchor property

Default:

Top / Left

Component/Class:

all visible components

Description:

Determines the method that will be used to align a component both vertically / horizontally within its parent's area.

See Also:

[Anchor Editor](#), [Bottom](#), [ExpandParent](#), [Left](#), [Right](#), [Top](#)

5.4 AutoSize property

Default:

True

Component/Class:

[all bar code components](#)

Description:

When set to true, bar code dimensions will automatically resize to the necessary size to display all encoded text.

See Also:

[Text](#)

5.5 BandStyle property

Default:

'none'

Component/Class:

[Band](#), [DataBand](#)

Description:

One of the powerful features of RAVE is the ability to set a band's behavior with the BandStyle property. Click on the ellipse in the BandStyle area, it will open an editor dialog window that allows you to set the behavior needed for the selected band. For information about "headers and footers" see FAQ (Frequently Asked Questions) [Design - Footers](#) and [Design - Header on ALL pages](#) .

See Also:

[ControllerBand](#), [BandStyle Editor](#)

5.6 BarCodeJustify property

Default:

pjLeft

Component/Class:

All bar code components

Description:

Determines where the bar code is printed relative to its bounding box.

pjLeft	Print the bar code left justified
pjCenter	Print the bar code centered
pjRight	Print the bar code right justified

See Also:

[Center](#), [Left](#), [Right](#)

5.7 BarCodeRotation property

Default:

Rot0

Component/Class:

all bar code components

Description:

Allows the bar code to be rotated to 4 different orientations. The pivot point for rotation is the top left corner of the bar code.

Rot0	no rotation
Rot90	rotate 90 degrees relative to page
Rot180	rotate 180 degrees relative to page
Rot270	rotate 270 degrees relative to page

See Also:

[Left](#), [Top](#), [Rotation](#)

5.8 BarHeight property

Default:

0.5 (PostNet component 0.125)

Component/Class:

all bar code components

Description:

Sets the height for the tallest bar.

See Also:

[BarWidth](#)

5.9 BarTop property

Default:

0

Component/Class:

all bar code components

Description:

Sets the location of the top of the bar code. The location of the readable text is controlled by PrintReadable and PrintTop properties.

See Also:

[PrintReadable](#), [PrintTop](#), [Top](#)

5.10 BarWidth property

Default:

0.01 (PostNet component 0.020)

Component/Class:

all bar code components

Description:

This property sets the width of the narrowest bar.

See Also:

[BarHeight](#), [Width](#)

5.11 Bin property

Default:

'Default' (windows system settings)

Component/Class:

[Page](#)

Description:

Specifies the paper tray that you want used for the document.

See Also:

[Collate](#), [Duplex](#), [Orientation](#), [PaperSize](#), [Printer](#), [Resolution](#)

5.12 BinCustom property

Default:

'' (empty)

Component/Class:

[Page](#)

Description:

If this property is not blank, the page will attempt to print on the first bin whose name contains the text defined by BinCustom. If BinCustom is blank, the bin selection is determined by the Bin property.

See Also:

[Bin](#)

5.13 BorderColor property

Default:

Black

Component/Class:

[Circle](#), [Ellipse](#), [Rectangle](#), [Square](#)

Description:

Sets the color to be used for the border of the graphic.

See Also:

[FillColor](#)

5.14 BorderStyle property

Default:

psSolid

Component/Class:

[Circle](#), [Ellipse](#), [Rectangle](#), [Square](#)

Description:

Sets style of border that appears as a frame around shapes.

psClear	No border is drawn
psDash	Creates a dashed border
psDashDot	Creates an alternating dash and dot border
psDashDotDot	Creates an alternating dash and double dot border pattern
psDot	Creates a dotted border
psInsideFrame	Creates a border that is inside the frame of closed shapes
psSolid	Creates a solid border

Note:

Only psSolid can have a pen width greater than 1.

See Also:

[BorderColor](#), [BorderWidth](#)

5.15 BorderWidth property

Default:

1

Component/Class:

[Circle](#), [Ellipse](#), [Rectangle](#), [Square](#)

Description:

Determines the width of the line used to draw the border around the graphic shape.

See Also:

[BorderStyle](#), [BorderWidthType](#)

5.16 BorderWidthType property

Default:

wtPixels

Component/Class:

[Circle](#), [Ellipse](#), [Rectangle](#), [Square](#)

Description:

Determines whether the *BorderWidth* property of the graphic shape is measure in pixels or points. If you want the thinnest line possible, use pixels and a *BorderWidth* of 1 (the defaults). For consistent thickness across all devices use points for all other *BorderWidth* values

<i>wtPixels</i>	<i>BorderWidth</i> is measured in pixels
<i>wtPoints</i>	<i>BorderWidth</i> is measured in points (1 point = 1/72nd of an inch)

See Also:[BorderWidth](#)

5.17 Bottom property

Default:

'None' <bottom edge of selected component>

Component/Class:

all bar code components

Description:

Sets the position for the bottom of the bar code. The value for this property includes the readable text if it is printed.

See Also:

[Anchor Editor](#), [Anchor](#), [ExpandParent](#), [Left](#), [Right](#), [PrintReadable](#), [PrintTop](#), [Top](#)

5.18 CalcType property

Default:

ctSum

Component/Class:

[CalcText](#), [CalcTotal](#)

Description:

Sets the type of calculation to be performed by the [CalcText](#) over the [DataField](#) property contents.

ctAverage	average value for a data field
ctCount	count the number of occurrences
ctMax	maximum value for a data field
ctMin	minimum value for a data field
ctSum	sum of a field

See Also:

[CountBlanks](#), [RunningTotal](#)

5.19 CalcVar property

Default:

'' (empty)

Component/Class:

[CalcTotal](#)

Description:

Defines the calculation component that will be used in the CalcType operation. If this property is defined, DataField will be ignored.

See Also:

[CalcType](#)

5.20 Categories property

Default:

'' (empty)

Component/Class:

[ProjectManager](#)

Description:

Defines the available categories that a report can belong to. For example, you could define categories called 'Accounting', 'General', 'Status' and 'System'.

See Also:

[Cursor](#), [DevLocked](#), [Parameters](#)

5.21 Category property

Default:

'' (empty)

Component/Class:

[Report](#)

Description:

Sets the category that a report can belong to. For example, you could define categories called 'Accounting', 'General', 'Status' and 'System'.

See Also:

[Cursor](#), [DevLocked](#), [Parameters](#)

5.22 Center property

Default:

'None' <center position of currently selected component>

Component/Class:

all bar code components

Description:

Sets or returns the position for the horizontal center of the bar code. When a value is assigned to Center the BarCodeJustify property is set to pjCenter as well.

See Also:

[Anchor](#), [BarCodeJustify](#), [Left](#), [Right](#)

5.23 CodePage property

Default:

cpCodeA

Component/Class:

[128 bar code](#) component

Description:

Specifies whether Code A, Code B or Code C is being used.

cpCodeA	sets 128 output to Code A
cpCodeB	sets 128 output to Code B
cpCodeC	sets 128 output to Code C

See Also:

[Bar Codes](#)

5.24 Collate property

Default:

False

Component/Class:

[Report](#)

Description:

Sets the collation style of the print job for the report.

See Also:

[Bin](#), [Duplex](#), [Orientation](#), [PaperSize](#), [Printer](#), [Resolution](#)

5.25 Color property

Default:

Black

Component/Class:

line and text components

Description:

Used to set the color of the component's output.

See Also:

[FillColor](#), [BorderColor](#)

5.26 Columns property

Default:

1

Component/Class:

[DataBand](#), [Region](#)

Description:

Determines the number of columns to be used, this property is component dependent.

DataBand

Defines how many columns the band will print. The width of each column is divided evenly across the width of the band and printing progresses from the left to right column before progressing to the next row.

Region

Sets the number of columns to be use in the region. When the region is printing, it will print it's bands contents in all columns from left to right in a snaking fashion.

See Also:

[ColumnSpacing](#)

5.27 ColumnSpacing property**Default:**

0

Component/Class:

[DataBand](#), [Region](#)

Description:

Defines the width of a buffer between each column. The spacing will not be applied before the first or after the last column.

See Also:

[Columns](#)

5.28 ConnectionName property**Default:**

'' (empty)

Component/Class:

[DataView](#)

Description:

Sets the name of the data connection from which data will be retrieved for the report.

See Also:

[Description](#), [FullName](#), [Name](#)

5.29 ContainsRTF property**Default:**

False

Component/Class:

[DataMemo](#)

Description:

Indicates whether the memo contains RTF or not. If ContainsRTF is true, then the output will be formatted according to the RTF codes contained in the memo.

5.30 Controller property**Default:**

'' (empty)

Component/Class:

[CalcText](#), [CalcTotal](#)

Description:

Defines the controller component that will execute the calculation in the CalcType property when the controller is printed. Normally, the controller is tied to a particular DataView and will signal when new data is available.

See Also:

[CalcType](#), [ControllerBand](#), [DataField](#), [DataView](#), [Initializer](#)

5.31 ControllerBand property

Default:

'' (empty)

Component/Class:

[Band](#), [DataBand](#)

Description:

Sets the controlling band for the current Band or DataBand. For example, a header or footer band style would assign the data of the DataBand that it is acting as a header or footer for. Detail bands in a master-detail style report would set their ControllerBand to the master DataBand.

See Also:

[BandStyle](#), [Controller](#)

5.32 Copies property

Default:

1

Component/Class:

[Report](#)

Description:

Sets the number of copies that the report will generate when it is printed.

See Also:

[Bin](#), [Collate](#), [Duplex](#), [Orientation](#), [PaperSize](#), [Printer](#), [Resolution](#)

5.33 CountBlanks property

Default:

True

Component/Class:

[CalcText](#), [CalcTotal](#)

Description:

This property sets whether the CalcText component will count blank field values for the ctAverage and ctCount calculation types.

See Also:

[CalcType](#), [CountNulls](#), [CountValue](#)

5.34 CountNulls property

Default:

False

Component/Class:

[CalcText](#), [CalcTotal](#)

Description:

Determines whether or not null (empty) values are included in the ctAverage and ctCount calculation types.

See Also:

[CalcType](#), [CountBlanks](#), [CountValue](#)

5.35 CountValue property

Default:

False

Component/Class:

[CalcText](#), [CalcTotal](#)

Description:

Defines the field value that will cause a count to occur if there is a match for CalcType = ctCount.

See Also:

[CalcType](#), [CountBlanks](#), [CountNulls](#)

5.36 Cursor property

Default:

crDefault

Component/Class:

all drawing

Description:

Determines the type of cursor used within that item's selection zone

- crAppStart
- crArrow
- crCross
- crDefault
- crDrag
- crHandPoint

crHelp
crHourGlass

See Also:

[Categories](#), [DevLocked](#), [Parameters](#)

5.37 DataField property

Default:

'' (empty)

Component/Class:

most data aware components

Description:

Defines the type of data that the component will display or process. The contents can be made up of data field names, [report variables](#), [project parameters](#) or even string constants surrounded by quotes. The [Data Text editor](#) which can be used to easily build a [DataField](#) value.

See Also:

[Data Text Editor](#), [DataView](#), [Variables](#)

5.38 DataView property

Default:

'' (empty)

Component/Class:

most data aware components

Description:

Defines the DataView that will be used if the DataField property is initialized to any field names.

See Also:

[DataField](#)

5.39 Description property

Default:

'' (empty)

Component/Class:

[Page](#), [Report](#), [DataView](#), [DataField](#)

Description:

Defines a multi-line memo that describes the component. This could be used in conjunction with the *FullName* property to document how the component is to be used.

See Also:

[FullName](#), [Name](#)

5.40 DesignerHide property

Default:

False

Component/Class:

[Band](#) and [DataBand](#)

Description:

This property controls the visibility of the band contents in the designer. You can reduce the "clutter" of other bands by setting their *DesignerHide* property to true. Then only the one(s) set to False will show. This might be needed if you have a large number of bands or a band that are occupying a large space.

Note:

This has NO effect on what will be printed, only on what is shown on the designer page.

See Also:

[DevLocked](#)

5.41 DestParam property

Default:

'' (empty)

Component/Class:

[CalcOp](#), [CalcTotal](#)

Description:

Defines the project parameter that the calculation will be written to when it is printed. This value can be blank if the calculation is only going to be used as an intermediate result for other calculation components.

See Also:

[DataField](#), [Operator](#)

5.42 DestPIVar property

Default:

'' (empty)

Component/Class:

[Page](#), [ProjectManager](#), [Report](#)

Description:

Initializes the value of a PIVar (Post Initialize Variable). Any PIVars of the same name that were previously printed will show this value. PIVars are normally printed using the Data Text Editor of the DataText component. The main difference between PIVars and parameters is that PIVars will use the value that is set after (in print order) the PIVar while parameters print the value that was set before. A common use for PIVars is to print a total in a header band that would be initialized later in the footer band. This works even across multiple pages.

Note:

Report . AlwaysGenerate must be true if you are using PIVars in your report.

See Also:

[AlwaysGenerate](#), [DataTextEditor](#), [PIVars](#), [DataText](#)

5.43 DetailKey property

Default:

'' (empty)

Component/Class:

[DataBand](#), [DataCycle](#)

Description:

Sets the detail key field(s) that will be matched to the master key field(s) in a master-detail report. Multiple fields should be separated with +'s, for example, "Customer Number" + "InvoiceNo".

See Also:

[MasterKey](#)

5.44 DevLocked property

Default:

False

Component/Class:

all components

Description:

Locks out the ability of anyone making changes to the property. May be used by the developer to prevent accidental changes being made to a stable part of a complex report structure.

See Also:

[Categories](#), [Cursor](#), [DesignerHide](#), [Parameters](#)

5.45 DisplayFormat property

Default:

'' (empty)

Component/Class:

[CalcOp](#), [CalcText](#), [CalcTotal](#), [All Numeric Field Types](#)

Description:

DisplayFormat formats the value given using the given format string. The various format specifiers are listed in the manual.

See Also:

[Format Codes](#)

5.46 DisplayOn property

Default:

doParent

Component/Class:

most components

Description:

This property controls whether this component will use during the preview display, printer output or both.

doAll	send this item to both preview & printer
doParent	use the parent setting for DisplayOn
doPreviewOnly	item will only be displayed on the preview
doPrinterOnly	component will only be displayed on the printer

See Also:

[Cursor](#)

5.47 DisplayType property

Default:

dtNumericFormat

Component/Class:

[CalcOp](#), [CalcText](#), [CalcTotal](#)

Description:

This defines whether the contents represent numeric or date/time information. The DisplayFormat property will be processed as either numeric or date/time format codes depending upon this setting.

dtDateTimeFormat	sets to Date / Time format
dtNumericFormat	sets to numeric format

5.48 Duplex property

Default:

pdDefault

Component/Class:

[Report](#)

Description:

Sets the duplex mode for the current printer.

<i>pdDefault</i>	Use current mode (Duplex mode NOT changed)
<i>pdSimplex</i>	Simplex mode (Duplex mode NOT initialized)
<i>pdHorizontal</i>	Duplex mode initialized - print Head to Toe
<i>pdVertical</i>	Duplex mode initialized - print Head to Head

Note:

Not all printers or printer drivers support duplex printing.

See Also:

[Bin](#), [Collate](#), [Orientation](#), [PaperSize](#), [Printer](#), [Resolution](#)

5.49 ExpandParent property

Default:

True

Component/Class:

[Memo](#)

Description:

When set to true, the height of the parent of the memo will be increased the same amount the height of the memo is increased.

Note:

In order to properly print dynamically sized memos, you will need to *set ExpandParent to true* and also *set the vertical anchor to stretch*.

See Also:

[Anchor Editor](#), [Anchor Property](#), [DataMemo component](#)

5.50 Extended property

Default:

False

Component/Class:

[Code39BarCode](#)

Description:

When set to true, Extended Code 39 format will be output instead of standard Code 39 format.

See Also:

[BarCode components](#)

5.51 FieldName property

Default:

'' (empty)

Component/Class:

all data field components

Description:

Defines the field name that this field component will retrieve data from.

See Also:

[ConnectionName](#)

5.52 FileLink property

Default:

'' (empty)

Component/Class:

[Bitmap](#), [Metafile](#)

Description:

Defines a filename to initialize the bitmap or metafile with a file on the hard disk.

See Also:

[Image](#)

5.53 FillColor property

Default:

White

Component/Class:

[Circle](#), [Ellipse](#), [Rectangle](#), [Square](#)

Description:

Sets the color that is used to fill the graphical shape.

See Also:

[BorderColor](#), [Color](#), [FillStyle](#)

5.54 FillStyle property

Default:

fsSolid

Component/Class:

[Circle](#), [Ellipse](#), [Rectangle](#), [Square](#)

Description:

Sets the style used to fill the graphical shape. You can use the Fill Toolbar to select the fill style desired or enter it from this properties drop list box.

fsBDiagonal
fsClear
fsCross
fsDiagCross
fsFDiagonal
fsHorizontal
fsNone
fsSolid
fsVertical

Note:

Setting the FillStyle to fsClear will cause the [FillColor](#) to be set to White.

See Also:

[BorderColor](#), [Color](#), [FillColor](#)

5.55 FinishNewPage property

Default:

False

Component/Class:

[Band](#), [DataBand](#)

Description:

Very similar to [StartNewPage](#), but if set to True, it will cause a new page to begin **AFTER** this band has finished printing.

Note:

The [StartNewPage](#) property would normally be used on header bands while the *FinishNewPage* property would normally be used on footer bands.

See Also:

[MaxRows](#), [StartNewPage](#)

5.56 FirstPage property

Default:

'' (empty)

Component/Class:

[Report](#)

Description:

Defines the first page that will be printed in a report if the *PageList* property is empty.

See Also:

[GotoMode](#), [GotoPage](#), [PageList](#)

5.57 Font property

Default:

System font

Component/Class:

all text components

Description:

Defines the font that will be used to draw the contents of a text component.

See Also:

[FontJustify](#)

5.58 FontJustify property

Default:

pjLeft

Component/Class:

most text components

Description:

Sets the horizontal justification of the text data in the box.

 pjBlock block justify the text

 pjLeft left justify the text

 pjCenter center the text

 pjRight right justify the text

See Also:

[Font](#), [Font Editor](#), [FontMaster](#), [Fonts Editor](#)

5.59 FontMirror property

Default:

System font

Component/Class:

most text components

Description:

This property sets the FontMaster component that is used to define the font for the text. Setting this property to a font master will override the Font property.

See Also:

[Font](#)

5.60 FullName property

Default:

'' (empty)

Component/Class:

[Page](#), [TRaveReport](#), [DataView](#), [DataField](#)

Description:

Defines the full name or long name of the project item component. The full name is for single line display and may contain special characters and spaces.

See Also:

[Description](#), [Name](#)

5.61 GotoMode property

Default:

gmGotoDone

Component/Class:

[Page](#)

Description:

This property determines the behavior of the [GotoPage](#) property.

gmGotoDone - Go to the [GotoPage](#) property setting when the page definition has completely finished printing. This may be several physical pages if you have a region defined on the page.

gmGotoNotDone - Go to the [GotoPage](#) property after a single physical page has printed, only if the current page definition has not finished printing. This option is usually only used with mirrored sections (odd/even layouts, different first page, ...).

gmCallEach - Call [GotoPage](#) property after each physical page has printed whether the page definition is finished printing or not. This is useful for inserting other pages after each page of a report prints. Control is returned to the calling page when the page chain ends (i.e., a blank [GotoPage](#) property is encountered)

See Also:

[GotoPage](#), [PageList](#)

5.62 GotoPage property

Default:

'' (empty)

Component/Class:

[Page](#)

Description:

Defines the page that will be printed after the current page according to the [GotoMode](#) property rule.

See Also:

[GotoMode](#), [PageList](#)

5.63 GridLines property

Default:

5

Component/Class:

[Page](#)

Description:

Controls the grid lines that are visible. The default setting of 5 means that every fifth grid line will be visible. If [GridSpacing](#) is set to 0.1, then it means that the visible grid lines will be every 1/2 inch but snap to grid will occur every 0.1 inches.

See Also:

[GridSpacing](#)

5.64 GridSpacing property

Default:

0.1

Component/Class:[Page](#)**Description:**

Sets the spacing between grid lines.

See Also:[GridLines](#)

5.65 GroupDataView property

Default:

'' (empty)

Component/Class:[Band](#), [DataBand](#)**Description:**

Defines the data view that will be used to calculate the [GroupKey](#) from.

See Also:[GroupKey](#)

5.66 GroupKey property

Default:

'' (empty)

Component/Class:[Band](#), [DataBand](#)**Description:**

Defines the field(s) that will be used to calculate the group key. When defining a report using grouping headers or footers, the *GroupKey* property is used to determine when a new group is encountered. Multiple fields should be separated with +'s, for example, "Customer Number" + "InvoiceNo".

See Also:[GroupDataView](#)

5.67 Height property

Default:

'None'

Component/Class:

all visible components

Description:

Defines the overall height of the component.

For bar codes this is a read only property that contains the height of the entire bar code.

If the bar code [PrintReadable](#) property is set to true, then the Height property contains the bar code height plus the line height of the current font.

See Also:

[BarHeight](#), [PrintReadable](#)

5.68 HRadius property

Default:

0

Component/Class:

[Rectangle](#), [Square](#)

Description:

Controls the horizontal radius of the rectangle corner. When used in combination with [VRadius](#), these properties allow you to round the corners of rectangles or squares.

See Also:

[VRadius](#)

5.69 Image property

Default:

'' (empty)

Component/Class:

[Bitmap](#), [Metafile](#)

Description:

Defines the image that will be printed with a bitmap or metafile component.

See Also:

[FileLink](#)

5.70 InitCalcVar property

Default:

'' (empty)

Component/Class:

[CalcController](#)

Description:

When this component is acting as an initializer, it defines the calculation component that will be used as the initializing value. If this property is defined, InitDataField and InitValue will be ignored.

See Also:

[InitDataField](#), [InitDataView](#), [InitValue](#)

5.71 InitDataField property

Default:

'' (empty)

Component/Class:

[CalcController](#), [PageNumInit](#)

Description:

CalcController - If the InitCalcVar property is blank, then this property defines the data field or project parameter that will be used as the initializing value for any components that this is an initializer for. If this property is defined, InitValue will be ignored.

PageNumInit - Defines the value that the relative page number will start from when this component is printed. If the text cannot be converted to a valid integer, a default value of 1 will be used.

See Also:

[InitCalcVar](#), [InitDataView](#), [InitValue](#)

5.72 InitDataView property

Default:

'' (empty)

Component/Class:

[CalcController](#), [PageNumInit](#)

Description:

Defines the default DataView for the InitDataField property.

See Also:

[InitDataField](#)

5.73 Initializer property

Default:

'' (empty)

Component/Class:

[CalcText](#), [CalcTotal](#)

Description:

Defines the initializer component. The CalcController is typically the initializer, which will initialize the calculation when the initializer is printed. Initializers are useful for setting a calculation to a specific value or initializing it at key points in a report.

See Also:

[Controller](#), [CalcController](#)

5.74 InitToFirst property

Default:

True

Component/Class:

[DataBand](#), [DataCyle](#)

Description:

Moves the DataSet to the Top of File or first record position.

See Also:

[ConnectionName](#)

5.75 InitValue property

Default:

0 for CalcController

1 for PageNumInit

Component/Class:

[CalcController](#), [PageNumInit](#)

Description:

CALCCONTROLLER - If the InitCalcVar and InitDataField properties are blank, this property defines a constant value that will be used as the initializing value for any components that this is an initializer for.

PAGENUMINIT - Defines the value that the relative page number will start from when this component is printed. Only the RelativePage report variable (not CurrentPage) will reflect the new page number.

See Also:

[InitCalcVar](#), [InitDataField](#), [InitDataView](#)

5.76 KeepBodyTogether property

Default:

False

Component/Class:

[DataBand](#)

Description:

This property, if true, causes the data band to attempt to keep the bands from the body header to the body footer together on the same page.

See Also:

[KeepRowTogether](#)

5.77 KeepRowTogether property

Default:

False

Component/Class:

[DataBand](#)

Description:

This property, if true, causes the data band to attempt to keep the bands from the row header to the row footer together on the same page.

See Also:

[KeepBodyTogether](#)

5.78 Left property

Default:

'None' <left edge of selected component>

Component/Class:

all visible components

Description:

Sets the position for the left edge of the component.

See Also:

[Anchor Editor](#), [Anchor](#), [Bottom](#), [ExpandParent](#), [Right](#), [Top](#), [Width](#)

5.79 LineStyle property

Default:

psSolid

Component/Class:

all line components

Description:

Sets style of line.

psClear	No line is drawn
psDash	Creates a dashed line
psDashDot	Creates an alternating dash and dot line
psDashDotDot	Creates an alternating dash and double dot line pattern
psDot	Creates a dotted pen
psInsideFrame	Creates a pen that draws a line inside the frame of closed shapes
psSolid	Creates a solid line

Note:

Only psSolid can have a pen width greater than 1 pixel.

See Also:

[BorderStyle](#), [BorderWidth](#), [LineWidth](#)

5.80 LineWidth property

Default:

1

Component/Class:[Drawing](#)**Description:**

Sets the width of the line. The units for this line thickness is controlled by the LineWidthType property. Line widths greater than 1 pixel can only be used with solid lines.

See Also:[LineStyle](#), [LineWidthType](#)

5.81 LineWidthType property

Default:

wtPixels

Component/Class:[Drawing](#)**Description:**

Determines whether the *LineWidth* property of the graphic shape is measure in pixels or points. Use pixels and a *LineWidth* of 1 (the defaults) if the thinnest line possible is needed. Also, to use points for all other *LineWidth* values for consistent thickness across devices.

wtPixels *LineWidth* is measured in pixels

wtPoints *LineWidth* is measured in points (1 point = 1/72nd of an inch)

See Also:[LineWidth](#)

5.82 Locked property

Default:

False

Component/Class:*all components***Description:**

Controls the state of the selected components properties and all of it children (if any). If True, then the properties can not selected or changed. The color of the property names and pips (if selected) will be red when this property is true.

Note:

This can be used to freeze part(s) of a page design so that the design can not accidentally be moved while completing a design in another area.

See Also:[DevLocked](#)

5.83 LookupDataView property

Default:

'' (empty)

Component/Class:

[DataText](#)

Description:

Specifies the data view that you want to perform the lookup on.

See Also:

[DataField](#), [LookupDisplay](#), [LookupField](#), [LookupInvalid](#)

5.84 LookupDisplay property

Default:

'' (empty)

Component/Class:

[DataText](#)

Description:

Specifies the field in *LookupDataView* that will actually be displayed in the report after the lookup is performed.

See Also:

[DataField](#), [LookupDataView](#), [LookupField](#), [LookupInvalid](#)

5.85 LookupField property

Default:

'' (empty)

Component/Class:

[DataText](#)

Description:

Specifies the field(s) in *LookupDataView* that will be matched to the value of the field(s) defined by *DataField*. Once a match is found, the value of *LookupDisplay* will be shown in the report.

See Also:

[DataField](#), [LookupDataView](#), [LookupDisplay](#), [LookupInvalid](#)

5.86 LookupInvalid property

Default:

'' (empty)

Component/Class:

[DataText](#)

Description:

Defines the text that will be displayed if no matches are found for the current value of [DataField](#) in the [LookupDataView](#).

See Also:

[DataField](#), [LookupDataView](#), [LookupDisplay](#), [LookupField](#)

5.87 MailMergeItems property

Default:

'' (empty)

Component/Class:

all memo components

Description:

Stores the tokens and replacement data text items that will be used in a search and replace during a mail merge session.

See Also:

Mail Merge Editor

5.88 MasterDataView property

Default:

'' (empty)

Component/Class:

[DataBand](#), [DataCycle](#)

Description:

Defines the DataView that will be used to calculate the *MasterKey* property from.

See Also:

[DetailKey](#), [MasterKey](#)

5.89 MasterKey property

Default:

'' (empty)

Component/Class:

[DataBand](#), [DataCycle](#)

Description:

Determines the master key field(s) for a master-detail style report. The contents of the MasterKey field(s) will be matched to the [DetailKey](#) property to set up the master-detail relationship in the detail DataBand. Multiple fields should be separated by +', for example, "Customer Number" + "InvoiceNo".

See Also:

[DetailKey](#), *MasterDataBand*

5.90 MatchSide property

Default:

msWidth

Component/Class:

[Bitmap](#), [Metafile](#)

Description:

Determines the method that the bitmap or metafile will use to resize itself.

msBoth The image will size to match both the designed width and height.

msHeight The image will match the designed height and adjust the width to maintain the correct proportions.

msInside The image will be drawn proportionally inside the designed area.

msWidth The image will match the designed width and adjust the height to maintain the correct proportions.

See Also:

[Height](#), [Width](#)

5.91 MaxPages property

Default:

0

Component/Class:

[Report](#)

Description:

Allows you to set (limit) the maximum number of pages to print in a report. This could be useful during the development cycle of large reports. If the value is 0 then all pages in the report will be printed.

See Also:

[Height](#), [Width](#)

5.92 MaxRows property

Default:

0

Component/Class:

[DataBand](#), [DataCycle](#)

Description:

Defines the maximum number of rows that will be printed for a data band. A value of 0 says to print all available rows in the data view.

See Also:

[StartNewPage](#)

5.93 MinHeightLeft property

Default:

0

Component/Class:

[Band](#)

Description:

Defines the minimum height that must remain in the region before this band will print. If the MinHeightLeft value is greater than the remaining height in the region, then band will be printed on the next page.

See Also:

[Region](#)

5.94 Mirror property

Default:

'' (empty)

Component/Class:

all components

Description:

Will cause the currently selected component to mirror (duplicate) the properties of the component entered from the list.

See Also:

[Global Page](#)

5.95 Name property

Default:

'None'

Component/Class:

all components

Description:

Defines the name of the component as referenced in the application's code. Use the Name property to assign a new name to the control or to find out what the name of the control is. By default, Rave assigns sequential names based on the type of the control, such as 'Rectangle1', 'Rectangle2', and so on. Change these to more meaningful names that make the code more readable. The Name must not contain any spaces or special characters. This is the name that will be used in the Project Tree Panel.

See Also:

[Description](#), [FullName](#)

5.96 NullText property

Default:

'' (empty)

Component/Class:

[DataField](#)

Description:

Sets the contents that will be printed if the [DataField](#) value is blank (empty).

See Also:

[DataField](#), [TextFalse](#), [TextTrue](#)

5.97 Operator property

Default:

coAdd

Component/Class:

[CalcOp](#)

Description:

Defines the type of operation that will be performed on the two source values. The result of this calculation can be placed in a project parameter using the DestParam property or used in other calculations as a CalcVar component.

coAdd	operation set to Source1 + Source2
coAverage	operation set to (Source1 + Source2) / 2.0
coDiv	operation set to Source1 / Source2
coExp	operation set to Source1 raised to the Source2 power
coGreater	operation set to the greater of Source1 and Source2
coLesser	operation set to the lesser of Source1 and Source2
coMod	operation set to Source1 mod Source2
coMul	operation set to Source1 * Source2
coSub	operation set to Source1 - Source2

See Also:

[DestParam](#), [ResultFunction](#), [Src1Xxxx](#), [Src2Xxxx](#)

5.98 Orientation property

Default:

poDefault

Component/Class:

[Page](#)

Description:

Sets the orientation for a page to landscape or portrait. The value of poDefault will use the default setting for the printer to determine the orientation.

See Also:

[Bin](#), [Collate](#), [Duplex](#), [PaperSize](#), [Printer](#), [Resolution](#)

5.99 OrphanRows property

Default:

0

Components:

[DataBand](#)

Description:

Lines of a paragraph that start at the bottom of a page or column but end on the next page or column are known as orphans. This property sets the minimum number of lines that can be by themselves at the **bottom** of a page. The default value of 0 and allows orphans (i.e. 1 row at the bottom of a page).

See Also:

[KeepBodyTogether](#), [KeepRowTogether](#), [WidowRows](#)

5.100 PageHeight property

Default:

11

Component/Class:

[Page](#)

Description:

Defines the height of the page. Normally this property will be modified via the [PaperSize](#) property.

See Also:

[PageWidth](#), [PaperSize](#)

5.101 PageList property

Default:

'' (empty)

Component/Class:

[Report](#)

Description:

Defines a list of pages to print when the report is executed. If no pages are defined in the PageList then the report will start with what is defined by the [FirstPage](#) property.

See Also:

[FirstPage](#), Batch Report example in manual

5.102 PageWidth property

Default:

'' (empty)

Component/Class:

[Page](#)

Description:

Defines the width of the page. Normally this property will be modified via the [PaperSize](#) property.

See Also:

[PageHeight](#), [PaperSize](#)

5.103 PaperSize property

Default:

"Letter, 8 ½ by 11 inches"

Component/Class:

[Page](#)

Description:

Provides a method to select the size of paper being used for a report node. Select one of the report nodes, the property panel will show the properties available at the report level. Select the PaperSize and there will be a drop list where you can select the paper size desired.

Note:

All pages within a report must be of the same size. If they are not, the first printed page will determine the size used for all pages within that report.

See Also:

[Orientation](#), [PageHeight](#), [PageWidth](#)

5.104 Parameters property

Default:

'' (empty)

Component/Class:

[Page](#), [ProjectManager](#), [Report](#)

Description:

Allows parameters to be defined that are available at the project, report and page level. Parameters can be used to print data that is passed from the application (e.g. ReportTitle or UserName) or to store calculation results. Parameters can be printed using the Data Text Editor of the DataText component. See FAQ (Frequently Asked Question) [Events - Getting & Setting Parameters](#) for example on us parameters in an event.

See Also:

[DataText](#), [DataText Editor](#), [DestParam](#), [DestPIVar](#), [PIVars](#), [Variables](#)

5.105 PIVars property

Default:

'' (empty)

Component/Class:

[Page](#), [ProjectManager](#), [Report](#)

Description:

Allows PIVars (Post Initialize Variables) to be defined at the project, report and page level. PIVars are initialized using the DestPIVar property of the calculation components. PIVars are normally printed using the Data Text Editor of the DataText component. The main difference between PIVars and parameters is that PIVars will use the value that is set after (*in print order*) the PIVar while parameters print the value that was set before. A common use for PIVars is to print a total in a header band that would be initialized later in the footer band. This works even across multiple pages. Report.AlwaysGenerate must be true if you are using PIVars in your report.

See Also:

[AlwaysGenerate](#), [DataText](#), [DestPIVar](#), [Parameters](#)

5.106 PositionMode property

Default:

pmOffset

Component/Class:

[Band](#), [DataBand](#)

Description:

Determines how the position of this band will be treated relative to the previously printed band.

pmAbsolute	distances measured from top of region
pmOffset	distances measured from bottom of last band
pmOverlay	does not advance band position

See Also:

[PositionValue](#)

5.107 PositionValue property

Default:

0

Component/Class:

[Band](#), [DataBand](#)

Description:

Sets the starting position of this band based upon the PositionMode property setting. If the setting is pmAbsolute, then this bands starting position is measured from the top of the controlling region for this band. If the setting is pmOffset, then the distance is from the bottom of the last printed band. The pmOverlay setting acts like pmOffset, however, it does not advance the "last" band printed setting.

See Also:

[PositionMode](#)

5.108 PrintChecksum property

Default:

False

Component/Class:

all bar code components

Description:

Determines if the readable text includes the checksum character.

Note:

It is possible that the checksum character may not be a printable character with some of the bar code types.

See Also:

[BarTop](#), [UseChecksum](#)

5.109 Printer property

Default:

' ' (empty)

Component/Class:

[Report](#)

Description:

Determines the printer that the report will be output to. Normally you will want to leave this field blank to print to the default printer.

See Also:

[Bin](#), [Collate](#), [Duplex](#), [Orientation](#), [PaperSize](#), [Resolution](#)

5.110 PrintReadable property

Default:

True

Component/Class:

all bar code components except UPC

Description:

Set this property to false if you do not want readable text to be printed along with the bar code.

Note:

For UPC bar codes, text is always printed.

See Also:

[PrintTop](#), [TextJustify](#)

5.111 PrintTop property

Default:

False

Component/Class:

all bar code components

Description:

Set this property to true if you want the readable text to be printed on top of the bar code. A false value means that the readable text will be printed below the bar code. This property has no effect when printing UPC codes, since the UPC text is always printed at the bottom of the bar code.

See Also:

[PrintReadable](#), [TextJustify](#)

5.112 ReprintLocs property

Default:

(ALL)

Component/Class:

[Band](#), [DataBand](#)

Description:

Determines whether the band reprints at the start of a new page. When a band is set to reprint on a new page, the type of band that caused the rollover to the new page will be checked against the band types in the *ReprintLocs* property. If the band type is found in *ReprintLocs*, then the band will be reprinted on the new page.

See Also:

[BandStyle](#)

5.113 Resolution property

Default:

prDefault

Component/Class:

[Report](#)

Description:

Determines the output resolution of the print job.

prDefault
prDraft
prHigh
prLow
prMedium

See Also:

[Bin](#), [Collate](#), [Duplex](#), [Orientation](#), [PaperSize](#), [Printer](#)

5.114 ResultFunction property

Default:

cfNone

Component/Class:

[CalcOp](#)

Description:

Defines the function that will be applied to the result of the calculation after the operation has been performed.

cfAbs	result is the absolute of the value
cfArcTan	result is the ArcTan of the value in radians
cfCos	result is the cosine of the value in radians
cfDec	result is the value - 1
cfFrac	result is the decimal portion of the value
cfHoursToTime	converts the value in hours to DateTime format
cfInc	result is the value + 1
cfMinsToTime	converts the value in minutes to DateTime format
cfNeg	result is the value * -1
cfNone	result is unchanged
cfPercent	result is multiplied by 100
cfRandom	Result is a random value between 0 and the value
cfRound	Result is rounded to the nearest integer
cfRound1	Result is rounded to the 1st decimal place
cfRound2	Result is rounded to the 2nd decimal place
cfRound3	Result is rounded to the 3rd decimal place
cfRound4	Result is rounded to the 4th decimal place
cfRound5	Result is rounded to the 5th decimal place
cfSecsToTime	Converts the value in seconds to datetime format
cfSin	Result is the Sine of the value in radians
cfSqr	Result is the square of the value
cfSqrt	Result is the square root of the value
cfTimeToHours	Converts the value in datetime format to hours
cfTimeToMins	Converts the value in DateTime format to minutes
cfTimeToSecs	Converts the value in DateTime format to seconds
cfTrunc	Result is the integer portion of the value

See Also:

[DestParam](#)

5.115 Right property

Default:

'None' <right edge of selected bar code component>

Component/Class:

all bar code components

Description:

Sets or returns the position for the right edge of the bar code. When a value is assigned to Right, the BarCodeJustify property is set to pjRight as well.

See Also:

[Anchor Editor](#), [Anchor](#), [BarCodeJustify](#), [Bottom](#), [Center](#), [ExpandParent](#), [Left](#), [Right](#), [Top](#)

5.116 Rotation property

Default:

0

Component/Class:

[CalcText](#), [DataText](#), [Text](#)

Description:

Defines the rotation in degrees of the selected text component. The text is rotated counter clockwise through the values 0 to 359.

See Also:

[BarCodeRotation](#)

5.117 RunningTotal property

Default:

False

Component/Class:

[CalcText](#), [CalcTotal](#)

Description:

Determines if a running total is kept when a NewPage occurs.

False means NO and the value is reset to 0 each time it is printed

True means Yes and the value is kept and will continue to total after it is printed

See Also:

[CalcType](#), [CountBlanks](#)

5.118 Size property

Default:

'none'

Component/Class:

[DataField](#)

Description:

Defines the character width of the data field. This is normally used with design time activities such as dragging and dropping DataText components to determine an approximate width that it will take to print the component.

5.119 SortKey property

Default:

'' (empty)

Component/Class:

[DataBand](#)

Description:

Defines the field(s) that will be passed to the data connection to set up a sort. The SortKey can be a composite key and it can be ascending (plus sign) or descending (minus sign). For example,

"Customer Number" - "InvoiceNo"

would be ascending order of customer numbers with descending invoice number within each customer number. The minus sign designated the descending order.

Note:

The underlying database must support the fields being passed as a sort key.

See Also:

[FieldName](#)

5.120 Src1CalcVar property

Default:

'' (empty)

Component/Class:

[CalcOp](#)

Description:

Defines a calculation component to use as the first source value for the calculation operation. If this property is defined, Src#DataField and Src#Value will be ignored.

See Also:

[Src#DataField](#), [Src#DataView](#), [Src#Function](#), [Src#Value](#)

5.121 Src1DataField property

Default:

'' (empty)

Component/Class:

[CalcOp](#)

Description:

If Src#CalcVar is blank, this defines a data field to use as the source value for that calculation operation. If this property is defined, Src#Value will be ignored.

See Also:

[Src#CalcVar](#), [Src#DataView](#), [Src#Function](#), [Src#Value](#)

5.122 Src1DataView property

Default:

'' (empty)

Component/Class:

[CalcOp](#)

Description:

Defines the default DataView that will be used for the Src#DataField property.

See Also:

[Src#DataField](#)

5.123 Src1Function property

Default:

cfNone

Component/Class:

[CalcOp](#)

Description:

Defines the function that will be performed on the source value before the calculation operation is performed.

Note:

see [ResultFunction](#) (for list of constants).

See Also:

[Src#CalcVar](#), [Src#DataField](#), [Src#Value](#)

5.124 Src1Value property

Default:

0

Component/Class:

[CalcOp](#)

Description:

If Src#CalcVar and Src#DataField are blank, defines a constant value that will be used as the first source value for the calculation operation.

See Also:

[Src#CalcVar](#), [Src#DataField](#), [Src#Function](#)

5.125 StartNewPage property

Default:

False

Component/Class:

[Band](#), [DataBand](#)

Description:

If true, will cause the region to resume on the next page if the band or DataBand has already been printed on the current page.

Note:

The *StartNewPage* property would normally be used on header bands while the [FinishNewPage](#) property would normally be used on footer bands.

See Also:

[FinishNewPage](#), [MaxRows](#)

5.126 Tag property

Default:

nil

Component/Class:

all components

Description:

Tag has no predefined meaning to Rave. The Tag property is provided for the convenience of storing additional integer value or pointer information for special needs in an application. For example, use the Tag property when implementing case statements with a component.

See Also:

[Name](#)

5.127 Text property

Default:

'None'

Component/Class:

text components and all bar code components

Description:

Defines the text to be printed.

Note:

For bar codes, do not include the check character. The check character will be automatically calculated and printed according to the state of the *UseChecksum* property.

Note:

For bar codes, any characters that are invalid for the bar code type will be deleted from the text property upon assignment.

See Also:

[Font](#), [FontJustify](#), [UseChecksum](#)

5.128 TextFalse property

Default:

'' (empty)

Component/Class:

[BooleanField](#)

Description:

Determines what will be printed if the field value is False. A blank value for this property will print the text "False".

See Also:

[TextTrue](#)

5.129 TextJustify property

Default:

pjCenter

Component/Class:

all bar code components

Description:

Determines how the readable text is justified in relation to the bar code.

pjBlock	Block justify the text portion
pjLeft	Left justify the text portion
pjCenter	Center justify the text portion
pjRight	Right justify the text portion

See Also:

[PrintReadable](#), [PrintTop](#), [Text](#)

5.130 TextTrue property

Default:

'' (empty)

Component/Class:

[BooleanField](#)

Description:

Determines what will be printed if the field value is True. A blank value for this property will print the text "True".

See Also:

[TextFalse](#)

5.131 Top property

Default:

'None' <top edge of selected component>

Component/Class:

all visible components

Description:

Sets or returns the position for the top edge of the component. For bar codes, the value for this property includes the readable text, if it is printed.

See Also:

[Anchor Editor](#), [Anchor](#), [BarTop](#), [Bottom](#), [Center](#), [ExpandParent](#), [Left](#), [PrintReadable](#), [PrintTop](#), [Right](#)

5.132 Truncate property

Default:

False for CalcText and Text
True for DataText

Component/Class:

[CalcText](#), [DataText](#), [Text](#)

Description:

When set to true, the text will be truncated to fit the [Width](#) property defined in the designer for that component. If the property is false, then all of the text will print without regard to the width property.

See Also:

[Width](#)

5.133 Units property

Default:

unInch

Component/Class:

[Project](#)

Description:

Sets the default units mode for this project. If the setting is unUser then units factor is determined by the value in [UnitsFactor](#).

unCM	Units are in centimeters
unInch	Units are in inches
unMM	Units are in millimeters
unPoint	Units are in points (1/72nd inch)
unUser	Unit are custom defined in <i>UnitsFactor</i>

See Also:

[UnitsFactor](#)

5.134 UnitsFactor property

Default:

1.0

Component/Class:

[Project](#)

Description:

Sets or returns the current conversion factor necessary to convert units to inches. Its value should equal the number of units that equal an inch.

(unCM = 2.54 since 2.54 centimeters equal an inch)

See Also:

[Units](#)

5.135 UseChecksum property

Default:

False (Code128 := true)

Component/Class:

all bar code components

Description:

Specifies whether a checksum character should be included in the bar code.

See Also:

[BarHeight](#), [BarWidth](#), [PrintReadable](#), [Text](#), [Width](#)

5.136 Visible property

Default:

True

Component/Class:

All

Description:

Determines whether the component will be printed or not. Typically set through scripting in the OnBeforePrint or OnBeforeReport events.

WARNING

Do not put a script that changes visibility in the same component you want to control as it will run until it is "false" but will not run after that as both the component and event "are no longer visible". The event script should be located in a parent to the component or in another component that "prints" before the component you want to control.

See Also:

[Event Scripting](#) , [Example](#)

5.137 VRadius property

Default:

0

Component/Class:

[Rectangle](#), [Square](#)

Description:

Controls the vertical radius of the rectangle corner. When used in combination with [HRadius](#), these properties round the corners of rectangles or squares.

See Also:

[HRadius](#)

5.138 WasteFit property

Default:

True

Component/Class:

[Page](#), [Section](#)

Description:

If WasteFit is true, then components using the "Anchor" property on the page or within the section will dynamically adjust themselves to fit within the printer's waste margins. If a component is not located within the waste area, no adjustment will occur unless it's parent component is located in the waste area.

It is **important** that components (within a page or section) use the "Anchor" property so that they will adjust as the margins change.

See Also:

[Adaptable Reports](#), [Anchor](#), [Anchor Editor](#), [PageHeight](#), [PageWidth](#), [PaperSize](#), [Waste Fit more](#)

5.139 WideFactor property

Default:

3.0

Component/Class:

all bar code components

Description:

The wide factor is the ratio of the wide bar to the narrow bar width.

See Also:

[BarHeight](#), [BarWidth](#), [Width](#)

5.140 WidowRows property

Default:

0

Component/Class:

[DataBand](#)

Description:

Lines of a paragraph at the **top** of a page or column that are part of the paragraph from the prior page or column are known as widows. This property sets the minimum number of lines that can be by themselves on the **top** of the page. The default setting is 0 and allows widows.

See Also:

[KeepBodyTogether](#), [KeepRowTogether](#), [OrphanRows](#)

5.141 Width property

Default:

<Width of current selected component>

Component/Class:

all components

Description:

Sets the width of the component. For bar codes, this will return the calculated width of the entire bar code for the current value of Text.

See Also:

[BarWidth](#), [Height](#), [Left](#), [Text](#), [WideFactor](#)

Appendix

Chapter



VI

6 Appendix

6.1 Lessons

6.1.1 Quick Start

RAVE (Report Authoring Visual Environment) is the visual report designer. This tutorial will quickly guide you through the minimum steps required to build your first visual report with the Rave visual designer and then give a brief overview of what makes up a Rave reporting project. Other tutorials will go into more detail on Rave so if you're ready, let's get started.

- 1) Start Delphi or C++Builder and create a new application.
- 2) Create a TTable, TQuery or some other database component and initialize it to a valid table. Set the component's Active property to true to insure that all properties are set correctly.
- 3) Create a TRvDataSetConnection component (located on the Report component tab) and set the DataSet property to the database component you created in Step 2. Change the name of this new component to "TutorialCXN".
- 4) Create a Project component (located on the Report component tab). Double-click on the new component or right click and select "Rave Visual Designer" to bring up the visual designer.
- 5) Once the Rave visual designer is finished loading, select Project | New Data View from the main menu to bring up the Data Connections dialog. Make sure that TutorialCXN is highlighted in the Active Data Connections listbox and press the OK button.
- 6) Locate the Project Tree (the treeview on the left side of the visual designer) and open up the Data View Dictionary. Select the new data view, DataView1, that was just created. Using the Property Panel (located below the Project Tree), change DataView1's Name property to TutorialDV.
- 7) Now we're ready to create a report. Select Tools | Report Wizards | Simple Table from the main menu to bring up the Simple Table wizard. Make sure TutorialDV is selected and press Next to advance. Select 2 or 3 fields in the listbox and press Next to advance. Change the order of the fields if you wish and press Next to advance. Change the report title to describe the contents of this report and the press Next to advance. Change the font sizes if you wish and the press Generate to create the report.
- 8) To preview this report, select Project | Execute Report to bring up the Report Setup dialog. Make sure "Preview" is selected as the report destination and press the OK button. You should now see a preview of your report.

Congratulations! You have now created your first Rave report. The following is a list of what typically makes up an application using Rave:

- 1) Data connection components - If you noticed in the above example, Rave uses data from your application. The standard data connection components, TRPCustomConnection, TRvDataSetConnection, TRPTableConnection and TRPQueryConnection provide a bridge between the data in your application and the Rave visual components. The TRPCustomConnection component can be used to access non-database data such as memory arrays or binary record files. TRvDataSetConnection can be used to provide access to TDataSet descendent components including 3rd party dataset components. TRPTableConnection and TRPQueryConnection are to be used specifically with TTable and TQuery components or their descendent's respectively. More detailed usage of data connection components is explained in much more detail in a later tutorial.
- 2) Project component - This component provides access to the reports and their components. The Project component contains many properties and methods that allow you to create, modify, print and design your reporting projects and will be explained in much more detail in a later tutorial. You will usually only require one Project component per application, but there is no limitation to having more. More detailed usage of the Project component is explained in much more detail in a later tutorial.
- 3) Report project file (.RAV file) - The report project file is where the report definitions are stored by the Rave visual designer. This is a binary file, similar to Delphi's .DFM files. All reports, global pages and data views for the reporting project are stored in this single file. You can export or import items from or to a report project file. Using methods of the Project component, you can also store the report project file in a database blob field or other location.
- 4) Reports - Reports are stored in the Report Library of the reporting project. A Rave report is made up of report pages and the visual reporting components stored on those pages. You can create as many page definitions as you want and combine them in a wide variety of methods.
- 5) Global Pages - Global pages are stored in the Global Page Catalog of the reporting project. Components contained on global pages, unlike those in report pages, are visible to all reports. global pages are a useful for storing templates that are mirrored on other report pages.
- 6) Data Views - Data views are stored in the Data View Dictionary of the reporting project. Data views provide an interface to data connection components. When creating new data views, you must have the data connection component active in either a running application or on a loaded Delphi or C++Builder form. The data view will then query the data connection component to retrieve meta-data information such as field names, data types, etc. Field components are contained within each data view allowing properties to be set for each data column.

6.1.2 CalcOp

For example, there could be two DataText components that need to be calculated together. Like $A + B = C$, where A and B represent the two DataText component values and C represents the result. This is where the CalcOp component would be used.

To choose the DataText components in the above example, use the Src1X and Src2X properties, where X is either CalcVar (a calculation variable), DataField, or DataValue. These are the three types of value sources that can be used for calculation in a CalcOp component. Note that the Src (source) can only use ONE of the three available source types, and there can only be two sources, Src1 or Src2.

Src1CalcVar	▼
DiscCalcOp2 DiscPriceCalcOp GrandCalcTotal SubCalcTotal SubTotalCalcOp TaxCalcOp TaxCalcTotal TotalPriceCalcOp	

Src1DataField	AmountPaid ▼ ...
Src1DataView	CustOrdDV

Src1Value	0.80
-----------	------

The three source types have many different values associated with them. For a CalcVar source, the drop down menu will list all the calculation variables available in that Page available for use. This calculation variable is just that, a variable for holding a value. This value can be from another CalcOp component or from some other calculation component. For the DataField source to get available values first designate what DataView the DataField will come from. In other words the DataField is a field in a table, which is the DataView. So, in order to choose a field, the DataView must first be chosen. For a Value source, typing in a numeric value is all that is needed to fill this property. Again, remember only one of these three sources can be assigned to a source.

Once the sources are chosen the operation to be used between them has to be chosen. To choose the desired operation, use the Operator property by using the drop down menu and making the appropriate choice. In the example, $A + B = C$, the operator would be "coAdd".

Operator	coAdd ▼
coAdd coAverage coDiv coExp coGreater coLesser coMod coMul coSub	

Src2Function	cfNone ▼
cfCos cfDec cfFrac cfHoursToTime cfInc cfMinsToTime cfNeg cfNone cfPercent	

There may be times when a function needs to be performed on a value before it is processed with the second value. This is where the property SrcYFunction, where Y is 1 or 2 for Src1 or Src2, will become handy. With SrcYFunction, the value can be converted (like from hours to minutes), or have a trigonometric function performed on it (like take the sin of the value), or have other functions performed on the value (like take the square root of the value, or take the absolute value).

Once the two values are chosen, it is now time to deal with the result. In the example, $A + B = C$, C is the value of the result. The result can either be written out to a project parameter (most likely a DataText component) or just held as an intermediate value. To write out the value to a parameter, use the DestParam property to choose the desired parameter to write the result write to. If the result is going to be in intermediate result, there is nothing else to do to the component. Although, it is highly suggested to rename the component, by using the Name property, to easily recognize the component for future use. This is because, as in intermediate values, you will most likely use the components in another CalcOp component, and a good reference will help to easily create the next step in the calculation process.

After setting the values and setting the result destination, it may be necessary to format the result or to even perform one last function on the result. Using the DisplayType and DisplayFormat properties to format the result into any necessary format. The DisplayType has two options, DateTimeFormat and NumericFormat. DisplayFormat has many options that must be typed into the edit box. To find these options look at the Formatting codes in this manual. Like the two values, A and B, in the example, $A + B = C$, the result C can also have a function performed on it before being written out to a parameter, or as a holding value. To select a function, use the ResultFunction property drop down menu.

$$\frac{(A+B)*(C*D)}{0.80} = E$$

$$A + B = Z$$

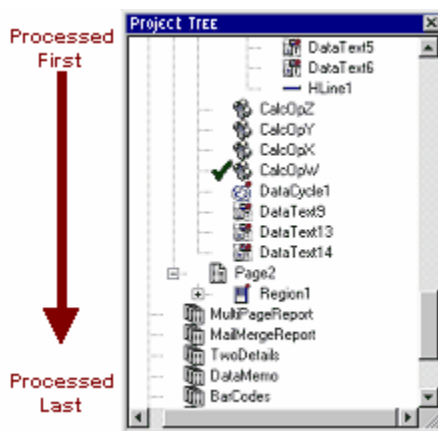
$$C * D = Y$$

$$Z * Y = X$$

$$\frac{X}{0.80} = W = E$$

The CalcOp components can also be chained together for more complex expressions using the Src#CalcVar properties, which can be set to other CalcOp or CalcTotal components. For example, to create the complex expression shown above in the top of box, it will be necessary to break up the expression into simple 2 value steps shown in the bottom of that box.

To break up the complex expression, four new expressions will need to be created and saved as CalcOp components. So the resulting components, Z, Y, X, and W will be four separate intermediate CalcOp's. But, as you can see from the flow of the expression, Z and Y will have to be completed before X can be created, as it is dependent upon the two previous operations. And finally, W will be completed last, after X, to get the final correct value.



Just as it is important to do the calculations in order, it is important to make sure the components are in order in the Project Tree. When a report is executed, the report executes components down the Project Tree. For CalcOp components or any calculation component, this means that they need to be in the correct order. In the example above, if Z, Y, X, and W were in the Project Tree, Z would have to be first in the tree and W would have to be last in the tree. This would mean that at execution, Z would be processed first, then Y, and so on. It is also important to note that if Z is dependent upon any other components (like other CalcOp components or DataText components), those components must come first in the Tree before the Z component is processed. Making sure the components are in correct order is known as setting the print order. To move the components up and down within the Project Tree, use the Alignment Palette to change the print order of the components.

6.1.3 Calculated Fields

You can create calculated fields in Delphi and use those in your reports. However, those fields are also available to Delphi visual form components such as DBGrids so that is not always a desirable option. Another problem is that you cannot define calculated fields for memo or graphic field types. Here is a method to define calculated fields in Delphi that will only be available to your reports, will not impact the rest of your application and works for memo or graphic fields. The following example could be done using DataMirrors, but we want to show a custom data connection solution. This example demonstrates a method for printing addresses where some of the lines may be blank and you do not want them printed.

1) Define an OnGetCols event

```
procedure TForm1.CustomerCXNGetCols(Connection: TRPCustomConnection);
begin
    With Connection do begin
        DoGetCols;
        WriteField('FullAddress', dtMemo, 30, '', '');
    end; { with }
end;
```

The DoGetCols call will create the default columns (fields) that would have been created if you did not define this event. The only step left for this event is to call WriteField to define the meta-information for each field that you want to define. See Lesson - "Custom Data Connection" for more information on WriteField.

2) Define an OnGetRow event

```
procedure TForm1.CustomerCXNGetRow(Connection: TRPCustomConnection);
var
    Stream: TMemoryStream;
    Memo: TMemo;
    s1, s2: string;
    i1: integer;
begin
    With Connection do begin
        DoGetRow; // get data for table fields

        For i1 := 0 to 3 do begin
            case i1 of
                0: s2 := Table1.FieldName('Company').AsString;
                1: s2 := Table1.FieldName('Addr1').AsString;
                2: s2 := Table1.FieldName('Addr2').AsString;
                3: begin
                    s2 := Table1.FieldName('City').AsString;
                    s2 := s2 + ', ' + Table1.FieldName('State').AsString +
                        ' ' + Table1.FieldName('Zip').AsString;
                end;
            end;
            // case
            if s2 <> '' then begin
                if s1 = '' then begin
                    s1 := s2;
                end else begin
                    s1 := s1 + #13#10 + s2;
                end;
            end; // else
        end; // if
    end;
```

```

end;          // for

Stream := TMemoryStream.Create;
Memo := TMemo.create(self);
try
    Memo.Text := s1;
    Memo.Lines.SaveToStream(Stream);
    Stream.Position := 0;
    WriteBlobData(Stream.Memory^, Stream.Size);
finally
    Stream.Free;
    Memo.Free;
end;          // try finally
end;          // with
end;

```

We need to force the default behavior for this event by calling `DoGetRow`. Then we get the address information and store it in a string variable. We can then write the data for our calculated field by calling `WriteBlobData` (since it is a memo field). See Lesson - "Custom Data Connections" for more information on `WriteBlobData` and the other `WriteXxxxData` methods. You can add more calculated fields to your data connection by adding more calls to `WriteField` and `WriteXxxxData` in each of these events.

As with custom data connections, you must have your application running and the data connection must be set to visible for these fields to show up.

6.1.4 Custom Data Connection

Through the events in the data connection components, you can customize how the data is sent to your Rave reports. For non-database data using the `TRvCustomConnection` component, you will need to provide all access to your data through these events. For database data connection components such as `TRvDataSetConnection`, you will normally only want to override the `OnValidateRow` event. The data connection events are as follows:

Event	Description
OnEOF	Used by Rave to determine if it is at the end of the data. A true value should be returned only if there are no rows or if a call to the <code>OnNext</code> event has moved past the last row.
OnFirst	Called when Rave wants the data cursor to be moved to the first row of the data. With Rave's advanced buffering system, this will usually only be called once at the beginning of a data session.
OnGetCols	Used by Rave to retrieve the meta-data of the data. This includes the field name, type, character size, full name and description. More details below.
OnGetRow	Used by Rave to retrieve the data for the current row. More details below.
OnGetSorts	Used by Rave to retrieve the available sorting methods.
OnNext	Used by Rave to move the data cursor to the next row.
OnOpen	Used by Rave to initialize the data session. The current state can be saved for a later call to <code>OnRestore</code> .
OnRestore	Used by Rave to restore the data session to its state before it was

	opened.
OnSetFilter	Used by Rave to filter the data for items such as Master-Detail reports.
OnSetSort	Used by Rave to sort the data. More details below.
OnValidateRow	Called for each row to allow custom filtering of the data. For a custom data connection, this event is normally not needed since filtering the data through the OnNext event is more efficient. However, this event can be very useful for other, more automated data connections such as TRvDataSetConnection. More details below.

The TRvCustomConnection component has a DataIndex and DataRows property of type integer. These are provided for use by custom connector events and if used, can alleviate the need to define the OnFirst, OnNext and OnEOF events. DataIndex is intended to be used as the data cursor position with 0 representing the first row. DataRows is intended to be used as the row count of the data. If the DataRows property is used, *then it should be located in the OnOpen event*. For example, if you were defining a custom data connection for a memory array, you would only need to initialize the Connection.DataRows property to the number of elements in the memory array and then let Rave handle the OnFirst, OnNext and OnEOF events. In the OnGetRow event you would then access the Connection.DataIndex property to determine which array element to pass back (remember that DataIndex is 0 for the first row).

OnGetCols Event

The OnGetCols event is called when Rave want to retrieve the meta-data information of the data. Inside this event you will want to call the Connection.WriteField method for each field (column) of your data. The definition of WriteField is as follows:

```
procedure WriteField(Name: string;
  DataType: TRPDataType;
  Width: integer;
  FullName: string;
  Description: string);
```

Name is the short name of the field and should only contain alphanumeric characters. DataType is the type of data that this field represents and should be one of the following values: dtString, dtInteger, dtBoolean, dtFloat, dtCurrency, dtBCD, dtDate, dtTime, dtDateTime, dtBlob, dtMemo or dtGraphic. Width is the relative character width of the field. Full name is a more descriptive name of the field and can include spaces and other non-alphanumeric characters. If FullName is blank then the short name is used as the field's full name. Description is a full description of the field and is typically edited with memo component so it can contain multiple lines. Use the description property to define how the field is to be used and provide any other information about the field's data.

Example:

```
procedure TDataForm.CustomCXNGetCols(Connection: TRvCustomConnection);
begin
  With Connection do begin
    WriteField('Index',dtInteger,8,'Index Field','Description 1');
    WriteField('Name',dtString,30,'Name Field','Description 2');
    WriteField('Amount',dtFloat,20,'Amount Field','Description
3');
  end; { with }
end;
```

OnOpen event

The OnOpen event is called to initialize a data session. In this event you can open up data files,

initialize variables and save the current state of the data for the OnRestore event which will be called to terminate the data session.

Example:

```
procedure TDataForm.CustomCXNOpen(Connection: TRvCustomConnection);
begin
    AssignFile(DataFile, 'DATAFILE.DAT');
    Reset(DataFile, 1);
end;
```

Example (from RaveDemo):

```
procedure TDataForm.CustomDetailCXNOpen(Connection:
TRvCustomConnection);
Begin
    Connection.DataRows := 100;
    Connection.DoFirst;
End;
```

OnFirst event

The OnFirst event is called to position the data cursor to the first row of the data.

Example:

```
procedure TDataForm.CustomCXNFirst(Connection: TRvCustomConnection);
begin
    Seek(DataFile, 0);
    BlockRead(DataFile, DataRecord, SizeOf(DataRecord), DataRead);
end;
```

OnNext event

The OnNext event is called to move the data cursor to the next row of data.

Example:

```
procedure TDataForm.CustomCXNNext(Connection: TRvCustomConnection);
begin
    BlockRead(DataFile, DataRecord, SizeOf(DataRecord), DataRead);
end;
```

OnEOF event

The OnEOF event is called to return whether the data cursor is beyond the EOF or not. A true value should be returned only if there are no rows or if a call to the OnNext event has moved past the last row.

Example:

```
procedure TMainForm.CustomCXNEOF(Connection: TRPCustomConnection; var
EOF: Boolean);
begin
    EOF := DataRead < SizeOf(DataRecord);
end;
```

OnGetRow event

The OnGetRow event is called to retrieve the data for the current row. There are several methods used to write the data to a special buffer used by Rave. The order and types of the fields written must match exactly the field definitions provided in the OnGetCols event.

The following is a list of the methods provided by the Connection object for writing data out to the

data buffer.

```

procedure WriteBCDDData(FormatData: string;
  NativeData: currency); { dtBCD }
procedure WriteBlobData(var Buffer;
  Len: longint); { dtBlob, dtMemo and dtGraphic }
procedure WriteBoolData(FormatData: string;
  NativeData: boolean); { dtBoolean }
procedure WriteCurrData(FormatData: string;
  NativeData: currency); { dtCurrency }
procedure WriteFloatData(FormatData: string;
  NativeData: extended); { dtFloat }
procedure WriteDateTimeData(FormatData: string;
  NativeData: TDateTime); { dtDate, dtTime and dtDateTime }
procedure WriteIntData(FormatData: string;
  NativeData: integer); { dtInteger }
procedure WriteStrData(FormatData: string;
  NativeData: string); { dtString }

```

There is also a special method called `WriteNullData` (no parameters) that can be called for any field that contains uninitialized or null data. The `FormatData` parameter is used to pass a preformatted string of the data for this field. The `NativeData` parameter is intended to pass the unformatted or raw data of the field. If special formatting is defined in the Rave report then the formatting will be applied to `NativeData`. If no special formatting is defined in the Rave report then the `FormatData` value will be used for printing.

Example:

```

procedure TDataForm.CustomCXNGetRow(Connection: TRvCustomConnection);
begin
  With Connection do begin
    WriteIntData('',DataRecord.IntField);
    WriteStrData('',DataRecord.StrField);
    WriteFloatData('',DataRecord.FloatField);
  end; { with }
end;

```

OnValidateRow event

The `OnValidateRow` event is called for each row and allows you to control whether the current row will be included in the report or not. This is usually the only event that will be defined for non-custom data connections.

Example:

```

procedure TDataForm.CustomCXNValidateRow(Connection: TRvCustomConnection;
var ValidRow: Boolean);
begin
  ValidRow := DataRecord.FloatField >= 0.0;
end;

```

OnRestore event

The `OnRestore` event is called to terminate and restore a data session to its previous state. In this event you can close data files, free resources and restore the state of the data to its state that it was before the `OnOpen` event was called.

Example:

```

procedure TDataForm.CustomCXNRestore(Connection: TRvCustomConnection);

```

```
begin
    CloseFile(DataFile);
end;
```

WARNING

In order for RAVE to be able to see your custom components, you will need to set the visible property to true and you will also need to be **running the application** that has the custom components. Custom components cannot be seen in RAVE while in design time only. So define your custom connections, run your application and then switch to the visual designer, while your application is still running, and then create/refresh your custom connection dataviews and your fields will then show up.

RAVEDEMO

The RaveDemo application included in the Rave\Demos folder has two custom connection master-detail style reports. The RaveDemo includes source and can be examined for how the OnEOF, OnGetCols, OnGetRow and OnOpen events were coded.

6.1.5 Data Connections

Data Connections provide a bridge between the data in your application and a Rave report. The first thing you have to do is choose the type of Rave data connection that you will need. This is determined by the type of database components that you are using. Use the table below to help you decide the best data connection component to use:

Data Connection Component	Best used with ...	How to connect
TRvCustomConnection	Memory arrays, non-database files or non-TDataSet database components	Define events such as OnFirst, OnNext, OnEOF, OnGetCols and OnGetRow.
TRvDataSetConnection	3 rd Party TDataSet descendent database components	Initialize the DataSet property to the TDataSet component
TRvTableConnection	TTable, replacement TTable or its descendents	Initialize the Table property to the TTable component
TRvQueryConnection	TQuery or its descendents	Initialize the Query property to the TQuery component

The Name property of the data connection component is used as the name of the data connection itself. **It is important to use unique names for your data connection components since there can be no duplication across your application.** It is also good practice to include a unique application identifier in your data connection names since data connections are visible from other applications. For example, if your application is called Wizbang Object Wizard you could prepend the letters WOW in front of each data connection name to help insure your data connection names are unique for your application.

The Visible property of the data connection component is used to define whether the data connection is visible to other applications or the end-user version of the Rave visual designer. The default value of False means that the data connection will only be visible to a Rave report being printed from within the same application as the data connection component or from the

programmer version of the Rave visual designer. A value of True for the Visible property means that the data connection will be visible to any Rave report being run from within any application or from the end-user or programmer version of the Rave visual designer. While data connections of this type are available to the end-user designer, they will not be displayed when then end-user attempts to create a new data view unless they belong to the application that started the Rave visual designer. If you want to make data connections from other applications available to your end-users it is best to create a report project with the programmer version of the Rave visual designer with the data views already defined for these external data connections.

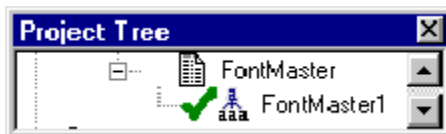
6.1.6 Font Master

Using Font Master

1. Create a new page in the report. Select New Report Page from the Project window.
2. Go to the Project Tree and select the Page.
3. In the Property Panel, type in "FontMaster" in the Name property, while the Page is still selected.



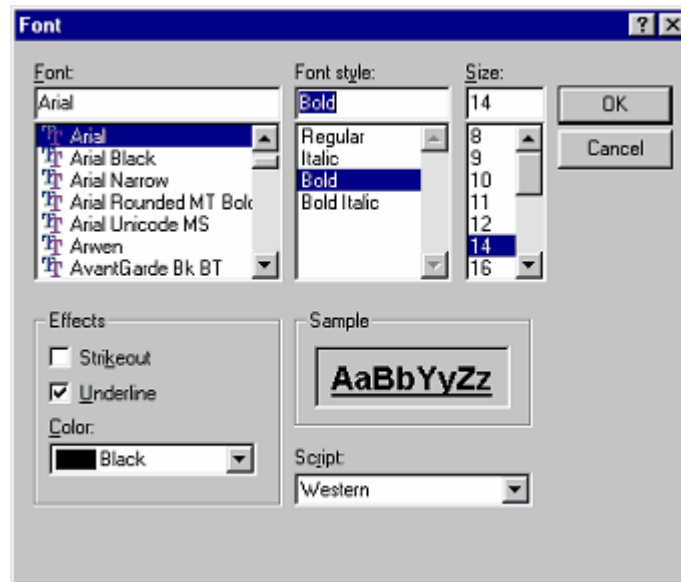
4. Go to the Standard Toolbar. Find the FontMaster component and click on the button. Then click on the Page.



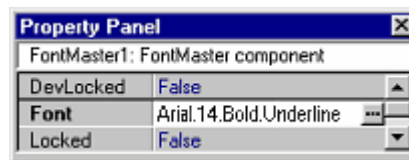
5. Look at the Project Tree Panel and notice that there is a new component underneath the Page called FontMaster1. But, when looking at the Page, there is no visual component. FontMaster is a non-visual component.
6. Make sure the FontMaster component is selected by clicking on it in the Project Tree. A green check mark will appear next to the component in the Project Tree to indicate that it is selected.

FontMaster1: FontMaster	
DevLocked	False
Font	Arial, 10
Locked	False
Name	FontMaster1
Tag	0
Visible	True

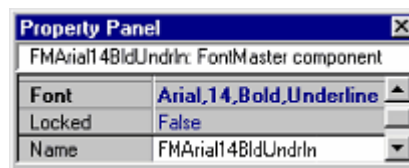
7. While the FontMaster component is selected, look at the Property Panel. In the Font Property, click on the ellipse button (the button with three dots).



8. The Font Dialog will appear after clicking the ellipse button. Use the scroll buttons and check mark boxes to make your desired selections. For this first FontMaster component, select Arial Font, Bold Font style, 14 size, and Underline in Effects.



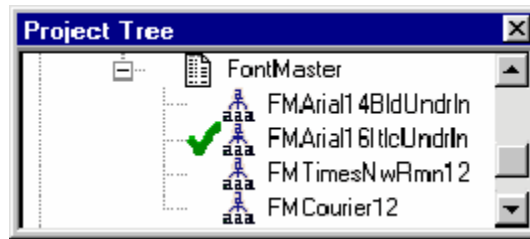
9. Look at the Property Panel, with the component still selected. The Font Property will reflect the selection made in the Font Dialog.



10. Now with the Font Property still selected, scroll to the Name Property and put "FMArial14BldUndrln" in the Name property edit box.



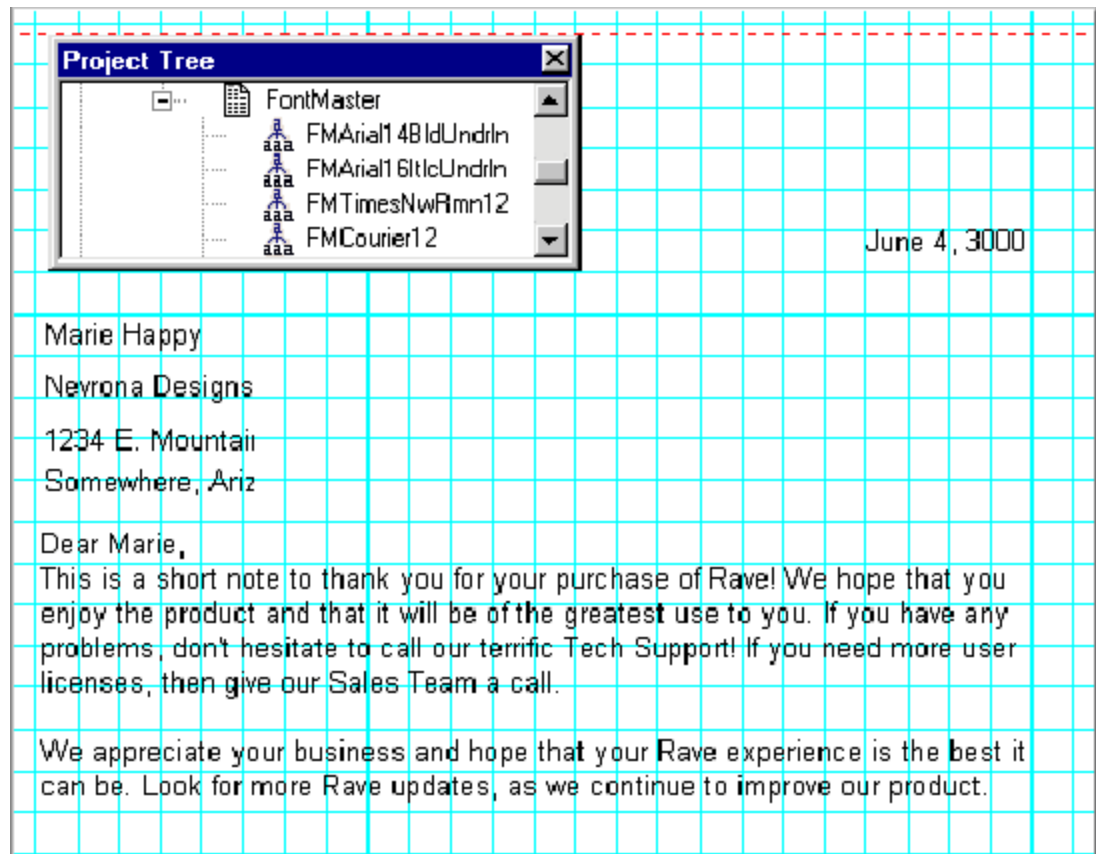
11. Next look at the Project Tree and notice the name change. It becomes very useful and helpful to rename the components in the Project Tree in order to distinguish each component from each other. This is why we demonstrate renaming of the FontMaster component and the Page component.



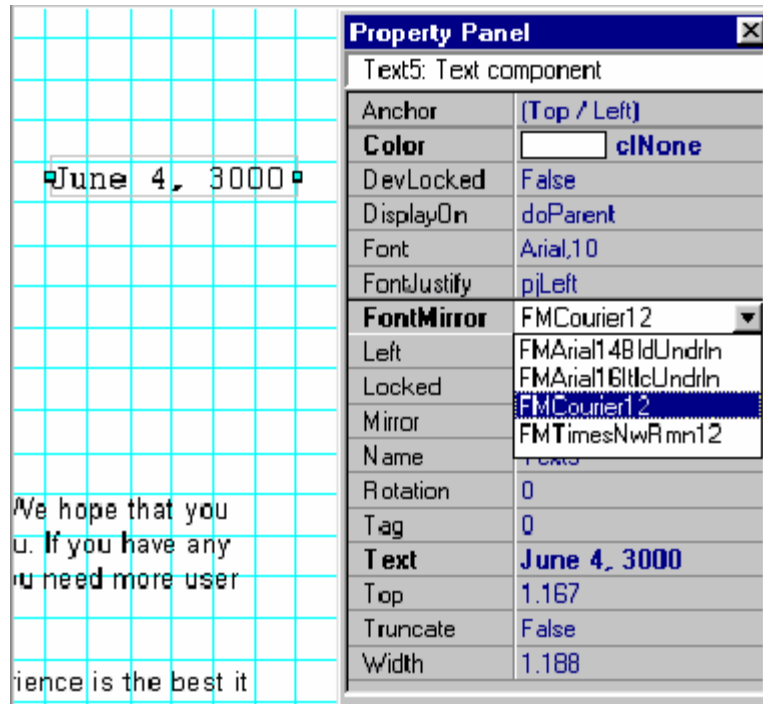
12. Now, complete steps 4 to 11 three times using the following names and settings. FMArial16ItlcUndrln: Arial Font, size 16, Italic, an Underlined. FMTimesNwRmn12: Times New Roman Font, and size 12. FMCourier12: Courier Font, and size 12. To make understanding the renaming of many components, sometimes it is helpful to use a specific naming convention. In these examples we use the following naming convention for the FontMaster component names:



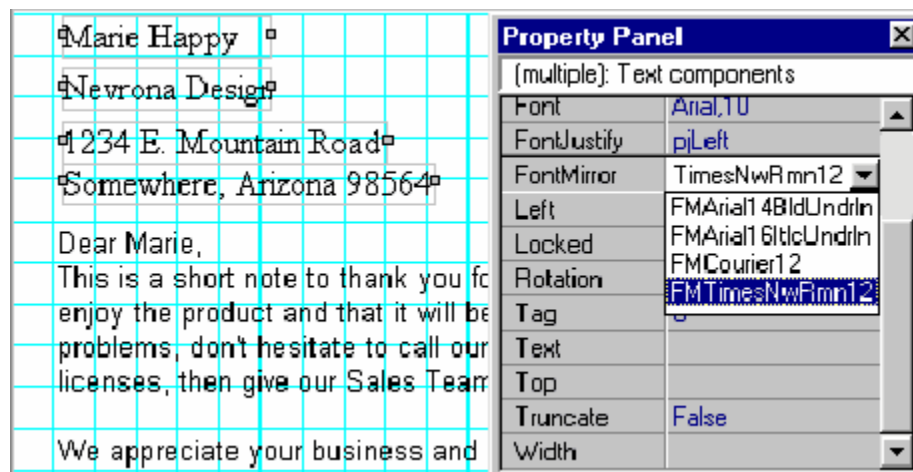
13. Next, drop down five Text components on the page, as well as one Memo component.
14. We will pretend to write a "letter" using components dropped on the Page.



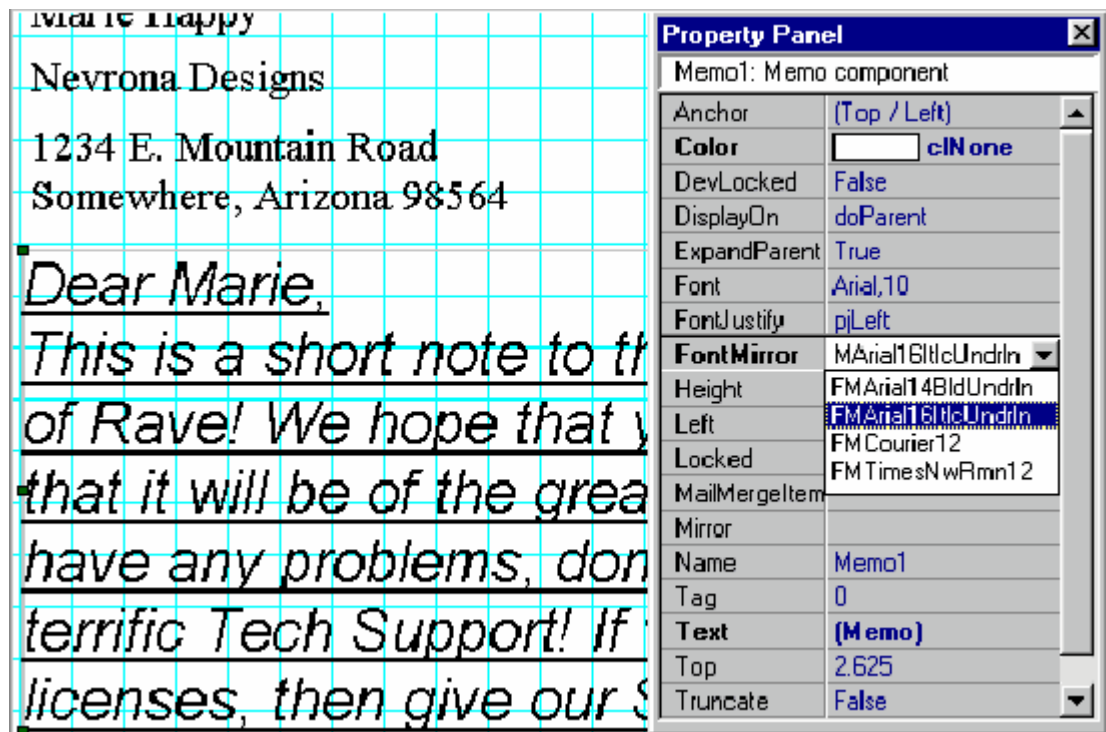
15. Use one Text component to use for a date holder, and use the other four Text components to make an address label. The Memo component can be used for the body of the message. So, fill your components appropriately. For more information on how to fill the components with your own text, see the Property Descriptions Listing, or the section on Standard Components. Feel free to use the alignment tools to get the "letter" components to align correctly. Also, after some of the following steps it may be necessary to resize the text components to display all the text correctly.



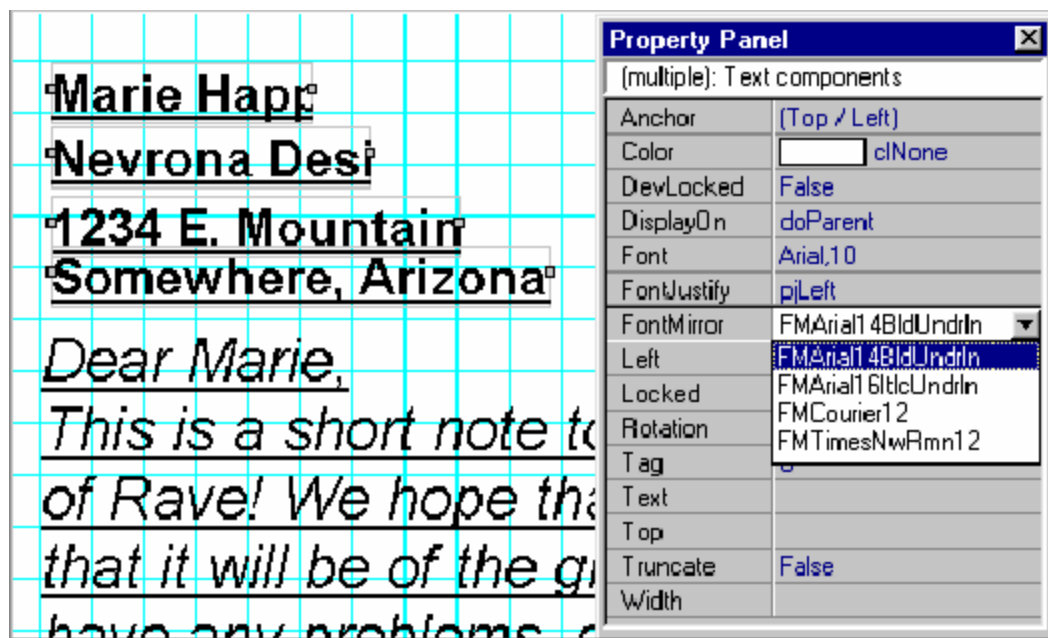
16. Next select the Text component that has the date. While selected, change the FontMirror Property to FMCourier12 by using the drop down menu. Notice that the font of the Text component changed to the pre-set font settings of the FMCourier12.



17. Next select the four address Text components. The Property Panel will show the designation that this is a multiple selection of Text components. In the FontMirror property, choose FMTimesNwRmn12.



18. Select the Memo component, which contains the body of the "letter". While it is selected, choose FMArial16ItlcUndrln in the FontMirror drop down menu.
19. This ends mirroring of fonts to components on the Page. One last thing to cover is what to do when you change your mind about the fonts?

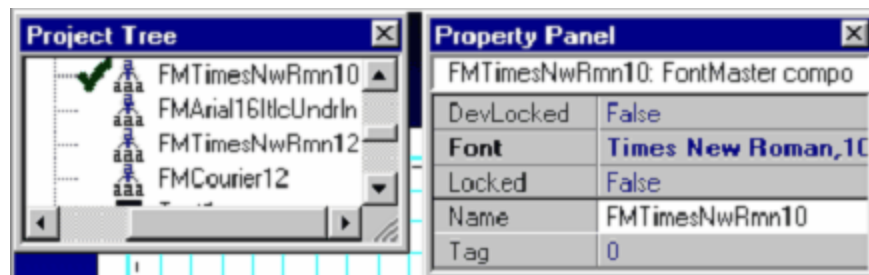


20. Select all the address Text components. Change the FontMirror property to FMArial14BldUndrln. Now, this would be a way to change many Text/Memo components from one font setting to another, without changing each Font property in each component.

21. Sometimes there may be many components linked to one FontMirror component, because they all relate to one specific area of the report. In this example, we can assume that all address headers would be linked to the FMArial14BldUnderln Font Mirror property. So, to change this we would have to redo the one FontMirror property.



22. Select the FMArial14BldUnderln FontMirror component in the Project Tree.
23. In the Property Panel, while the component is still selected, click on the Font property ellipse button.
24. Change the Font Dialog properties to the following: Times New Roman, size 10 font, with no other Font Effects.



25. In the Name property of the FontMirror property, put the following name: FMTimesNwRmn10.

6.2 FAQ

Frequently Asked Questions

6.2.1 Design - Footers

QUESTION:

I need to design a report so that the footer is at the physical bottom of each page, regardless of how much or how little is in the detail section.

ANSWER:

You need to set the Band's "PositionMode" property to pmAbsolute. Then set the "PositionValue" to desired starting location. With pmAbsolute the starting position is measured from the top of the controlling region for this band.

[Band Style Editor](#) [PositionMode](#) [PositionValue](#)

6.2.2 Design - Header on ALL pages

QUESTION:

My report MUST have the same header and footer on EACH page, regardless of how many detail records there are for the master record.

ANSWER:

The solution is put the items you want on every page directly ON THE PAGE not in a band. Pardon the caps. Just wanting to emphasis that you do not have to do anything or everything with bands in a region. The main thing to get across is that Rave is a Page-Based designer and exactly what that means.

For example, if you are doing a form style report, like one of the IRS forms, etc, you don't need to use a region at all. We have seen some RAV files that were ALL form style and none of the reports had a region. Of course, if you are doing a "form" report, you may need to use the "DataCycle" component to move through the data tables. Remember, the "DataCycle" should be the first component in the tree list.

If you are doing a band style report then you use the region component as a "holder" for band components.

If a component has to be in a region, it will not let you put it outside of a region. Try and put a band directly on the page - can not do it. Try other components, Text, DataText, DataMemo etc. and they will go where you like. Of course, you have to assign the DataView and DataField properties on the data driven components as you normally would.

6.2.3 Design - Page Numbers

QUESTION:

How can I put Page Number and Date Information on my Reports?

ANSWER:

Drop a DataText component directly on the Page where you want one of your desired informational items. Start the DataField-Editor by clicking on the ellipse (3 dots) on the right side of the DataField Property. There you can define several things other than the normal "Database" type components. What we want to focus on is the "Report Variables" section. The drop box in that section is where you can select from many of the commonly needed report variables like "CurrentPage", "TotalPages" and a variety of date strings. Down at the bottom of the "Data Text Editor" dialog is a "Data Text" section. Once you have selected a "Report Variable", you have to transfer your selection to the "Data Text" area. This is done with the "Insert Report Var" button. As you transfer your selections, you will note that they appear in the "Data Text" area. If you know the exact format, you can actually type or modify the information directly in the Data Text box. You should end up with something like the following:

For a "Page 1 of 99 Pages" format use:

```
"Page" & Report.CurrentPage & "of" & Report.TotalPages & "Pages"
```

For a "Report printed on: mm/dd/yy" use:

```
"Report printed on:" & Report.DateShort "
```

"+" and "&" signs are for concatenation. The "+" sign is for no spaces and the "&" sign will give you a space between.

6.2.4 Error - Duplicate License

QUESTION:

I am getting a message "Duplicate License Key Found", what should I do?

ANSWER:

The error that you are getting indicates that multiple copies of Rave Reports is being run on the network at the same time with the same serial number. This is not allowed. If you have more than one developer that needs to use Rave then you will need to purchase additional copies of Rave Reports and reinstall each copy with the individual serial numbers. That would allow multiple copies of Rave Reports to be run at the same time.

6.2.5 Events - Getting & Setting Parameters

QUESTION

How would I set / get parameters while using the Rave Language?

SOLUTION

You must first define your [parameter](#)(s) in Rave to be able to set or get parameter(s) while in the Rave Event Editor. Using the parameter "MyParam" as an example, the following code would then set and then use the value from the specified parameter. Remember that parameters must always be a "string" value. This code was tested from the OnGetText event of a text component.

```
// To GET your parameter within an event

Value := RaveProject.GetParam('MyParam');

// To SET your parameter within an event

RaveProject.SetParam('MyParam', 'NewValueAsString');

// Sample event that "increments" a parameter named "LineCount"

{ Event for DataTextCount1.OnGetText }
function DataTextCount1_OnGetText(Self: TRaveDataText; var Value: string);
var
    iCount: Integer;
    sCount: String;
begin
    iCount := StrToInt(RaveProject.GetParam('LineCount'));
    sCount := IntToStr(iCount + 1);
    Value := sCount;
    RaveProject.SetParam('LineCount', sCount);
end OnGetText;
```

6.2.6 Events - Hiding Bands or Components

QUESTION:

How do I hide a band based upon a DataText value?

SOLUTION:

There is a "[Visible](#)" property for almost all of the components, including the band components. The description of this property is that it "*determines whether the component will be printed or not. Typically, set through scripting in the OnBeforePrint or OnBeforeReport events*". You do this by putting a script that sets the "visible" property in a parent of the component you want to control (or in another component). Do not put a script that changes visibility in the same component you want to control as it will run until it is "false" but will not run after that as both the component and event "are no longer visible". This example shows a header band event that controlling the visibility of a footer band.

```
{ Event for HeaderBand.OnBeforePrint }
function Header_OnBeforePrint(Self: TRaveBand);
begin
    if Frac(DvInventoryLastEdit.AsFloat) < 0.25 then
        BandRowFooter.visible := True;
    elseif Frac(DvInventoryLastEdit.AsFloat) > 0.75 then
        BandRowFooter.visible := True;
    else
        BandRowFooter.visible := False;
    end if;
end OnBeforePrint;
```

6.2.7 Font - Dynamically Changing Fonts

QUESTION

How can I dynamically change a Font color, name, size or style?

SOLUTION

Some report designs can have one or more fields that would be enhanced if the you could control the font attributes of those fields to emphasis them when a certain condition occurred. There are a couple of ways to do this in Rave Reports.

1) One choice would be to "mirror" different sections. Each section would probably look very similar and would contain Text or DataText components with the font attributes you want. These sections to be mirrored should be on a Global Page. One section could contain all your components with font attributes set as "Arial" "10 points" "Normal" "Black". Another section would also contain the same components with the font attributes set as "Arial" "10 points" "Bold" "Red".

2) Another choice is to use the [FontMaster](#) component and a Rave event . to control which FontMaster is used. First, drop several FontMaster components on a Global Page. Set one FontMaster name to "FontArial10NormalBlack" with the font attributes set as "Arial" "10 points" "Normal" "Black". Then set the other FontMaster name to "FontArial10BoldRed" with the font attributes set as "Arial" "10 points" "Bold" "Red". The following is an example of an event that will change the FontMirror property of a DataText component to point to the FontMaster you want based upon the contents of a DataField. This could be used in an OnBeforePrint or OnGetText event of your Text or DataText component(s).

```
{ Event for DataTextFishName.OnGetText }
function DataTextFishName_OnGetText(Self: TRaveDataText; var Value: string);
begin
    if UpperCase(BioLifeDVCategory.AsString) = 'SHARK' then
        Self.FontMirror := FontArial10BoldRed;
    else
        Self.FontMirror := FontArial10NormalBlack;
    end if;
end OnGetText;
```

6.2.8 Font - Missing Text

QUESTION:

When I save a report in RTF format, some of the text appears to be chopped off. What should I do?

OR

When I preview my report on screen it does not look the same as the printed output.

ANSWER:

This problem is usually caused by using a font which is not true-type. For example, "MS Sans Serif" is not a True-Type font and often causes a problem. Fonts which are not true-type do not scale very well and will look quite different when viewed on screen as compared with how they will look when printed. For this reason, we do not recommend that you use any fonts that are not true-type fonts. Usually, switching your font(s) to a true-type will solve the problem that you are having.

6.2.9 Font - Overlapping Words

QUESTION:

During preview, I have a problem where the spaces are not correct between words. The words often overlap each other.

ANSWER:

Rave Reports main focus is to output reports as accurately as possible to paper. A preview problem can occur since most printers have very high resolutions when compared to your display screen. Printers are 600 dpi or better while screens are 96 dpi. Please note that 96 does NOT divide into 600 nicely. That means that the preview scaling factors are often fractional values and can cause problems when trying to simulate the printed output on your screen. This can result in the overlapping text problems that you may notice during the report preview process. To minimize this preview problem on your display, make sure you only use True-Type fonts as they scale better. For example, "MS Sans Serif" is NOT a True-Type font and should not be used.

6.2.10 Font - Unicode Support

QUESTION:

I can not make any of my fonts work. How do I solve that?

ANSWER:

The .NET versions of Rave Reports does support composite and Unicode fonts. Rave Reports for .NET support for "symbolic languages" like Japanese, Chinese and Korean is much better than with Windows versions since all strings are stored internally as two byte characters.

The current Windows versions of Rave Reports do not fully support Multi-Byte Character Set (MBCS), composite or Unicode fonts. This is especially true for our render HTML and PDF components. This is primarily due to a widely discussed problem with handling "widestrings" that is deep within the core of all versions of Delphi for Win32. That makes supporting these "symbolic Language" fonts very difficult to impossible.

Rave Reports does have limited support for MBCS (Multi-Byte Character Sets) basically because of the nature of the way the characters are stored. There is some limited support for Japanese and Chinese, basically when they use the special MS Arial Unicode font. Rave Reports has been able to get around part of the display problem for these "symbolic" fonts for the normal print and preview processes. We do this by avoiding the Delphi areas having problems and accessing some Windows functionality. However, both HTML and PDF do not have nor provide any reliable / useable path for our rendering components to use.

CodeGear has announced that they are working on a version of Delphi for Win32 (Tiburon) that will support Unicode. We are looking forward to that release. Once that is released, we plan to work on all of our rendering components, especially HTML and PDF, to resolve any display issues with these "symbolic" fonts.

6.3 Format Codes

The DisplayFormat property allows you to format the value given using a format string. The following format specifiers are supported in the format string:

6.3.1 Alphanumeric Items

The following is a list of the different alphanumeric format codes and what they accomplish for each output type.

Examples:

<u>Format String</u>	<u>123456.78</u>	<u>-123.0</u>	<u>0.5</u>	<u>0.0</u>
#,##0.00	123,456.78	-123.00	0.50	0.00
##	123456.8	-123	0.5	0
\$.00	\$123,456.78	\$-123.00	\$0.50	\$0.00
0.00;(0.00);'-'	123456.78	(123.00)	0.50	-----

Specifier

Represents

0

Digit place holder. If value being formatted has a digit where the '0' appears, then the digit is copied to the output string. Otherwise, a '0' is in the output string.

#

Digit place holder. If value being formatted has a digit where the '#' appears, then the digit is copied to the output string. Otherwise, nothing appears in that position.

.

Decimal point. The first '.' character in the format string determines the location of the decimal separator in the formatted value. The actual character used as the decimal separator in the output string is determined by the Number Format of the International section in the Windows Control Panel.

,

Thousand separator. If format string contains a ',' characters, the output will have thousand separators inserted between each group of three digits to left of decimal point. The actual character used as a thousand separator in the output is determined by the Number Format of the International section in the Windows Control Panel.

E+

Scientific notation. If any of the strings 'E+', 'E-', 'e+', or 'e-' are contained in the format string, the number is formatted using scientific notation. A group of up to four '0' characters can immediately follow the 'E+', 'E-', 'e+', or 'e-' to determine the minimum number of digits in the exponent. The 'E+' and 'e+' formats cause a plus sign to be output for positive exponents and a minus sign to be output for negative exponents. The 'E-' and 'e-' formats output a sign character only for negative exponents.

'xx'/'"xx"

Characters enclosed in single or double quotes are output as-is, and do not affect formatting.

;

Separates sections for positive, negative, and zero numbers in the format string.

The locations of the leftmost '0' before the decimal point in the format string and the rightmost '0' after the decimal point in the format string determine the range of digits that are always present in the output string.

The number being formatted is always rounded to as many decimal places as there are digit placeholder's ('0' or '#') to the right of the decimal point. If the format string contains no decimal point, the value being formatted is rounded to the nearest whole number.

If the number being formatted has more digits to the left of the decimal separator than there are digit placeholder's to the left of the '.' character in the format string, the extra digits are output before the first digit placeholder.

To allow different formats for positive, negative, and zero values, the format string can contain between one and three sections separated by semicolons.

One section: The format string applies to all values.

Two sections: The first section applies to positive values and zeros, and the second section applies to negative values.

Three sections: The first section applies to positive values, the second applies to negative values, and the third applies to zeros.

If the section for negative values or the section for zero values is empty, that is if there is nothing between the semicolons that delimit the section, the section for positive values is used instead.

If the section for positive values is empty, or if the entire format string is empty, the value is formatted using general floating-point formatting with 15 significant digits.

6.3.2 Date Time Items

Items that are either a date or time field can use the following format codes. The format specifiers are not case sensitive. If the format parameter is blank then the value is formatted as if a 'c' specifier had been given. The following format specifiers are supported:

Examples:

dddd, mmmm d, yyyy => Monday, September 21 2004

d mmm yy => 21 Sep 04

<u>Specifier</u>	<u>Displays</u>
c	Displays date using format given by ShortDateFormat global variable, followed by time using format given by LongTimeFormat global variable. The time is not displayed if fractional part of the DateTime value is zero.
d	Displays the day as a number without a leading zero (1-31).
dd	Displays the day as a number with a leading zero (01-31).
ddd	Displays the day as an abbreviation (Sun-Sat) using the strings given by the ShortDayNames global variable.
dddd	Displays the day as a full name (Sunday-Saturday) using the strings given by the LongDayNames global variable.
dddddd	Displays date using format given by the ShortDateFormat global variable.
dddddd	Displays date using format given by the LongDateFormat global variable.
m	Displays month as number without leading zero (1-12). If m specifier immediately follows h or hh specifier, the minute rather than month is displayed.
mm	Displays month as number with leading zero (01-12). If mm specifier immediately follows h or hh specifier, the minute rather than month is displayed.
mmm	Displays the month as an abbreviation (Jan-Dec) using the strings given by the ShortMonthNames global variable.
mmmm	Displays the month as a full name (January-December) using the strings given by the LongMonthNames global variable.
yy	Displays the year as a two-digit number (00-99).
yyyy	Displays the year as a four-digit number (0000-9999).
h	Displays the hour without a leading zero (0-23).
hh	Displays the hour with a leading zero (00-23).
n	Displays the minute without a leading zero (0-59).
nn	Displays the minute with a leading zero (00-59).
s	Displays the second without a leading zero (0-59).
ss	Displays the second with a leading zero (00-59).
t	Displays time using format given by the ShortTimeFormat global variable.
tt	Displays time using format given by the LongTimeFormat global variable.
am/pm	Uses the 12-hour clock for the preceding h or hh specifier, and displays 'am' for any hour before noon, and 'pm' for any hour after noon. The am/pm specifier can use lower, upper, or mixed case, and the result is displayed accordingly.
a/p	Uses the 12-hour clock for the preceding h or hh specifier, and displays 'a' for any hour before noon, and 'p' for any hour after noon. The a/p specifier can use lower, upper, or mixed case, and the result is displayed accordingly.
ampm	Uses the 12-hour clock for the preceding h or hh specifier, and displays the contents of the TimeAMString global variable for any hour before noon, and the

contents of the TimePMString global variable for any hour after noon.

/ Displays date separator character given by DateSeparator global variable.

: Displays time separator character given by TimeSeparator global variable.

'xx'/"xx" Characters enclosed in single or double quotes are displayed as-is, and do not affect formatting.

6.4 Shortcuts

Page Designer or Project Tree

Click	on a component selects that component
Right Click	shows context menu for that component
Shift Alt Click	adds all components of the same type as the component clicked on to the selection list of the current page designer
Shift Ctrl Click	adds all children of clicked component to selection list
Shift Click	on a component toggles the selection for that component. This can be used to select multiple components.

Page Designer Only

Click	in blank area of Page Designer removes selection on all component(s)
Ctrl + arrow key	taps (moves) selected component (s) in direction of arrow key
Ctrl C / Ctrl Ins	copies selection to clipboard
Ctrl Click	centers the design window to location clicked
Ctrl F4	unloads current global page
Ctrl V / Shift Ins	paste clipboard to page designer
Ctrl X / Shift Del	cuts selection to clipboards
Ctrl Shift Alt	then Click-drag to select a group of components within a parent control
Delete	deletes all currently selected component(s)
Escape	changes selection to parent of current component
F9	executes the current report*
F11	toggles between page designer and property panel*
Shift + arrow key	increase size in direction of arrow key
Shift + Up	= decrement height
Shift + Down	= increment height
Shift + Left	= decrement width
Shift + Right	= increment width

Tree View Only

Alt Drag	DataView datafield to designer - creates text component
Alt Drag	selected component to container component in Project Tree - makes the destination component (must be a container component like sections or regions) the parent of all selected components.
Ctrl Drag	DataView DataField to page designer - creates datatext component
Ctrl Drag	component to page designer - creates a mirror of component
Double Click	on Global Page node - loads selected page into page designer
Double Click	on Report node - activates selected report

* ALL of these shortcuts can be modified with the keyboard shortcut assignment feature.

See Also:

Preferences --> [Defaults](#), [Designer](#), [Environment](#), [Printing](#), [Shortcuts](#)

6.5 Variables

The following is a list of pre-defined Report variables that can be used with the DataText component to display several standard items like page numbers or dates..

'CurrentPage' 'FirstPage' 'LastPage' 'RelativePage' 'TotalPages'
'DateInter' 'DateLong' 'DateShort' 'DateUS'
'DriverName' 'PortName' 'PrinterName'
'Time24' 'TimeAMPM' 'TimeLong' 'TimeShort'

See Also:

[Data Text Editor](#), [Parameters](#)

Index

- A -

Adaptable 90
Anchors 90

- B -

BarCode 54
 2 of 5 58
 BarCodeJustify 101
 BarCodeRotation 102
 BarHeight 102
 BarTop 102
 BarWidth 102
 Bottom 105
 Center 106
 Code 128 60
 Code 39 59
 Components 56
 EAN 62
 EAN-128 60
 Extended 115
 Height 120
 Left 124
 PostNet 57
 PrintChecksum 133
 PrintReadable 134
 PrintTop 134
 Right 136
 Top 141
 UCC-128 60
 UPC 61
 UseCheckSum 143
 WideFactor 144
 Width 145
Batch 94
 Calling Pages 95
 Chain Pages 97
 Different 97, 98
 First Page 97
 Odd / Even 98
Border
 Color 103
 Style 104
 Width 104
 WidthType 104

- C -

Component 48
 Band 68
 BarCode 56
 Bitmap 81
 CalcController 69
 CalcOp 70
 CalcText 71
 CalcTotal 72
 Circle 64
 DataBand 73
 DataCycle 74
 DataMemo 75
 DataMirrorSection 76
 DataText 78
 DataView 66
 Drawing 63
 Ellipse 64
 FontMaster 81
 Memo 82
 MetaFile 84
 Page 66
 PageNumInit 84
 ProjectManager 67
 Rectangle 64
 Region 79
 Report 67, 68
 Section 84
 Square 65
 Standard 80
 Text 86
Constants
 cfAbs 135
 cfArcTan 135
 cfCos 135
 cfDec 135
 cfFrac 135
 cfHoursToTime 135
 cfInc 135
 cfMinsToTime 135
 cfNeg 135
 cfNone 135
 cfPercent 135
 cfRandom 135
 cfRound 135
 cfRound1 135
 cfRound2 135
 cfRound3 135
 cfRound4 135
 cfRound5 135
 cfSecsToTime 135

Constants

cfSin	135	msInside	128
cfSqr	135	msWidth	128
cfSqrt	135	pdDefault	114
cfTimeToHours	135	pdHorizontal	114
cfTimeToMins	135	pdSimplex	114
cfTimeToSecs	135	pdVertical	114
cfTrunc	135	pjBlock	117
coAdd	130	pjBock	141
coAverage	130	pjCenter	101, 117, 141
coDiv	130	pjLeft	101, 117, 141
coExp	130	pjRight	101, 117, 141
coGreater	130	pmAbsolute	133
coLesser	130	pmOffset	133
coMod	130	pmOverlay	133
coMul	130	prDefault	135
coSub	130	prDraft	135
cpCodeA	106	prHigh	135
cpCodeB	106	prLow	135
cpCodeC	106	prMedium	135
crAppStart	110	psClear	104, 124
crArrow	110	psDash	104, 124
crCross	110	psDashDot	104, 124
crDefault	110	psDashDotDot	104, 124
crDrag	110	psDot	104, 124
crHandPoint	110	psInsideFrame	104, 124
crHelp	110	psSolid	104, 124
crHourGlass	110	Rot0	102
ctAverage	105	Rot180	102
ctCount	105	Rot270	102
ctMax	105	Rot90	102
ctMin	105	unCM	142
ctSum	105	unInch	142
doAll	114	unMM	142
doParent	114	unPoint	142
doPreviewOnly	114	unUser	142
doPrinterOnly	114	wtPixels	104, 125
dtDateTimeFormat	114	wtPoints	104, 125
dtNumericFormat	114		
fsBDiagonal	116		
fsClear	116		
fsCross	116		
fsDiagCross	116		
fsFDiagonal	116		
fsHorizontal	116		
fsNone	116		
fsSolid	116		
fsVertical	116		
gmCallEach	118		
gmGotoDone	118		
gmGotoNotDone	118		
msBoth	128		
msHeight	128		

- D -

DataField

BCD	49
Blob	49
Boolean	49
Currency	49
Date	49
DateTime	49
Float	49
Graphic	49
Integer	49
Memo	49
String	49
Time	49

Drawing

- Circle 64
- Ellipse 64
- Rectangle 64
- Square 65

- E -

Editor

- Anchor 7
- Band Style 8
- Color 9
- DataText 11
- Event 37
- Font 13
- Fonts 52
- Lines 52
- Page List 94
- Scripting 37

Events 36

- Scripting 40

Example Scripting

- GreenBar 41
- Hiding Components 167
- OnBeforePrint 42
- OnGetText 43
- OnPrint 43
- Parameters 42
- Parameters, Get or Set 167
- Section Component 43
- Visibility 42

- F -

FAQ

- Changing Fonts 168
- Dup License 166
- Footers 165
- Headers 165
- Hiding Components 167
- MBCS 169
- Missing Text 168
- Overlapping Words 169
- Page Numbers 166
- Parameters, Get or Set 167
- Unicode 169

First Glance 3

Format 169

- Alphanumeric 170
- Date 173
- DisplayFormat 170, 173
- Time 173

Functions

Scripting 39

- H -

Hide Components 143

- I -

Introduction 2

- L -

Lesson

- CalcOp 149
- Calculated Fields 153
- Calling Pages 95
- Chain Pages 97
- Custom Data Connection 154
- Data Connection 158
- Different First Page 98
- Different Odd / Even Page 98
- FontMaster 159
- Quick Start 148

- M -

Main Screen 13

Master Detail 33

- N -

Navigation 4

- O -

Output 17

- Execute Report 17
- HTML 24
- PDF 25
- Preferences 18
- Preview 18
- Printer 22
- PRN 23

- P -

Panel

- Project Tree 5, 14
- Property 6

Preferences

- Default 28
- Designer 27
- Environment 26
- Printing 29

Preferences

Shortcuts 31

ProjectTree

DataView Dictionary 16

Global Page 16

Naming Components 17

Report Library 15

Property 100

AllowSplit 100

AlwaysGenerate 100

Anchor 100

AutoSize 101

BandStyle 101

BarCodeJustify 101

BarCodeRotation 102

BarHeight 102

BarTop 102

BarWidth 102

Bin 103

BinCustom 103

BorderColor 103

BorderStyle 104

BorderWidth 104

BorderWidthType 104

Bottom 105

CalcType 105

CalcVar 105

Categories 106

Category 106

Center 106

CodePage 106

Collate 107

Color 107

Columns 107

ColumnSpacing 108

ConnectionName 108

ContainsRTF 108

Controller 108

ControllerBand 109

Copies 109

CountBlanks 109

CountNulls 110

CountValue 110

Cursor 110

DataField 111

DataView 111

Description 111

DesignerHide 112

DestParam 112

DestPIVar 112

DetailKey 113

DevLocked 113

DisplayFormat 113

DisplayOn 114

DisplayType 114

Duplex 114

ExpandParent 115

Extended 115

FieldName 115

FileLink 116

FillColor 116

FillStyle 116

FinishNewPage 117

FirstPage 117

Font 117

FontJustify 117

FontMirror 118

Format 113

FullName 118

GotoMode 118

GotoPage 119

GridLines 119

GridSpacing 119

GroupDataView 120

GroupKey 120

Height 120

HRadius 121

Image 121

InitCalcVar 121

InitDataField 122

InitDataView 122

Initializer 122

InitToFirst 123

InitValue 123

KeepBodyTogether 123

KeepRowTogether 124

Left 124

LineStyle 124

LineWidth 125

LineWidthType 125

Locked 125

LookupDataField 126

LookupDisplay 126

LookupField 126

LookupInvalid 126

MailMergeItems 127

MasterDataView 127

MasterKey 127

MatchSide 128

MaxPages 128

MaxRows 128

MinHeightLeft 129

Mirror 129

Name 129

NullText 130

Operator 130

Property 100
 Orientation 130
 OrphanRows 131
 PageHeight 131
 PageList 131
 PageWidth 131
 PaperSize 132
 Parameters 132
 PIVars 132
 PositionMode 133
 PositionValue 133
 PrintChecksum 133
 Printer 134
 PrintReadable 134
 PrintTop 134
 ReprintLocs 135
 Resolution 135
 ResultFunction 135
 Right 136
 Rotation 136
 RunningTotal 137
 Scr1DataView 138
 Size 137
 SortKey 137
 Src1CalcVar 138
 Src1DataField 138
 Src1Function 139
 Src1Value 139
 StartNewPage 139
 Tag 140
 Text 140
 TextFalse 140
 TextJustify 141
 TextTrue 141
 Top 141
 Truncate 142
 UnitFactor 142
 Units 142
 UseCheckSum 143
 Visible 143
 VRadius 143
 WasteFit 144
 WideFactor 144
 WidowRows 144
 Width 145
 Property Panel 14

- R -

Report
 Adaptable 90
 Master Detail 33
 Simple 32

Report Expert 35

- S -

Scripting 36
 Classes 41
 Event Editor 37
 Events 40
 Functions 39
 Syntax 38
 Scripting Example
 GreenBar 41
 Hiding Components 167
 OnBeforePrint 42
 OnGetText 43
 OnPrint 43
 Parameters 42
 Parameters, Get or Set 167
 Section Component 43
 Visibility 42
 Section
 OnPrint Event 43
 Shortcuts 176

- T -

Toolbar 48
 Alignment 50
 Colors 51
 Designer 52
 Fonts 52
 Lines 52
 Project 53
 Zoom 53

- V -

Variables 177

- W -

Waste Fit
 Adaptable 91
 Wizard 31
 Master Detail 33
 Report Expert 35
 Simple 32

