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
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.
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.


To access the Nevrona Designs newsgroups, create a new server entry in your newsgroup reader with the Host address of 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

Prepared: December 2008 in Arizona
# Table of Contents

## Part I  Introduction
- 1 TechSupport .......................................................... 2

## Part II  Classes
- 1 TBaseReport .......................................................... 4
- 2 TCanvasReport ....................................................... 5
- 3 TDbMemoBuf .......................................................... 6
- 4 TMemoBuf .............................................................. 6
- 5 TRpBarsBase ......................................................... 7
- 6 TRpBaseComponent ............................................... 7
- 7 TRpComponent ....................................................... 7
- 8 TRpRender ............................................................ 8
- 9 TRpRenderCanvas .................................................. 8
- 10 TRpRenderStream ................................................. 9

## Part III  Components
- 1 TRvCustomConnection .......................................... 12
- 2 TRvDataSetConnection ........................................... 13
- 3 TRvNDRWriter ....................................................... 14
- 4 TRvProject ........................................................... 15
- 5 TRvQueryConnection ............................................. 16
- 6 TRvRenderBitmap .................................................. 17
- 7 TRvRenderHTML ................................................... 17
- 8 TRvRenderJPEG .................................................... 18
- 9 TRvRenderMetafile ................................................. 18
- 10 TRvRenderPDF .................................................... 19
- 11 TRvRenderPreview ............................................... 20
- 12 TRvRenderPrinter ............................................... 21
- 13 TRvRenderRTF ..................................................... 23
- 14 TRvRenderText .................................................... 23
- 15 TRvSystem .......................................................... 24
- 16 TRvTableConnection ............................................ 25

## Part IV  Events
- 1 OnAfterClose ...................................................... 28
- 2 OnAfterOpen ....................................................... 28
- 3 OnAfterPrint ........................................................ 28
- 4 OnBeforeClose ..................................................... 29
- 5 OnBeforeOpen ...................................................... 29
- 6 OnBeforePrint ...................................................... 29
- 7 OnCreate ............................................................ 30
- 8 OnDecodeImage ..................................................... 30
- 9 OnDesignerSave .................................................. 31
| 10 | OnDesignerSaveAs | 31 |
| 11 | OnDesignerShow  | 31 |
| 12 | OnDestroy      | 32 |
| 13 | OnEOF          | 32 |
| 14 | OnFirst        | 32 |
| 15 | OnGetCols      | 33 |
| 16 | OnGetRow       | 33 |
| 17 | OnGetSorts     | 33 |
| 18 | OnNewColumn    | 33 |
| 19 | OnNewPage      | 34 |
| 20 | OnNext         | 35 |
| 21 | OnOpen         | 35 |
| 22 | OnPageChange   | 35 |
| 23 | OnPreviewSetup | 36 |
| 24 | OnPreviewShow  | 37 |
| 25 | OnPrint        | 37 |
| 26 | OnPrintFooter  | 37 |
| 27 | OnPrintHeader  | 38 |
| 28 | OnPrintPage    | 39 |
| 29 | OnRestore      | 39 |
| 30 | OnSetFilter    | 40 |
| 31 | OnSetSort      | 40 |
| 32 | OnValidateRow  | 40 |
| 33 | OnZoomChange   | 40 |
| 34 | OverridePreview| 41 |
| 35 | OverrideSetup  | 41 |
| 36 | OverrideStatus | 42 |

**Part V Methods**

| 1  | Abort             | 44 |
| 2  | AbortPage        | 44 |
| 3  | AdjustLine       | 44 |
| 4  | AllowAll         | 45 |
| 5  | AllowPreviewOnly | 45 |
| 6  | AllowPrinterOnly | 46 |
| 7  | Append           | 46 |
| 8  | AppendMemoBuf    | 46 |
| 9  | Arc              | 47 |
| 10 | AssignFont       | 47 |
| 11 | BrushCopy        | 47 |
| 12 | CalcGraphicHeight| 48 |
| 13 | CalcGraphicWidth | 48 |
| 14 | Chord            | 49 |
| 15 | Clear            | 49 |
| 16 | ClearAllTabs     | 49 |
| 17 | ClearColumns     | 50 |
18 ClearRaveBlob ................................................................. 50
19 ClearTabs ........................................................................ 51
20 Close .................................................................................. 51
21 ConstraintHeightLeft ......................................................... 51
22 CopyRect ........................................................................... 52
23 CR ....................................................................................... 52
24 Create (TBaseReport) ........................................................ 52
25 Create (TRpBarsBase) ......................................................... 53
26 CreateBrush ....................................................................... 54
27 CreateFont .......................................................................... 54
28 CreatePen ........................................................................... 55
29 CreatePoint ......................................................................... 55
30 CreateRect .......................................................................... 56
31 Delete .................................................................................. 56
32 Design ................................................................................ 56
33 DesignReport ..................................................................... 57
34 Destroy ............................................................................... 57
35 Draw .................................................................................... 57
36 DrawFocusRect .................................................................... 58
37 Ellipse ............................................................................... 58
38 Empty .................................................................................. 59
39 Execute (TBaseReport) ....................................................... 59
40 Execute (TRvProject) ......................................................... 60
41 ExecuteCustom .................................................................... 60
42 ExecuteReport ..................................................................... 60
43 FillRect ............................................................................... 61
44 Finish .................................................................................. 61
45 FinishTabBox ..................................................................... 61
46 FloodFill ............................................................................ 62
47 FrameRect .......................................................................... 62
48 FreeSaved .......................................................................... 63
49 GetMemoLine ...................................................................... 63
50 GetNextLine ........................................................................ 63
51 GetParam ........................................................................... 64
52 GetReportCategoryList .................................................... 64
53 GetReportList ..................................................................... 65
54 GetTab ............................................................................... 65
55 GotoFooter .......................................................................... 65
56 GotoHeader ......................................................................... 65
57 GotoXY ............................................................................... 66
58 GraphicFieldToBitmap ...................................................... 66
59 Home .................................................................................. 67
60 Insert ................................................................................... 67
61 InsertMemoBuf ................................................................. 67
62 IsValidChar ........................................................................ 68
<table>
<thead>
<tr>
<th></th>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>63</td>
<td>LF</td>
<td>68</td>
</tr>
<tr>
<td>64</td>
<td>LinesLeft</td>
<td>68</td>
</tr>
<tr>
<td>65</td>
<td>LineTo</td>
<td>69</td>
</tr>
<tr>
<td>66</td>
<td>LoadFromFile (TMemoBuf)</td>
<td>69</td>
</tr>
<tr>
<td>67</td>
<td>LoadFromFile (TRvProject)</td>
<td>70</td>
</tr>
<tr>
<td>68</td>
<td>LoadFromStream (TMemoBuf)</td>
<td>70</td>
</tr>
<tr>
<td>69</td>
<td>LoadFromStream (TRvProject)</td>
<td>70</td>
</tr>
<tr>
<td>70</td>
<td>LoadRaveBlob</td>
<td>71</td>
</tr>
<tr>
<td>71</td>
<td>Macro</td>
<td>71</td>
</tr>
<tr>
<td>72</td>
<td>MakeLink</td>
<td>71</td>
</tr>
<tr>
<td>73</td>
<td>MemoHeightLeft</td>
<td>72</td>
</tr>
<tr>
<td>74</td>
<td>MemoLines</td>
<td>72</td>
</tr>
<tr>
<td>75</td>
<td>MemoLinesLeft</td>
<td>73</td>
</tr>
<tr>
<td>76</td>
<td>MoveTo</td>
<td>73</td>
</tr>
<tr>
<td>77</td>
<td>NewColumn</td>
<td>73</td>
</tr>
<tr>
<td>78</td>
<td>NewLine</td>
<td>74</td>
</tr>
<tr>
<td>79</td>
<td>NewPage</td>
<td>74</td>
</tr>
<tr>
<td>80</td>
<td>NewPara</td>
<td>74</td>
</tr>
<tr>
<td>81</td>
<td>NextPage</td>
<td>75</td>
</tr>
<tr>
<td>82</td>
<td>NoPrinters</td>
<td>75</td>
</tr>
<tr>
<td>83</td>
<td>Open</td>
<td>76</td>
</tr>
<tr>
<td>84</td>
<td>Pie</td>
<td>76</td>
</tr>
<tr>
<td>85</td>
<td>Polygon</td>
<td>76</td>
</tr>
<tr>
<td>86</td>
<td>Polyline</td>
<td>77</td>
</tr>
<tr>
<td>87</td>
<td>PopFont</td>
<td>77</td>
</tr>
<tr>
<td>88</td>
<td>PopPos</td>
<td>78</td>
</tr>
<tr>
<td>89</td>
<td>PopTabs</td>
<td>78</td>
</tr>
<tr>
<td>90</td>
<td>PrevPage</td>
<td>79</td>
</tr>
<tr>
<td>91</td>
<td>Print (TBaseReport)</td>
<td>79</td>
</tr>
<tr>
<td>92</td>
<td>Print (TRpBarsBase)</td>
<td>79</td>
</tr>
<tr>
<td>93</td>
<td>PrintBitmap</td>
<td>80</td>
</tr>
<tr>
<td>94</td>
<td>PrintBitmapRect</td>
<td>80</td>
</tr>
<tr>
<td>95</td>
<td>PrintBlock</td>
<td>80</td>
</tr>
<tr>
<td>96</td>
<td>PrintCenter</td>
<td>81</td>
</tr>
<tr>
<td>97</td>
<td>PrintCharJustify</td>
<td>81</td>
</tr>
<tr>
<td>98</td>
<td>PrintData</td>
<td>81</td>
</tr>
<tr>
<td>99</td>
<td>PrintDataStream</td>
<td>82</td>
</tr>
<tr>
<td>100</td>
<td>PrintFimA</td>
<td>82</td>
</tr>
<tr>
<td>101</td>
<td>PrintFimB</td>
<td>83</td>
</tr>
<tr>
<td>102</td>
<td>PrintFimC</td>
<td>83</td>
</tr>
<tr>
<td>103</td>
<td>PrintFooter</td>
<td>83</td>
</tr>
<tr>
<td>104</td>
<td>PrintHeader</td>
<td>84</td>
</tr>
<tr>
<td>105</td>
<td>PrintHeight</td>
<td>84</td>
</tr>
<tr>
<td>106</td>
<td>PrintImageRect</td>
<td>84</td>
</tr>
<tr>
<td>107</td>
<td>PrintJustify</td>
<td>85</td>
</tr>
<tr>
<td>Page</td>
<td>Function</td>
<td>Start Page</td>
</tr>
<tr>
<td>------</td>
<td>-----------------------------------</td>
<td>------------</td>
</tr>
<tr>
<td>108</td>
<td>PrintLeft</td>
<td>86</td>
</tr>
<tr>
<td>109</td>
<td>PrintLines</td>
<td>86</td>
</tr>
<tr>
<td>110</td>
<td>PrintLn</td>
<td>86</td>
</tr>
<tr>
<td>111</td>
<td>PrintMemo</td>
<td>87</td>
</tr>
<tr>
<td>112</td>
<td>PrintPage</td>
<td>87</td>
</tr>
<tr>
<td>113</td>
<td>PrintRight</td>
<td>88</td>
</tr>
<tr>
<td>114</td>
<td>PrintTab</td>
<td>88</td>
</tr>
<tr>
<td>115</td>
<td>PrintXY (TBaseReport)</td>
<td>88</td>
</tr>
<tr>
<td>116</td>
<td>PrintXY (TRpBarsBase)</td>
<td>89</td>
</tr>
<tr>
<td>117</td>
<td>PushFont</td>
<td>89</td>
</tr>
<tr>
<td>118</td>
<td>PushPos</td>
<td>89</td>
</tr>
<tr>
<td>119</td>
<td>PushTabs</td>
<td>90</td>
</tr>
<tr>
<td>120</td>
<td>RecoverPrinter</td>
<td>90</td>
</tr>
<tr>
<td>121</td>
<td>Rectangle</td>
<td>90</td>
</tr>
<tr>
<td>122</td>
<td>RedrawPage</td>
<td>91</td>
</tr>
<tr>
<td>123</td>
<td>RegisterGraphic</td>
<td>91</td>
</tr>
<tr>
<td>124</td>
<td>ReleasePrinter</td>
<td>92</td>
</tr>
<tr>
<td>125</td>
<td>ReplaceAll</td>
<td>92</td>
</tr>
<tr>
<td>126</td>
<td>ReportDescToMemo</td>
<td>93</td>
</tr>
<tr>
<td>127</td>
<td>Reset (TBaseReport)</td>
<td>93</td>
</tr>
<tr>
<td>128</td>
<td>Reset (TMemoBuf)</td>
<td>93</td>
</tr>
<tr>
<td>129</td>
<td>ResetLineHeight</td>
<td>94</td>
</tr>
<tr>
<td>130</td>
<td>ResetPrinter</td>
<td>94</td>
</tr>
<tr>
<td>131</td>
<td>ResetSection</td>
<td>94</td>
</tr>
<tr>
<td>132</td>
<td>ResetTabs</td>
<td>95</td>
</tr>
<tr>
<td>133</td>
<td>RestoreBuffer</td>
<td>95</td>
</tr>
<tr>
<td>134</td>
<td>RestoreFont</td>
<td>95</td>
</tr>
<tr>
<td>135</td>
<td>RestorePos</td>
<td>96</td>
</tr>
<tr>
<td>136</td>
<td>RestoreState</td>
<td>96</td>
</tr>
<tr>
<td>137</td>
<td>RestoreTabs</td>
<td>96</td>
</tr>
<tr>
<td>138</td>
<td>ReuseGraphic</td>
<td>97</td>
</tr>
<tr>
<td>139</td>
<td>RoundRect</td>
<td>97</td>
</tr>
<tr>
<td>140</td>
<td>RTFLoadFromFile</td>
<td>97</td>
</tr>
<tr>
<td>141</td>
<td>RTFLoadFromStream</td>
<td>98</td>
</tr>
<tr>
<td>142</td>
<td>Save</td>
<td>98</td>
</tr>
<tr>
<td>143</td>
<td>SaveBuffer</td>
<td>98</td>
</tr>
<tr>
<td>144</td>
<td>SaveFont</td>
<td>98</td>
</tr>
<tr>
<td>145</td>
<td>SavePos</td>
<td>99</td>
</tr>
<tr>
<td>146</td>
<td>SaveRaveBlob</td>
<td>99</td>
</tr>
<tr>
<td>147</td>
<td>SaveState</td>
<td>100</td>
</tr>
<tr>
<td>148</td>
<td>SaveTabs</td>
<td>100</td>
</tr>
<tr>
<td>149</td>
<td>SaveToFile</td>
<td>100</td>
</tr>
<tr>
<td>150</td>
<td>SaveToStream (TMemoBuf)</td>
<td>100</td>
</tr>
<tr>
<td>151</td>
<td>SaveToStream (TRvProject)</td>
<td>101</td>
</tr>
<tr>
<td>152</td>
<td>SearchFirst</td>
<td>101</td>
</tr>
<tr>
<td>Number</td>
<td>Function</td>
<td>Page</td>
</tr>
<tr>
<td>--------</td>
<td>---------------------------</td>
<td>------</td>
</tr>
<tr>
<td>153</td>
<td>SearchNext</td>
<td>102</td>
</tr>
<tr>
<td>154</td>
<td>SelectBin</td>
<td>102</td>
</tr>
<tr>
<td>155</td>
<td>SelectPaper</td>
<td>102</td>
</tr>
<tr>
<td>156</td>
<td>SelectPrinter</td>
<td>103</td>
</tr>
<tr>
<td>157</td>
<td>SelectReport</td>
<td>103</td>
</tr>
<tr>
<td>158</td>
<td>SetBrush</td>
<td>103</td>
</tr>
<tr>
<td>159</td>
<td>SetColumns</td>
<td>104</td>
</tr>
<tr>
<td>160</td>
<td>SetColumnWidth</td>
<td>104</td>
</tr>
<tr>
<td>161</td>
<td>SetData</td>
<td>105</td>
</tr>
<tr>
<td>162</td>
<td>SetFont</td>
<td>105</td>
</tr>
<tr>
<td>163</td>
<td>SetPageSize</td>
<td>105</td>
</tr>
<tr>
<td>164</td>
<td>SetParam</td>
<td>106</td>
</tr>
<tr>
<td>165</td>
<td>SetPen</td>
<td>106</td>
</tr>
<tr>
<td>166</td>
<td>SetPIVar</td>
<td>107</td>
</tr>
<tr>
<td>167</td>
<td>SetRTF</td>
<td>107</td>
</tr>
<tr>
<td>168</td>
<td>SetTab</td>
<td>107</td>
</tr>
<tr>
<td>169</td>
<td>SetTopOfPage</td>
<td>108</td>
</tr>
<tr>
<td>170</td>
<td>ShadeToColor</td>
<td>109</td>
</tr>
<tr>
<td>171</td>
<td>ShowPrintDialog</td>
<td>109</td>
</tr>
<tr>
<td>172</td>
<td>ShowPrinterSetupDialog</td>
<td>109</td>
</tr>
<tr>
<td>173</td>
<td>SoftLine</td>
<td>110</td>
</tr>
<tr>
<td>174</td>
<td>Start</td>
<td>110</td>
</tr>
<tr>
<td>175</td>
<td>StretchDraw</td>
<td>110</td>
</tr>
<tr>
<td>176</td>
<td>SupportBin</td>
<td>111</td>
</tr>
<tr>
<td>177</td>
<td>SupportCollate</td>
<td>111</td>
</tr>
<tr>
<td>178</td>
<td>SupportDuplex</td>
<td>111</td>
</tr>
<tr>
<td>179</td>
<td>SupportOrientation</td>
<td>111</td>
</tr>
<tr>
<td>180</td>
<td>SupportPaper</td>
<td>112</td>
</tr>
<tr>
<td>181</td>
<td>Tab</td>
<td>112</td>
</tr>
<tr>
<td>182</td>
<td>TabEnd</td>
<td>112</td>
</tr>
<tr>
<td>183</td>
<td>TabStart</td>
<td>113</td>
</tr>
<tr>
<td>184</td>
<td>TabWidth</td>
<td>113</td>
</tr>
<tr>
<td>185</td>
<td>TextRect</td>
<td>114</td>
</tr>
<tr>
<td>186</td>
<td>TextWidth</td>
<td>114</td>
</tr>
<tr>
<td>187</td>
<td>UnregisterGraphic</td>
<td>115</td>
</tr>
<tr>
<td>188</td>
<td>UpdateStatus</td>
<td>115</td>
</tr>
<tr>
<td>189</td>
<td>WriteBCDData</td>
<td>115</td>
</tr>
<tr>
<td>190</td>
<td>WriteBlobData</td>
<td>116</td>
</tr>
<tr>
<td>191</td>
<td>WriteBoolData</td>
<td>116</td>
</tr>
<tr>
<td>192</td>
<td>WriteCurrData</td>
<td>117</td>
</tr>
<tr>
<td>193</td>
<td>WriteDateTime</td>
<td>117</td>
</tr>
<tr>
<td>194</td>
<td>WriteFloatData</td>
<td>117</td>
</tr>
<tr>
<td>195</td>
<td>WriteIntData</td>
<td>118</td>
</tr>
<tr>
<td>196</td>
<td>WriteNullData</td>
<td>118</td>
</tr>
<tr>
<td>197</td>
<td>WriteStrData</td>
<td>119</td>
</tr>
</tbody>
</table>
Part VI  Properties

1  Aborted  ................................................................. 126
2  AccuracyMethod .................................................. 126
3  Active (TRpRender) .............................................. 127
4  Active (TRvProject) .............................................. 127
5  AscentHeight ....................................................... 127
6  BarBottom ............................................................ 128
7  BarCodeJustify .................................................... 128
8  BarCodeRotation .................................................. 129
9  BarHeight ............................................................. 129
10 BarTop ................................................................. 129
11 BarWidth .............................................................. 130
12 BaseReport (TMemoBuf) ........................................ 130
13 BaseReport (TRpBarsBase) ................................. 131
14 BaseReport (TRvSystem) ...................................... 131
15 Bins ...................................................................... 131
16 BKColor ............................................................... 132
17 Bold ..................................................................... 132
18 Bottom ................................................................. 133
19 BottomWaste ...................................................... 133
20 BoxLineColor ........................................................ 133
21 BoxLineColor constants ........................................ 134
22 Buffer ................................................................. 134
23 BufferInc ............................................................. 134
24 CacheDir ............................................................... 135
25 Canvas .................................................................. 135
26 Center .................................................................... 135
27 CheckSum ............................................................ 136
28 CodePage ............................................................. 136
29 Collate ................................................................. 137
30 ColumnEnd .......................................................... 137
<table>
<thead>
<tr>
<th></th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>ColumnLinesLeft</td>
<td>138</td>
</tr>
<tr>
<td>32</td>
<td>ColumnNum</td>
<td>138</td>
</tr>
<tr>
<td>33</td>
<td>Columns</td>
<td>138</td>
</tr>
<tr>
<td>34</td>
<td>ColumnStart</td>
<td>139</td>
</tr>
<tr>
<td>35</td>
<td>ColumnWidth</td>
<td>139</td>
</tr>
<tr>
<td>36</td>
<td>Copies</td>
<td>139</td>
</tr>
<tr>
<td>37</td>
<td>CPI</td>
<td>140</td>
</tr>
<tr>
<td>38</td>
<td>CurrentPage</td>
<td>141</td>
</tr>
<tr>
<td>39</td>
<td>CurrentPass</td>
<td>141</td>
</tr>
<tr>
<td>40</td>
<td>CursorXPos</td>
<td>141</td>
</tr>
<tr>
<td>41</td>
<td>CursorYPos</td>
<td>142</td>
</tr>
<tr>
<td>42</td>
<td>DataSet</td>
<td>142</td>
</tr>
<tr>
<td>43</td>
<td>DefaultDest</td>
<td>142</td>
</tr>
<tr>
<td>44</td>
<td>DescentHeight</td>
<td>143</td>
</tr>
<tr>
<td>45</td>
<td>DriverName</td>
<td>143</td>
</tr>
<tr>
<td>46</td>
<td>DeviceName</td>
<td>143</td>
</tr>
<tr>
<td>47</td>
<td>DisplayName</td>
<td>144</td>
</tr>
<tr>
<td>48</td>
<td>DLLFile</td>
<td>144</td>
</tr>
<tr>
<td>49</td>
<td>FileName</td>
<td>144</td>
</tr>
<tr>
<td>50</td>
<td>Duplex</td>
<td>145</td>
</tr>
<tr>
<td>51</td>
<td>Engine</td>
<td>145</td>
</tr>
<tr>
<td>52</td>
<td>Extended</td>
<td>146</td>
</tr>
<tr>
<td>53</td>
<td>ExtendedText</td>
<td>146</td>
</tr>
<tr>
<td>54</td>
<td>Field</td>
<td>147</td>
</tr>
<tr>
<td>55</td>
<td>FieldAliasList</td>
<td>147</td>
</tr>
<tr>
<td>56</td>
<td>FileName</td>
<td>147</td>
</tr>
<tr>
<td>57</td>
<td>FirstPage</td>
<td>148</td>
</tr>
<tr>
<td>58</td>
<td>FontAlign</td>
<td>148</td>
</tr>
<tr>
<td>59</td>
<td>FontBaseline</td>
<td>149</td>
</tr>
<tr>
<td>60</td>
<td>FontBottom</td>
<td>149</td>
</tr>
<tr>
<td>61</td>
<td>FontCharset</td>
<td>149</td>
</tr>
<tr>
<td>62</td>
<td>FontColor</td>
<td>150</td>
</tr>
<tr>
<td>63</td>
<td>FontHandle</td>
<td>150</td>
</tr>
<tr>
<td>64</td>
<td>FontHeight</td>
<td>151</td>
</tr>
<tr>
<td>65</td>
<td>FontName</td>
<td>151</td>
</tr>
<tr>
<td>66</td>
<td>FontPitch</td>
<td>151</td>
</tr>
<tr>
<td>67</td>
<td>FontRotation</td>
<td>152</td>
</tr>
<tr>
<td>68</td>
<td>Fonts</td>
<td>152</td>
</tr>
<tr>
<td>69</td>
<td>FontSize</td>
<td>153</td>
</tr>
<tr>
<td>70</td>
<td>FontTop</td>
<td>153</td>
</tr>
<tr>
<td>71</td>
<td>FontWidth</td>
<td>154</td>
</tr>
<tr>
<td>72</td>
<td>FrameMode</td>
<td>154</td>
</tr>
<tr>
<td>73</td>
<td>GridHoriz</td>
<td>155</td>
</tr>
<tr>
<td>74</td>
<td>GridPen</td>
<td>155</td>
</tr>
<tr>
<td>75</td>
<td>GridVert</td>
<td>156</td>
</tr>
<tr>
<td>Category</td>
<td>Page</td>
<td></td>
</tr>
<tr>
<td>-------------------------------</td>
<td>------</td>
<td></td>
</tr>
<tr>
<td>Height</td>
<td>156</td>
<td></td>
</tr>
<tr>
<td>IgnoreFileSettings</td>
<td>157</td>
<td></td>
</tr>
<tr>
<td>ImageQuality</td>
<td>157</td>
<td></td>
</tr>
<tr>
<td>Italic</td>
<td>158</td>
<td></td>
</tr>
<tr>
<td>Justify</td>
<td>158</td>
<td></td>
</tr>
<tr>
<td>LastPage</td>
<td>158</td>
<td></td>
</tr>
<tr>
<td>Left</td>
<td>159</td>
<td></td>
</tr>
<tr>
<td>LeftWaste</td>
<td>159</td>
<td></td>
</tr>
<tr>
<td>LineBottom</td>
<td>160</td>
<td></td>
</tr>
<tr>
<td>LineHeight</td>
<td>160</td>
<td></td>
</tr>
<tr>
<td>LineHeightMethod</td>
<td>161</td>
<td></td>
</tr>
<tr>
<td>LineMiddle</td>
<td>161</td>
<td></td>
</tr>
<tr>
<td>LineNum</td>
<td>162</td>
<td></td>
</tr>
<tr>
<td>LinesPerInch</td>
<td>162</td>
<td></td>
</tr>
<tr>
<td>LineTop</td>
<td>163</td>
<td></td>
</tr>
<tr>
<td>LoadDesigner</td>
<td>163</td>
<td></td>
</tr>
<tr>
<td>LocalFilter</td>
<td>163</td>
<td></td>
</tr>
<tr>
<td>LPI</td>
<td>164</td>
<td></td>
</tr>
<tr>
<td>MacroData</td>
<td>164</td>
<td></td>
</tr>
<tr>
<td>MarginBottom</td>
<td>165</td>
<td></td>
</tr>
<tr>
<td>MarginLeft</td>
<td>165</td>
<td></td>
</tr>
<tr>
<td>MarginMethod</td>
<td>166</td>
<td></td>
</tr>
<tr>
<td>MarginPercent</td>
<td>166</td>
<td></td>
</tr>
<tr>
<td>MarginRight</td>
<td>167</td>
<td></td>
</tr>
<tr>
<td>MarginTop</td>
<td>167</td>
<td></td>
</tr>
<tr>
<td>MaxCopies</td>
<td>167</td>
<td></td>
</tr>
<tr>
<td>MaxSize</td>
<td>168</td>
<td></td>
</tr>
<tr>
<td>Memo</td>
<td>168</td>
<td></td>
</tr>
<tr>
<td>MetafileDPI</td>
<td>169</td>
<td></td>
</tr>
<tr>
<td>Monochrome</td>
<td>169</td>
<td></td>
</tr>
<tr>
<td>NoBufferLine</td>
<td>169</td>
<td></td>
</tr>
<tr>
<td>NoCRLF</td>
<td>170</td>
<td></td>
</tr>
<tr>
<td>NoNewLine</td>
<td>170</td>
<td></td>
</tr>
<tr>
<td>NoNTColorFix</td>
<td>171</td>
<td></td>
</tr>
<tr>
<td>NoPrinterPageHeight</td>
<td>171</td>
<td></td>
</tr>
<tr>
<td>NoPrinterPageWidth</td>
<td>171</td>
<td></td>
</tr>
<tr>
<td>OnCompress</td>
<td>172</td>
<td></td>
</tr>
<tr>
<td>Orientation</td>
<td>172</td>
<td></td>
</tr>
<tr>
<td>OriginX</td>
<td>173</td>
<td></td>
</tr>
<tr>
<td>OriginY</td>
<td>173</td>
<td></td>
</tr>
<tr>
<td>OutputFileName</td>
<td>174</td>
<td></td>
</tr>
<tr>
<td>OutputInvalid</td>
<td>174</td>
<td></td>
</tr>
<tr>
<td>OutputName</td>
<td>174</td>
<td></td>
</tr>
<tr>
<td>PageHeight</td>
<td>175</td>
<td></td>
</tr>
<tr>
<td>PageInc</td>
<td>175</td>
<td></td>
</tr>
</tbody>
</table>
121 PagInvalid ................................................................. 175
122 Pages ...................................................................... 176
123 PageWidth ............................................................... 176
124 Papers ...................................................................... 176
125 ParaJustify ............................................................... 177
126 PLVar ...................................................................... 177
127 Port .......................................................................... 178
128 Pos ........................................................................... 178
129 Position ..................................................................... 179
130 PrintChecksum .......................................................... 179
131 PrintEnd .................................................................... 179
132 PrinterIndex .............................................................. 180
133 Printers .................................................................... 180
134 Printing .................................................................... 181
135 PrintReadable ............................................................ 181
136 PrintStart ................................................................... 181
137 PrintTop ..................................................................... 182
138 ProjectFile ................................................................. 182
139 Query ........................................................................ 182
140 RaveBlobDateTime .................................................... 183
141 ReadableHeight .......................................................... 183
142 ReportDateTime .......................................................... 183
143 ReportDesc ............................................................... 184
144 ReportDest ............................................................... 184
145 ReportFullName ........................................................ 184
146 ReportName .............................................................. 185
147 RichEdit ...................................................................... 185
148 Right ......................................................................... 185
149 RightWaste .............................................................. 186
150 RTFField .................................................................... 186
151 RTFText ...................................................................... 186
152 RulerType ................................................................... 186
153 RuntimeVisibility ...................................................... 187
154 ScaleX ....................................................................... 187
155 ScaleY ....................................................................... 188
156 ScrollBox .................................................................... 188
157 SectionBottom ........................................................... 189
158 SectionLeft ............................................................... 189
159 SectionRight ............................................................ 190
160 SectionTop .................................................................. 190
161 Selection .................................................................... 191
162 ServerMode ............................................................... 191
163 ShadowDepth ............................................................ 192
164 Size .......................................................................... 192
165 StatusFormat ............................................................. 192
<table>
<thead>
<tr>
<th>Key</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>166</td>
<td>StatusLabel</td>
<td>193</td>
</tr>
<tr>
<td>167</td>
<td>StatusText</td>
<td>193</td>
</tr>
<tr>
<td>168</td>
<td>StoreRAV</td>
<td>194</td>
</tr>
<tr>
<td>169</td>
<td>Stream</td>
<td>194</td>
</tr>
<tr>
<td>170</td>
<td>StreamMode</td>
<td>195</td>
</tr>
<tr>
<td>171</td>
<td>Strikeout</td>
<td>196</td>
</tr>
<tr>
<td>172</td>
<td>Subscript</td>
<td>196</td>
</tr>
<tr>
<td>173</td>
<td>Superscript</td>
<td>197</td>
</tr>
<tr>
<td>174</td>
<td>SystemFiler</td>
<td>197</td>
</tr>
<tr>
<td>175</td>
<td>SystemOptions</td>
<td>197</td>
</tr>
<tr>
<td>176</td>
<td>SystemPreview</td>
<td>198</td>
</tr>
<tr>
<td>177</td>
<td>SystemPrinter</td>
<td>199</td>
</tr>
<tr>
<td>178</td>
<td>SystemSetups</td>
<td>199</td>
</tr>
<tr>
<td>179</td>
<td>TabColor</td>
<td>199</td>
</tr>
<tr>
<td>180</td>
<td>TabJustify</td>
<td>200</td>
</tr>
<tr>
<td>181</td>
<td>Table</td>
<td>200</td>
</tr>
<tr>
<td>182</td>
<td>TabShade</td>
<td>200</td>
</tr>
<tr>
<td>183</td>
<td>Text (TMemoBuf)</td>
<td>201</td>
</tr>
<tr>
<td>184</td>
<td>Text (TRpBarsBase)</td>
<td>201</td>
</tr>
<tr>
<td>185</td>
<td>TextBKMode</td>
<td>202</td>
</tr>
<tr>
<td>186</td>
<td>TextJustify</td>
<td>202</td>
</tr>
<tr>
<td>187</td>
<td>Title</td>
<td>203</td>
</tr>
<tr>
<td>188</td>
<td>TitlePreview</td>
<td>203</td>
</tr>
<tr>
<td>189</td>
<td>TitleSetup</td>
<td>203</td>
</tr>
<tr>
<td>190</td>
<td>TitleStatus</td>
<td>204</td>
</tr>
<tr>
<td>191</td>
<td>Top</td>
<td>204</td>
</tr>
<tr>
<td>192</td>
<td>TopWaste</td>
<td>204</td>
</tr>
<tr>
<td>193</td>
<td>TotalPasses</td>
<td>205</td>
</tr>
<tr>
<td>194</td>
<td>TransparentBitmaps</td>
<td>205</td>
</tr>
<tr>
<td>195</td>
<td>TruncateText</td>
<td>205</td>
</tr>
<tr>
<td>196</td>
<td>Underline</td>
<td>206</td>
</tr>
<tr>
<td>197</td>
<td>Units</td>
<td>206</td>
</tr>
<tr>
<td>198</td>
<td>UnitsFactor</td>
<td>207</td>
</tr>
<tr>
<td>199</td>
<td>UseBreakingSpaces</td>
<td>207</td>
</tr>
<tr>
<td>200</td>
<td>UseChecksum</td>
<td>207</td>
</tr>
<tr>
<td>201</td>
<td>UseCompression</td>
<td>208</td>
</tr>
<tr>
<td>202</td>
<td>UseSetRange</td>
<td>208</td>
</tr>
<tr>
<td>203</td>
<td>Version</td>
<td>208</td>
</tr>
<tr>
<td>204</td>
<td>WideFactor</td>
<td>209</td>
</tr>
<tr>
<td>205</td>
<td>Width</td>
<td>209</td>
</tr>
<tr>
<td>206</td>
<td>XDPI</td>
<td>209</td>
</tr>
<tr>
<td>207</td>
<td>XPos</td>
<td>210</td>
</tr>
<tr>
<td>208</td>
<td>YDPI</td>
<td>210</td>
</tr>
<tr>
<td>209</td>
<td>YPos</td>
<td>210</td>
</tr>
<tr>
<td>210</td>
<td>ZoomFactor</td>
<td>211</td>
</tr>
</tbody>
</table>
Part VII  Types  216
1  TAccuracyMethod ................................................................. 216
2  TBKMode ............................................................................. 216
3  TFontAlign .......................................................................... 216
4  TLineHeightMethod ............................................................ 217
5  TMacroID ............................................................................ 217
6  TMarginMethod ................................................................. 217
7  TOrientation ........................................................................ 218
8  TPrintJustify ....................................................................... 218
9  TPrintUnits ......................................................................... 218
10  TReportDest ....................................................................... 219
11  TStreamMode ...................................................................... 219
12  TSystemOption .................................................................... 219
13  TSystemOptions ............................................................... 220
14  TSystemSetup ..................................................................... 220
15  TSystemSetups .................................................................... 220
16  TTabJustify ........................................................................ 221

Part VIII  Archived  224
1  Components .......................................................................... 225
   TDbTablePrinter .................................................................. 225
   TDetailShell ......................................................................... 225
   TLabelShell ........................................................................... 226
   TMasterShell ....................................................................... 226
   TReportShell ........................................................................ 227
   TTablePrinter ....................................................................... 227
2  Events .................................................................................. 230
   OnAddTotal event .................................................................. 230
   OnBodyAfter ........................................................................ 230
   OnBodyBefore ....................................................................... 230
   OnBodyFooter ...................................................................... 230
   OnBodyHeader ...................................................................... 231
   OnDetailAfter ....................................................................... 231
   OnDetailBefore ..................................................................... 231
   OnEndOfSection .................................................................... 232
   OnGroupAfter ........................................................................ 232
   OnGroupAfterLast ............................................................... 232
   OnGroupBefore ..................................................................... 232
   OnGroupBeforeFirst ............................................................. 233
   OnGroupFooter ...................................................................... 233
   OnGroupHeader ..................................................................... 234
   OnLabelAfter event ............................................................... 234
   OnLabelBefore ....................................................................... 234
   OnLabelPrint event ............................................................... 235
   OnOverFlow event ............................................................... 235
   OnPageAfter .......................................................................... 235
   OnPageBefore ........................................................................ 236
   OnPageFooter ........................................................................ 236
   OnPageHeader ........................................................................ 236
3 Methods .............................................................................................................................................. 241
  Default ................................................................. 241
  PrintBodyFooter .......................................................... 241
  PrintBodyHeader .......................................................... 241
  PrintDetail ................................................................. 242
  PrintGroupFooter .......................................................... 242
  PrintGroupHeader .......................................................... 242
  PrintPageFooter .......................................................... 243
  PrintPageHeader .......................................................... 243
  PrintReportFooter ......................................................... 244
  PrintReportHeader ......................................................... 244
  PrintRow ................................................................. 244
  SetupSection ............................................................. 245
4 Properties ........................................................................................................................................... 245
  AsFloat ........................................................................ 245
  AsInteger ........................................................................ 246
  Border ........................................................................ 246
  Bottom ........................................................................ 246
  BottomMethod ............................................................. 247
  BoxLines ........................................................................ 247
  Col ........................................................................ 248
  Description .................................................................... 248
  DetailReport .................................................................... 249
  DetailTablePrinter .......................................................... 249
  DisplayFormat .................................................................... 250
  DrawExtents .................................................................... 250
  DrawPen ........................................................................ 251
  DrawPreviewOnly ........................................................ 251
  Enabled ........................................................................ 251
  Font ........................................................................ 252
  FontIndex ........................................................................ 252
  Height .......................................................................... 253
  HeightMethod .................................................................... 253
  isNewPage ....................................................................... 254
  IsReprint ......................................................................... 254
  LabelBrand ....................................................................... 255
  LabelHeight ...................................................................... 255
  LabelShape ...................................................................... 255
  LabelWidth ...................................................................... 256
  Left ........................................................................ 256
  LeftMethod ....................................................................... 257
  MinHeight ....................................................................... 257
  NumAcross ....................................................................... 258
  NumDown ........................................................................ 258
  PrintByRow ....................................................................... 259
  Reprint ........................................................................ 259
  Right ........................................................................ 259
  RightMethod ....................................................................... 259
  Row ........................................................................ 260
  SectionBodyFooter .......................................................... 260
  SectionBodyHeader .......................................................... 261
  SectionGroupFooter .......................................................... 261
  SectionGroupHeader .......................................................... 262
Part IX  By Category  276
  1 Category BarCode ................................................................. 276
  2 Category Column ................................................................. 276
  3 Category Control ................................................................. 277
  4 Category Font ................................................................. 277
  5 Category Graphics ................................................................. 278
  6 Category Label ................................................................. 278
  7 Category Memo ................................................................. 279
  8 Category Misc - ................................................................. 279
  9 Category Position ................................................................. 280
 10 Category Preview ................................................................. 280
 11 Category Printer ................................................................. 281
 12 Category Printing ................................................................. 281
 13 Category Rave - ................................................................. 282
 14 Category Render - ................................................................. 283
 15 Category ReportSection ................................................................. 283
 16 Category ReportSystem ................................................................. 283
 17 Category RTF ................................................................. 283
 18 Category Shell ................................................................. 284
 19 Category TablePrinter - ................................................................. 285
 20 Category Tabs ................................................................. 285
 21 Category Units ................................................................. 286

Part X  Format Codes  288
  1 Alphanumeric Items ................................................................. 288
  2 Date Time Items ................................................................. 289

Index  291
Introduction
1 Introduction

Congratulations! You've made an excellent choice. Rave was designed to give the power of reporting back to the programmer where it belongs. Rave does this by offering a powerful suite of printing components that simplify the task of creating professional reports. Rave does not need any extra .DLL's, .VBX's or .EXE's. Reports can be written entirely with code and compiled in your application for easier distribution and faster execution. Or you can use the visual designer and its components for creating your reports. The visually designed reports are normally stored in one or more file(s) that are saved as an external RAV file. If desired, these report definitions can be saved in the application EXE. Read through this manual and the examples on the accompanying disk and you'll soon be turning nightmare printing jobs into dream reports.

See Rave Reports Developer Guide for information on upgrading from a bundled version of Rave Reports to the BEX version.

1.1 TechSupport

Contacting Nevrona Designs

| Web page: | http://www.nevrona.com |
| Sales: | Please email sales@nevrona.com or call (480) 491-5492. |
| News Groups: | news://news.nevrona.com |
| Mailing Address: | 5301 S Superstition Mountain Dr Ste 104-345 |
| | Gold Canyon AZ 85218-1917 |
| Voice Phone: | (480) 491-5492 |
| Fax: | (602) 296-0189 |

Please include both your Rave version and serial number in all messages to Nevrona. If it is a technical support request then it would also help if you included your operating system and language you are using.

Operating Systems including SP (service pack level)

Windows Linux .NET other

Language including version and service pack

Delphi C++Builder VB C# other
Classes

Chapter
2 Classes

A class, or class type, defines a structure consisting of fields, methods, and properties. Instances of a class type are called objects. The fields, methods, and properties of a class are called its components or members.

2.1 TBaseReport

Unit
RpBase

Hierarchy

TComponent
  TRpComponent
  TRpBaseComponent
  TBaseReport

Description
TBaseReport is the primary ancestor class for all report generation classes. TBaseReport defines the methods, properties and events used by all output components. While most interaction with TBaseReport is for code-based reporting, all of the visual components in Rave are built to use the functionality stored in this cornerstone class of Rave.

Events Derived from TBaseReport

Methods Derived from TBaseReport
Abort, AbortPage, AdjustLine, AllowAll, AllowPreviewOnly, AllowPrinterOnly, Arc, AssignFont, BrushCopy, CalcGraphicHeight, CalcGraphicWidth, Chord, ClearAllTabs, ClearColumns, ClearTabs, CopyRect, CR, Create, CreateBrush, CreateFont, CreatePen, CreatePoint, CreateRect, Destroy, DrawFocusRect, Draw, Ellipse, EndLink, Execute, FillRect, Finish, FinishTabBox, FloodFill, FrameRect, GetMemoLine, GetNextLine, GetTab, GotoFooter, GotoHeader, GotoXY, GraphicFieldToBitmap, Home, LF, LinesLeft, LineTo, Macro, MakeLink, MemoLines, MoveTo, NewColumn, NewLine, NewPage, NoPrinters, Pie, Polygon, Polyline, PopFont, PopPos, PopTabs, Print, PrintBitmap, PrintBitmapRect, PrintBlock, PrintCenter, PrintCharJustify, PrintData, PrintDataStream, PrintFooter, PrintHeader, PrintImageRect, PrintJustify, PrintLeft, PrintLn, PrintMemo, PrintRight, PrintTab, PrintXY, PushFont, PushPos, PushTabs, RecoverPrinter, Rectangle, RegisterGraphic, ReleasePrinter, Reset, ResetLineHeight, ResetPrinter, ResetSection, ResetTabs, RestoreFont, RestorePos, RestoreTabs, ReuseGraphic, RoundRect, SaveFont, SavePos, SaveTabs, SelectBin, SelectPaper, SelectPrinter, SetBrush, SetColumns, SetColumnWidth, SetFont, SetPaperSize, SetPen, SetPIVar, SetTab, SetTopOfPage, ShadeToColor, ShowPrintDialog, ShowPrinterSetupDialog, Start, StartLink, StretchDraw, SupportBin, SupportCollate, SupportDuplex, SupportOrientation, SupportPaper, Tab, TabEnd, TabStart, TabWidth, TextRect, TextWidth, UnregisterGraphic, UpdateStatus, XD2U, XI2D, XI2U, XU2D, XU2I, YD2I, YD2U, YI2D, YI2U, YU2D, YU2I

Properties Derived from TBaseReport
Abort, AccuracyMethod, AscentHeight, Bins, BKColor, Bold, BottomWaste, BoxLineColor, Canvas, Collate, ColumnEnd, ColumnLinesLeft, ColumnNum, Columns, ColumnStart, ColumnWidth, Copies, CurrentPage, CurrentPass, CursorXPos, CursorYPos, DescentHeight, DeviceName, DevMode, DriverName, Duplex, FileName, FirstPage, FontAlign, FontBaseline, FontBottom, FontCharset, FontColor, FontHandle, FontHeight, FontName, FontPitch, FontRotation, Fonts, FontSize, FontTop, FontWidth, FrameMode, GridVert, Italic, LastPage, LeftWaste, LineBottom, LineHeight, LineHeightMethod, LineMiddle, LineNum, LinesPerInch, LineTop, MacroData, MarginBottom, MarginLeft, MarginRight, MarginTop, MaxCopies, NoBufferLine, NoNTColorFix, NoPrinterPageHeight, NoPrinterPageWidth, Orientation, OriginX, OriginY, OutputInvalid, OutputName, PageHeight, PagelInvalid, PageWidth, Papers, PIVar, Port, PrinterIndex, Printers, Printing, ReportDateTime, RightWaste, ScaleX, ScaleY, SectionBottom, SectionLeft, SectionRight, SectionTop, Selection, ShadowDepth, StatusFormat, StatusLabel, StatusText, Stream, StreamMode, Strikeout, Subscript, Superscript, TabColor, TabJustify, TabShade, TextBKMode, Title, TopWaste, TotalPages
2.2 TCanvasReport

Properties Derived from TRpComponent

Version

TCanvasReport attaches many of the abstract methods in TBaseReport to TCanvas methods and is used by all output components that write to a canvas. TRVRenderPrinter (printer canvas) and TRVRenderPreview (preview canvas) are two examples of components that descend from TCanvasReport.

Events Derived from TBaseReport


Methods Derived from TBaseReport

Abort, AbortPage, AdjustLine, AllowAll, AllowPreviewOnly, AllowPrinterOnly, Arc, AssignFont, BrushCopy, CalcGraphicHeight, CalcGraphicWidth, Chord, ClearAllTabs, ClearColumns, ClearTabs, CopyRect, CR, Create, CreateBrush, CreateFont, CreatePen, CreatePoint, CreateRect, Destroy, DrawFocusRect, Draw, Ellipse, Execute, FillRect, Finish, FinishTabBox, FloodFill, FrameRect, GetMemoLine, GetNextLine, GetTab, GotoFooter, GotoHeader, GotoXY, GraphicFieldToBitmap, Home, LF, LinesLeft, LineTo, Macro, MemoLines, MoveTo, NewColumn, NewLine, NewPage, NoPrinters, Pie, Polygon, Polyline, PopFont, PopPos, PopTabs, Print, PrintBitmap, PrintBitmapRect, PrintBlock, PrintCenter, PrintCharJustify, PrintData, PrintDataStream, PrintFooter, PrintHeader, PrintImageRect, PrintJustify, PrintLeft, PrintLn, PrintMemo, PrintRight, PrintTab, PrintXY, PushFont, PushPos, PushTabs, RecoverPrinter, Rectangle, RegisterGraphic, ReleasePrinter, Reset, ResetLineHeight, ResetPrinter, ResetSection, ResetTabs, RestoreFont, RestorePos, RestoreTabs, ReuseGraphic, RoundRect, SaveFont, SavePos, SaveTabs, SelectBin, SelectPaper, SelectPrinter, SetBrush, SetColumns, SetColumnWidth, SetFont, SetPaperSize, SetPen, SetPIVar, SetTab, SetTopOfPage, ShadeToColor, ShowPrintDialog, ShowPrinterSetupDialog, Start, StretchDraw, SupportBin, SupportCollate, SupportDuplex, SupportOrientation, SupportPaper, Tab, TabEnd, TabStart, TabWidth, TextRect, TextWidth, UnregisterGraphic, UpdateStatus, X2D, X2D, X2D, X2D, X2D, X2D, X2D, Y2D, Y2D, Y2D, Y2D, Y2D, Y2D, Y2D, Y2D

Properties Derived from TBaseReport

AbortOpt, AccuracyMethod, AscentHeight, Bins, BKColor, Bold, BottomWaste, BoxLineColor, Canvas, Collate, ColumnEnd, ColumnLinesLeft, ColumnNum, Columns, ColumnStart, ColumnWidth, Copies, CurrentPage, CurrentPass, CursorXPos, CursorYPos, DescentHeight, DeviceName, DevMode, DriverName, Duplex, FileName, FirstPage, FontAlign, FontBaseline, FontBottom, FontCharSet, FontColor, FontHandle, FontHeight, FontName, FontPitch, FontRotation, Fonts, FontSize, FontTop, FontWidth, FrameMode, GridVert, Italic, LastPage, LeftWaste, LineBottom, LineHeight, LineHeightMethod, LineMiddle, LineNum, LinesPerInch, LineTop, MacroData, MarginBottom, MarginLeft, MarginRight, MarginTop, MaxCopies, NoBufferLine, NoNTColorFix, NoPrinterPageHeight, NoPrinterPageSize, Orientation, OriginX, OriginY, OutputInvalid, OutputName, PageHeight, PagelInvalid, PageWidth, Papers, PIVar, Port, PrinterIndex, Printers, Printing, ReportDateTime, RightWaste, ScaleX, ScaleY, SectionBottom, SectionLeft, SectionRight, SectionTop, Selection, ShadowDepth, StatusFormat, StatusLabel, StatusText, Stream, StreamMode, Strikeout, Subscript, Superscript, TabColor, TabJustify, TabShade, TextBKMode, Title, TopWaste, TotalPages, TransparentBitmaps, TruncateText, Underline, Units, UnitsFactor, XDPI, XPos, YDPI, YPos
2.3 **TDbMemoBuf**

**Unit**
RpDBUtil

**Hierarchy**

```
TComponent
  |__ TRpComponent
  |__ TMemoBuf
  |   |__ TDbMemoBuf
```

**Description**
This class adds TMemoField processing to the TMemoBuf class through the Field and RTFField properties.

**Methods Derived from TMemoBuf**
- Append
- AppendMemoBuf
- ConstraintHeightLeft
- Delete
- Empty
- FreeSaved
- InsertMemoBuf
- Insert
- LoadFromFile
- LoadFromStream
- MemoHeightLeft
- MemoLinesLeft
- PrintHeight
- PrintLines
- ReplaceAll
- Reset
- RestoreBuffer
- RestoreState
- RTFLoadFromFile
- RTFLoadFromStream
- SaveBuffer
- SaveState
- SaveToStream
- SearchFirst
- SearchNext
- SetData

**Properties Derived from TMemoBuf**
- BaseReport
- Buffer
- BufferInc
- Field
- Justify
- MaxSize
- Memo
- NoCRLF
- NoNewLine
- Pos
- PrintEnd
- PrintStart
- RichEdit
- RTFField
- RTFText
- Size
- Text

2.4 **TMemoBuf**

**Unit**
RpMemo

**Hierarchy**

```
TComponent
  |__ TRpComponent
  |__ TMemoBuf
```

**Description**
TMemoBuf provides access to the code based word wrapping functionality of Rave. TMemoBuf allows text to be loaded into it via several different properties and methods. Output can then be processed using methods such as PrintLines or PrintHeight.

**Methods Derived from TMemoBuf**
- Append
- AppendMemoBuf
- ConstraintHeightLeft
- Delete
- Empty
- FreeSaved
- InsertMemoBuf
- Insert
- LoadFromFile
- LoadFromStream
- MemoHeightLeft
- MemoLinesLeft
- PrintHeight
- PrintLines
- ReplaceAll
- Reset
- RestoreBuffer
- RestoreState
- RTFLoadFromFile
- RTFLoadFromStream
- SaveBuffer
- SaveState
- SaveToStream
- SearchFirst
- SearchNext
- SetData

**Properties Derived from TMemoBuf**
- BaseReport
- Buffer
- BufferInc
- Field
- Justify
- MaxSize
- Memo
- NoCRLF
- NoNewLine
- Pos
- PrintEnd
- PrintStart
- RichEdit
- RTFField
- RTFText
- Size
- Text
2.5  **TRpBarsBase**

Unit
RpBars

Hierarchy

```
TObject
  TRpBarsBase
```

Description
This is the base class for all bar code output classes and provides basic bar processing and drawing functionality.

Methods Derived from TRpBarsBase
```
Create, IsValidChar, Print, PrintFimA, PrintFimB, PrintFimC, PrintXY
```

Properties Derived from TRpBarsBase
```
BarBottom, BarCodeJustify, BarCodeRotation, BarHeight, BarTop, BarWidth, BaseReport, Bottom, Center, CheckSum, CodePage, Extended, ExtendedText, Height, Left, Position, PrintChecksum, PrintReadable, PrintTop, ReadableHeight, Right, Text, TextJustify, Top, UseCheckSum, WideFactor, Width
```

2.6  **TRpBaseComponent**

Unit
RpBase

Hierarchy

```
TComponent
  TRpComponent
    TRpBaseComponent
```

Description
TRpBaseComponent is the ancestor class for all output related components in Rave. Non-output related components will descend from TRpComponent instead of TRpBaseComponent.

Property Derived from TRpComponent
```
Version
```

2.7  **TRpComponent**

Unit
RpDefine

Hierarchy

```
TComponent
  TRpComponent
```

Description
This is the base class for all Rave components. TRpComponent defines the Version property.

Properties Derived from TRpComponent
```
Version
```
2.8 TRpRender

Unit
RpRender

Hierarchy
TComponent
   | TRpComponent
   |   | TRpRender
Description
This is the base class for all rendering components and provides basic connectivity to the NDR conversion functions and output methods.

Properties Derived from TRpRender
Active, BufferDocument, CacheDir, DisplayName, ImageQuality, MetafileDPI, OnCompress, ServerMode, UseCompression

Properties Derived from TRpComponent
Version

2.9 TRpRenderCanvas

Unit
RpRenderCanvas

Hierarchy
TComponent
   | TRpComponent
   |   | TRpRender
   |   |   | TRpRenderStream
   |   |   |   | TRpRenderCanvas
Description
This class provides basic connectivity to the output methods to a TCanvas object. TRvRenderPrinter and TRvRenderPreview descend from this class.

Derived from TRpRender
Active, CacheDir, DisplayName, ImageQuality, MetafileDPI, OnCompress, ServerMode, UseCompression

Derived from TRpComponent
Version
2.10 TRpRenderStream

Unit
RpRender

Hierarchy

TComponent
  TRpComponent
  TRpRender
  TRpRenderStream

Description
This class provides basic streaming functionality to the basic TRpRender class.

Derived from TRpRender
Active, CacheDir, DisplayName, ImageQuality, MetafileDPI, OnCompress, ServerMode, UseCompression

Derived from TRpComponent
Version
Components
3 Components

A component is defined as something placed on the page, such as a barcode, line, region, shape, etc. The components available in Rave can be found on any of the component toolbars (e.g., Standard, Drawing, Report and Barcode).

The toolbars are made available by clicking on the Tools menu followed by the Toolbars menu. The available toolbars will then be shown in another submenu and will have check marks showing the toolbars that are currently visible. Once a component toolbar is active and selected, a component can be selected and placed on the page. The Page is a special base component, and more details are given in the Page Designer chapter.

Special properties are associated with each component. These component properties can be seen using the Property Panel. Set the properties of each component to the desired setting by either typing the setting in a text dialog, using a drop down menu, or by using the special ellipse (...) button to get to the property dialog box.

There are many properties associated with each component, but don’t be intimidated by the number of properties. The properties are there to allow adjustment for a component's behavior and in many cases the default settings are adequate. Also, please note that the number of properties listed with each component may vary depending on the user level that has been set under preferences. To adjust the user's level, please visit the environment tab in the preferences dialog. See the Preferences chapter for more details.

Since there are many properties associated with each component, this chapter will focus mainly on the component toolbars rather than their associated properties. Property details are listed in Appendix D. The current chapter provides a good overview of what each component toolbar does without too much detail about the property specifics. Do note that many components share common properties, so once the common properties are learned for one component, they can be applied to other properties.

Components are also defined by their relationship relative to other components. This relationship is defined by a parent-child relationship. When a text component is placed on the page, the parent is the page component and the child is the text component. Another way to look at it is that the page contains the text component, thus the parent component contains the child component.

The parent-child relationship also extends into the positioning of the components. All positions are relative to the upper left corner of the parent, thus the Left property and Top property are used to define the relative position of a component. If the parent is like a Section component, which can contain any number of other components; then as the parent component is moved around, it’s children components will move accordingly. If the parent component is deleted, all of its children will also be deleted.

3.1 TRvCustomConnection

Unit

\[
\text{RpCon}
\]

Hierarchy

```
TComponent
    | TRpComponent
    | TRvCustomConnection
```

Object Inspector

```
properties

FieldAliasList:  (TStrings)
LocalFilter:     True
Name:            TRvCustomConnection1
RuntimeVisibility: rTDeveloper
Tag:             0
Version:         6.0.3 [VCL7]
All shown
```
Description
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.

NOTE:
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. 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).

Properties from TRvCustomConnection
FieldAliasList LocalFilter RuntimeVisibility

Properties from TRpComponent
Version

Methods from TRvCustomConnection
WriteBCDData WriteBlobData WriteBoolData WriteCurrData WriteDateTime WriteFloatData WriteIntData WriteNullData WriteStrData

Events from TRvCustomConnection

3.2 TRvDataSetConnection

Unit
RpConDS

Hierarchy
TComponent
   TRpComponent
      TRvCustomConnection
         TRvDataSetConnection

Description
Data connection components - Rave uses data from your application. This is accomplished with data connection components, TRvCustomConnection, TRvDataSetConnection, TRvTableConnection and TRvQueryConnection to provide a bridge between the data in your application and the Rave visual components.

TRvCustomConnection 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.

TRvTableConnection is to be used specifically with TTable components or their descendent's respectively.

TRvQueryConnection is to be used specifically with TQuery components or their descendent's respectively.
Events Derived from TRvCustomConnection

Methods Derived from TRvCustomConnection
WriteBCDData, WriteBlobData, WriteBoolData, WriteCurrData, WriteDateTime, WriteFloatData, WriteIntData, WriteNullData, WriteStrData

Properties Derived from TRvDataSetConnection
DataSet

Properties Derived from TRvCustomConnection
FieldAliasList, LocalFilter, RuntimeVisibility

Properties Derived from TRpComponent
Version

3.3 TRvNDRWriter

Unit

RPFIler

Hierarchy

TComponent
| TRpComponent
| TRpBaseComponent
| TBaseReport
| TRvNDRWriter

Description

The TRvNDRWriter component is used in conjunction with TRvRenderPrinter and TRvRenderPreview to store a report in a special binary format until it is ready to be printed or previewed.

Properties and Events

TRvNDRWriter has properties and events to control file output. AccuracyMethod determines the way that strings are output for more accurate print preview. FileName is the file that will be created if StreamMode is anything other than smUser. Use smFile for large reports (>10 pages or lots of bitmaps) and smMemory for smaller reports (< 10 pages). To send a report to a file call the Execute method.

NOTE:
The binary file that TRvNDRWriter creates does not contain actual printer commands (such as PCL) but rather a custom tokenized version of the report. This format is not officially documented but the source code that writes the file is located in RpBASE.PAS and RpFILER.PAS and the source code that reads the file is located in RpPRINT.PAS. These files should be located in \RAVE\SOURCE.

Events Derived from TBaseReport
3.4 TRvProject

The TRvProject component is the key to providing access to the visual reports you create with Rave. Normally you will have a single TRvProject component in your application, although you can have more if necessary. The ProjectFile property defines the report project file that your application uses to hold the report definitions. This file will have an extension of .RAV and even though it is a single file, it can contain as many report definitions as you need. When the Open method of TRaveReport is called, this report project file will be loaded into memory to prepare for printing or end-user design changes. You should make sure that the Close method is called when you no longer need the report project or before you close your application. If any changes are made to the report project you can save them by calling the Save method. TRvProject also has several properties and methods, such as SelectReport, GetReportList, ReportDescToMemo, ReportDesc, ReportName and ReportFullName to make it easy to create an efficient interface for your users. See the RAVEDEMO project for a good example of how to define a Rave interface.
The Engine property of TRvProject allows you to define an alternate output engine to be used. This allows you
to output Rave reports through the Text, RTF or HTML filer components or to define custom setup and
preview screens through the TRvSystem component.

**TRvProject Events**


**TRvProject Methods**


**TRvProject Properties**

*Active, DLLFile, Engine, LoadDesigner, ProjectFile, RaveBlobDateTime, ReportDesc, ReportFullName, ReportName, StoreRAV*

**TRpComponent Properties**

*Version*

### 3.5 TRvQueryConnection

**Unit**

![RpConBDE](image)

**Hierarchy**

```
TRvCustomDataSetConnection
  └── TRvQueryConnection
```

**Description**

Data connection components - Rave uses data from your application. This is accomplished with data
collection components, TRvCustomConnection, TRvDataSetConnection, TRvTableConnection and
TRvQueryConnection to provide a bridge between the data in your application and the Rave visual
components.

**TRvCustomConnection** 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.

**TRvTableConnection** is to be used specifically with **TTable** components or their descendent's respectively.

**TRvQueryConnection** is to be used specifically with **TQuery** components or their descendent's respectively.

**Properties Derived from TRvQueryConnection**

*Query*
3.6 **TRvRenderBitmap**

Unit

![BMP](image1.png) RvRenderBitmap

Hierarchy

TComponent

<table>
<thead>
<tr>
<th>TRpComponent</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRpRender</td>
</tr>
<tr>
<td>TRpRenderStream</td>
</tr>
<tr>
<td>TRvRenderBitmap</td>
</tr>
</tbody>
</table>

Description

TRvRenderBitmap will convert an NDR stream or file to a bitmap graphic image format. Each page in the report will be generated to a separate formatted file.

Properties Derived from TRpRender

*Active*, *DisplayName*, *FileExtension*, *ImageDPI*

Properties Derived from TRpComponent

*Version*

---

3.7 **TRvRenderHTML**

Unit

![HTML](image2.png) RvRenderHTML

Hierarchy

TComponent

<table>
<thead>
<tr>
<th>TRpComponent</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRpRender</td>
</tr>
<tr>
<td>TRpRenderStream</td>
</tr>
<tr>
<td>TRvRenderHTML</td>
</tr>
</tbody>
</table>

Description

TRvRenderHTML will convert an NDR stream or file to HTML format. Each page in the report will be generated to an HTML 4.0 formatted file. Text, graphic, line and rectangle objects are supported.

Properties Derived from TRpRender

*Active*, *CacheDir*, *DisplayName*, *FileExtension*, *OnDecodeImage*, *ServerMode*, *UseBreakingSpaces*

Properties Derived from TRpComponent

*Version*

removed

*ImageQuality*, *MetafileDPI*, *OnCompress*, *UseCompression*
3.8  TRvRenderJPEG

Unit

RvRenderJPEG

Hierarchy

TComponent

TRpComponent

TRpRender

TRpRenderStream

TRvRenderJPEG

Description

TRvRenderJPEG will convert an NDR stream or file to a JPEG graphic image format. Each page in the report will be generated to a separate formatted file.

Properties Derived from TRpRender

Active, CompressionQuality, DisplayName, FileExtension, ImageDPI

Properties Derived from TRpComponent

Version

3.9  TRvRenderMetafile

Unit

RvRenderMetafile

Hierarchy

TComponent

TRpComponent

TRpRender

TRpRenderStream

TRvRenderMetafile

Description

TRvRenderMetafile will convert an NDR stream or file to a Windows Metafile graphic image format. Each page in the report will be generated to a separate formatted file.

Properties Derived from TRpRender

Active, DisplayName, Enhanced, FileExtension, ImageDPI

Properties Derived from TRpComponent

Version
3.10 TRvRenderPDF

Unit

RvRenderPDF

Hierarchy

TComponent
  TRpComponent
  TRpRender
  TRpRenderStream
  TRvRenderPDF

Description
TRvRenderPDF will convert an NDR stream or file to PDF format. Text, graphic, line and shape (rectangle, ellipse) objects are supported.

Properties Derived from TRpRender
Active, BufferDocument, DisableHyperlinks, DisplayName, DocInfo, EmbedFonts, FileExtension, FontEncoding, ImageQuality, MetafileDPI, OnCompress, OnDecodeImage, UseCompression

Properties Derived from TRpComponent
Version, CacheDir, ServerMode
3.11 TRvRenderPreview

Unit

RvRenderPreview

Hierarchy

TComponent
   TRpComponent
   TRpBaseComponent
      TBaseReport
      TCanvasReport
      TRvRenderPrinter
      TRvRenderPreview

Description

The TRvRenderPreview component takes a file generated by a TRvNDRWriter component and sends it to the screen for previewing. TRvRenderPreview has many methods and events that allow the programmer to create a completely customized user interface.

Properties

ScrollBox defines the TScrollBox component that the report will be drawn in. FileName and StreamMode are used in the same manner as TRvNDRWriter and TRvRenderPreview. GridHoriz and GridVert define the horizontal and vertical spacing, in inches or metric, between each grid marking drawn with GridPen. RulerType along with the grid settings can be useful during report development for determining accurate placement of items without having to produce printed output. MarginMethod and MarginPercent determine the method and size of the blank margin around the page image. ShadowDepth defines the number of pixels for the page shadow. Monochrome defines whether the output is drawn on a monochrome or color bitmap. are skipped when calling NextPage or PrevPage. ZoomInc defines the amount that ZoomIn and ZoomOut will use to modify the current zoom percentage, ZoomFactor.

Events

OnPageChange is called whenever the current page is changed and allows the programmer to update the user interface with the new current page number. OnZoomChange is called whenever the current zoom factor, ZoomFactor, is changed and allows the programmer to update the user interface with the new zoom factor.

Events Derived from TRvRenderPreview

OnPageChange, OnZoomChange

Events Derived from TBaseReport


Methods Derived from TRvRenderPreview

Clear, ExecuteCustom, NextPage, PrevPage, PrintPage, RedrawPage, XD2I, ZoomIn, ZoomOut
### 3.12 TRvRenderPrinter

#### Unit

![RvRenderPrinter](image)

#### Hierarchy

```
TComponent
  TRpComponent
    TRpBaseComponent
      TBaseReport
        TCanvasReport
          TRvRenderPrinter
```

### Methods Derived from TBaseReport

- Abort, AbortPage, AdjustLine, AllowAll, AllowPreviewOnly, AllowPrinterOnly, Arc, AssignFont, BrushCopy, CalcGraphicHeight, CalcGraphicWidth, Chord, ClearAllTabs, ClearColumns, ClearTabs, CopyRect, CR, Create, CreateBrush, CreateFont, CreatePen, CreatePoint, CreateRect, Destroy, DrawFocusRect, Draw, Ellipse, Execute, FillRect, Finish, FinishTabBox, FloodFill, FrameRect, GetMemoLine, GetNextLine, GetTab, GotoFooter, GotoHeader, GotoXY, GraphicFieldToBitmap, Home, LF, LinesLeft, LineTo, Macro, MemoLines, MoveTo, NewColumn, NewLine, NewPage, NoPrinters, Pie, Polygon, Polyline, PopFont, PopPos, PopTabs, Print, PrintBitmap, PrintBitmapRect, PrintBlock, PrintCenter, PrintCharJustify, PrintData, PrintDataStream, PrintFooter, PrintHeader, PrintImageRect, PrintJustify, PrintLeft, PrintLn, PrintMemo, PrintRight, PrintTab, PrintXY, PushFont, PushPos, PushTabs, RecoverPrinter, Rectangle, RegisterGraphic, ReleasePrinter, Reset, ResetLineHeight, ResetPrinter, ResetSection, ResetTabs, RestoreFont, RestorePos, RestoreTabs, ReuseGraphic, RoundRect, SaveFont, SavePos, SaveTabs, SelectBin, SelectPaper, SelectPrinter, SetBrush, SetColumns, SetColumnWidth, SetFont, SetPaperSize, SetPen, SetPIVar, SetTab, SetTopOfPage, ShadeToColor, ShowPrintDialog, ShowPrinterSetupDialog, Start, StretchDraw, SupportBin, SupportCollate, SupportDuplex, SupportOrientation, SupportPaper, Tab, TabEnd, TabStart, TabWidth, TextRect, TextWidth, UnregisterGraphic, UpdateStatus, XD2U, XI2D, X2I, XU2D, XU2I, YD2I, YD2U, YI2D, YI2U, YU2D, YU2I

### Properties Derived from TRvRenderPreview

- MarginMethod, MarginPercent, Monochrome, PageInc, Pages, ScrollBox, ZoomFactor, ZoomInc, ZoomPageFactor, ZoomPageWidthFactor

### Properties Derived from TRvRenderPrinter

- IgnoreFileSettings

### Properties Derived from TRpComponent

- Version

Components
Description
The TRvRenderPrinter component takes a file generated by a TRvNDRWriter component and sends it to the
current printer. TRvRenderPrinter is often used to do a print from the preview screen. TRvRenderPrinter is a
simple component but does have methods and properties to customize the selection of what gets printed.

Properties and Events
FileName is the name of the report file generated by TRvNDRWriter if StreamMode is smMemory or smFile. A
stream mode of smUser is used when the programmer wants to provide their own stream object (any
descendant of TStream will work) by assigning it to the Stream property of TRvNDRWriter, TRvRenderPrinter
and/or TRvRenderPreview. There are no events for TRvRenderPrinter. To send a report file to the printer call
the Execute or ExecuteCustom methods.

Events Derived from TBaseReport
OnPrintHeader, OnPrintPage

Methods Derived from TBaseReport
Abort, AbortPage, AdjustLine, AllowAll, AllowPreviewOnly, AllowPrinterOnly, Arc, AssignFont, BrushCopy,
CalcGraphicHeight, CalcGraphicWidth, Chord, ClearAllTabs, ClearColumns, ClearTabs, CopyRect, CR,
Create, CreateBrush, CreateFont, CreatePen, CreatePoint, CreateRect, Destroy, DrawFocusRect, Draw,
Ellipse, Execute, FillRect, Finish, FinishTabBox, FloodFill, FrameRect, GetMemoLine, GetNextLine, GetTab,
GotoFooter, GotoHeader, GotoXY, GraphicFieldToBitmap, Home, LF, LinesLeft, LineTo, Macro, MemoLines,
MoveTo, NewColumn, NewLine, NewPage, NoPrinters, Pie, Polygon, Polyline, PopFont, PopPos, PopTabs,
Print, PrintBitmap, PrintBitmapRect, PrintBlock, PrintCenter, PrintCharJustify, PrintData, PrintDataStream,
PrintFooter, PrintHeader, PrintImageRect, PrintJustify, PrintLeft, PrintLn, PrintMemo, PrintRight, PrintTab,
PrintXY, PushFont, PushPos, PushTabs, RecoverPrinter, Rectangle, RegisterGraphic, ReleasePrinter, Reset,
ResetLineHeight, ResetPrinter, ResetSection, ResetTabs, RestoreFont, RestorePos, RestoreTabs,
ReuseGraphic, RoundRect, SaveFont, SavePos, SaveTabs, SelectBin, SelectPaper, SelectPrinter, SetBrush,
SetColumns, SetColumnWidth, SetFont, SetPaperSize, SetPen, SetPIVar, SetTab, SetTopOfPage,
ShadeToColor, ShowPrintDialog, ShowPrinterSetupDialog, Start, StretchDraw, SupportBin, SupportCollate,
SupportDuplex, SupportOrientation, SupportPaper, Tab, TabEnd, TabStart, TabWidth, TextRect, TextStyle,
UnregisterGraphic, UpdateStatus, XD2U, XI2D, XI2U, XU2D, XU2I, YD2I, YD2U, YI2D, YI2U, YU2D, YU2I

Properties Derived from TRvRenderPrinter
IgnoreFileSettings

Properties Derived from TBaseReport
Abort, AccuracyMethod, AscentHeight, Bins, BKColor, Bold, BottomWaste, BoxLineColor, Canvas, Collate,
ColumnEnd, ColumnLinesLeft, ColumnNum, ColumnStart, ColumnWidth, Copies, CurrentPage,
CurrentPass, CursorXPos, CursorYPos, DescentHeight, DeviceName, DevMode, DriverName, Duplex,
FileName, FirstPage, FontAlign, FontBaseline, FontBottom, FontCharSet, FontColor, FontHandle, FontHeight,
FontName, FontPitch, FontRotation, Fonts, FontSize, FontTop, FontWidth, FrameMode, GridVert, Italic,
LastPage, LeftWaste, LineBottom, LineHeight, LineHeightMethod, LineMiddle, LineNum, LinesPerInch,
LineTop, MacroData, MarginBottom, MarginLeft, MarginRight, MarginTop, MaxCopies, NoBufferLine,
NoNTColorFix, NoPrinterPageHeight, NoPrinterPageWidth, Orientation, OriginX, OriginY, OutputInvalid,
OutputName, PageHeight, PageInvalid, PageWidth, Papers, PIVar, Port, PrinterIndex, Printers, Printing,
ReportDateTime, RightWaste, ScaleX, ScaleY, SectionBottom, SectionLeft, SectionRight, SectionTop,
Selection, ShadowDepth, StatusFormat, StatusLabel, StatusText, Stream, StreamMode, Strikeout, Subscript,
Superscript, TabColor, TabJustify, TabShade, TextBkMode, Title, TopWaste, TotalPasses,
TransparentBitmaps, TruncateText, Underline, Units, UnitsFactor, XDPJ, XPos, YDPJ, YPos

Properties Derived from TComponent
Version
3.13 TRvRenderRTF

Unit

Hierarchy

Description
TRvRenderRTF will convert an NDR stream or file to an RTF document. Text, graphic, line and rectangle objects are supported.

Properties Derived from TRpRender
Active, DisplayName, FileExtension, ImageEncoding, ImageOutput, OnDecodeImage

Properties Derived from TRpComponent
Version

3.14 TRvRenderText

Unit

Hierarchy

Description
TRvRenderText will convert an NDR stream or file to an text document. Only text objects are supported in the output, all other objects will be ignored.

TRpRender Properties
Active, CPI, DisplayName, FileExtension, FormFeed, LeftBorder, LPI, TopBorder

TRpComponent Properties
Version

removed
CacheDir, ImageQuality, MetafileDPI, OnCompress, ServerMode, UseCompression
3.15 TRvSystem

Unit

\[
\text{RpSystem}
\]

Hierarchy

\[
\text{TComponent}
\]

\[
\text{TRpComponent}
\]

\[
\text{TRvSystem}
\]

Description

The TRvSystem component is a very powerful component that integrates the functionality of the previous three components, TRvRenderPreview, TRvRenderPrinter and TRvNDRWriter in one easy to use system. TRvSystem can send a report to the printer or a preview screen and can display a setup and status screen as well.

Properties

DefaultDest is where the report will be sent if no setup screen is used or is the default during setup. SystemFiler, which can be accessed by double-clicking on the left column in the Object Inspector, will display all of the filer type options from TRvNDRWriter, TRvRenderPreview and TRvRenderPrinter. All SystemFiler options operate the same as the other components except for a stream mode of smMemory which does not require a filename and will use a TMemoryStream to contain the report.

The SystemOptions properties control the configuration of the TRvSystem component. soUseFiler will always send the report to a report file. This can be very useful if the Macro method has been used in the report. soWaitForOK will determine whether the user has to press the OK button once the report has been generated for output. soShowStatus will determine whether or not the status screen is displayed when the report is being generated or printing. soAllowPrintFromPreview will determine whether the user can print from the preview screen. soPreviewModal determines the modal mode that the preview window is brought up in soNoGenerate will skip over the generation phase of the report and proceed straight to the screen. This options should only be used with a StreamMode of smFile where the report file has been previously generated and needs only to be viewed or printed.

SystemPreview displays all of the preview type options found in TRvRenderPreview. SystemPrinter displays all of the printer type options found in TRvNDRWriter.

The SystemSetups properties control the configuration of the standard setup screen for TRvSystem. ssAllowSetup will determine whether or not the setup screen is displayed. ssAllowCopies, ssAllowCollate and ssAllowDuplex will enable those options in the setup screen. ssAllowDestPreview, ssAllowDestPrinter and ssAllowDestFile will determine which destination options the user has access to. ssAllowPrinterSetup will determine whether the user can select the printer setup dialog which allows the selection of alternate printers and other printer options. ssAllowPreviewSetup determines whether the user will be allowed to select the printer setup dialog after preview.

from TRvSystem

\[
\text{BaseReport DefaultDest GridHoriz GridPen OutputFileName ReportDest RulerType SystemFiler SystemOptions SystemPreview SystemPrinter SystemSetups TitlePreview TitleSetup TitleStatus}
\]

from TRpComponent

\[
\text{Version}
\]
Components

Events
All of the OnXxxx events for TRvSystem operate exactly like they do for TRvNDRWriter. The override events, OverridePreview, OverrideSetup and OverrideStatus allow the programmer to replace the default screens provided with Rave with their own. There is no printed documentation on how to do this but the TRvSystem component uses the same method as a user would have to. Reference the methods OverridePreviewProc, OverrideStatusProc and OverrideSetupProc for how to create an override event method. The units RpFormPreview, RpFormStatus and RpFormSetup located in \RAVE\SOURCE will also show how to interface with TRvSystem and can be used as starting points for customized versions of the different forms.

from TRvSystem
OnPreviewSetup  OnPreviewShow  OverridePreview  OverrideSetupOverrideStatus

3.16  TRvTableConnection

Unit

RpConBDE

Hierarchy
TComponent

TRpComponent

TRvCustomConnection

TRvTableConnection

Description
Data connection components - Rave uses data from your application. This is accomplished with data connection components, TRvCustomConnection, TRvDataSetConnection, TRvTableConnection and TRvQueryConnection to provide a bridge between the data in your application and the Rave visual components.

TRvCustomConnection 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.

TRvTableConnection is to be used specifically with TTable components or their descendents respectively.

TRvQueryConnection is to be used specifically with TQuery components or their descendents respectively.

Events Derived from TRvCustomConnection

Methods Derived from TRvCustomConnection
WriteBCDData, WriteBlobData, WriteBoolData, WriteCurrData, WriteDateTime, WriteFloatData, WriteIntData, WriteNullData, WriteStrData

Properties Derived from TRvTableConnection
Table, UseSetRange

Properties Derived from TRvCustomConnection
FieldAliasList, LocalFilter, RuntimeVisibility

Properties Derived from TRpComponent
Version
Events

Chapter IV
4 Events

An event is a mechanism that links an occurrence to some code. More specifically, an event is a method pointer that points to a method in a specific class instance.

4.1 OnAfterClose

Declaration

procedure OnAfterClose(Sender: TObject);

Category

Rave

Description

This event will be called immediately after the Rave project is closed.

See also

TRvProject Class, Active, Close, OnAfterOpen, OnBeforeClose, OnBeforeOpen, Open

4.2 OnAfterOpen

Declaration

procedure OnAfterOpen(Sender: TObject);

Category

Rave

Description

This event will be called immediately after the Rave project is opened.

See also

TRvProject Class, Active, Close, OnAfterClose, OnBeforeClose, OnBeforeOpen, Open

4.3 OnAfterPrint

Declaration

procedure OnAfterPrint(Sender: TObject);

Category

Control

Description

This event will be called after each print job has finished printing, even if the print job was aborted or an exception has been generated. This can be useful for cleaning up resources that were allocated in OnBeforePrint.

See also

TBaseReport Class, Execute, OnBeforePrint

Example (Delphi)

procedure TReportForm.AfterPrintReport2(Sender: TObject);
begin { AfterPrintReport2 }
  CustomerTable.Close;
end; { AfterPrintReport2 }

Example (C++Builder)

void __fastcall TReportForm:: AfterPrintReport2 ( TObject *Sender )
{
   CustomerTable->Close();
}
4.4 **OnBeforeClose**

**Declaration**

```delphi
procedure OnBeforeClose(Sender: TObject);
```

**Category**

*Rave*

**Description**

This event will be called immediately before the Rave project is closed.

**See also**

*TRvProject Class, Active, Close, OnAfterClose, OnAfterOpen, OnBeforeOpen, Open*

4.5 **OnBeforeOpen**

**Declaration**

```delphi
procedure OnBeforeOpen(Sender: TObject);
```

**Category**

*Rave*

**Description**

This event will be called immediately before the Rave project is opened.

**See also**

*TRvProject Class, Active, Close, OnAfterClose, OnAfterOpen, OnBeforeClose, Open*

4.6 **OnBeforePrint**

**Declaration**

```delphi
procedure OnBeforePrint(Sender: TObject);
```

**Category**

*Control*

**Description**

This event is called before the print job has begun. This can be useful to initialize non-report items such as table record pointers. This event can also be useful to set report items that must be set before the print job begins (such as paper size and orientation).

**See also**

*TBaseReport Class, Execute, OnAfterPrint*

**Example (Delphi)**

```delphi
procedure TReportForm.BeforePrintReport5(Sender: TObject);
begin
  { BeforePrintReport5 }  
  with Sender as TBaseReport do begin
    StatusFormat := 'Printing Page $#13''#13';
    StatusText.Add('');
    StatusText.Add('');
    end;  
  CustomerTable.First;
end;  
```


Example (C++Builder)

```cpp
void __fastcall TReportForm:: BeforePrintReport5 (TObject *Sender)
{
    TBaseReport* rp = dynamic_cast<TBaseReport*>(Sender);
    rp1->StatusFormat = "Printing Page \n\n";
    rp1->StatusText->Add("");
    rp1->StatusText->Add("");
    CustomerTable->First();
} / BeforePrintReport5
```

### 4.7 OnCreate

**Declaration**

```delphi```
procedure OnCreate(Sender: TObject);
```

**Category**

Rave

**Description**

This event is called when the TRvProject is created. This is the normal place to register custom Rave components by calling the RaveRegister procedure for the unit containing the custom Rave components. See the tutorials for more information.

**See also**

TRvProject Class, OnDestroy

### 4.8 OnDecodeImage

**Declaration**

```delphi```
procedure OnDecodeImage( Sender: TObject; ImageStream: TStream; ImageType: String; Bitmap: TBitmap );
```

**Category**

Graphics HTML PDF RTF

**Description**

This event is called when Rave needs to convert image data (created from the PrintImageRect method) to a bitmap for printing. This would normally appear on a TRvRenderPrinter or TRvRenderPreview component, but could also be defined in a TRvSystem component.

**See also**

TBaseReport Class, PrintImageRect

**Example (Delphi)**

```delphi```
```
vary
    Image: TJPEGImage;
    Format: word;
    Data: THandle;
    Palette: HPalette;
if ImageType = 'JPG' then begin
    Image := TJPEGImage.Create; // Create a TJPEGImage class
    Image.LoadFromStream(ImageStream); // Load JPEG image from ImageStream
    Image.DIBNeeded; // Convert JPEG to bitmap format
    // Save JPEG to clipboard in bitmap format
    Image.SaveToClipboardFormat(Format,Data,Palette);
    Image.Free; // Free the image
    // Load bitmap from clipboard
    Bitmap.LoadFromClipboardFormat(Format,Data,Palette);
end; { if}
```
Example (C++Builder)

```cpp
if (ImageType == "JPG") {
    Image = new TJPEGImage(); // Create a JPEGImage class
    Image->LoadFromStream(ImageStream); // Load JPEG image from ImageStream
    Image->DIBNeeded(); // Convert JPEG to bitmap format
    // Save JPEG to clipboard in bitmap format
    Image->SaveToClipboardFormat(Format, Data, Palette);
    delete Image; // Free the image
    // Load bitmap from clipboard
    Bitmap->LoadFromClipboardFormat(Format, Data, Palette);
}
```

4.9 OnDesignerSave

Declaration

```cpp```
procedure OnDesignerSave(Sender: TObject);
```cpp```

Category

Rave

Description

When this event is defined, a save button and save menu item will be displayed in the end user version of the Rave visual designer to allow the end user to perform intermediate saves. In this event, you will normally call RvProject.Save or whatever code you are using to save the project (i.e., RvProject1.SaveToStream(BlobStream)). The Sender parameter is the TRvProject component that generated the event.

NOTE:

This feature is only available with a Rave EUDL license. See the Nevrona website at http://www.nevrona.com for more information on obtaining an EUDL license.

See also

TRvProject Class, OnDesignerSaveAs, OnDesignerShow, SaveToStream

4.10 OnDesignerSaveAs

Declaration

```cpp```
procedure OnDesignerSaveAs(Sender: TObject);
```cpp```

Category

Rave

Description

When this event is defined, a Save As menu item will be displayed in the end user version of the Rave visual designer to allow the end user to perform saves to alternate destinations. In this event, you will normally prompt the user for an alternate destination and then call RvProject.Save or whatever code you are using to save the project (i.e., RvProject1.SaveToStream(BlobStream)). The Sender parameter is the TRvProject component that generated the event.

NOTE:

This feature is only available with a Rave EUDL license. See the Nevrona website at http://www.nevrona.com for more information on obtaining an EUDL license.

See also

TRvProject Class, OnDesignerSave, OnDesignerShow, SaveToStream

4.11 OnDesignerShow

Declaration

```cpp```
procedure OnDesignerShow(Sender: TObject);
```cpp```
Rave Reports Developer Reference

Category
Rave

Description
This event will be called after the Rave visual designer is initialized but immediately before it is displayed. This will allow you to show a splash screen or change the mouse cursor while the designer is loading, then restore everything just before Rave is displayed. The Sender parameter is the TRvProject component that generated the event.

NOTE:
This feature is only available with a Rave EUDL license. See the Nevrona website at http://www.nevrona.com for more information on obtaining an EUDL license.

See also
TRvProject Class, OnDesignerSave

4.12 OnDestroy

Declaration
procedure OnDestroy(MyPrinter: Trave);

Category
Rave

Description
This event is called when the TRvProject component is being destroyed. This is useful for freeing up resources that were allocated in the OnCreate event.

See also
TRvProject Class, OnCreate

4.13 OnEOF

Declaration
procedure OnEOF(Connection: TRvCustomConnection; var Eof: Boolean);

Category
Rave

Description
This event is called when the Rave data system wants the EOF status for the data. See the tutorial on customizing data connections for more information.

See also
TRvCustomConnection Class, OnFirst, OnNext

4.14 OnFirst

Declaration
procedure OnFirst(Connection: TRvCustomConnection);

Category
Rave

Description
This event is called when the Rave data system wants the data cursor to be positioned to the beginning of the data. See the tutorial on customizing data connections for more information.

See also
TRvCustomConnection Class, OnEOF, OnNext
4.15 **OnGetCols**

**Declaration**

```plaintext
procedure OnGetCols(Connection: TRvCustomConnection);
```

**Category**

Rave

**Description**

This event is called when the Rave data system wants to retrieve the meta-data information (field names, types, sizes and descriptions) for the data. See the tutorial on customizing data connections for more information.

See also

TRvCustomConnection Class, OnGetRow

4.16 **OnGetRow**

**Declaration**

```plaintext
procedure OnGetRow(Connection: TRvCustomConnection);
```

**Category**

Rave

**Description**

This event is called when the Rave data system wants to retrieve the data for the current row of the data. See the tutorial on customizing data connections for more information.

See also

TRvCustomConnection Class, OnFirst, onNext

4.17 **OnGetSorts**

**Declaration**

```plaintext
procedure OnGetSorts(Connection: TRvCustomConnection);
```

**Category**

Rave

**Description**

This event is called when the Rave data system wants the available sorting methods available for the data. See the tutorial on customizing data connections for more information.

See also

TRvCustomConnection Class, OnSetSort

4.18 **OnNewColumn**

**Declaration**

```plaintext
procedure OnNewColumn(Sender: TObject);
```

**Category**

Control

**Description**

This event will be called whenever a new column has begun (after a call to PrintLn, NewLine, SetColumns or SetColumnWidth). This can be useful for printing column headers.

See also

TBaseReport Class, NewLine, PrintLn, SetColumns, SetColumnWidth
Example (Delphi)
procedure TReportForm.OnNewColumnReport10(Sender: TObject);
begin
  with Sender as TBaseReport do begin
    Underline := true;
    PrintLn('Column Titles');
    Underline := false;
  end; { with }
end;

Example (C++Builder)
void __fastcall TReportForm::OnNewColumnReport10 (TObject *Sender)
{
  TBaseReport* rp = dynamic_cast<TBaseReport*>(Sender);
  rp1->Underline = true;
  rp1->PrintLn("Column Titles");
  rp1->Underline = false;
}

4.19 OnNewPage

Declaration
procedure OnNewPage(Sender: TObject);

Category
Control

Description
This event will be called whenever a new page is generated. This can be useful to initialize page related items.

See also
TBaseReport Class, NewPage, SelectBin

Example (Delphi)
procedure TRpForm.RvNDRWriter1NewPage(Sender: TObject);
begin
  with Sender as TBaseReport do begin
    MarginTop := 0.5;
    Home;
    SetFont('Arial',24);
    PrintHeader('Report Title', pjCenter);
    MarginTop := 1.0;
    Home;
    SetFont('Arial',10);
    PrintHeader(FormatDateTime(DateFormat, now), pjRight);
  end; { with }
end;
Example (C++Builder)

```cpp
void __fastcall TRpForm::RvNDRWriter1NewPage (TObject *Sender)
{
    TBaseReport* rp = dynamic_cast<TBaseReport*>(Sender);
    rp1->PrintBitmapRect(0.5, 0.5, 1.20, 1.20, Logo);
    rp1->MarginTop = 0.5;
    rp1->Home();
    rp1->SetFont("Arial", 24);
    rp1->PrintHeader("Report Title", pjCenter);
    rp1->MarginTop = 1.0;
    rp1->Home();
    rp1->SetFont("Arial", 10);
    rp1->PrintHeader(FormatDateTime("ddd, dd mmm yyyy hh:mm:ss", Now()), pjRight);
}
```

### 4.20 OnNext

**Declaration**

```delphi
procedure OnNext(Connection: TRvCustomConnection);
```

**Category**

Rave

**Description**

This event is called when the Rave data system wants the data cursor to be moved to the next row of the data. See the tutorial on customizing data connections for more information.

See also

TRvCustomConnection Class, OnEOF, OnFirst

### 4.21 OnOpen

**Declaration**

```delphi
procedure OnOpen(Connection: TRvCustomConnection);
```

**Category**

Rave

**Description**

This event is called when the Rave data system wants to initialize the data session. See the tutorial on customizing data connections for more information.

See also

TRvCustomConnection Class, OnRestore

### 4.22 OnPageChange

**Declaration**

```delphi
procedure OnPageChange(Sender: TObject);
```

**Category**

Preview

**Description**

This event will be called whenever the current page changes on the preview screen. This can be useful for updating the current page number on visual controls on the preview screen.

See also

TRvRenderPreview Class, NextPage, PrevPage, PrintPage

**Example (Delphi)**

```delphi
procedure TPreForm.RvRenderPreview1PageChange(Sender: TObject);
```
begin
  with RvRenderPreview1 do begin
    PageEdit.Text := IntToStr(CurrentPage);
    PageLabel.Caption := 'Page ' + IntToStr(CurrentPage - FirstPage + 1) + ' of ' + IntToStr(Pages);
  end; { with }
end;

Example (C++Builder)
void __fastcall TForm1::RvSystem1PreviewSetup(TObject *Sender)
{  
  TRvRenderPreview* fp = dynamic_cast<TRvRenderPreview*>(Sender);
  fp->ZoomFactor = 50;
  TForm* pf = dynamic_cast<TForm*>(fp->Owner);
  fp->Position = poDesigned;
  fp->Top = 10;
  fp->Left = 10;
}

4.23 OnPreviewSetup

Declaration
procedure OnPreviewSetup( Sender: TObject );

Category Preview

Description
This will allow you to modify the TRvRenderPreview component on a preview form as well as the preview form itself. Some functions, such as ZoomPageWidthFactor will need to be called in the OnPreviewShow event.

NOTE:
OnPreviewSetup is called before the form is shown and TRvRenderPreview is started.

See also TRvSystem Class, OnPreviewShow

Example (Delphi)
Procedure TForm1.RvSystem1PreviewSetup( Sender: TObject );
begin
  with Sender as TRvRenderPreview do begin
    ZoomFactor := 50;
    with Owner as TForm do begin
      Position := poDesigned;
      Top := 10;
      Left := 10;
    end; { with }
  end; { with }
end;

Example (C++Builder)
void __fastcall TForm1::RvSystem1PreviewSetup(TObject *Sender)
{  
  TRvRenderPreview* fp = dynamic_cast<TRvRenderPreview*>(Sender);
  fp->ZoomFactor = 50;
  TForm* pf = dynamic_cast<TForm*>(fp->Owner);
  fp->Position = poDesigned;
  fp->Top = 10;
  fp->Left = 10;
}
4.24 OnPreviewShow

**Declaration**

procedure OnPreviewShow( Sender: TObject );

**Category**

Preview

**Description**

This will allow you to modify the TRvRenderPreview component on the preview form itself.

**NOTE:**

This event is called during the OnShow event of the preview form.

**See also**

TRvSystem Class, OnPreviewSetup

**Example (Delphi)**

```
procedure TForm1.RvSystem1PreviewShow( Sender: TObject );
begin
  with Sender as TRvRenderPreview do begin
    ZoomFactor := ZoomPageWidthFactor;
  end;  { with }
end;
```

**Example (C++Builder)**

```
void __fastcall TForm1::RvSystem1PreviewShow(TObject *Sender)
{
  TRvRenderPreview* fp = dynamic_cast<TRvRenderPreview*>(Sender);
  fp->ZoomFactor = fp->ZoomPageWidthFactor;
}
```

4.25 OnPrint

**Declaration**

procedure OnPrint(Sender: TObject);

**Category**

Control

**Description**

This event will be called when it is time to print the body of the report. To begin a new page call the NewPage method. To finish the report just exit this event. The event is useful for more complicated reports that are different from page to page.

**See also**

TBaseReport Class, Execute, NewPage, OnPrintPage

4.26 OnPrintFooter

**Declaration**

procedure OnPrintFooter(Sender: TObject);

**Category**

Control

**Description**

This event will be called after the body for each page that has been printed. This can be useful for printing similar footers for each page.

**See also**

TBaseReport Class, GotoFooter, PrintFooter, OnPrintHeader
Example (Delphi)

```delphi
procedure TReportForm.PrintFooterReport5(Sender: TObject);
begin { PrintFooterReport5 }
  with Sender as TBaseReport do begin
    SetFont('Times New Roman',8);
    MarginBottom := 0.5;
    PrintFooter('Page ' + IntToStr(CurrentPage),pjLeft);
    PrintFooter('Date 01/20/95',pjRight);
    MarginBottom := 1.0;
  end; { with }
end; { PrintFooterReport5 }
```

Example (C++Builder)

```cpp
void __fastcall TReportForm::PrintFooterReport5 (TObject *Sender)
{
  TBaseReport* rp = dynamic_cast<TBaseReport*>(Sender);
  rp1->SetFont("Times New Roman",8);
  rp1->MarginBottom = 0.5;
  rp1->PrintFooter("Page " + IntToStr(rp1->CurrentPage),pjLeft);
  rp1->PrintFooter("Date 01/20/95",pjRight);
  rp1->MarginBottom = 1.0;
}
```

### 4.27 OnPrintHeader

**Declaration**

```delphi
procedure OnPrintHeader(Sender: TObject);
```

**Category**

Control

**Description**

This event will be called before the body for each page that has been printed. This can be useful for printing similar headers for each page.

**See also**

TBaseReport Class, GotoHeader, OnPrintFooter, PrintHeader

Example (Delphi)

```delphi
procedure TReportForm.PrintHeaderReport5(Sender: TObject);
begin { PrintHeaderReport5 }
  with Sender as TBaseReport do begin
    MarginTop := 0.5;
    SetFont('Arial',24);
    Underline := true;
    Home;
    PrintCenter('Customer List', PageWidth / 2);
    MarginTop := 1.0;
  end; { with }
end; { PrintHeaderReport5 }
```

Example (C++Builder)

```cpp
void __fastcall TReportForm::PrintHeaderReport5 (TObject *Sender)
{
  TBaseReport* rp = dynamic_cast<TBaseReport*>(Sender);
  rp1->MarginTop = 0.5;
  rp1->SetFont("Arial",24);
  rp1->Underline = true;
  rp1->Home();
  rp1->PrintCenter("Customer List", rp1->PageWidth / 2);
  rp1->MarginTop = 1.0;
}
```
4.28 OnPrintPage

Declaration

function OnPrintPage( Sender: TObject; var PageNum: Integer): Boolean;

Category

Control

Description

This event will be called when it is time to print the body of a page for the report. This event will only be called if an OnPrint event handler does not already exist for this report. To begin a new page, return a result of true; otherwise, to finish the report just exit this event with a result of false. This event is useful for reports that are the same from page to page.

See also

TBaseReport Class, Execute, OnPrint

Example (Delphi)

function TReportForm.PrintPageReport3(Sender: TObject;
var PageNum: integer): Boolean;
begin { PrintPageReport3 }
  with Sender as TBaseReport do begin
    SetFont('Times New Roman',10);
    Home;

    { Print memo buffer }
    SetColumns(3,0.25);
    MemoBuf.PrintStart := ColumnStart;
    MemoBuf.PrintEnd := ColumnEnd;
    PrintMemo(MemoBuf, ColumnLinesLeft, false);
    ClearColumns;

    Result := not MemoBuf.Empty;
  end; { with }
end; { PrintPageReport3 }

Example (C++Builder)

bool __fastcall TReportForm::PrintPageReport3 (TObject *Sender, int &PageNum)
{
  TBaseReport* rp = dynamic_cast<TBaseReport*>(Sender);
  rp->SetFont("Times New Roman",10);
  rp->Home();

  // Print memo buffer
  rp->SetColumns(3,0.25);
  MemoBuf->PrintStart = rp->ColumnStart;
  MemoBuf->PrintEnd = rp->ColumnEnd;
  rp->PrintMemo(MemoBuf, rp->ColumnLinesLeft(), false);
  rp->ClearColumns();

  return !MemoBuf->Empty();
}

4.29 OnRestore

Declaration

procedure OnRestore(Connection: TRvCustomConnection);

Category

Rave
40 Rave Reports Developer Reference

Description
This event is called when the Rave data system wants to restore the data session to its state before the OnOpen event was called. See the tutorial on customizing data connections for more information.

See also
TRvCustomConnection Class, OnOpen

4.30 OnSetFilter

Declaration
procedure OnSetFilter(Connection: TRvCustomConnection);

Category
Rave

Description
This event is called when the Rave data system wants to filter the data based on field criteria. See the tutorial on customizing data connections for more information.

See also
TRvCustomConnection Class, OnSetSort

4.31 OnSetSort

Declaration
procedure OnSetSort(Connection: TRvCustomConnection);

Category
Rave

Description
This event is called when the Rave data system wants to sort the data. See the tutorial on customizing data connections for more information.

See also
TRvCustomConnection Class, OnSetFilter

4.32 OnValidateRow

Declaration
procedure OnValidateRow(Connection: TRvCustomConnection; var ValidRow: Boolean);

Category
Rave

Description
This event is called for each row in the data and allows the custom selection of which records will be included in the report by setting ValueRow to true or false. See the tutorial on customizing data connections for more information.

See also
TRvCustomConnection Class, OnSetFilter

4.33 OnZoomChange

Declaration
procedure OnZoomChange(Sender: TObject);

Category
Preview
Events

41

Description
This event will be called whenever the current zoom factor changes for the preview screen. This can be useful for updating the current zoom factor on visual controls on the preview screen.

NOTE:
If an OnZoomChange event handler is created, it is responsible for redrawing the page by calling RedrawPage.

See also
TRvRenderPreview Class, RedrawPage, ZoomIn, ZoomOut

Example Delphi
procedure TRpPreviewForm.RvRenderPreview1ZoomChange(Sender: TObject);
var
  S1: string[10];
begin
  Str(RvRenderPreview1.ZoomFactor:1:1,S1);
  ZoomEdit.Text := S1;
  RvRenderPreview1.RedrawPage;
end;

Example (C++Builder)
void __fastcall TForm1::RvRenderPreview1ZoomChange(TObject *Sender)
{
  AnsiString S1;
  S1 = FloatToStrF(RvRenderPreview1->ZoomFactor, ffGeneral,1,1);
  ZoomEdit->Text = S1;
  RvRenderPreview1->RedrawPage();
}

4.34 OverridePreview

Declaration
procedure OverridePreview(RvSystem: TRvSystem; OverrideMode: TOverrideMode; var OverrideForm: TForm);

Category
ReportSystem

Description
This event allows the programmer to replace the default preview screen with a custom preview screen. See RpSYSTEM.PAS for more information.

See also
TRvSystem Class, OverridePreviewProc

4.35 OverrideSetup

Declaration
procedure OverrideSetup( RvSystem: TRvSystem; OverrideMode: TOverrideMode; var OverrideForm: TForm);

Category
ReportSystem

Description
This event allows the programmer to replace the default preview screen with a custom preview screen. See RpSYSTEM.PAS for more information.

See also
TRvSystem Class, OverrideSetupProc
4.36 OverrideStatus

Declaration

procedure OverrideStatus( RvSystem: TRvSystem; OverrideMode: TOverrideMode; var OverrideForm: TForm);

Category

ReportSystem

Description

This event allows the programmer to replace the default preview screen with a custom preview screen. See RpSYSTEM.PAS for more information.

See also

TRvSystem Class, OverrideStatusProc
Methods

Chapter V
5 Methods

A method is a procedure or function associated with a class. A call to a method specifies the object (or, if it is a class method, the class) that the method should operate on.

5.1 Abort

Declaration

procedure Abort;

Category

Control

Description

This method will abort the printing of the report and set the property Aborted to true.

NOTE:

Abort raises the silent exception Abort that will cease the current thread of execution. Make sure to use exception handling (try...finally) to restore any resources that you may allocate in your reporting code.

See Also

TBaseReport Class, Aborted, Execute

Example (Delphi)

procedure TRpStatusForm.CancelButtonClick(Sender:TObject);
begin
  RvNDRWriter1.Abort;
end;

Example (C++Builder)

void __fastcall TRpStatusForm::CancelButtonClick(TObject* Sender)
{
  RvNDRWriter1->Abort();
}

5.2 AbortPage

Declaration

procedure AbortPage;

Category

Control

Description

This method will abort the printing of the current page and start printing a new page.

See also

TBaseReport Class, Abort

Example (Delphi)

RvNDRWriter1.AbortPage;

Example (C++Builder)

rp1->AbortPage();

5.3 AdjustLine

Declaration

procedure AdjustLine;
## Category

**Position**

### Description

This method will adjust the current text cursor so that the current line is placed correctly below the previous line after a change in font size. Use `AdjustLine` when you want to reset the line height and line font after the cursor is already on the next line.

### See also

`TBaseReport Class`, `ResetLineHeight`

### Example (Delphi)

```delphi
SetFont('Arial',14);
PrintLn('This is the first line of text');
SetFont('Arial',10);
AdjustLine;
PrintLn('This is the second line of text');
```

### Example (C++Builder)

```cpp
rp1->SetFont("Arial",14);
rp1->PrintLn("This is the first line of text");
rp1->SetFont("Arial",10);
rp1->AdjustLine();
rp1->PrintLn("This is the second line of text");
```

### 5.4 AllowAll

#### Declaration

```delphi
procedure AllowAll;
```

#### Category

**Control**

#### Description

This method will reset the valid destinations to all after they have been modified by `AllowPreviewOnly` or `AllowPrinterOnly`.

#### See also

`TBaseReport Class`, `AllowPreviewOnly`, `AllowPrinterOnly`

### Example (Delphi)

```delphi
// Draw a line on the preview screen only
AllowPreviewOnly;
MoveTo(1.5,1.5);
LineTo(6.5,1.5);
AllowAll;
```

### Example (C++Builder)

```cpp
rp1->AllowPreviewOnly();
rp1->MoveTo(1.5,1.5);
rp1->LineTo(6.5,1.5);
rp1->AllowAll();
```

### 5.5 AllowPreviewOnly

#### Declaration

```delphi
procedure AllowPreviewOnly;
```

#### Category

**Control**
Description
This method will set the valid destinations to preview only. Any printing commands that follow will only be sent to the preview screen. The method can be very useful to print items that you want to appear on the preview screen but not the printer (Such as the label extents for the TLabelShell component).

See also
TBaseReport Class, AllowAll, AllowPrinterOnly

Example
See AllowAll

5.6 AllowPrinterOnly

Declaration
procedure AllowPrinterOnly;

Category
Control

Description
This method will set the valid destinations to printer only. Any printing commands that follow will only be sent to the printer. This method can be very useful to print items that you want to appear on the printer but not the preview screen.

See also
TBaseReport Class, AllowAll, AllowPreviewOnly

Example
See AllowAll

5.7 Append

Declaration
procedure Append(Text: string);

Category
Memo

Description
This method will append Text to the end of the memo buffer.

See also
TMemoBuf Class, Insert

Example (Delphi)
MemoBuf.Append(' This is a new sentence on the end.');</n

Example (C++Builder)
MemoBuf->Append(" This is a new sentence on the end.");

5.8 AppendMemoBuf

Declaration
procedure AppendMemoBuf(MemoBuf: TMemoBuf);

Category
Memo

Description
Will append MemoBuf to the current memo buffer.
See also

**TMemoBuf Class, InsertMemoBuf**

**Example (Delphi)**

```delphi
MemoBuf1.AppendMemoBuf(MemoBuf2);
```

**Example (C++Builder)**

```cpp
MemoBuf1->AppendMemoBuf(MemoBuf2);
```

### 5.9 Arc

**Declaration**

```delphi
procedure Arc(X1,Y1,X2,Y2,X3,Y3,X4,Y4: double);
```

**Category**

*Graphics*

**Description**

This method draws an arc inside an ellipse bounded by the rectangle defined by \((X1,Y1)\) and \((X2,Y2)\). The arc starts at the intersection of the line drawn between the ellipse center \(((X1+X2)/2.0,(Y1+Y2)/2.0)\) and the point \((X3,Y3)\) and is drawn counterclockwise until it reaches the intersection of the line drawn between the ellipse center and the point \((X4,Y4)\).

See also

**TBaseReport Class, Ellipse, Pie**

**Example (Delphi)**

```delphi
RvNDRWriter1.Arc(1.0,1.0,3.0,3.0,3.0,2.0,0.0,0.0);
```

**Example (C++Builder)**

```cpp
RvNDRWriter1->Arc(1.0,1.0,3.0,3.0,3.0,2.0,0.0,0.0);
```

### 5.10 AssignFont

**Declaration**

```delphi
procedure AssignFont(Font: TFont);
```

**Category**

*Font*

**Description**

Selects current font to the TFont object from list.

See also

**TBaseReport Class, SetFont**

**Example (Delphi)**

```delphi
RvNDRWriter1.AssignFont(FontDialog1.Font);
```

**Example (C++Builder)**

```cpp
RvNDRWriter1->AssignFont(FontDialog1->Font);
```

### 5.11 BrushCopy

**Declaration**

```delphi
procedure BrushCopy(const Dest: TRect; Bitmap: TBitmap; const Source: TRect; Color: TColor);
```

**Category**

*Graphics*
Description
Copies a portion of Bitmap specified by the rectangle Source to the printer canvas. Color of Bitmap is replaced by the brush color of the destination canvas. The rectangle Dest defines the region to copy the bitmap to.

See also
TBaseReport Class, CreateRect, TColor, TRect

Example (Delphi)
RvNDRWriter1.BrushCopy(DestRect, UserBMP, SrcRect, clBlack);

Example (C++Builder)
RvNDRWriter1->BrushCopy(DestRect, UserBMP, SrcRect, clBlack);

5.12 CalcGraphicHeight

Declaration
function CalcGraphicHeight(Width: double; Graphic: TGraphic): double;

Category
Graphics

Description
This method will calculate and return the value for the new Height of the Graphic based on the Width value while maintaining the original ratio of the Graphic. This could be used to see if there is enough room left on the page before attempting to print the graphic. This can be used for both bitmaps and metafiles.

See also
TBaseReport Class, CalcGraphicWidth, PrintBitmap, PrintBitmapRect, StretchDraw

Example (Delphi)
Bitmap := TBitmap.Create;
Bitmap.LoadFromFile('RpDEMO.BMP');
PrintBitmapRect(X1, Y1, X1 + 3.0, Y1 + CalcGraphicHeight(3.0, Bitmap), Bitmap);
Bitmap.Free;

Example (C++Builder)
Graphics::TBitmap* Bitmap;
Bitmap = new Graphics::TBitmap();
Bitmap->LoadFromFile("RpDEMO.BMP");
rp1->PrintBitmapRect(X1, Y1, X1 + 3.0, Y1 + rp1->CalcGraphicHeight(3.0, Bitmap), Bitmap);
delete Bitmap;

5.13 CalcGraphicWidth

Declaration
function CalcGraphicWidth(Height: double; Graphic: TGraphic): double;

Category
Graphics

Description
This method will calculate and return the value for the new Width of the Graphic based on the Height value while maintaining the original ratio of the Graphic. This can be used for both bitmaps and metafiles.

See also
TBaseReport Class, CalcGraphicHeight, PrintBitmap, PrintBitmapRect, StretchDraw

Example (Delphi)
Bitmap := TBitmap.Create;
Bitmap.LoadFromFile('RpDEMO.BMP');
PrintBitmapRect(X1, Y1,
        X1 + CalcGraphicWidth(3.0, Bitmap), Y1 + 3.0, Bitmap);
Bitmap.Free;

Example (C++Builder)
Graphics::TBitmap* Bitmap;
Bitmap = new Graphics::TBitmap();
rp1->PrintBitmapRect(X1, Y1,
        X1 + rp1->CalcGraphicHeight(3.0, Bitmap), 3.0, Bitmap);
delete Bitmap;

5.14 Chord

Declaration
procedure Chord(X1, Y1, X2, Y2, X3, Y3, X4, Y4: double);

Category Graphics

Description
This method draws a chord inside an ellipse bounded by the rectangle defined by (X1,Y1) and (X2,Y2). The chord starts at the intersection of the line drawn between the ellipse center ((X1+X2)/2.0,(Y1+Y2)/2.0) and the point (X3,Y3) and is drawn to the line drawn between the ellipse center and the point (X4,Y4).

See also TBaseReport Class, Ellipse

Example (Delphi)
RvNDRWriter1.Chord(1.0,1.0,3.0,3.0,0.0,0.8,3.0,2.0);

Example (C++Builder)
RvNDRWriter1->Chord(1.0,1.0,3.0,3.0,0.0,0.8,3.0,2.0);

5.15 Clear

Declaration
procedure Clear;

Category Preview

Description
This method will remove the TImage from the preview TScrollBox and refresh the display. This method can be useful for clearing the preview screen without having to destroy the preview form.

See also TRvRenderPreview Class, ScrollBox

Example (Delphi)
// Clear the preview screen
RvRenderPreview1.Clear;

Example (C++Builder)
RvRenderPreview1->Clear();

5.16 ClearAllTabs

Declaration
procedure ClearAllTabs;
Category
Tabs

Description
This method will clear the current tab settings as well as all saved tab settings. This call is normally not
needed since the tabs are cleared once the report is finished.

See also
TBaseReport Class, ClearTabs, SaveTabs

Example (Delphi)
// Clear all tabs, including saved tabs
ClearAllTabs;

Example (C++Builder)
rpl->ClearAllTabs();

5.17 ClearColumns

Declaration
procedure ClearColumns;

Category
Column

Description
This method removes all current column settings.

See also
TBaseReport Class, SetColumns, SetColumnWidth

Example (Delphi)
RvNDRWriter1.ClearColumns;

Example (C++Builder)
RvNDRWriter1->ClearColumns();

5.18 ClearRaveBlob

Declaration
procedure ClearRaveBlob;

Category
Rave

Description
This method will clear the currently loaded report project from the application form. You should not need to
call this function since the normal method of clearing the loaded report project is through the
TRvProject.StoreRAV property editor.

See also
TRvProject Class, LoadRaveBlob, RaveBlobDateTime, SaveRaveBlob, StoreRAV

Example (Delphi)
RvProject1.ClearRaveBlob;

Example (C++Builder)
RvProject1->ClearRaveBlob();
5.19 ClearTabs

Declaration

procedure ClearTabs;

Category

Tabs

Description

This method removes all current tab settings but will leave saved tab settings as they were.

See also

TBaseReport Class, ResetTabs, SetTab

Example (Delphi)

RvNDWriter1.ClearTabs;

Example (C++Builder)

RvNDWriter1->ClearTabs();

5.20 Close

Declaration

procedure Close;

Category

Rave

Description

This method will close the report project and unload it from memory. If you call the Open method of TRvProject, you should insure that this method is called before the application terminates.

See also

TRvProject Class, Active, OnAfterClose, OnAfterOpen, OnBeforeClose, OnBeforeOpen, Open

Example (Delphi)

RvProject1.Close;

Example (C++Builder)

RvProject1->Close();

5.21 ConstraintHeightLeft

Declaration

function ConstraintHeightLeft(Constraint: double): double;

Category

Memo

Description

This method will return the height necessary to print the memo buffer for the current font between PrintStart and PrintEnd. However, for speed purposes, this method will stop processing when the height exceeds the Constraint parameter.

NOTE:

You must initialize the TMemoBuf.BaseReport before calling this method.

See also

TMemoBuf Class, MemoHeightLeft, PrintEnd, PrintMemo, PrintStart, TMemoBuf
Example (Delphi)
MemoBuf.BaseReport := Sender as TBaseReport;
HeightLeft := MemoBuf.ConstraintHeightLeft(5.0);

Example (C++Builder)
MemoBuf->BaseReport = rp;
HeightLeft = MemoBuf->ConstraintHeightLeft(5.0);

5.22 CopyRect
Declaration
procedure CopyRect(const Dest: TRect; Canvas: TCanvas; const Source: TRect);

Category
Graphics

Description
This method copies part of an image defined by the rectangle Source from another canvas to the area on
the printer canvas defined by the rectangle Dest.

See also
TBaseReport Class, CreateRect, TCanvas, TRect

Example (Delphi)
RvNDRWriter1.CopyRect( DestRect, DestCanvas, SrcRect);

Example (C++Builder)
RvNDRWriter1->CopyRect( DestRect, DestCanvas, SrcRect);

5.23 CR
Declaration
procedure CR;

Category
Position

Description
This method performs a carriage return which moves the horizontal text cursor position to the beginning of
the current line. The beginning of the current line is defined by either the current SectionLeft setting or the
setting of ColumnStart if columns are in use.

See also
TBaseReport Class, ColumnStart, LF, NewLine, SectionLeft

Example (Delphi)
with RvNDRWriter1 do begin
  SectionLeft := 3.0;
  PrintLn('This text is 3 inches from left');
  SectionLeft := 1.0;
  CR;
end; { with }

Example (C++Builder)
rp1->SectionLeft = 3.0;
rp1->PrintLn("This text is 3 inches from left");
rp1->SectionLeft = 1.0;
rp1->CR();

5.24 Create (TBaseReport)
Declaration
constructor Create(AOwner: TComponent);
Methods

5.25 Create (TRpBarsBase)

Declaration

```delphi
constructor Create( BaseRpt: TBaseReport );
```

Category

BarCode

Description

This constructor is called to create an instance of the Bar Code Class. The current reporting object should be passed into the BaseRpt parameter.

See also

TRpBarsBase Class, BaseReport (bar code)

Example (Delphi)

```delphi
BarCode1 := TRpBarsPostNet.Create(Sender as TBaseReport);
```

```delphi
with BarCode1 do begin
  BarHeight := 0.125;
end;
```
BarWidth := 0.020;
UseChecksum := True;
Text := '85283-3558'; {'-' will be stripped}
Left := MarginLeft + 1.0;
Print;
end; {if}
Barcode1.Free;

Example (C++Builder)
TBaseReport* rp = dynamic_cast<TBaseReport*>(Sender);
TRpBarsPostNet* bc1 = new TRpBarsPostNet(rp);
bc1->BarHeight = 0.125;
bcl->BarWidth = 0.020;
bcl->UseChecksum = true;
bcl->Text = "85283-3558"; / "-" will be stripped
bc1->Left = rp1->MarginLeft + 1.0;
bcl->Print();
delete bc1;

5.26 CreateBrush

Declaration
function CreateBrush(NewColor: TColor; NewStyle: TBrushStyle; NewBitmap: TBitmap): TBrush;

Category
Graphics

Description
This method will create a TBrush object for the given parameters. If a bitmap is not desired, pass in the value of nil. You can assign this brush to the canvas to change the current brush.

NOTE:
The brush object returned must be released by calling the free method of TBrush.

See also
TBaseReport Class, SetBrush, TBrush, TBrushStyle, TColor

Example (Delphi)
var MyBrush: TBrush;
begin
MyBrush := CreateBrush(clRed, bsSolid, nil);
end;

Example (C++Builder)
TBrush* MyBrush;
MyBrush = rp1->CreateBrush(clRed, bsSolid, NULL);
MyBrush->Free();

5.27 CreateFont

Declaration
function CreateFont(NewName: string; NewSize: integer): TFont;

Category
Font

Description
This method will create a TFont object for the given parameters. NewSize is the point size of the font (1/72nds of an inch). You can assign this font to the canvas to change the current font.

NOTE:
The font object returned must be released by calling the free method of TFont. Also, it is preferable to use
SaveFont and RestoreFont.

See also
TBaseReport Class, RestoreFont, SaveFont, SetFont, TFont

Example (Delphi)
var MyFont: TFont;
begin
  MyFont := CreateFont('Times New Roman',8.00);
end;

Example (C++Builder)
TFont* MyFont;
MyFont = rp1->CreateFont("Times New Roman",8.00);

5.28 CreatePen

Declaration
function CreatePen(NewColor: TColor; NewStyle: TPenStyle; NewWidth: integer; NewMode: TPenMode): TPen;

Category
Graphics

Description
This method will create a TPen object for the given parameters. The NewWidth parameter, if positive, is the width of the pen in printer units (dots) and if negative, is the width of the pen in 1/100ths of an inch. You can assign this pen to the canvas to change the current pen.

NOTE:
The pen object returned must be released by calling the free method of TPen.

See also
TBaseReport Class, SetPen, TColor, TPen, TPenMode, TPenStyle

Example (Delphi)
MyPen := CreatePen(clBlack,psSolid,1,pmBlack);

Example (C++Builder)
MyPen = rp1->CreatePen(clBlack,psSolid,1,pmBlack);

5.29 CreatePoint

Declaration
function CreatePoint(X,Y: double): TPoint;

Category
Graphics

Description
This method will return a TPoint record initialized to the point (X,Y).

See also
TBaseReport Class, TPoint

Example (Delphi)
MyPoint := CreatePoint(1.00,6.00);

Example (C++Builder)
MyPoint = rp1->CreatePoint(1.00,6.00);
5.30 CreateRect

Declaration

function CreateRect(X1,Y1,X2,Y2: double): TRect;

Category

Graphics

Description

This method will return a TRect record initialized to the rectangle defined by the points (X1,Y1) and (X2,Y2).

See also

TBaseReport Class, CopyRect, TextRect, TRect

Example (Delphi)

MyRect := CreateRect(1.00,6.00,3.00,8.00);

Example (C++Builder)

MyRect = rp1->CreateRect(1.00,6.00,3.00,8.00);

5.31 Delete

Declaration

procedure Delete(BufPos: longint; DelLen: longint);

Category

Memo

Description

This method will delete DelLen characters starting at BufPos in the memo buffer.

See also

TMemoBuf Class, Insert

Example (Delphi)

// Delete 5 characters at current position
MemoBuf.Delete(MemoBuf.Pos,5);

Example (C++Builder)

MemoBuf->Delete(MemoBuf->Pos,5);

5.32 Design

Declaration

procedure Design;

Category

Rave

Description

This method will start the execution of the Rave visual designer for the currently selected report.

NOTE:

This feature is only available with a Rave EUDL license. See the Nevrona website at http://www.nevrona.com for more information on obtaining an EUDL license.

See also

TRvProject Class, DesignReport, Execute, ExecuteReport, SelectReport
5.33 **DesignReport**

**Declaration**

```delphi
procedure DesignReport(ReportName: string);
```

**Category**

Rave

**Description**

This method will start the execution of the Rave visual designer for the specified report. `ReportName` is the short name of the report as defined in the report project. If you want to design the report by its full name you will need to call the `SelectReport` and `Design` methods.

**NOTE:**

This feature is only available with a Rave EUDL license. See the Nevrona website at http://www.nevrona.com for more information on obtaining an EUDL license.

**See also**

TRvProject Class, Design, Execute, ExecuteReport

**Example (Delphi)**

```delphi
RvProject1.DesignReport('CustomerListing');
```

**Example (C++Builder)**

```cpp
RvProject1->DesignReport("Customer Listing");
```

5.34 **Destroy**

**Declaration**

```delphi
Destructor Destroy;
```

**Category**

Misc

**Description**

The Destroy destructor should never be called directly. To destroy a component created with Create, call the Free method.

**See also**

TBaseReport Class, Create

**Example**

see Create

5.35 **Draw**

**Declaration**

```delphi
procedure Draw(X,Y: double; Graphic: TGraphic);
```

**Category**

Graphics

**Description**

This method draws Graphic to the printer canvas at the location (X,Y).
NOTE:
Do not use Draw for bitmaps. Use PrintBitmap or PrintBitmapRect instead.

See also
TBaseReport Class, PrintBitmap, PrintBitmapRect, StretchDraw, TGraphic

Example (Delphi)
var   MyLogo: TGraphic;
begin  MyLogo := TMetafile.Create;
    try    MyLogo.LoadFromFile('MYLOGO.WMF');
            RvNDRWriter1.Draw(1.0,2.0,MyLogo);
    finally    MyLogo.Free;
    end;  { tryf }
end;

Example (C++Builder)
TGraphic* MyLogo;
MyLogo = new TMetafile();
try {    MyLogo->LoadFromFile("MYLOGO.WMF");
            RvNDRWriter1->Draw(1.0,2.0,MyLogo);
}  __finally {    delete MyLogo;
};  tryf

5.36 DrawFocusRect

Declaration
procedure DrawFocusRect(const Rect: TRect);

Category
Graphics

Description
This method will draw a rectangle, defined by Rect, in the style used to indicate that the rectangle has focus.

See also
TBaseReport Class, CreateRect, TRect

Example (Delphi)
RvNDRWriter1.DrawFocusRect(CreateRect(1.0,1.0,2.0,3.0));

Example (C++Builder)
RvNDRWriter1->DrawFocusRect(rp1->CreateRect(1.0,1.0,2.0,3.0));

5.37 Ellipse

Declaration
procedure Ellipse(X1,Y1,X2,Y2: double);

Category
Graphics

Description
This method draws an ellipse bounded by the rectangle defined by (X1,Y1) and (X2,Y2).
See also  
*TBaseReport Class, Arc, Pie*

Example (Delphi)  
Ellipse(5.375,1.25,7.375,2.75);

Example (C++Builder)  
rpl->Ellipse(5.375,1.25,7.375,2.75);

5.38 Empty

Declaration  
function Empty: Boolean;

Category  
*Memo*

Description  
This method will return true if the memo buffer does not have anything in it or if the current position, Pos, is beyond the end of the buffer.

See also  
*TMemoBuf Class, Pos, Size*

Example (Delphi)  
if not MemoBuf1.Empty then begin  
  PrintMemo(MemoBuf1,0,false);  
end; { if }

Example (C++Builder)  
if (!MemoBuf1->Empty()) {  
  rpl->PrintMemo(MemoBuf1,0,false);  
} / if

5.39 Execute (TBaseReport)

Declaration  
procedure Execute;

Category  
*Control*

Description  
This method will begin the printing task assigned to the component. For report generation components (*TRvSystem, TRvNDRWriter*) the event handlers OnBeforePrint, OnPrint, OnPrintPage, OnNewPage, OnNewColumn, OnPrintHeader, OnPrintFooter and OnAfterPrint will be called at their appropriate times. For *TRvRenderPrinter* or *TRvRenderPreview* the contents of the report stream from a *TRvNDRWriter* will be sent to either the printer or the preview screen. See *Start* for printing the report for a *TRvRenderPreview* component.

See also  
*TBaseReport Class, Abort, Painting, All printing event handlers*

Example (Delphi)  
RvNDRWriter1.Execute;

Example (C++Builder)  
RvNDRWriter1->Execute();
5.40 **Execute (TRvProject)**

**Declaration**

```delphi
procedure Execute;
```

**Category**  
Rave

**Description**  
This method will start the printing of the currently selected Rave report. This method can be called while a printing job is in progress from a TRvNDRWriter component (typically inside of the OnPrint event) to add in the Rave report to the current code generated report.

**See also**  
TRvProject Class, ExecuteReport, SelectReport

**Example (Delphi)**

```delphi
RvProject1.Execute;
```

**Example (C++Builder)**

```c++
RvProject1->Execute();
```

5.41 **ExecuteCustom**

**Declaration**

```delphi
procedure ExecuteCustom(NewFirstPage: integer; NewLastPage: integer; NewCopies: integer);
```

**Category**  
Control

**Description**  
This method will print the report but only for the specified parameters. NewCopies, if non-zero, will override the copies setting in the report file. NewFirstPage and NewLastPage, if non-zero, will only print the report file for that page range.

**See also**  
TRvRenderPreview Class, Copies, Execute

**Example (Delphi)**

```delphi
// Print 2 copies of only the first four pages
RvRenderPrinter1.ExecuteCustom( 1, 4, 2);
```

**Example (C++Builder)**

```c++
RvRenderPrinter1->ExecuteCustom( 1, 4, 2);
```

5.42 **ExecuteReport**

**Declaration**

```delphi
procedure ExecuteReport(ReportName: string);
```

**Category**  
Rave

**Description**  
This method will start the execution of the named Rave report. This method can be called while a printing job is in progress from a TRvNDRWriter component (typically inside of the OnPrint event) to add in the Rave report to the current code generated report.

**See also**  
TRvProject Class, Execute
**Example** (Delphi)
RvProject1.ExecuteReport('CustomerListing');

**Example** (C++Builder)
RvProject1->ExecuteReport("CustomerListing");

### 5.43 FillRect

**Declaration**
procedure FillRect(const Rect: TRect);

**Category**
Graphics

**Description**
This method fills the rectangle defined by `Rect` with the current brush.

See also
* TBaseReport Class, CreateRect, TRect

**Example** (Delphi)
FillRect( CreateRect( 1.0, 1.0, 2.0, 3.0 ) );

**Example** (C++Builder)
rp1->FillRect(rp1->CreateRect(1.0, 1.0, 2.0, 3.0));

### 5.44 Finish

**Declaration**
procedure Finish;

**Category**
Control

**Description**
This method finishes a preview session for the TRvRenderPreview component or finishes a print job for TRvNDRWriter. `Start` must have been called first before `Finish` will be a valid call.

See also
* TBaseReport Class, Start

**Example** (Delphi)
RvRenderPreview1.Finish;

**Example** (C++Builder)
RvRenderPreview1->Finish();

### 5.45 FinishTabBox

**Declaration**
procedure FinishTabBox(Width: integer);

**Category**
Tabs

**Description**
Draws the top line for the current set of tabs using a line width of `Width`. Useful when printing a table drawn with the setting of BOXLINELEFTRIGHT to finish the bottom of each tab box. This function can also be called at the beginning to draw the top line of the table.
See also

*TBaseReport Class, SetTab*

**Example (Delphi)**

```delphi
ClearTabs;
SetTab(0.5, pjLeft, 1.5, 5, BOXLINELEFTRIGHT, 0);
SetTab(NA, pjLeft, 1.5, 5, BOXLINELEFTRIGHT, 0);
SetTab(NA, pjLeft, 4.5, 5, BOXLINELEFTRIGHT, 0);
FinishTabBox(1);
PrintTab('Name');
PrintTab('Picture');
PrintTab('Description');
NewLine;
FinishTabBox(1);
```

**Example (C++Builder)**

```cpp
rp1->ClearTabs();
rp1->SetTab(0.5, pjLeft, 1.5, 5, BOXLINELEFTRIGHT, 0);
rp1->SetTab(NA, pjLeft, 1.5, 5, BOXLINELEFTRIGHT, 0);
rp1->SetTab(NA, pjLeft, 4.5, 5, BOXLINELEFTRIGHT, 0);
rp1->FinishTabBox(1);
rp1->PrintTab("Name");
rp1->PrintTab("Picture");
rp1->PrintTab("Description");
rp1->NewLine();
rp1->FinishTabBox(1);
```

### 5.46 FloodFill

**Declaration**

```delphi
procedure FloodFill(X, Y: double; Color: TColor; FillStyle: TFillStyle);
```

**Category**  
*Graphics*

**Description**

This method fills an area of the printer canvas using the current brush. *FloodFill* begins at the point \((X, Y)\) and fills until the boundary specified by the color, \(\text{Color}\), is encountered. *FillStyle* defines the method of fill used. (\(fsBorder\) will fill until the color, \(\text{Color}\), is encountered and \(fsSurface\) will fill while the color, \(\text{Color}\), is still encountered.)

See also

*TBaseReport Class, PageInvalid, TColor*

**Example (Delphi)**

```delphi
FloodFill(2.0, 3.0, clRed, fsBorder);
```

**Example (C++Builder)**

```cpp
FloodFill(2.0, 3.0, clRed, fsBorder);
```

### 5.47 FrameRect

**Declaration**

```delphi
procedure FrameRect(const Rect: TRect);
```

**Category**  
*Graphics*

**Description**

This method draws the rectangle \(\text{Rect}\) using the current brush to draw the border of the rectangle. *FrameRect* does not fill the rectangle with the current brush.
5.48 FreeSaved

**Declaration**

```
procedure FreeSaved;
```

**Category**

Memo

**Description**

This method will free the memory allocated by a previous call to `SaveBuffer`. This method is normally not needed as the saved buffer is freed when the memo buffer is freed.

**See also**

TBaseReport Class, CreateRect, TRect

**Example (Delphi)**

```
RVNDRWriter1.FrameRect( CreateRect( 1.0,1.0, 2.0,3.0 ) );
```

**Example (C++Builder)**

```
RvNDRWriter1->FrameRect( rp1->CreateRect(1.0,1.0,2.0,3.0) );
```

5.49 GetMemoLine

**Declaration**

```
function GetMemoLine( MemoBuf: TMemoBuf; var EOL: Boolean): string;
```

**Category**

Memo

**Description**

This method will return a single line from the memo buffer each time it is called. You can print the memo buffer line by line by placing this function inside a `PrintLn` statement. `EOL` returns true when it encounters a carriage return or the end of the memo buffer.

**See also**

TBaseReport Class, MemoLines, PrintMemo, TMemoBuf

**Example (Delphi)**

```
PrintLn(GetMemoLine(MemoBuf, EOL));
```

**Example (C++Builder)**

```
rp1->PrintLn(rp1->GetMemoLine(MemoBuf, EOL));
```

5.50 GetNextLine

**Declaration**

```
function GetNextLine(var EOL: Boolean): string;
```

**Category**

Memo

**Description**

This method will return a single line from the memo buffer each time it is called. You can print the memo buffer line by line by placing this function inside a `PrintLn` statement. `EOL` returns true when it encounters a carriage return or the end of the memo buffer.
buffer line by line by placing this function inside a *PrintLn* statement. *EOL* returns true when it encounters a carriage return or the end of the memo buffer.

**NOTE:**
You must initialize the *TMemoBuf.BaseReport* before calling this method.

See also
*TBaseReport Class, MemoLines, PrintMemo, TMemoBuf*

**Example** (Delphi)

```delphi
PrintLn(GetNextLine(EOL));
```

**Example** (C++Builder)

```cpp
rp1->PrintLn(rp1->GetNextLine(EOL));
```

### 5.51 GetParam

**Declaration**

```delphi
procedure GetParam(ParamName: string);
```

**Category**

*Rave*

**Description**

GetParam allows an event in the Visual Designer to get a parameter that was passed from the application to the currently loaded Rave project. These parameters can be used to control dynamic layouts, SQL parameters or other items to print in a visually designed report.

See also
*TRvProject Class, SetParam*

**Example** (in Visual Designer Event)

```delphi
RaveProject.GetParam('UserName', UserName);
```

### 5.52 GetReportCategoryList

**Declaration**

```delphi
procedure GetReportCategoryList(ReportList: TStrings; Categories: string; FullName: Boolean);
```

**Category**

*Rave*

**Description**

This method will allow you to get all of the reports matching specific categories. If you had categories called Accounting, General, Status and System. Now if you want to get a list of all reports except System, then you would call RvProject1.GetReportCategoryList(ReportList, 'Accounting; Status; General;');. If FullName is true, this will return the full names of all reports in the current report project and if it is false, it will return the short names of the reports.

**NOTE:**

The double "",";" at the end of the category list is to include all reports where the category is not defined (the default value).

See also
*TRvProject Class, SelectReport*
5.53 GetReportList

Declaration

procedure GetReportList(ReportList: TStrings; FullName: Boolean);

Category
Rave

Description
This method will fill ReportList with a list of Rave defined reports that could then be used in a list box or other TStrings compatible object. ReportList must be an already created TStrings object. If FullName is true, this will return the full names of all reports in the current report project and if it is false it will return the short names of the reports.

See also
TRvProject Class, SelectReport

5.54 GetTab

Declaration

function GetTab(Index: integer): PTab;

Category
Tabs

Description
This method will return the tab setting specified by Index. If Index is 0 then GetTab will return the current tab setting and if Index is greater than the number of defined tabs then a value of nil will be returned. See RpDEFINE.PAS for information on the PTab structure.

See also
TBaseReport Class, TabIndex

5.55 GotoFooter

Declaration

procedure GotoFooter;

Category
Position

Description
This method will position the text cursor just above the current SectionBottom.

See also
TBaseReport Class, MarginBottom, PrintFooter, SectionBottom

Example (Delphi)

GotoFooter;
Print('Line just above SectionBottom');

Example (C++Builder)

rpl->GotoFooter();
rpl->Print("Line just above SectionBottom");

5.56 GotoHeader

Declaration

procedure GotoHeader;

Category
Position
Description
This method will position the text cursor just below the current SectionTop.

See also
TBaseReport Class, MarginTop, PrintHeader, SectionTop

Example (Delphi)
RvNDRWriter1.GotoHeader;
RvNDRWriter1.Print('Line just below SectionTop');

Example (C++Builder)
RvNDRWriter1->GotoHeader();
RvNDRWriter1->Print("Line just below SectionTop");

5.57 GotoXY

Declaration
procedure GotoXY(NewXPos: double; NewYPos: double);

Category
Position

Description
This method will move the text cursor to the position NewXPos, NewYPos.

See also
TBaseReport Class, XPos, YPos

Example (Delphi)
// This code shows how to position the output at specific coordinates.
GotoXY(1.0,8.5);
Print('Text at 1.0,8.5');

Example (C++Builder)
rp1->GotoXY(1.0,8.5);
rp1->Print("Text at 1.0,8.5");

5.58 GraphicFieldToBitmap

Declaration
procedure GraphicFieldToBitmap(GraphicField: TGraphicField; Bitmap: TBitmap);

Category
Graphics

Description
This method will convert a TGraphicField (graphical data from a database) to a bitmap.

NOTE:
You must include RpDBUTIL in your Uses statement to access this procedure.

See also
TBaseReport Class, PrintBitmap, PrintBitmapRect, TGraphicField

Example (Delphi)
// Convert and print a TGraphicField
Bitmap := TBitmap.Create;
GraphicFieldToBitmap(Table1Graphic,Bitmap);
PrintBitmapRect(5.375,3.5,7.375,5.5,Bitmap);
Bitmap.Free;
5.59 Home

Declaration
procedure Home;

Category
Position

Description
This method will move the text cursor to the beginning of line 1.

See also
TBaseReport Class

Example (Delphi)
SetFont('Arial',10);
Home;
Print('Text in the Home position');

Example (C++Builder)
rpl->SetFont("Arial",10);
rpl->Home();
Print("Text in the Home position");

5.60 Insert

Declaration
procedure Insert(BufPos: longint; Text: string);

Category
Memo

Description
This method will insert Text into the memo buffer at BufPos. BufPos should be 0 to insert before the entire buffer.

See also
TMemoBuf Class, Append

Example (Delphi)
MemoBuf.Insert(0,'This text will now be first');

Example (C++Builder)
MemoBuf->Insert(0,"This text will now be first");

5.61 InsertMemoBuf

Declaration
procedure InsertMemoBuf(BufPos: longint; MemoBuf: TMemoBuf);

Category
Memo

Description
Will insert a MemoBuf at BufPos into the current memo buffer.
See also

`TMemoBuf Class, AppendMemoBuf`

**Example (Delphi)**

```delphi
MemoBuf1.InsertMemoBuf(10,MemoBuf2);
```

**Example (C++Builder)**

```c++
MemoBuf1->InsertMemoBuf(10,MemoBuf2);
```

### 5.62 IsValidChar

**Declaration**

```delphi
function IsValidChar( Ch: char ): Boolean;
```

**Category**

`BarCode`

**Description**

Is used to determine whether a character is a valid character for the particular bar code being printed.

**See also**

`TRpBarsBase Class`

**Example (Delphi)**

```delphi
// following will return false because 2of5 only support numbers
Code2of5.IsValidChar('A')
```

**Example (C++Builder)**

```c++
Code2of5->IsValidChar('A')
```

### 5.63 LF

**Declaration**

```delphi
procedure LF;
```

**Category**

`Position`

**Description**

This method performs a line feed which moves the vertical text cursor position down by the distance specified by the property `LineHeight`. It also increments the property `LineNum`. If Columns are in use, and the text cursor is moved below the current `SectionBottom`, the text cursor is placed at the top of the next column. The top of the next column is defined by the setting of `SectionTop`.

**See also**

`TBaseReport Class, CR, LineHeight, LineNum, NewLine, SectionBottom, SectionTop`

**Example (Delphi)**

```delphi
RVNDRWriter1.LF;
```

**Example (C++Builder)**

```c++
RVNDRWriter1->LF();
```

### 5.64 LinesLeft

**Declaration**

```delphi
function LinesLeft: integer;
```

**Category**

`Position`
Description
This method will return the number of lines that can be printed above the current SectionBottom including the current line.

See also
TBaseReport Class, ColumnLinesLeft, SectionBottom

Example (Delphi)
if RvNDRWriter1.LinesLeft < 3 then begin
  RvNDRWriter1.NewPage;
end; { if }

Example (C++Builder)
if (RvNDRWriter1->LinesLeft() < 3) {
  RvNDRWriter1->NewPage();
} / if

5.65 LineTo

Declaration
procedure LineTo(X,Y: double);

Category
Graphics

Description
This method will draw a line using the current pen from the previous graphic cursor position to the point specified by (X,Y).

See also
TBaseReport Class, MoveTo

Example (Delphi)
with RvNDRWriter1 do begin
  MoveTo( 1.0, 1.0 );
  LineTo( 3.0, 3.0 );
  MoveTo( 1.0, 3.0 );
  LineTo( 3.0, 1.0 );
end; { with }

Example (C++Builder)
rp1->MoveTo( 1.0, 1.0 );
rp1->LineTo( 3.0, 3.0 );
rp1->MoveTo( 1.0, 3.0 );
rp1->LineTo( 3.0, 1.0 );

5.66 LoadFromFile (TMemoBuf)

Declaration
function LoadFromFile( FileName: String);

Category
Memo

Description
This method will load a memo buffer with the contents of a text file. To load RTF text, use RTFLoadFile.

See also
TMemoBuf Class, LoadFromStream, RTFLoadFromFile, SaveToStream

Example (Delphi)
MemoBuf1.LoadFromFile('Letter.Txt');
Example (C++Builder)
MemoBuf1->LoadFromFile("Letter.Txt");

5.67 LoadFromFile (TRvProject)

Declaration
procedure LoadFromFile(Filename: string);

Category
Rave

Description
This method will load the report project file specified by the FileName parameter as the current Rave project.

See also
TRvProject Class, LoadFromStream, SaveToFile, SaveToStream

Example (Delphi)
RvProject1.LoadFromFile('Project1.Rav');

Example (C++Builder)
RvProject1->LoadFromFile("Project1.Rav");

5.68 LoadFromStream (TMemoBuf)

Declaration
procedure LoadFromStream(Stream: TStream; BufSize: longint);

Category
Memo

Description
This method will load the memo buffer from the stream for BufSize number of bytes.

See also
TMemoBuf Class, SaveToStream

Example (Delphi)
MemoBuf1.LoadFromStream(MyStream, StreamSize);

Example (C++Builder)
MemoBuf1->LoadFromStream(MyStream, StreamSize);

5.69 LoadFromStream (TRvProject)

Declaration
procedure LoadFromStream(Stream: TStream);

Category
Rave

Description
This method will load the report project store in Stream as the current report project.

See also
TRvProject Class, LoadFromFile, SaveToFile, SaveToStream

Example (Delphi)
RvProject1.LoadFromStream(BlobStream);
Example (C++Builder)
RvProject1->LoadFromStream(BlobStream);

5.70 LoadRaveBlob

Declaration
procedure LoadRaveBlob(Stream: TStream);

Category  Rave

Description
This method will load the report project stored in Stream into the application form. You should not need to call this function since the normal method of loading a report project is through the TRvProject. StoreRAV property editor.

See also  TRvProject Class, ClearRaveBlob, RaveBlobDateTime, SaveRaveBlob, StoreRAV

Example (Delphi)
RvProject1.LoadRaveBlob( MyStream );

Example (C++Builder)
RvProject1->LoadRaveBlob( MyStream );

5.71 Macro

Declaration
function Macro(MacroID: TMacroID): string;

Default  6

Category  Misc, Printing

Description
This function inserts a macro into your report. The macro will be inserted at the time of report output (to preview or printer) and not at report generation time. Use this method with all printing methods. For a list of MacroID see the type definition of TMacroID.

See also  TBaseReport Class, MacroData, TMacroID

Example (Delphi)
// Print the current page and total pages
PrintRight(Macro(midCurrentPage) + ' of ' + Macro(midTotalPages), 8.0);

Example (C++Builder)
rp1->PrintRight(rp1->Macro(midCurrentPage) + " of " + rp1->Macro(midTotalPages), 8.0);

5.72 MakeLink

Declaration
function MakeLink(aDisplayText:String ; aHyperlink: String): string;

function StartLink(aHyperlink: String);

function EndLink;
Default

nil

Category

Control, Printing

Description

This function creates a text string which contains an embedded link. This text string is a "hot area" and clicking within that "hot area" will jump you to the HyperLink location. This method only works when your reports are rendered in HTML or PDF formats.

See also

TBaseReport Class, StartLink, EndLink

Example (Delphi)

// Print your text with a Hyperlink
PrintLeft(MakeLink('Tutorials', www.nevrona.com/Default.aspx?tabid=179), 2.5);

Example (C++Builder)

rp1->PrintLeft(rp1->MakeLink('Tutorials', www.nevrona.com/Default.aspx?tabid=179), 2.5);

5.73 MemoHeightLeft

Declaration

function MemoHeightLeft: double;

Category

Memo

Description

This method will return the height necessary to print the memo buffer for the current font between PrintStart and PrintEnd.

NOTE:

You must initialize the TMemoBuf.BaseReport before calling this method.

See also

TMemoBuf Class, ConstraintHeightLeft, MemoLinesLeft, PrintEnd, PrintMemo, PrintStart, TMemoBuf

Example (Delphi)

MemoBuf.BaseReport := Sender as TBaseReport;
HeightLeft := MemoBuf.MemoHeightLeft;

Example (C++Builder)

MemoBuf->BaseReport = rp;
HeightLeft = MemoBuf->MemoHeightLeft();

5.74 MemoLines

Declaration

function MemoLines(MemoBuf: TMemoBuf): longint;

Category

Memo

Description

This method will return the number of lines necessary to print the memo buffer MemoBuf for the current font between PrintStart and PrintEnd.

See also

TBaseReport Class, PrintEnd, PrintMemo, PrintStart, TMemoBuf
**Example (Delphi)**

```delphi
// Save number of lines needed to print memo
LinesLeft := RvNDRWriter1.MemoLines(MyMemo);
```

**Example (C++Builder)**

```cpp
LinesLeft = RvNDRWriter1->MemoLines(MyMemo);
```

### 5.75 MemoLinesLeft

**Declaration**

```delphi```
function MemoLinesLeft: longint;
```

**Category**

Memo

**Description**

This method will return the number of lines necessary to print the memo buffer for the current font between `PrintStart` and `PrintEnd`.

**NOTE:**

You must initialize the `TMemoBuf.BaseReport` before calling this method

**See also**

`TMemoBuf Class`, `PrintEnd`, `PrintMemo`, `PrintStart`, `MemoHeightLeft`, `TMemoBuf`

**Example (Delphi)**

```delphi
MemoBuf.BaseReport := Sender as TBaseReport;
LinesLeft := MemoBuf.MemoLinesLeft;
```

**Example (C++Builder)**

```cpp
MemoBuf->BaseReport = rp;
LinesLeft = MemoBuf->MemoLinesLeft();
```

### 5.76 MoveTo

**Declaration**

```delphi```
procedure MoveTo(X,Y: double);
```

**Category**

Graphics

**Description**

This method will move the current graphic cursor position to the point specified by (X,Y).

**See also**

`TBaseReport Class`, `LineTo`

**Example (Delphi)**

```delphi
RvNDRWriter1.MoveTo( NewX, NewY );
```

**Example (C++Builder)**

```cpp
RvNDRWriter1->MoveTo( NewX, NewY );
```

### 5.77 NewColumn

**Declaration**

```delphi```
procedure NewColumn;
```

**Category**

Column, Control
Description
Creates a new column in addition to the columns that already exist (that were set using the SetColumns or SetColumnWidth methods). If there is not enough space on the current page, it will create one with the current settings on the next page.

See also
TBaseReport Class, SetColumns, SetColumnWidth

Example (Delphi)
RvNDRWriter1.NewColumn;

Example (C++Builder)
RvNDRWriter1->NewColumn();

5.78 NewLine

Declaration
procedure NewLine;

Category
Column, Position

Description
This method performs a carriage return (CR) followed by a line feed (LF), then resets the tabs.

See also
TBaseReport Class, ColumnStart, CR, LF, ResetTabs

Example (Delphi)
RvNDRWriter1.NewLine;

Example (C++Builder)
RvNDRWriter1->NewLine();

5.79 NewPage

Declaration
procedure NewPage;

Category
Control

Description
This method will end the current page and start printing on a new page. The OnPrintFooter event handler will be called before the current page is finished. The OnPrintHeader and OnNewPage event handlers will be called after the new page has been created.

See also

Example (Delphi)
RvNDRWriter1.NewPage;

Example (C++Builder)
RvNDRWriter1->NewPage();

5.80 NewPara

Declaration
procedure NewPara;
5.81 NextPage

Declaration

procedure NextPage;

Description

This method will go to and print the next page to the preview window. The OnPageChange event handler will be called if the current page number changes.

See also

TRvRenderPreview Class, CurrentPage, PrevPage, OnPageChange

Example (Delphi)

RvRenderPreview1.NextPage;

Example (C++Builder)

RvRenderPreview1->NextPage();

5.82 NoPrinters

Declaration

function NoPrinters: Boolean;

Description

This function will return true if there are no printers defined in the current Windows system and false if there are. TRvRenderPrinter will not function without an installed printer driver; however, TRvNDRWriter and TRvRenderPreview will still work.

See also

TBaseReport Class, NoPrinterPageHeight, NoPrinterPageWidth

Example (Delphi)

// Set up for landscape paper
if NoPrinters then begin
Example (C++Builder)
if (rp1->NoPrinters()) {
    rp1->NoPrinterPageHeight = 8.5;
    rp1->NoPrinterPageWidth = 11.0;
} / if

5.83 Open

Declaration
procedure Open;

Category
Rave

Description
This method will open the report project file defined by ProjectFile to make it available for printing or modification.

See also
TRvProject Class, Close, LoadDesigner, OnAfterOpen, OnBeforeOpen, ProjectFile, Save

5.84 Pie

Declaration
procedure Pie(X1,Y1,X2,Y2,X3,Y3,X4,Y4: double);

Category
Graphics

Description
This method draws a pie slice inside an ellipse bounded by the rectangle defined by (X1,Y1) and (X2,Y2). The slice starts at the intersection of the line drawn between the ellipse center ((X1+X2) / 2.0,(Y1+Y2) / 2.0) and the point (X3,Y3) and is drawn counterclockwise until it reaches the intersection of the line drawn between the ellipse center and the point (X4,Y4).

See also
TBaseReport Class, Arc, Ellipse

Example (Delphi)
SetBrush(clBlack, bsHorizontal, nil);
Pie(3.25,1.0,5.25,3.0,5.25,2.0,0.0,0.0);
SetBrush(clBlack, bsVertical, nil);
Pie(3.25,1.0,5.25,3.0,0.0,0.0,0.0,3.25,7.0);
SetBrush(clBlack, bsBDiagonal, nil);
Pie(3.25,1.0,5.25,3.0,3.25,7.0,5.25,2.0);

Example (C++Builder)
rp1->SetBrush(clBlack, bsHorizontal, NULL);
rp1->Pie(3.25,1.0,5.25,3.0,5.25,2.0,0.0,0.0);
rp1->SetBrush(clBlack, bsVertical, NULL);
rp1->Pie(3.25,1.0,5.25,3.0,0.0,0.0,0.0,3.25,7.0);
rp1->SetBrush(clBlack, bsBDiagonal, NULL);
rp1->Pie(3.25,1.0,5.25,3.0,3.25,7.0,5.25,2.0);

5.85 Polygon

Declaration
procedure Polygon(const Points: array of TPoint);
Methods

5.86 Polyline

Declaration

procedure Polyline(const Points: array of TPoint);

Category

Graphics

Description

This method will draw a series of lines using the current pen connecting the points defined in the open array Points.

See also

TBaseReport Class, CreatePoint, TPoint

Example (Delphi)

PolyLineArr[1] := CreatePoint( 0 , -1 );
PolyLineArr[2] := CreatePoint( 0.59,  0.81);
PolyLineArr[3] := CreatePoint( 0.95, -0.31);
PolyLineArr[4] := CreatePoint( -0.95, -0.31);
PolyLineArr[5] := CreatePoint( 0.59,  0.81);
PolyLineArr[6] := CreatePoint( 0  , -1);
Polyline(PolyLineArr);

Example (C++Builder)

POINT PolyLineArr[7];
PolyLineArr[1] = rp1->CreatePoint( 0 , -1 );
PolyLineArr[2] = rp1->CreatePoint( 0.59,  0.81);
PolyLineArr[3] := CreatePoint( 0.95, -0.31);
PolyLineArr[4] := CreatePoint( -0.95, -0.31);
PolyLineArr[5] := CreatePoint( 0.59,  0.81);
PolyLineArr[6] := CreatePoint( 0  , -1);
rp1->Polyline(PolyLineArr,6);

5.87 PopFont

Declaration

function PopFont: Boolean;
Category
Font

Description
This method will set the font to the setting that was last pushed by PushFont. PopFont will return false if no more fonts exist on the stack.

See also
TBaseReport Class, PushFont

Example (Delphi)
PushFont;
SetFont('Arial',10);
PrintLn('This is in Arial');
PopFont;

Example (C++Builder)
rp1->PushFont();
rp1->SetFont("Arial",10);
rp1->PrintLn("This is in Arial");
rp1->PopFont();

5.88 PopPos

Declaration
function PopPos: Boolean;

Category
Position

Description
This method will set the text cursor position to the setting that was last pushed by PushPos. PopPos will return false if no more positions exist on the stack.

See also
TBaseReport Class, PushPos

Example (Delphi)
PushPos;
PrintXY(4,1.5,'Name');
PopPos;

Example (C++Builder)
rp1->PushPos();
rp1->PrintXY(4,1.5,"Name");
rp1->PopPos();

5.89 PopTabs

Declaration
function PopTabs: Boolean;

Category
Tabs

Description
This method will set the tabs to the setting that was last pushed by PushTabs. PopTabs will return false if no more tabs exist on the stack.

See also
TBaseReport Class, PushTabs
5.90 **PrevPage**

**Declaration**

```delphi
procedure PrevPage;
```

**Category**

Preview

**Description**

This method will go to and print the previous page to the preview window. The `OnPageChange` event handler will be called if the current page number changes.

**See also**

*TRvRenderPreview Class, CurrentPage, NextPage, OnPageChange*

**Example (Delphi)**

```delphi
RvRenderPreview1.PrevPage;
```

**Example (C++Builder)**

```cpp
RvRenderPreview1->PrevPage();
```

### 5.91 Print (TBaseReport)

**Declaration**

```delphi
procedure Print(Text: string);
```

**Category**

Printing

**Description**

This method will print the string, `Text`, at the current text cursor position. If the string contains any tab characters (9) the `Tab` method will be called with the default parameters. The text cursor is left at the end of the string that is printed.

**See also**

*TBaseReport Class, all other print functions*

**Example (Delphi)**

```delphi
RvNDRWriter1.Print('Hello World!');
```

**Example (C++Builder)**

```cpp
RvNDRWriter1->Print("Hello World!");
```

### 5.92 Print (TRpBarsBase)

**Declaration**

```delphi
procedure Print;
```

**Category**

BarCode

**Description**

This method will print the bar code at the current text cursor position. The text cursor is left at the end of the string that is printed.

**See also**

*TRpBarsBase Class, GotoXY, PrintReadable, PrintTop, PrintXY, Text*

**Example (Delphi)**

```delphi
BarCode1.Text := '12345';
BarCode1.Print;
```
Example (C++Builder)
BarCode1->Text = "12345";
BarCode1->Print();

5.93  PrintBitmap

Declaration
procedure PrintBitmap(X,Y: double; ScaleX, ScaleY: double; Bitmap: TBitmap);

Category
Graphics

Description
This method will draw Bitmap on the printer canvas at the point defined by (X,Y). The bitmap will be scaled by the factors ScaleX and ScaleY. (Example (Delphi) A scaling factor of 2 would draw each pixel in the bitmap as 2 pixels on the printer canvas.)

See also
TBaseReport Class, TBaseReport Class, PrintBitmapRect

Example (Delphi)
// Print MyBitmap in upper left corner four times its size
RvNDRWriter1.PrintBitmap( 1.0, 1.0, 2.0, 2.0, MyBitmap );

Example (C++Builder)
RvNDRWriter1->PrintBitmap( 1.0, 1.0, 2.0, 2.0, MyBitmap );

5.94  PrintBitmapRect

Declaration
procedure PrintBitmapRect(X1,Y1,X2,Y2: double; Bitmap: TBitmap);

Category
Graphics

Description
This method will draw Bitmap on the printer canvas stretched or shrunken to fit within the rectangle defined by the points (X1,Y1) and (X2,Y2).

See also
TBaseReport Class, CalcGraphicHeight, CalcGraphicWidth, PrintBitmap, StretchDraw

Example (Delphi)
Bitmap := TBitmap.Create;
Bitmap.LoadFromFile('RpDEMO.BMP');
PrintBitmapRect(5.375,3.5,7.375,5.5,Bitmap);
Bitmap.Free;

Example (C++Builder)
TBitmap* Bitmap = new TBitmap();
Bitmap.LoadFromFile("RpDEMO.BMP");
rp1->PrintBitmapRect(5.375,3.5,7.375,5.5,Bitmap);
delete Bitmap;

5.95  PrintBlock

Declaration
procedure PrintBlock(Text: string; Pos: double; Width: double);

Category
Printing
Methods

Description
This method will print Text on the current line starting at Pos. The text will be block justified within the area defined by Width.

See also
TBaseReport Class, All other print functions

Example (Delphi)
PrintBlock('This is block justified text', 0.5, 4.0);

Example (C++Builder)
rp1->PrintBlock("This is block justified text", 0.5, 4.0);

5.96 PrintCenter

Declaration
procedure PrintCenter(Text: string; Pos: double);

Category
Printing

Description
This method will print the string, Text, on the current line centered horizontally at the position, Pos.

See also
TBaseReport Class, all other print functions

Example (Delphi)
PrintCenter('Text centered at 2.0', 2.0);

Example (C++Builder)
rp1->PrintCenter("Text centered at 2.0", 2.0);

5.97 PrintCharJustify

Declaration
procedure PrintCharJustify(Text: string; Ch: char; Pos: double);

Category
Printing

Description
This method will print a text string out, justified at Pos with respect to the first occurrence of Ch in Text. This can be useful for printing columns of numbers, aligned by the decimal point, when there can be a variable number of digits after the decimal point.

See also
TBaseReport Class, PrintLeft, PrintRight

Example (Delphi)
// Print the number justified by the decimal point
PrintCharJustify(NumStr,'.',4.25);

Example (C++Builder)
rp1->PrintCharJustify(NumStr,".",4.25);

5.98 PrintData

Declaration
procedure PrintData(Value: string);
Category

**Printer**

Description

This method will print the string Value directly to the printer. This can be useful for sending printer specific commands to do things not normally supported by the Windows printer driver (Example (Delphi) electronic forms or HP-GL commands).

**WARNING:**

Including any printer specific commands in your reports may render the reports unusable on other computer systems. Use this method only on a limited basis.

**NOTE:**

This property may be used to send raw HTML tags and text out to the page which is not altered in any way by Rave.

See also

*TBaseReport Class*, *All other print functions*, *PrintDataStream*

**Example** (Delphi)

```delphi
RvNDRWriter1.PrintData( SpecialCodes );
```

**Example** (C++Builder)

```cpp
RvNDRWriter1->PrintData( SpecialCodes );
```

### 5.99 PrintDataStream

**Declaration**

```delphi
procedure PrintDataStream(Stream: TStream; BufSize: longint);
```

<table>
<thead>
<tr>
<th>Category</th>
<th>Printer</th>
</tr>
</thead>
</table>

**Description**

This procedure will send *BufSize* bytes from *Stream* directly to the printer. If *BufSize* is 0 the remaining contents of *Stream* will be send.

**NOTE:**

Depending upon the content of the data sent to the printer, this command may cause your reports to be incompatible across different brands of printers. There are also many printer functions that are incompatible with the Windows printer driver and should not be used.

See also

*TBaseReport Class*, *PrintData*

**Example** (Delphi)

```delphi
MyFileStream := TFileStream.Create('PAGE.PCL', fmOpenRead);
PrintDataStream(MyFileStream,0);
MyFileStream.Free;
```

**Example** (C++Builder)

```cpp
MyFileStream = new TFileStream("PAGE.PCL", fmOpenRead);
rp1->PrintDataStream(MyFileStream,0);
delete MyFileStream;
```

### 5.100 PrintFimA

**Declaration**

```delphi
procedure PrintFimA( X, Y: double );
```

<table>
<thead>
<tr>
<th>Category</th>
<th>BarCode</th>
</tr>
</thead>
</table>

---

82 Rave Reports Developer Reference
**Description**
This method prints a PostNet FIM A at the given X, Y location.

See also
- TRpBarsBase Class, PrintFimB, PrintFimC

**Example (Delphi)**
PostNetBC1.PrintFimA(3.5,0.5);

**Example (C++Builder)**
PostNetBC1->PrintFimA(3.5,0.5);

### 5.101 PrintFimB

**Declaration**
procedures PrintFimB( X,Y: double );

**Category**
BarCode

**Description**
This method prints a PostNet FIM B at the given X, Y location.

See also
- TRpBarsBase Class, PrintFimA, PrintFimC

**Example (Delphi)**
PostNetBC1.PrintFimB(3.5,0.5);

**Example (C++Builder)**
PostNetBC1->PrintFimB(3.5,0.5);

### 5.102 PrintFimC

**Declaration**
procedures PrintFimC( X,Y: double );

**Category**
BarCode

**Description**
This method prints a PostNet FIM C at the given X, Y location.

See also
- TRpBarsBase Class, PrintFimA, PrintFimB

**Example (Delphi)**
PostNetBC1.PrintFimC(3.5,0.5);

**Example (C++Builder)**
PostNetBC1->PrintFimC(3.5,0.5);

### 5.103 PrintFooter

**Declaration**
procedure PrintFooter(Text: string; Justify: TPrintJustify);

**Category**
Printing
Description
This method will print the string, Text, just above the current SectionBottom justified by, Justify, between the current SectionLeft and SectionRight.

See also
TBaseReport Class, All other print functions, GotoFooter

Example (Delphi)
PrintFooter('Date 01/20/95', pjRight);

Example (C++Builder)
PrintFooter("Date 01/20/95", pjRight);

5.104 PrintHeader

Declaration
procedure PrintHeader(Text: string; Justify: TPrintJustify);

Category
Printing

Description
This method will print the string, Text, just below the current SectionTop justified by, Justify, between the current SectionLeft and SectionRight.

See also
TBaseReport Class, All other print functions, GotoHeader

Example (Delphi)
PrintHeader('Report Header Text', pjCenter);

Example (C++Builder)
PrintHeader("Report Header Text", pjCenter);

5.105 PrintHeight

Declaration
procedure PrintHeight(Height: double; PrintTabs: Boolean);

Category
Memo

Description
This method will print the memo buffer for the height specified by the Height parameter. If Height is 0 then all lines in the memo buffer will be printed. If PrintTabs is true, then PrintHeight will print lines of empty tabs for each line that the memo buffer is printed on.

NOTE:
If the entire memo buffer is not printed, the internal position of MemoBuf will be set to the last character that was printed. This will allow the memo buffer to be continued on another page.

NOTE:
You must initialize the TMemoBuf.BaseReport before calling this method.

See also
TMemoBuf Class, BaseReport, TMemoBuf, MemoHeightLeft

5.106 PrintImageRect

Declaration
procedure PrintImageRect(X1,Y1,X2,Y2: double; ImageStream: TStream; ImageType: string);
Category  
Graphics

Description  
This method will draw ImageStream on the printer canvas stretched or shrunken to fit within the rectangle defined by the points (X1,Y1) and (X2,Y2).

See also  
TBaseReport Class, CalcGraphicHeight, CalcGraphicWidth, OnDecodeImage, PrintBitmap, StretchDraw

Example (Delphi)

```delphi
with Sender as TBaseReport do begin  Stream := TMemoryStream.Create;  Image := TJPEGImage.Create;  try    ... := 0;    PrintImageRect(1,1,3,3,Stream,'JPG');  finally    Image.Free;    Stream.Free  end; {tryf}end; {with}
```

Example (C++Builder)

```cpp
TBaseReport *rp = dynamic_cast<TBaseReport*>(Sender);  Stream = new TMemoryStream->Create();  Image = new TJPEGImage->Create();  try {    Image->LoadFromFile("image1.jpg");    Image->SaveToStream(Stream);    Stream->Position = 0;    rp1->PrintImageRect(1,1,3,3,Stream, "JPG");  } finally {    delete Image;    delete Stream;  } {tryf}
```

5.107 PrintJustify

Declaration  
procedure PrintJustify(Text: string; Pos: double; Justify: TPrintJustify; Margin: double; Width: double);

Category  
BarCode, Printing

Description  
This method will print left, right, center or block justified text. The text will be justified inside a measurement rectangle starting at Pos and with a horizontal size of Width. Margin is the spacing between the text and the sides of the measurement rectangle in units.

See also  
TBaseReport Class, PrintBlock, PrintCenter, PrintLeft, PrintRight
5.108 PrintLeft

Declaration
procedure PrintLeft(Text: string; Pos: double);

Category
Printing

Description
This method will print the string Text on the current line left justified at the position Pos.

See also
TBaseReport Class, All other print functions

Example (Delphi)
RVNDRWriter1.PrintLeft('Text left at 4.0', 4.0);

Example (C++Builder)
RvNDRWriter1->PrintLeft("Text left at 4.0", 4.0);

5.109 PrintLines

Declaration
procedure PrintLines(Lines: longint; PrintTabs: Boolean);

Category
Memo

Description
This method will print the memo buffer for the number of lines specified by Lines. If Lines is 0 then all lines in the memo buffer will be printed. If PrintTabs is true, then PrintMemo will print lines of empty tabs for each line that the memo buffer is printed on.

NOTE:
If the entire memo buffer is not printed, the internal position of MemoBuf will be set to the last character that was printed. This will allow the memo buffer to be continued on another page.

NOTE:
You must initialize the TMemoBuf.BaseReport before calling this method.

See also
TMemoBuf Class, BaseReport, MemoLinesLeft, TMemoBuf

5.110 PrintLn

Declaration
procedure PrintLn(Text: string);
Methods

Category
Printing

Description
This method will print the string Text just like the Print method does; however, it also calls NewLine to go to the next line.

See also
TBaseReport Class, All other print functions, NewLine

Example (Delphi)

RvNDRWriter1.Println('Text on a line');
RvNDRWriter1.Println('Text on another line');

Example (C++Builder)

RvNDRWriter1->PrintLn("Text on a line");
RvNDRWriter1->Println("Text on another line");

5.111 PrintMemo

Declaration

procedure PrintMemo(MemoBuf: TMemoBuf; Lines: longint; PrintTabs: Boolean);

Category
Memo

Description
This method will print the memo buffer, MemoBuf, for the number of lines specified by Lines. If Lines is 0 then all lines in the memo buffer will be printed. If PrintTabs is true, then PrintMemo will print lines of empty tabs for each line that the memo buffer is printed on.

NOTE:
If the entire memo buffer is not printed, the internal position of MemoBuf will be set to the last character that was printed. This will allow the memo buffer to be continued on another page.

See also
TBaseReport Class, MemoLines, TMemoBuf

Example (Delphi)

SetColumns(3,0.25);
MemoBuf.PrintStart := ColumnStart;
MemoBuf.PrintEnd := ColumnEnd;
PrintMemo(MemoBuf, ColumnLinesLeft, false);
ClearColumns;

Example (C++Builder)

rp1->SetColumns(3,0.25);
MemoBuf->PrintStart = rp1->ColumnStart;
MemoBuf->PrintEnd := rp1->ColumnEnd;
rp1->PrintLines(MemoBuf, rp1->ColumnLinesLeft, false);
rp1->ClearColumns();

5.112 PrintPage

Declaration

procedure PrintPage(PageNum: word);

Category
Preview
Description
This method will print the page specified by PageNum to the preview window. The OnPageChange event handler will be called if the current page number changes.

See also
TRvRenderPreview Class, OnPageChange, RedrawPage

Example (Delphi)
RvRenderPreview1.PrintPage( 2);

Example (C++Builder)
RvRenderPreview1->PrintPage( 2);

5.113 PrintRight

Declaration
procedure PrintRight(Text: string; Pos: double);

Category
Printing

Description
This method will print the string, Text, on the current line right justified at the position, Pos.

See also
TBaseReport Class, all other print functions

Example (Delphi)
RvNDRWriter1.PrintRight('Right justified at 3.0',3.0 );

Example (C++Builder)
RvNDRWriter1->PrintRight("Right justified at 3.0",3.0 );

5.114 PrintTab

Declaration
procedure PrintTab(Text: string);

Category
Printing

Description
This method will print the next tab setting and then print Text within that tab box. This is equivalent to Print( #9 + Text); with the exception that Text is truncated if it is too long.

See also
TBaseReport Class, Print, PrintLn, Tab

Example (Delphi)
PrintTab(FieldByName('Name'));

Example (C++Builder)
PrintTab(FieldByName("Name"));

5.115 PrintXY (TBaseReport)

Declaration
procedure PrintXY(X,Y: double; Text: string);

Category
Printing
Description
This method will print the string, Text, at the location specified by the point (X,Y).

NOTE:
The Y position will determine the location of the baseline of the printed text.

See also
TBaseReport Class, All other print functions, GotoXY

Example (Delphi)
RvNDRWriter1.PrintXY( 1.0, 2.0, 'Text above (1.0, 2.0)');

Example (C++Builder)
RvNDRWriter1->PrintXY( 1.0, 2.0, "Text above (1.0, 2.0)" );

5.116 PrintXY (TRpBarsBase)

Declaration
procedure PrintXY( X, Y: double );

Category
BarCode

Description
This method will print the bar code at the location specified by the point (X,Y).

NOTE:
The Y position will determine the location of the top of the bar code.

See also
TRpBarsBase Class, Print, PrintReadable, PrintTop, Text

Example (Delphi)
Code2of5.Text := '12345';
Code2of5.PrintXY( 1.0, 2.0 );

Example (C++Builder)
Code2of5->Text = "12345";
Code2of5->PrintXY( 1.0, 2.0 );

5.117 PushFont

Declaration
function PushFont: Boolean;

Category
Font

Description
This method will push the current font onto an internal stack for later retrieval by PopFont.

See also
TBaseReport Class, PopFont

Example
see PopFont

5.118 PushPos

Declaration
function PushPos: Boolean;
**PushTabs**

**Declaration**

```delphi
function PushTabs: Boolean;
```

**Category**

Tabs

**Description**

This method will push the current tab settings onto an internal stack for later retrieval by `PopTabs`.

**See also**

`TBaseReport Class, PopTabs`

**Example**

See `PopTabs`

**RecoverPrinter**

**Declaration**

```delphi
procedure RecoverPrinter;
```

**Category**

Printer

**Description**

This method will recover the printer handle that was released by a prior call to `ReleasePrinter`.

**See also**

`TBaseReport Class, ReleasePrinter`

**Example**

See `ReleasePrinter`

**Rectangle**

**Declaration**

```delphi
procedure Rectangle(X1,Y1,X2,Y2: double);
```

**Category**

Graphics

**Description**

This method will draw a rectangle defined by the points `(X1,Y1)` and `(X2,Y2)`. The rectangle will be drawn with a border of the current `pen` and filled with the current `brush`.

**See also**

`TBaseReport Class, RoundRect`

**Example** (Delphi)

```delphi
RVNDRWriter1.Rectangle(1.0, 1.0, 4.0, 5.0);
```
Example (C++Builder)
RvNDRWriter1->Rectangle(1.0, 1.0, 4.0, 5.0);

5.122 RedrawPage

Declaration
procedure RedrawPage;

Category
Preview

Description
This method will redraw the current page for the preview screen.

See also
TRvRenderPreview Class, PrintPage

Example (Delphi)
RvRenderPreview1.RedrawPage;

Example (C++Builder)
RvRenderPreview1->RedrawPage();

5.123 RegisterGraphic

Declaration
procedure RegisterGraphic( index: integer);

Category
Graphics

Description
This method will help manage repeating, large bitmaps in a print job. You can register up to 10 bitmaps at once by passing in the index value from 1 to 10. With this method only one copy of the bitmap would be stored in the file with all other print functions referencing the same copy.

NOTE:
Use UnregisterGraphic( n ) to make sure that the graphic index that you are using is cleared.

NOTE:
This method will only optimize the execution of a report through TRvNDRWriter.

See also
TBaseReport Class, ReuseGraphic, UnregisterGraphic

Example (Delphi)
Bitmap := TBitmap.Create;
with Sender as TBaseReport do try
  Bitmap.LoadFromFile( 'LOGO.BMP' );
  UnregisterGraphic( 1 );
  while not Table1.EOF do begin
    ReuseGraphic( 1 );
    PrintBitmapRect( 1,1,2,2,Bitmap );
    RegisterGraphic( 1 );
    { other printing code }
  end; { while }
finally
  Bitmap.Free;
end; { with }
Example (C++Builder)

```cpp
Bitmap = new TBitmap();
try {
    Bitmap->LoadFromFile( "LOGO.BMP" );
    rpl->UnregisterGraphic( 1 );
    while (!Table1->Eof) {
        rpl->ReuseGraphic( 1 );
        rpl->PrintBitmapRect( 1,1,2,2, Bitmap );
        rpl->RegisterGraphic( 1 );
    // other printing code
    } while
} __finally {
    delete Bitmap;
} // try
```

### 5.124 ReleasePrinter

**Declaration**

```delphi```
procedure ReleasePrinter;
```delphi```

**Category**

 Printer

**Description**

This method will release the printer handle from Rave so that other components, such as `TPrinterSetupDialog`, can access the printer. Use `RecoverPrinter` to re-initialize Rave and recover the printer handle.

**See also**

 `TBaseReport Class`, `RecoverPrinter`

**Example (Delphi)**

```delphi```
RvNDRWriter1.ReleasePrinter;
PrinterSetupDialog1.Execute;
RvNDRWriter1.RecoverPrinter;
```delphi```

**Example (C++Builder)**

```cpp```
RvNDRWriter1->ReleasePrinter();
PrinterSetupDialog1->Execute();
RvNDRWriter1->RecoverPrinter();
```cpp```

### 5.125 ReplaceAll

**Declaration**

```delphi```
procedure ReplaceAll(SearchText: string; ReplaceText: string; CaseMatters: Boolean);
```delphi```

**Category**

 Memo

**Description**

This method will replace all occurrences of `SearchText` with `ReplaceText`. If `CaseMatters` is true then the case of the characters must match; otherwise, case will not be a factor for a match.

**See also**

 `TMemoBuf Class`, `SearchFirst`, `SearchNext`

**Example (Delphi)**

```delphi```
MemoBuf.ReplaceAll('ame, Name, false);
MemoBuf.ReplaceAll('ddress, Address, false);
```delphi```
5.126 ReportDescToMemo

Declaration
procedure ReportDescToMemo(Memo: TCustomMemo);

Category
Rave

Description
Initializes the memo component, Memo, to the contents of the currently selected report description.

See also
TRvProject Class, ReportDesc, SelectReport

5.127 Reset (TBaseReport)

Declaration
procedure Reset;

Category
Control

Description
This method will reset certain settings (Pen, Brush, Origins, Columns, Tabs, Sections and Text Cursor position) to their default values.

See also
TBaseReport Class, ResetPrinter

Example (Delphi)
RvNDRWriter1.Reset;

Example (C++Builder)
RvNDRWriter1->Reset();

5.128 Reset (TMemoBuf)

Declaration
procedure Reset;

Category
Memo

Description
This method will reset the memo buffer back to the beginning position. Use this method if you have printed a portion of a memo buffer, but want to start at the beginning again.

See also
TMemoBuf Class, Pos

Example (Delphi)
MemoBuf1.Reset;

Example (C++Builder)
MemoBuf1->Reset();
5.129 ResetLineHeight

Declaration
procedure ResetLineHeight;

Category
Position

Description
This method will reset the property LineHeight to the current font if the LineHeightMethod property is equal to IhmFont. Otherwise, ResetLineHeight sets LineHeight to the value of 1.0 LinesPerInch or leaves it alone if LineHeightMethod is IhmUser.

See also
TBaseReport Class, LineHeight, LineHeightMethod

Example (Delphi)
RvNDRWriter1.ResetLineHeight;

Example (C++Builder)
RvNDRWriter1->ResetLineHeight();

5.130 ResetPrinter

Declaration
procedure ResetPrinter;

Category
Printer

Description
This method will reset the current printer for the settings given in the DevMode structure as well as other printer related settings. This function is called automatically whenever you change the current printer or change the orientation.

See also
TBaseReport Class, DevMode

Example (Delphi)
RvNDRWriter1.ResetPrinter;

Example (C++Builder)
RvNDRWriter1->ResetPrinter();

5.131 ResetSection

Declaration
procedure ResetSection;

Category
Position

Description
This method will reset the section values, SectionLeft, SectionRight, SectionTop and SectionBottom to be equal to the current margin settings.

See also
TBaseReport Class, All Margin and Section properties

Example (Delphi)
RvNDRWriter1.ResetSection;
5.132 ResetTabs

Declaration
procedure ResetTabs;

Category
Tabs

Description
This method resets the current tab to the beginning. NewLine calls this function to reset the current tab.

See also
TBaseReport Class, ClearTabs, SetTab

Example (Delphi)
RVNDRWriter1.ResetTabs;

Example (C++Builder)
RvNDRWriter1->ResetTabs();

5.133 RestoreBuffer

Declaration
procedure RestoreBuffer;

Category
Memo

Description
This method will restore the memo buffer to the state it was in during the last call to SaveBuffer.

See also
TMemoBuf Class, SaveBuffer

5.134 RestoreFont

Declaration
function RestoreFont(Index: integer): Boolean;

Category
Font

Description
This method will restore the font settings, saved by a previous SaveFont call, using an Index from 1 to 10. The result of this function will be true if the call was successful.

See also
TBaseReport Class, SaveFont

Example (Delphi)
// Restore the font saved in position 10
RestoreFont (10);

Example (C++Builder)
rp1->RestoreFont (10);
5.135 **RestorePos**

**Declaration**

```delphi
function RestorePos(Index: byte): Boolean;
```

**Category**

Position

**Description**

This method will set the text cursor position to the setting that was last stored at index, \( \text{Index} \), by \( \text{SavePos} \). The valid values for \( \text{Index} \) are 1 to 10.

**See also**

*TBaseReport Class, SavePos*

**Example (Delphi)**

```delphi
RvNDRWriter1.RestorePos(1);
```

**Example (C++Builder)**

```delphi
RvNDRWriter1->RestorePos(1);
```

5.136 **RestoreState**

**Declaration**

```delphi
procedure RestoreState;
```

**Category**

Memo

**Description**

This method restores the cursor position and other state information of the memo buffer back to what it was when \( \text{SaveState} \) was called.

**NOTE:**

This does not effect the contents of the memo buffer.

**See also**

*TMemoBuf Class, Pos, RestoreBuffer, SaveState*

5.137 **RestoreTabs**

**Declaration**

```delphi
function RestoreTabs(Index: integer): Boolean;
```

**Category**

Tabs

**Description**

This method will restore the tab settings, saved by a previous \( \text{SaveTabs} \) call, using an \( \text{Index} \) from 1 to 10. The result of this function will be true if the call was successful.

**See also**

*TBaseReport Class, RestoreTabs, SetTab*

**Example (Delphi)**

```delphi
// Restore the tab settings in position 3
RestoreTabs(3);
```

**Example (C++Builder)**

```delphi
RestoreTabs(3);
```
5.138 **ReuseGraphic**

**Declaration**

```delphi
procedure ReuseGraphic;
```

**Category**

Graphics

**Description**

This method allows the use of a repeating, large bitmaps in a print job that has been registered with the `RegisterGraphic` method. With this method only one copy of the bitmap would be stored in the file with all other print functions referencing the same copy.

**NOTE:**

This method will only optimize the execution of a report through TRvNDRWriter.

**See also**

`TBaseReport Class, RegisterGraphic, UnregisterGraphic`

**Example**

See `RegisterGraphic`

5.139 **RoundRect**

**Declaration**

```delphi
procedure RoundRect(X1,Y1,X2,Y2,X3,Y3: double);
```

**Category**

Graphics

**Description**

This method will draw a rectangle defined by the points (X1,Y1) and (X2,Y2). The corners of the rectangle will be drawn as quarters of an ellipse with a width of X3 and a height of Y3. The rectangle will be drawn with a border of the current `pen` and filled with the current `brush`.

**See also**

`TBaseReport Class, Ellipse, Rectangle`

**Example (Delphi)**

```delphi
RoundRect(1.125,3.5,3.125,5.0,0.25,0.25);
```

**Example (C++Builder)**

```delphi
rp1->RoundRect(1.125,3.5,3.125,5.0,0.25,0.25);
```

5.140 **RTFLoadFromFile**

**Declaration**

```delphi
procedure RTFLoadFromFile( FileName: String);
```

**Category**

Memo

**Description**

Load an RTF text file into the memo buffer.

**See also**

`TMemoBuf Class, LoadFromFile, RTFLoadFromStream`

**Example (Delphi)**

```delphi
MemoBuf1.RTFLoadFromFile('Letter.RTF');
```
Example (C++Builder)
MemoBuf1->RTFLoadFromFile("Letter.RTF");

5.141 RTFLoadFromStream

Declaration
procedure RTFLoadFromStream( stream: TStream; BufSize: longint);

Category
Memo

Description
Loads a RTF text from a stream into the memo buffer. If BufSize is 0 then remaining length of the stream
is read in, otherwise, BufSize bytes are read in.

See also
TMemoBuf Class, LoadFromFile, RTFLoadFromFile

5.142 Save

Declaration
procedure Save;

Category
Rave

Description
This method will save the current report project to the file specified by the ProjectFile property.

See also
TRvProject Class, Close, Open, ProjectFile

5.143 SaveBuffer

Declaration
procedure SaveBuffer;

Category
Memo

Description
This method will save the current memo buffer to a saved buffer that can later be restored with
RestoreBuffer. This can be useful for printing form letters that you need to modify for each print run, but
want to return to the original settings at the beginning of each page.

See also
TMemoBuf Class, FreeSaved, RestoreBuffer

Example (Delphi)
// Save original contents
MemoBuf.SaveBuffer;

Example (C++Builder)
MemoBuf->SaveBuffer();

5.144 SaveFont

Declaration
function SaveFont(Index: integer): Boolean;

Category
Font
### SaveFont

**Description**
This method will save the current font settings using a value of Index from 1 to 10. These settings can later be restored with a call to RestoreFont. The result of this function will be true if the call was successful.

**See also**
[TBaseReport Class, RestoreFont]

**Example (Delphi)**
```delphi
// Save the current font settings in position 2
SaveFont(2);
```

**Example (C++Builder)**
```cpp
rp1->SaveFont(2);
```

### SavePos

**Declaration**

```delphi
function SavePos(Index: byte): Boolean;
```

**Category**

Position

**Description**
This method will store the current text cursor position into an array at index, Index. The valid values for Index are 1 to 10.

**See also**
[TBaseReport Class, RestorePos]

**Example (Delphi)**
```delphi
RvNDWriter1.SavePos(1);
```

**Example (C++Builder)**
```cpp
RvNDWriter1->SavePos(1);
```

### SaveRaveBlob

**Declaration**

```delphi
function SaveRaveBlob(Stream: TStream);
```

**Category**

Rave

**Description**
This method will save the currently loaded report project from the application form to Stream. You should not need to call this function since the normal method of saving the loaded report project is through the TRvProject.StoreRAV property editor.

**See also**
[TRvProject Class, ClearRaveBlob, LoadRaveBlob, RaveBlobDateTime, StoreRAV]

**Example (Delphi)**
```delphi
RvProject1.SaveRaveBlob( MyStream );
```

**Example (C++Builder)**
```cpp
RvProject1->SaveRaveBlob( MyStream );
```
5.147  SaveState

Declaration
procedure SaveState;

Category
Memo

Description
This method saves the current cursor position, Pos, and other state information. You can restore the memo buffer state back by calling RestoreState.

See also
TMemoBuf Class, Pos, RestoreState, SaveBuffer

5.148  SaveTabs

Declaration
function SaveTabs(Index: integer): Boolean;

Category
Tabs

Description
This method will save the current tab settings using a value of Index from 1 to 10. These settings can later be restored with a call to RestoreTabs. The result of this function will be true if the call was successful.

See also
TBaseReport Class, RestoreTabs, SetTab

Example (Delphi)
// Save the current tab settings in position 5
SaveTabs(5);

Example (C++Builder)
SaveTabs(5);

5.149  SaveToFile

Declaration
function SaveToFile(FileName: String);

Category
Rave

Description
This method will save the report project to the file specified by FileName.

See also
TRvProject Class, LoadFromStream, Save, SaveToStream

Example (Delphi)
RvProject1.SaveToFile('Project1.Rav');

Example (C++Builder)
RvProject1->SaveToFile("Project1.Rav");

5.150  SaveToStream (TMemoBuf)

Declaration
procedure SaveToStream(Stream: TStream);
Methods

5.151 SaveToStream (TRvProject)

Declaration

procedure SaveToStream(Stream: TStream);

Category
Rave

Description
This method will save the report project to Stream.

See also
TRvProject Class, LoadFromFile, LoadFromStream, Save, SaveToFile

Example (Delphi)
RvProject1.SaveToStream(RaveStream);

Example (C++Builder)
RvProject1->SaveToStream(RaveStream);

5.152 SearchFirst

Declaration

function SearchFirst(SearchText: string; CaseMatters: Boolean): Boolean;

Category
Memo

Description
This method will start a search process, looking for SearchText from the beginning of the buffer. If CaseMatters is true then the case of the characters must match; otherwise, case will not be a factor for the match. This function will return true if it finds a match and false if it doesn't. Use SearchNext to continue the search after the first occurrence.

See also
TMemoBuf Class, Pos, SearchNext

Example (Delphi)

// Store the number of occurrences of 'APPLE' in apples
Apples := 0;
Found := MemoBuf1.SearchFirst('APPLE', false);
while Found do begin
  Inc(Apples);
  Found := MemoBuf1.SearchNext;
end; { while }
Example (C++Builder)
Apples := 0;
Found := MemoBuf->SearchFirst("APPLE", false);
while (Found == true) {
  Apples++;
  Found = MemoBuf->SearchNext();
}/ while

5.153 SearchNext

Declaration
function SearchNext: Boolean;

Category
Memo

Description
This method will continue a search initiated by SearchFirst. This function will return true if it finds a match and false if it doesn't.

See also
TMemoBuf Class, Pos, SearchFirst

Example
See SearchFirst

5.154 SelectBin

Declaration
function SelectBin(BinName: string): Boolean;

Category
Printer

Description
This method will select a bin containing BinName in its description and return a Boolean value of whether it was successful or not.

NOTE:
This method must be called before any calls to the OnNewPage event.

See also
TBaseReport Class, Bins, OnNewPage, SupportBin

Example (Delphi)
SelectBin('UPPER');

Example (C++Builder)
SelectBin("UPPER");

5.155 SelectPaper

Declaration
function SelectPaper(PaperName: string): Boolean;

Category
Printer

Description
This method will select a paper size containing PaperName in its description and return a Boolean value of whether it was successful or not.
See also

_TBaseReport Class, Papers, SupportPaper_

**Example (Delphi)**

<SelectPaper('LEGAL');>

**Example (C++Builder)**

<SelectPaper("LEGAL");>

### 5.156 SelectPrinter

**Declaration**

```plaintext
function SelectPrinter(SubStr: string; ExactMatch: Boolean): Boolean;
```

**Category**

Printer

**Description**

This method will set the current printer to the first printer in Printers that contains the substring SubStr in its name. ExactMatch determines whether you need an exact match or not on the printer name. If no printer is found then the current printer is not changed and a false value is returned.

See also

_TBaseReport Class, PrinterIndex_

**Example (Delphi)**

<SelectPrinter('Laser', false);>

**Example (C++Builder)**

<SelectPrinter("Laser", false);>

### 5.157 SelectReport

**Declaration**

```plaintext
function SelectReport(ReportName: string; FullName: Boolean): Boolean;
```

**Category**

Rave

**Description**

This method will select the report specified by ReportName. If FullName is true, the function will search the report whose full name matches, otherwise it will search the short names. The result of the function is whether the selection of the report, ReportName, was successful or not.

See also

TRvProject Class, GetReportList, ReportFullName, ReportName

### 5.158 SetBrush

**Declaration**

```plaintext
procedure SetBrush(NewColor: TColor; NewStyle: TBrushStyle; NewBitmap: TBitmap);
```

**Category**

Graphics

**Description**

This method will set the current brush for the given parameters. If a bitmap is not desired, pass in the value of nil.

See also

_TBaseReport Class, CreateBrush, TBrushStyle, TColor_
**Example (Delphi)**

RvNDRWriter1.SetBrush(clBlack, bsClear, nil);

**Example (C++Builder)**

RvNDRWriter1->SetBrush(clBlack, bsClear, NULL);

### 5.159 SetColumns

**Declaration**

procedure SetColumns(NewColumns: integer; Between: double);

**Category**

Column

**Description**

This method sets up a specific number of columns, NewColumns, with a separation, Between, between each column. The column width is calculated to fit within the current SectionLeft and SectionRight.

**See also**

TBaseReport Class, ColumnWidth, SectionLeft, SectionRight, SetColumnWidth

**Example (Delphi)**

// This code shows how to create 4 columns and send output to them. Also see PrintMemo. { with 0.5" between each }
SetColumns(4, 0.5);
while ColumnLinesLeft > 0 do begin
  PrintLn(IntToStr(LinesLeft) + '/' +
          IntToStr(ColumnLinesLeft) + '/' +
          IntToStr(LineNum) + '/' +
          IntToStr(ColumnNum));
end; { while }

**Example (C++Builder)**

rp1->SetColumns(4, 0.5);
while (rp1->ColumnLinesLeft() > 0) {
  rp1->PrintLn( IntToStr(rp1->LinesLeft()) + "/" +
                IntToStr(rp1->ColumnLinesLeft()) + "/" +
                IntToStr(rp1->LineNum) + "/" +
                IntToStr(rp1->ColumnNum));
}

### 5.160 SetColumnWidth

**Declaration**

procedure SetColumnWidth(Width: double; Between: double);

**Category**

Column

**Description**

This method sets the columns to a specific width, Width, with a separation, Between, between each column. The number of columns is calculated to fit within the current SectionLeft and SectionRight.

**See also**

TBaseReport Class, Columns, SectionLeft, SectionRight, SetColumns

**Example (Delphi)**

// Create columns 2 inches wide and a half of an inch apart
RvNDRWriter1.SetColumnWidth( 2.0, 0.5 );
5.161 SetData

**Declaration**

```delphi
procedure SetData(var Buffer; BufSize: longint);
```

**Category**

Memo

**Description**

This method will assign the data in Buffer (for BufSize bytes) to the memo buffer. This can be useful for long strings that are more than 255 characters.

**See also**

TMemoBuf Class, Text

**Example (Delphi)**

```delphi
// Assign a PChar to a memo buffer
MemoBuf.SetData(PCharVar^, StrLen(PCharVar));
```

**Example (C++Builder)**

```cpp
RvNDRWriter1->SetColumnWidth( 2.0, 0.5 );
```

5.162 SetFont

**Declaration**

```delphi
procedure SetFont(NewName: string; NewSize: integer);
```

**Category**

Font

**Description**

This method will set the current font for the given parameters. NewSize is the point size of the font (1/72nds of an inch).

**NOTE:**

If you are using a symbol set, be sure to use FontCharSet after the SetFont method.

**See also**

TBaseReport Class, AssignFont, CreateFont, FontCharSet

**Example (Delphi)**

```delphi
RvNDRWriter1.SetFont( 'Arial', 10 );
```

**Example (C++Builder)**

```cpp
RvNDRWriter1->SetFont( "Arial", 10 );
```

5.163 SetPaperSize

**Declaration**

```delphi
procedure SetPaperSize(Size: integer; Width: double; Height: double);
```

**Category**

Printer

**Description**

This method will set the current paper size for the selected printer to the settings of either the Windows API constant, Size (see TDevMode.dmPaperSize) or if Width and Height are non-zero then it will attempt to set a custom paper size.
NOTE:
Not all printer drivers support custom page sizes and most have minimum and maximum acceptable values.

See also

TBaseReport Class

Example (Delphi)

```delphi
// Set papersize to 10" wide by 12" high then set papersize to 8.5 wide by 14" high
RvNDRWriter1.SetPaperSize(0,10,12);
RvNDRWriter1.SetPaperSize(DMPAPER_LEGAL,0,0);
```

Example (C++Builder)

```cpp
RvNDRWriter1->SetPaperSize(0,10,12);
RvNDRWriter1->SetPaperSize(DMPAPER_LEGAL,0,0);
```

5.164 SetParam

Declaration

```delphi
procedure SetParam(PropertyName: string; ParamValue: string);
```

Category

Rave

Description

SetParam allows the application to pass project parameters to the currently loaded Rave project. These parameters can be used to control dynamic layouts, SQL parameters or other items to print in a visually designed report.

See also

TRvProject Class, GetParam

Example (Delphi)

```delphi
RvProject1.SetParam('UserName',UserName);
```

Example (C++Builder)

```cpp
RvProject1->SetParam("UserName",UserName);
```

Example (in Visual Designer Event)

```delphi
RaveProject.SetParam('UserName',UserName);
```

5.165 SetPen

Declaration

```delphi
procedure SetPen(NewColor: TColor; NewStyle: TPenStyle; NewWidth: integer; NewMode: TPenMode);
```

Category

Graphics

Description

This method will set the current pen for the given parameters. The NewWidth parameter, if positive, is the width of the pen in printer units (dots) and if negative, is the width on the pen in 1/100ths of an inch.

See also

TBaseReport Class, CreatePen, TColor, TPenMode, TPenStyle

Example (Delphi)

```delphi
RvNDRWriter1.SetPen(clBlack,psSolid,-2,pmCopy);
```
5.166 SetPIVar

Declaration
procedure SetPIVar(PIVarName: string; PIVarValue: string);

Category
Printing

Description
This method allows you to initialize the value of a PIVar (Post Initialize Variable). Any PIVars of the same name that were previously printed will show this value. 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. TRvSystem.SystemOptions.soUserFiler must be true if you are using PIVars in your report.

See also
TBaseReport Class, PIVar

Example
see PIVar

5.167 SetRTF

Declaration
procedure SetRTF(var Buffer; BufSize: longint);

Category
Memo, RTF

Components
TRvRenderRTF

Description
Works exactly like SetData, except the data stored in Buffer is RTF text.

See also
RTFText, SetData

5.168 SetTab

Declaration
procedure SetTab(NewPos: double; NewJustify: TPrintJustify; NewWidth: double; NewMargin: double; NewLines: byte; NewShade: byte);

Category
Tabs

Description
This method adds a tab setting.

- **NewPos** defines the starting position of the tab. If NewPos is set to the constant, NA, then the tab will start immediately after the previous tab box.
- **NewJustify** defines whether the tab is left (pjLeft), right (pjRight) or center (pjCenter) justified. If a non-zero width is given, then a tab box is defined and the text will be justified within the tab box rather than justified at the tab position.
- **NewMargin** defines the distance between the tab box side and the text in 1/100ths of an inch.
- **NewLines** uses the BoxLineXxxx constants to define where lines are to be drawn around the tab box.
- **NewShade** defines the percent of background shading to use for this tab box.
Example (Delphi)

ClearTabs;
SetPen(clBlack, psSolid, 1, pmCopy);
SetTab(0.5, pjCenter, 3.5, 0, BOXLINEALL, 0);
SetTab(NA, pjCenter, 1.0, 0, BOXLINEALL, 0);
SetTab(NA, pjCenter, 1.5, 0, BOXLINEALL, 0);
SetTab(NA, pjCenter, 1.5, 0, BOXLINEALL, 0);
Bold := true;
Tab(-2, NA, -2, -2, NA);
Print('Name');
Tab(NA, NA, -2, -2, NA);
Print('Number');
Tab(NA, NA, -2, -2, NA);
Print('Amount 1');
Tab(NA, -2, -2, -2, NA);
PrintLn('Amount 2');
Bold := false;

Example (C++Builder)

rp1->ClearTabs();
rp1->SetPen(clBlack, psSolid, 1, pmCopy);
rp1->SetTab(0.5, pjCenter, 3.5, 0, BOXLINEALL, 0);
rp1->SetTab(NA, pjCenter, 1.0, 0, BOXLINEALL, 0);
rp1->SetTab(NA, pjCenter, 1.5, 0, BOXLINEALL, 0);
rp1->SetTab(NA, pjCenter, 1.5, 0, BOXLINEALL, 0);
rp1->Bold = true;
rp1->Tab(-2, NA, -2, -2, NA);
rp1->Print("Name");
rp1->Tab(NA, NA, -2, -2, NA);
rp1->Print("Number");
rp1->Tab(NA, NA, -2, -2, NA);
rp1->Print("Amount 1");
rp1->Tab(NA, -2, -2, -2, NA);
rp1->PrintLn("Amount 2");
rp1->Bold = false;

5.169 SetTopOfPage

Declaration

procedure SetTopOfPage;

Category

Position

Description

This method will set SectionTop to the bottom of the current line.

See also

TBaseReport Class, MarginTop, SectionTop

Example (Delphi)

RVNDRWriter1.SetTopOfPage;

Example (C++Builder)

RVNDRWriter1->SetTopOfPage();
5.170 ShadeToColor

Declaration
function ShadeToColor(ShadeColor: TColor; ShadePercent: byte): TColor;

Category
Graphics

Description
This function will create a color that only has ShadePercent amount of ShadeColor.

See also
TBaseReport Class, SetBrush, TColor

5.171 ShowPrintDialog

Declaration
function ShowPrintDialog: Boolean;

Category
Printer

Description
Brings up the standard Windows PrintDialog. Use this function instead of Delphi’s TPprintDialog component.

See also
TBaseReport Class, ShowPrinterSetupDialog

Example (Delphi)
if RVNDRWriter1.ShowPrintDialog then begin
  RVNDRWriter1.Execute;
end; { if }

Example (C++Builder)
if (RVNDRWriter1->ShowPrintDialog()) {
  RVNDRWriter1->Execute();
} / if

5.172 ShowPrinterSetupDialog

Declaration
function ShowPrinterSetupDialog: Boolean;

Category
Printer

Description
Brings up the standard Windows PrinterSetupDialog. Use this function instead of Delphi’s TPrinterSetupDialog component.

See also
TBaseReport Class, ShowPrintDialog

Example (Delphi)
if RVNDRWriter1.ShowPrinterSetupDialog then begin
  RVNDRWriter1.Execute;
end; { if }

Example (C++Builder)
if (RVNDRWriter1->ShowPrinterSetupDialog()) {
5.173 SoftLine

Declaration
procedure SoftLine;

Category
RTF

Components
TRvRenderRTF

Description
This method should be called to go to the next line in an RTF exported document without inserting a hard carriage return. For printer based output (TRvRenderPrinter, TRvNDRWriter) this method performs the same as NewLine.

See also
NewLine, NewPara

5.174 Start

Declaration
procedure Start;

Category
Control

Description
For TRvRenderPreview, this method starts a preview session and draws the first page to the preview screen. Use the methods, PrevPage, NextPage, PrintPage, ZoomIn and ZoomOut to interact with the user of the preview screen after Start has been called. For TRvNDRWriter, these methods start a printing job that should be terminated later with a call to Finish. All event handlers are active except for OnPrint and OnPrintPage which are used only with Execute.

See also
TBaseReport Class, Execute, Finish

Example (Delphi)
RvRenderPreview1.Start;

Example (C++Builder)
RvRenderPreview1->Start();

5.175 StretchDraw

Declaration
procedure StretchDraw(const Rect: TRect; Graphic: TGraphic);

Category
Graphics

Description
This method draws the graphic object, Graphic, to the printer canvas stretched or shrunken to fit within the rectangle, Rect.

NOTE:
Do not use StretchDraw for bitmaps, instead use PrintBitmap or PrintBitmapRect.
See also TBaseReport Class, CreateRect, Draw, PrintBitmap, PrintBitmapRect, TGraphic, TRect

5.176 SupportBin

Declaration
function SupportBin(BinNum: integer): Boolean;

Category Printer

Description This method will return true if the bin number (see TDevMode.dmDefaultSource in the Windows API help) specified by BinNum is supported by the printer, otherwise it will return false.

See also TBaseReport Class, SelectBin, other Support methods, TDevMode in Windows API help

5.177 SupportCollate

Declaration
function SupportCollate: Boolean;

Category Printer

Description This method will return true if the printer supports collation, otherwise it will return false.

See also TBaseReport Class, Other Support methods

5.178 SupportDuplex

Declaration
function SupportDuplex: Boolean;

Category Printer

Description This method will return true if the current printer supports duplex (double sided) printing.

See also TBaseReport Class, Duplex, Other Support methods

5.179 SupportOrientation

Declaration
function SupportOrientation: Boolean;

Category Printer

Description This method will return true if the current printer supports orientation changes.

See also TBaseReport Class, Other Support methods
5.180 SupportPaper

Declaration

Category
Printer

Description
This method will return true if the paper number (see TDevMode.dmPaperSize in the Windows API help) specified by PaperNum is supported by the printer, otherwise it will return false.

See also
TBaseReport Class, SelectPaper, SupportPaper, Other Support methods, TDevMode in Windows API help

5.181 Tab

Declaration
procedure Tab(LeftWidth: integer; RightWidth: integer; TopWidth: integer; BottomWidth: integer; ShadeOverride: integer);

Category
Tabs

Description
This method sets the current tab settings to the next available tab. If the next tab is a tab box, then the lines for that tab are drawn at this time as well as any shading that might apply. The LeftWidth, RightWidth, TopWidth and BottomWidth are overrides for the width of the side of the tab box in 1/100ths of an inch, but should be passed as the constant, NA, for the default pen width. If the LeftWidth, RightWidth, TopWidth or BottomWidth parameter(s) are positive, then it is the width of the pen in printer units (dots) and if negative, it is the width on the pen in 1/100ths of an inch. ShadeOverride is a percent of shading to draw the background of the tab box in and will override TabShade or the original setting of the tab box shading.

See also
TBaseReport Class, SetTab, TabShade

Example (Delphi)
with RvNDRWriter1 do begin
  Tab(-2,NA,-2,-2,NA);
  Print('First tab');
  Tab(NA,NA,-2,-2,NA);
  Print('Second tab');
end; { with }

Example (C++Builder)
rpl->Tab(-2,NA,-2,-2,NA);
rpl->Print("First tab");
rpl->Tab(NA,NA,-2,-2,NA);
rpl->Print("Second tab");

5.182 TabEnd

Declaration
function TabEnd(Index: integer): double;

Category
Tabs
Description
This method will return the horizontal ending position of the tab box specified by Index. If Index is 0 then the result will be for the current tab and if Index is greater than the number of defined tabs then a value of 0.0 will be returned.

See also
TBaseReport Class, GetTab, TabStart, TabWidth

Example (Delphi)
// End of current tab region
CurrEnd := RvNDRWriter1.TabEnd( 0);

Example (C++Builder)
CurrEnd = RvNDRWriter1->TabEnd( 0);

5.183 TabStart

Declaration
function TabStart(Index: integer): double;

Category
Tabs

Description
This method will return the horizontal starting position of the tab box specified by Index. If Index is 0 then the result will be for the current tab and if Index is greater than the number of defined tabs then a value of 0.0 will be returned.

See also
TBaseReport Class, GetTab, TabEnd, TabWidth

Example (Delphi)
// Start of current tab region
CurrStart := RvNDRWriter1.TabStart( 0);

Example (C++Builder)
CurrStart = RvNDRWriter1->TabStart( 0);

5.184 TabWidth

Declaration
function TabWidth(Index: integer): double;

Category
Tabs

Description
This method will return the width of the tab box specified by Index. If Index is 0 then the result will be for the current tab and if Index is greater than the number of defined tabs then a value of 0.0 will be returned.

See also
TBaseReport Class, TabEnd, TabStart

Example (Delphi)
// Width of current tab region
CurrWidth := RvNDRWriter1.TabWidth( 0);

Example (C++Builder)
CurrWidth = RvNDRWriter1->TabWidth( 0);
5.185 TextRect

Declaration

procedure TextRect( Rect: TRect; X,Y: double; const Text: string);

Category

Graphics

Description

This method will draw Text clipped within the rectangle defined by Rect. The point (X,Y) defines the starting point of the text. Use CreateRect to initialize Rect.

See also

TBaseReport Class, CreateRect, All print methods, TRect

Example (Delphi)

var   TxtRect: TRect;     .TxtXPos: double;     .TxtYPos: double;     .Txt: string;begin   TxtRect := ...   TxtXPos := 0.95;   TxtYPos := 0.95;   Txt := 'Text is clipped off!';   TextRect(TxtRect, TxtXPos, TxtYPos, Txt);end;

Example (C++Builder)

TRect TxtRect;double TxtXPos;double TxtYPos;AnsiString Txt;TxtRect = rp1->CreateRect(1.00,1.00,3.00,3.00);TxtXPos = 0.95;TxtYPos = 0.95;Txt := "Text is clipped off!";rp1->TextRect(TxtRect, TxtXPos, TxtYPos, Txt);

5.186 TextWidth

Declaration

function TextWidth(Text: string): double;

Category

Position

Description

This method will return the length of the string, Text.

See also

TBaseReport Class

Example (Delphi)

var   TxtLen: double;begin  TxtLen := TextWidth( "How long am I?" );end;

Example (C++Builder)

double TxtLen = rp1->TextWidth("How long am I?");
5.187 UnregisterGraphic

Declaration

procedure UnregisterGraphic( index: integer );

Category

Graphics

Description

This method will help manage repeating, large bitmaps in a print job. This method is used to insure that the index used by RegisterGraphic is clear. You must call this method if you have previously registered a graphic in that index. However, it is safe and recommended to always call UnregisterGraphic before using these graphic index methods.

See also

TBaseReport Class, RegisterGraphic, ReuseGraphic

Example

See RegisterGraphic

5.188 UpdateStatus

Declaration

procedure UpdateStatus;

Category

Misc

Description

This method will update the label defined by StatusLabel with the current information defined by the report status or the items contained in StatusText.

See also

TBaseReport Class, StatusLabel, StatusText

Example (Delphi)

// After report execution, depending on whether the user aborted the report's creation or not, the status bar is updated with the appropriate message.
if Aborted then begin
  StatusFormat := #13'Report Canceled!';
  UpdateStatus;
end else begin
  StatusFormat := #13'Report Completed!';
  UpdateStatus;
end; { else }

Example (C++Builder)

if (rp1->Aborted) {
  rp1->StatusFormat = "\nReport Canceled!";
  rp1->UpdateStatus();
} else {
  rp1->StatusFormat = "\nReport Completed!";
  rp1->UpdateStatus();
}// else

5.189 WriteBCDData

Declaration

function WriteBCDData(FormatData: String; NativeData: Currency): String;
Rave Reports Developer Reference

**Category**
Rave

**Description**
This method writes the contents of a custom BCD field (of type dtBCD) inside of the OnGetRow event of a data connection component. The data for custom fields must be written in the same order as the fields were defined in the OnGetCols event. The FormatData parameter defines the formatted value of the field, but can be blank if no pre-formatted output is needed. The NativeData parameter should contain the unmodified contents of the field.

**See also**
*TRvCustomConnection Class, OnGetCols, OnGetRow, other WriteXxxxData methods*

**Example (Delphi)**
```
Connection.WriteBCDData( ' ',InvoiceAmount );
```

**Example (C++Builder)**
```
Connection->WriteBCDData( " ",InvoiceAmount );
```

### 5.190 WriteBlobData

**Declaration**
```
function WriteBlobData(var: Buffer; Len: Longint): String;
```

**Category**
Rave

**Description**
This method writes the contents of a custom blob field (of type dtBlob / dtGraphic / dtMemo) inside of the OnGetRow event of a data connection component. The data for custom fields must be written in the same order as the fields were defined in the OnGetCols event.

**See also**
*TRvCustomConnection Class, OnGetCols, OnGetRow, other WriteXxxxData methods*

**Example (Delphi)**
```
Connection.WriteBlobData( ' ',CustomerPict );
```

**Example (C++Builder)**
```
Connection->WriteBlobData( " ",CustomerPict );
```

### 5.191 WriteBoolData

**Declaration**
```
function WriteBoolData(FormatData: String; NativeData: Boolean): String;
```

**Category**
Rave

**Description**
This method writes the contents of a custom Boolean field (of type dtBoolean) inside of the OnGetRow event of a data connection component. The data for custom fields must be written in the same order as the fields were defined in the OnGetCols event. The FormatData parameter defines the formatted value of the field, but can be blank if no pre-formatted output is needed. The NativeData parameter should contain the unmodified contents of the field.

**See also**
*TRvCustomConnection Class, OnGetCols, OnGetRow, other WriteXxxxData methods*
Example (Delphi)
Connection.WriteBoolData('',CustomerActive);

Example (C++Builder)
Connection->WriteBoolData("",CustomerActive);

5.192 WriteCurrData

Declaration
function WriteCurrData(FormatData: String; NativeData: Currency): String;

Category
Rave

Description
This method writes the contents of a custom Currency field (of type dtFloat) inside of the OnGetRow event of a data connection component. The data for custom fields must be written in the same order as the fields were defined in the OnGetCols event. FormatData parameter defines the formatted value of the field, but can be blank if no pre-formatted output is needed. NativeData parameter should contain the unmodified contents of the field

See also
TRvCustomConnection Class, OnGetCols, OnGetRow, other WriteXxxxData methods

Example (Delphi)
Connection.WriteCurrData('',InvoiceAmount);

Example (C++Builder)
Connection->WriteCurrData("",InvoiceAmount);

5.193 WriteDateTime

Declaration
function WriteDateTime(FormatData: String; NativeData: TDateTime);

Category
Rave

Description
This method writes the contents of a custom DateTime field (of type dtDate / dtTime / dtDateTime) inside of the OnGetRow event of a data connection component. The data for custom fields must be written in the same order as the fields were defined in the OnGetCols event. FormatData parameter defines the formatted value of the field, but can be blank if no pre-formatted output is needed. NativeData parameter should contain the unmodified contents of the field

See also
TRvCustomConnection Class, OnGetCols, OnGetRow, other WriteXxxxData methods

Example (Delphi)
Connection.WriteDateTime('',Now);

Example (C++Builder)
Connection->WriteDateTime("",Now);

5.194 WriteFloatData

Declaration
function WriteFloatData(FormatData: String; NativeData: Extended): String;

Category
Rave
Description
This method writes the contents of a custom BCD field (of type dtFloat) inside of the OnGetRow event of a data connection component. The data for custom fields must be written in the same order as the fields were defined in the OnGetCols event. FormatData parameter defines the formatted value of the field, but can be blank if no pre-formatted output is needed. NativeData parameter should contain the unmodified contents of the field.

See also
TRvCustomConnection Class, OnGetCols, OnGetRow, other WriteXxxxData methods

Example (Delphi)
Connection.WriteFloatData( "",CustomerBudget );

Example (C++Builder)
Connection->WriteFloatData( "",CustomerBudget );

5.195 WriteIntData

Declaration
function WriteIntData(FormatData: String; NativeData: Integer): String;

Category
Rave

Description
This method writes the contents of a custom integer field (of type dtInteger) inside of the OnGetRow event of a data connection component. The data for custom fields must be written in the same order as the fields were defined in the OnGetCols event. FormatData parameter defines the formatted value of the field, but can be blank if no pre-formatted output is needed. NativeData parameter should contain the unmodified contents of the field.

See also
TRvCustomConnection Class, OnGetCols, OnGetRow, other WriteXxxxData methods

Example (Delphi)
Connection.WriteIntData( "",CustomerCount );

Example (C++Builder)
Connection->WriteIntData( "",CustomerCount );

5.196 WriteNullData

Declaration
function WriteNullData( no parameters );

Category
Rave

Description
This method writes a null inside of the OnGetRow event of a data connection component. The data for custom fields must be written in the same order as the fields were defined in the OnGetCols event.

See also
TRvCustomConnection Class, OnGetCols, OnGetRow, other WriteXxxxData methods

Example (Delphi)
Connection.WriteNullData( );

Example (C++Builder)
Connection->WriteNullData( );
5.197 WriteStrData

Declaration

function WriteStrData(FormatData: String; NativeData: String): String;

Category

Rave

Description

This method writes the contents of a custom String field (of type dtString) inside of the OnGetRow event of a data connection component. The data for custom fields must be written in the same order as the fields were defined in the OnGetCols event. FormatData parameter defines the formatted value of the field, but can be blank if no pre-formatted output is needed. NativeData parameter should contain the unmodified contents of the field.

Please see WriteBlobData for type dtMemo data fields.

See also

TRvCustomConnection Class, OnGetCols, OnGetRow, other WriteXxxxData methods

Example (Delphi)

Connection.WriteStrData( '',CustomerName );

Example (C++Builder)

Connection->WriteStrData( '',CustomerName );

5.198 XD2I

Declaration

function XD2I(Pos: longint): double;

Category

Units

Description

This method will convert horizontal printer canvas measurements (dots) to inch measurements.

See also

TRvRenderPreview Class, All other units conversion functions

Example (Delphi)

// With Units currently set to unInch
XPos := RvNDRWriter1.XD2I( LastXDots );

Example (C++Builder)

XPos = RvNDRWriter1->XD2I( LastXDots );

5.199 XD2U

Declaration

function XD2U(Pos: longint): double;

Category

Units

Description

This method will convert horizontal printer canvas measurements (dots) to unit measurements (defined by Units and UnitsFactor).
See also
   TBaseReport Class, Units, UnitsFactor, All other units conversion functions

**Example (Delphi)**
```delphi
XPos := RvNDRWriter1.XD2U( LastXDots );
```

**Example (C++Builder)**
```cpp
XPos = RvNDRWriter1->XD2U( LastXDots );
```

### 5.200 XI2D

**Declaration**
```plaintext
function XI2D(Pos: double): longint;
```

**Category**
Units

**Description**
This method will convert horizontal inch measurements to printer canvas measurements (dots).

See also
   TBaseReport Class, All other units conversion functions

**Example (Delphi)**
```delphi
// With Units currently set to UnInch
CurrXDots := RvNDRWriter1.XI2D( RvNDRWriter1.XPos );
```

**Example (C++Builder)**
```cpp
CurrXDots = RvNDRWriter1->XI2D( RvNDRWriter1->XPos );
```

### 5.201 XI2U

**Declaration**
```plaintext
function XI2U(Pos: double): double;
```

**Category**
Units

**Description**
This method will convert horizontal inch measurements to unit measurements (defined by Units and UnitsFactor).

See also
   TBaseReport Class, Units, UnitsFactor, All other units conversion functions

**Example (Delphi)**
```delphi
XPos := RvNDRWriter1.XI2U( LastXInch );
```

**Example (C++Builder)**
```cpp
XPos = RvNDRWriter1->XI2U( LastXInch );
```

### 5.202 XU2D

**Declaration**
```plaintext
function XU2D(Pos: double): longint;
```

**Category**
Units

**Description**
This method will convert horizontal unit measurements (defined by Units and UnitsFactor) to printer canvas measurements (dots).
See also [TBaseReport Class, Units, UnitsFactor, All other units conversion functions]

**Example (Delphi)**
```delphi
CurrXDots := RvNDRWriter1.XU2D( RvNDRWriter1.XPos );
```

**Example (C++Builder)**
```cpp
CurrXDots = RvNDRWriter1->XU2D(RvNDRWriter1->XPos );
```

### 5.203 XU2I

**Declaration**
```delphi
function XU2I(Pos: double): double;
```

**Category**

*Units*

**Description**

This method will convert horizontal unit measurements (defined by *Units* and *UnitsFactor*) to inch measurements.

See also [TBaseReport Class, Units, UnitsFactor, All other units conversion functions]

**Example (Delphi)**
```delphi
// With units set to unCM
CurrXInch := RvNDRWriter1.XU2I( RvNDRWriter1.XPos );
```

**Example (C++Builder)**
```cpp
CurrXInch = RvNDRWriter1->XU2I( RvNDRWriter1->XPos );
```

### 5.204 YD2I

**Declaration**
```delphi
function YD2I(Pos: longint): double;
```

**Category**

*Units*

**Description**

This method will convert vertical printer canvas measurements (dots) to inch measurements.

See also [TBaseReport Class, All other units conversion functions]

**Example (Delphi)**
```delphi
// With Units currently set to unInch
YPos := RvNDRWriter1.YD2I( LastYDots );
```

**Example (C++Builder)**
```cpp
YPos = RvNDRWriter1->YD2I( LastYDots );
```

### 5.205 YD2U

**Declaration**
```delphi
function YD2U(Pos: longint): double;
```

**Category**

*Units*
### Description
This method will convert vertical printer canvas measurements (dots) to unit measurements (defined by `Units` and `UnitsFactor`).

See also
- `TBaseReport Class`, `Units`, `UnitsFactor`, All other units conversion functions

**Example (Delphi)**
```delphi
RvNDRWriter1.YPos = RvNDRWriter1.YD2U(LastYDots);
```

**Example (C++Builder)**
```cpp
RvNDRWriter1->YPos = RvNDRWriter1->YD2U(LastYDots);
```

### 5.206 YI2D

**Declaration**
```delphi
function YI2D(Pos: double): longint;
```

**Category**
`Units`

**Description**
This method will convert vertical inch measurements to printer canvas measurements (dots).

See also
- `TBaseReport Class`, All other units conversion functions

**Example (Delphi)**
```delphi
// With Units currently set to unInch
CurrYDots := RvNDRWriter1.YI2D(YPos);
```

**Example (C++Builder)**
```cpp
CurrYDots = RvNDRWriter1->YI2D(RvNDRWriter1->YPos);
```

### 5.207 YI2U

**Declaration**
```delphi
function YI2U(Pos: double): double;
```

**Category**
`Units`

**Description**
This method will convert vertical inch measurements to unit measurements (defined by `Units` and `UnitsFactor`).

See also
- `TBaseReport Class`, `Units`, `UnitsFactor`, All other units conversion functions

**Example (Delphi)**
```delphi
RVNDRWriter1.YPos := RVNDRWriter1.YI2U(LastYInch);
```

**Example (C++Builder)**
```cpp
RVNDRWriter1->YPos = RVNDRWriter1->YI2U(LastYInch);
```

### 5.208 YU2D

**Declaration**
```delphi
function YU2D(Pos: double): longint;
```
Category
Units

Description
This method will convert vertical unit measurements (defined by Units and UnitsFactor) to printer canvas measurements (dots).

See also
TBaseReport Class, Units, UnitsFactor, All other units conversion functions

Example (Delphi)
CurrYDots := RvNDRWriter1.YU2D( RvNDRWriter1.YPos );

Example (C++Builder)
CurrYDots = RvNDRWriter1->YU2D( RvNDRWriter1->YPos );

5.209 YU2I

Declaration
function YU2I(Pos: double): double;

Category
Units

Description
This method will convert vertical unit measurements (defined by Units and UnitsFactor) to inch measurements.

See also
TBaseReport Class, Units, UnitsFactor, All other units conversion functions

Example (Delphi)
// With units set to unCM
CurrYInch := RvNDRWriter1.YU2I( RvNDRWriter1.YPos );

Example (C++Builder)
CurrYInch = RvNDRWriter1->YU2I( RvNDRWriter1->YPos );

5.210 ZoomIn

Declaration
procedure ZoomIn;

Category
Preview

Description
This method will add ZoomInc to the current ZoomFactor and will make the image larger on the screen. If an OnZoomChange event handler is defined, then that event handler will be called and is responsible for redrawing the page otherwise the page is redrawn.

See also
TRvRenderPreview Class, ZoomOut, ZoomInc, ZoomFactor, OnZoomChange

Example (Delphi)
// This code causes the ZoomFactor to be incremented by ZoomInc percent.
RvRenderPreview1.ZoomIn;

Example (C++Builder)
RvRenderPreview1->ZoomIn();
5.211 ZoomOut

Declaration

procedure ZoomOut;

Category

Preview

Description

This method will subtract \textit{ZoomInc} from the current \textit{ZoomFactor} and will make the image smaller on the screen. If an \textit{OnZoomChange} event handler is defined, then that event handler will be called and is responsible for redrawing the page, otherwise the page is redrawn.

See also

\textit{TRvRenderPreview Class, ZoomIn, ZoomInc, ZoomFactor, OnZoomChange}

Example (Delphi)

\begin{verbatim}
RvRenderPreview1.ZoomOut;
\end{verbatim}

Example (C++Builder)

\begin{verbatim}
RvRenderPreview1->ZoomOut();
\end{verbatim}
Properties

Chapter VI
6  Properties

A property defines an attribute of an object. But a property associates specific actions with reading or modifying its data. Properties provide control over access to an object's attributes, and they allow attributes to be computed.

6.1  Aborted

Declaration

property Aborted: Boolean;

Category  Control

Description  This property will be set to true after a call to \texttt{Abort} has been made.

See also  \texttt{TBaseReport Class, Abort}

Example (Delphi)

\begin{verbatim}
RVNDRWriter1.Execute;
if RVNDRWriter1.Aborted then begin
  StatusFormat := #13 + 'Report Canceled!';
end else begin
  StatusFormat := #13 + 'Report Completed!';
end;
UpdateStatus;
\end{verbatim}

Example (C++Builder)

\begin{verbatim}
rp1->Execute();
if (rp1->Aborted) {
  rp1->StatusFormat = "\nReport Canceled!";
} else {
  rp1->StatusFormat = "\nReport Completed!";
}
rp1->UpdateStatus();
\end{verbatim}

6.2  AccuracyMethod

Declaration

property AccuracyMethod: TAccuracyMethod;

Default  \begin{verbatim}
\texttt{amAppearance \{TRvNDRWriter\}}
\texttt{amPositioning \{TRvSystem\}}
\end{verbatim}

Category  Control

Description  This property controls how text is written to the report file. If \textit{AccuracyMethod} is equal to \texttt{amPositioning} then the text is written out in a manner that will be reproduced as accurately as possible on the screen or any printers. If it is equal to \texttt{amAppearance} then the text string is written out as a complete string in the normal fashion. The problem with \texttt{amAppearance} is that screen fonts often do not size the same as printer fonts. Therefore, text strings may appear shorter or longer on the preview screen than they do on the printer.
See also

TBaseReport Class

Example (Delphi)

RvNDRWriter1.AccuracyMethod := amAppearance;

Example (C++Builder)

RvNDRWriter1->AccuracyMethod = amAppearance;

6.3 Active (TRpRender)

Declaration

property Active: Boolean read FActive write FActive

Default

true

Category

Render

Description

From the Print Setup dialog box, select the option to print to file. File types may then be selected from the combobox. Setting the active property to true, which is the default, will cause the component to be listed as one of the file formats to print to.

See also

TRpRender Class, DisplayName

6.4 Active (TRvProject)

Declaration

property Active: Boolean;

Default

false

Category

Rave

Description

You can change or retrieve the active state of a report project with this property. Setting Active to true is the same as calling the Open method while setting Active to false is the same as calling the Close method.

See also


Example (Delphi)

// Same as RaveProject1.Open;
RvProject1.Active := True; { Same as RvProject1.Open; }

Example (C++Builder)

RvProject1->Active = true;

6.5 AscentHeight

Declaration

property AscentHeight: double;

Category

Position
Description
Returns the height of the line font above the baseline.

NOTE:
This applies to the line font only and not to the current text font.

See also
TBaseReport Class, DescentHeight, FontHeight, LineHeight

6.6 BarBottom

Declaration
property BarBottom: double;

Default
pjLeft

Category
BarCode

Description
Sets or returns the location of the bottom of the bar portion of the bar code. The location of the readable text is controlled by PrintReadable and PrintTop properties.

See also
TRpBarsBase Class, BarTop, Bottom, PrintReadable, PrintTop

Example
See Create { bar code }

6.7 BarCodeJustify

Declaration
property BarCodeJustify: TPrintJustify

Default
pjLeft

Category
BarCode

Description
This determines where the bar code is printed relative to the Position property.
- pjLeft: Print the bar code left justified at Position
- pjCenter: Print the bar code centered at Position
- pjRight: Print the bar code right justified at Position

See also
TRpBarsBase Class, Center, Left, Position, Right

Example (Delphi)
// equivalent to Center := 2.5;
Position := 2.5;
BarcodeJustify := pjCenter;

Example (C++Builder)
rpl->Position = 2.5;
rpl->BarcodeJustify = pjCenter;
6.8 BarCodeRotation

Declaration

property BarCodeRotation: TBarCodeRotation;

Default

Rot0

Category

BarCode

Description

This property 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.

<table>
<thead>
<tr>
<th>Rotation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rot0</td>
<td>no rotation</td>
</tr>
<tr>
<td>Rot90</td>
<td>rotate 90 degrees relative to page</td>
</tr>
<tr>
<td>Rot180</td>
<td>rotate 180 degrees relative to page</td>
</tr>
<tr>
<td>Rot270</td>
<td>rotate 270 degrees relative to page</td>
</tr>
</tbody>
</table>

See also

TRpBarsBase Class, Left, Top

Example (Delphi)

// print Bar Code upside down
BarCodeRotation := Rot180;

Example (C++Builder)

rp1->BarCodeRotation = Rot180;

6.9 BarHeight

Declaration

property BarHeight: double;

Default

0.5 (PostNet 0.125)

Category

BarCode

Description

Sets or returns the value for the tallest bar.

See also

TRpBarsBase Class, BarWidth

Example (Delphi)

// Bars will be 3/10 inch tall
BarHeight := 0.3;

Example (C++Builder)

rp1->BarHeight = 0.3;

6.10 BarTop

Declaration

property BarTop: double;

Default

0
Category

BarCode

Description

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

See also

TRpBarsBase Class, BarBottom, PrintReadable, PrintTop, Top

Example (Delphi)

BarCode1.BarTop := 0.5;

Example (C++Builder)

BarCode1->BarTop = 0.5;

6.11 BarWidth

Declaration

property BarWidth: double

Default

0.01 (PostNet 0.020)

Category

BarCode

Description

Sets or returns the value of the narrow bar width.

See also

TRpBarsBase Class, BarHeight, Width

Example (Delphi)

// set narrow bar width to 2/100 ths
BarWidth := 0.02;

Example (C++Builder)

Bar1->BarWidth = 0.02;

6.12 BaseReport (TMemoBuf)

Declaration

property BaseReport: TBaseReport

Default

nil

Category

Memo

Description

Sets or returns the reporting object that the memo will be printed through. There are certain methods that require this property to be initialized before the will print.

See also

TMemoBuf Class, MemoHeightLeft, MemoLinesLeft, PrintHeight, PrintLines

Example (Delphi)

MemoBuf.BaseReport := Sender as TBaseReport;
Example (C++Builder)
MemoBuf->BaseReport = dynamic_cast<TBaseReport*>(Sender);

6.13 **BaseReport (TRpBarsBase)**

Declaration

```plaintext
property BaseReport: TBaseReport
```

Default

nil

Category

*BarCode*

Description

Sets or returns the reporting object that the bar code will be printed through. This property is normally set through the constructor, `Create`.

See also

*TRpBarsBase Class, Create*

Example (Delphi)
```
Barcode1.BaseReport := ( Sender as TBaseReport );
```

Example (C++Builder)
```
Barcode1->BaseReport = dynamic_cast<TBaseReport*>(Sender);
```

6.14 **BaseReport (TRvSystem)**

Declaration

```plaintext
property BaseReport: TBaseReport
```

Default

nil

Category

*Control*

Description

Provides access to the `TBaseReport` object that is created by `RvSystem`, the base class of all output classes. This property will be nil until the `Execute` method is called. It is normally not necessary to access this property since the `TBaseReport` object is passed as the `Sender` parameter for all printing events.

See also

*TRvSystem Class, Execute*

Example (Delphi)
```
RvSystem1.BaseReport.Print('This is a test');
```

or
```
with Sender as TBaseReport do begin
  Print('This is a test'); { Equivalent code inside OnPrint event }
end; { with }
```

Example (C++Builder)
```
r1->BaseReport->Print("This is a test");
```

6.15 **Bins**

Declaration

```plaintext
property Bins: TStrings;
```
Default

(the list of bins for the default printer)

Category

Printer

Description

This property will return a TStringList containing all of the valid printer bins for the current printer.

See also

TBaseReport Class, SelectBin, SupportBin, TStrings

Example (Delphi)

// Display the printer bins in a list box
ListBox1.Items := RvNDRWriter1.Bins;

Example (C++Builder)

ListBox1->Items = RvNDRWriter1->Bins;

6.16 BKColor

Declaration

property BKColor: TColor;

Default
clWhite

Category

Graphics

Description

This property returns or sets the current background color for text output.

See also

TBaseReport Class, TColor, TextBKMode

Example (Delphi)

RvNDRWriter1.BKColor := clWhite;

Example (C++Builder)

RvNDRWriter1->BKColor = clWhite;

6.17 Bold

Declaration

property Bold: Boolean;

Default

false

Category

Font

Description

This property returns or sets the bold attribute for the current font

See also

TBaseReport Class, Italic, Strikeout, Underline
**Example (Delphi)**

```delphi
with RvNDRWriter1 do begin
  Bold := true;
  Print( 'Bold Text' );
  Bold := false;
end; { with }
```

**Example (C++Builder)**

```cpp
rp1->Bold = true;
rp1->Print( "Bold Text" );
rp1->Bold = false;
```

### 6.18 Bottom

**Declaration**

```delphi
property Bottom: double;
```

**Category**

*BarCode*

**Description**

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

**See also**

*TRpBarsBase Class, BarBottom, PrintReadable, PrintTop*

### 6.19 BottomWaste

**Declaration**

```delphi
property BottomWaste: double;
```

**Category**

*Printer*

**Description**

This property returns the waste area on the bottom side of the page that the printer cannot print into. It is a good idea to make sure that the report's margins are greater than or equal to its waste areas.

**See also**

*TBaseReport Class, LeftWaste, MarginBottom, RightWaste, TopWaste*

**Example**

See *LeftWaste*

### 6.20 BoxLineColor

**Declaration**

```delphi
property BoxLineColor: TColor;
```

**Default**

clBlack

**Category**

*Tabs*

**Description**

This property will define the color used to draw the sides of tab boxes defined with SetTab.

**See also**

*TBaseReport Class, SetTab, Tab, TabColor, TColor*
Example (Delphi)
RVNDRWriter1.BoxLineColor := clGreen;

Example (C++Builder)
RVNDRWriter1->BoxLineColor = clGreen;

6.21 BoxLineXxxx constants

Declaration
const BoxLineXxxx

Category
Tabs

Description
BOXLINEONE: No lines drawn.
BOXLINELEFT: Line drawn on left only.
BOXLINERIGHT: Line drawn on right only.
BOXLINETOP: Line drawn on top only.
BOXLINEBOTTOM: Line drawn on bottom only.
BOXLINEALL: Lines drawn on all sides.
BOXLINELEFTRIGHT: Lines drawn on left and right.
BOXLINETOPBOTTOM: Lines drawn on top and bottom.
BOXLINEBOTTON: All lines except indicated are drawn.
BOXLINENOTOP
BOXLINEOBOTTOM
BOXLINENOLEFT
BOXLINENORIGHT

See also
TBaseReport Class, SetTab

Example
see SetTab

6.22 Buffer

Declaration
property Buffer: ^Array[ 0..MaxBufSize ] of Char;

Category
Memo

Description
This property is a pointer to memory buffer used by TMemoBuf.

NOTE:
Not normally necessary to access this property.

See also
TMemoBuf Class, LoadFromFile, SetData, Text

6.23 BufferInc

Declaration
property BufferInc: longint;

Default
256

Category
Memo
Description
This property controls the granularity of the memo buffer when its size changes. Setting this property to 1 will keep the buffer size exactly equal to the size of the text but will be inefficient when the buffer grows or shrinks. Setting this property to a larger value will make editing the memo buffer more efficient.

See also
TMemoBuf Class, MaxSize

Example (Delphi)
MemoBuf.BufferInc := 128;

Example (C++Builder)
MemoBuf->BufferInc = 128;

6.24 CacheDir

Declaration
property CacheDir: String read FCacheDir write FCacheDir

Category
Render HTML PDF

Description
If you are running the HTML component from a server, setting the CacheDir will allow you to specify where the temporary image files will be stored.

See also
TRpRender Class, ServerMode

6.25 Canvas

Declaration
property Canvas: TCanvas;

Category
Printer

Description
This method returns the TCanvas object that is being printed on.

NOTE:
Direct manipulation of the canvas is not supported or captured by TRvNDRWriter (and thus TRvRenderPrinter and TRvRenderPreview).

See also
TBaseReport Class, RpDev, TCanvas

Example (Delphi)
// Save the current canvas
RvNDRWriter1.Canvas.Pen := SavePen;

Example (C++Builder)
RvNDRWriter1->Canvas->Pen = SavePen;

6.26 Center

Declaration
property Center: double;

Default
relative to Left and Right properties
### Category

**BarCode**

### 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

- TRpBarsBase Class, `BarCodeJustify`, `Left`, `Position`, `Right`

**Example** (Delphi)

```delphi
Barcode1.Center := (SectionLeft + SectionRight) / 2.0;
```

**Example** (C++Builder)

```cpp
Barcode1->Center = (rp1->SectionLeft + rp1->SectionRight)/2.0;
```

### 6.27 CheckSum

**Declaration**

```delphi
property CheckSum: Boolean;
```

### Category

**BarCode**

### Description

This property returns the checksum character(s) that is/are calculated using the current value of the Text property. If UseChecksum is true, this value will be automatically included in the bar code.

See also

- TRpBarsBase Class, `UseChecksum`

### 6.28 CodePage

**Declaration**

```delphi
property CodePage: TCodePage128;
```

**Default**

`cpCodeA`

### Category

**BarCode**

### 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

- TRpBarsBase Class

**Example** (Delphi)

```delphi
// set 128 code output to C
CodePage := cpCodeC;
Text := '125692';
```

**Example** (C++Builder)

```cpp
Barcode1->CodePage = cpCodeC;
Barcode1->Text = "125692";
```
6.29 Collate

Declaration

```
property Collate: Boolean
```

Default

(will be equal to the collation setting for the default printer)

Category

Printer

Description

This property will enable or disable collation.

NOTE:

This property is only supported in Delphi 2.0 and will always return false in Delphi 1.0. Not all printer drivers support collation, use SupportCollate to determine availability.

See also

TBaseReport Class, SupportCollate

Example (Delphi)

```
if SupportCollate then begin
  Collate := true;
end; { if }
```

Example (C++Builder)

```
if (rp1->SupportCollate()) {
  rp1->Collate = true;
}
```

6.30 ColumnEnd

Declaration

```
property ColumnEnd: double;
```

Category

Column

Description

This property will return the horizontal ending position of the current column. This can be useful for printing memo buffers inside of a column.

See also

TBaseReport Class, ColumnNum, SetColumns, SetColumnWidth

Example (Delphi)

```
// Print memo buffer
SetColumns(3,0.25);
MemoBuf.PrintStart := ColumnStart;
MemoBuf.PrintEnd := ColumnEnd;
PrintMemo(MemoBuf, ColumnLinesLeft, false);
```

Example (C++Builder)

```
rp1->SetColumns(3,0.25);
MemoBuf->PrintStart = rp1->ColumnStart;
MemoBuf->PrintEnd = rp1->ColumnEnd;
rp1->PrintMemo(MemoBuf, rp1->ColumnLinesLeft(), false);
```
6.31 ColumnLinesLeft

Declaration

function ColumnLinesLeft: integer;

Category

Column

Description
This method returns the number of lines that can be printed above the current SectionBottom for the current column plus all lines that are in remaining columns. This count includes the current line.

See also

TBaseReport Class, all column methods, LinesLeft, SectionBottom

Example (Delphi)

SetColumns(4, 0.5);
while ColumnLinesLeft > 0 do begin
  Println(IntToStr(LinesLeft) + '/' +
  IntToStr(ColumnLinesLeft) + '/' +
  IntToStr(LineNum) + '/' +
  IntToStr(ColumnNum));
end;

Example (C++Builder)

rp1->SetColumns(4, 0.5);
while (rp1->ColumnLinesLeft() > 0) {
  rp1->PrintLn(IntToStr(rp1->LinesLeft()) + AnsiString("/") +
  IntToStr(rp1->ColumnLinesLeft()) +
  AnsiString("/") +
  IntToStr(rp1->LineNum) + AnsiString("/") +
  IntToStr(rp1->ColumnNum));
}/ while

6.32 ColumnNum

Declaration

property ColumnNum: integer;

Default

1

Category

Column

Description
This property will return or set the current column number that the text cursor is on.

See also

TBaseReport Class, Columns, SetColumns, SetColumnWidth

Example (Delphi)

CurrColNum := RvNDRWriter1.ColumnNum;

Example (C++Builder)

CurrColNum = RvNDRWriter1->ColumnNum;

6.33 Columns

Declaration

property Columns: integer;
Properties

**6.34 ColumnStart**

**Declaration**

```delphi
class TBaseReport
begin
  property ColumnStart: double;
end;
```

**Category**

Column

**Description**

This property will return the horizontal starting position of the current column. This can be useful for printing memo buffers inside of a column.

**See also**

TBaseReport Class, ColumnNum, SetColumns, SetColumnWidth

**Example (Delphi)**

```delphi
CurrColStart := RvNDRWriter1.ColumnStart;
```

**Example (C++Builder)**

```delphi
CurrColStart := RvNDRWriter1->ColumnStart;
```

**6.35 ColumnWidth**

**Declaration**

```delphi
class TBaseReport
begin
  property ColumnWidth: double;
end;
```

**Category**

Column

**Description**

This property returns the width of the current column.

**See also**

TBaseReport Class, SetColumns, SetColumnWidth

**Example (Delphi)**

```delphi
CurrColWidth := RvNDRWriter1.ColumnWidth;
```

**Example (C++Builder)**

```delphi
CurrColWidth := RvNDRWriter1->ColumnWidth;
```

**6.36 Copies**

**Declaration**

```delphi
class TBaseReport
begin
  property Copies: integer;
end;
```

**Category**

Column

**Description**

This property returns the number of columns that are available from the last call to SetColumns or SetColumnWidth.

**See also**

TBaseReport Class, ColumnNum, SetColumns, SetColumnWidth

**Example (Delphi)**

```delphi
CurrColumns := RvNDRWriter1.Columns;
```

**Example (C++Builder)**

```delphi
CurrColumns = RvNDRWriter1->Columns;
```
Default
1

Category
Printer

Description
This property returns or sets the current number of copies of the report that will be printed by the printer.

NOTE:
Not all printers support this function, especially non-laserjet printers. Use MaxCopies to determine availability. For these printers, just call the report multiple times or use TRvNDRWriter and TRvRenderPrinter to speed up report generation. Use a value of 0 to retain the setting defined by TPrinterSetupDialog.

See also
TBaseReport Class, MaxCopies

Example (Delphi)
// Print three copies
RvNDRWriter1.Copies := 3;

Example (C++Builder)
RvNDRWriter1->Copies = 3;

6.37 CPI

Declaration
property CPI: double;

Default
10

Category
Misc, Render

Components
TRvRenderText

Description
Sets the Characters Per Inch for translation from horizontal units to text columns.

See also
LeftBorder, LPI, NewPage

Example (Delphi)
WITH RvRenderText1 do begin
  CPI := 16;
  LPI := 8;
  PrintLn('This text is 16 characters per inch');
  PrintLn('With 8 Lines per inch');
end; { with }

Example (C++Builder)
RvRenderText1->CPI = 16;
RvRenderText1->LPI = 8;
RvRenderText1->PrintLn("This text is 16 characters per inch");
RvRenderText1->PrintLn("With 8 Lines per inch");
6.38 CurrentPage

Declaration

property CurrentPage: integer;

Category

Control

Description

This property returns the current page number.

See also

TBaseReport Class

Example (Delphi)

with RvRenderPreview1 do begin
    PageEdit.Text := IntToStr(CurrentPage);
    PageLabel.Caption := 'Page ' +
        IntToStr(CurrentPage-FirstPage+1) +
        ' of ' + IntToStr(Pages);
end; { with }

Example (C++Builder)

PageEdit->Text = IntToStr( RvRenderPreview1->CurrentPage);
PageLabel->Caption = AnsiString("Page ") +
    IntToStr(RvRenderPreview1->CurrentPage -
        RvRenderPreview1->FirstPage+1) +
    AnsiString(" of ") +
    IntToStr( RvRenderPreview1->Pages);

6.39 CurrentPass

Declaration

property CurrentPass: Integer;

Category

Misc

Description

This is the value that will be returned when a %c is encountered in a StatusFormat string. Normally set by Rave and used when printing multiple copies on a printer that does not support that option.

See also

TBaseReport Class, StatusFormat, StatusLabel, StatusText, TotalPasses, UpdateStatus

Example (Delphi)

RvNDRWriter1.StatusFormat := 'Printing page (Pass of )';

Example (C++Builder)

RvNDRWriter1->StatusFormat = "Printing page (Pass of )";

6.40 CursorXPos

Declaration

property CursorXPos: longint;

Category

Position

Description

This property returns the horizontal text cursor position in printer units (dots).
See also
   TBaseReport Class, CursorYPos, XPos, YPos

Example (Delphi)
  CurrentYDots := RvNDRWriter1.CursorYPos;

Example (C++Builder)
  CurrentYDots = RvNDRWriter1->CursorYPos;

6.41 CursorYPos

Declaration
  property CursorYPos: longint;

Category
   Position

Description
  This property returns the vertical text cursor position in printer units (dots).

See also
   TBaseReport Class, CursorXPos, XPos, YPos

Example (Delphi)
  CurrentYDots := RvNDRWriter1.CursorYPos;

Example (C++Builder)
  CurrentYDots = RvNDRWriter1->CursorYPos;

6.42 DataSet

Declaration
  property DataSet: TDataSet;

Default
   nil

Category
   Rave

Description
  Specifies the dataset to use with the current TRvDataSetConnection component.

See also
   TRvDataSetConnection Class

Example (Delphi)
  CustomerCXN.DataSet := CustomerTable;

Example (C++Builder)
  CustomerCXN->DataSet = CustomerTable;

6.43 DefaultDest

Declaration
  property DefaultDest: TReportDest;

Default
   rdPreview
Category  
**ReportSystem**

Description  
This property will determine the default report destination that appears in the setup dialog. If the setup dialog is disabled then `DefaultDest` will determine where the report is sent. Valid values are `rdFile`, `rdPreview` and `rdPrinter`.

See also  
[TRvSystem Class](#), [ReportDest](#), [TReportDest](#)

**Example (Delphi)**  
```delphi
RvSystem1.DefaultDest := rdPrinter;
```

**Example (C++Builder)**  
```cpp
RvSystem1->DefaultDest = rdPrinter;
```

### 6.44 DescentHeight

**Declaration**  
```delphi
property DescentHeight: double;
```

**Category**  
**Position**

**Description**  
Returns the height of the line font below the baseline.

**NOTE:**  
This applies to the line font only and not to the current text font.

See also  
[TBaseReport Class](#), [AscentHeight](#), [FontHeight](#), [LineHeight](#)

### 6.45 DeviceName

**Declaration**  
```delphi
property DeviceName: string;
```

**Category**  
**Printer**

**Description**  
This property will return the device name for the currently selected printer.

See also  
[TBaseReport Class](#), [PrinterIndex](#)

**Example (Delphi)**  
```delphi
// Save current device name
CurrDeviceName := RvNDRWriter1.DeviceName;
```

**Example (C++Builder)**  
```cpp
CurrDeviceName = RvNDRWriter1->DeviceName;
```

### 6.46 DevMode

**Declaration**  
```delphi
property DevMode: PDevMode;
```
Category **Printer**

Description
This property provides access to the `TDevMode` structure for the current printer. After any changes to `DevMode` are made, `ResetPrinter` should be called.

See also
[TBaseReport Class], TDevMode structure in Windows API help.

**Example** (Delphi)

```delphi
// Save current printer device mode and set the print resolution to low
CurrDevMode := RvNDRWriter1.DevMode;
RvNDRWriter1.DevMode^.dmPrintQuality := DMRES_LOW;
```

**Example** (C++Builder)

```cpp
PDevMode CurrDevMode = RvNDRWriter1->DevMode;
RvNDRWriter1->DevMode->dmPrintQuality = DMRES_LOW;
```

### 6.47 **DisplayName**

**Declaration**

```delphi
property DisplayName: string read FDisplayName write SetDisplayName;
```

**Category** **Render**

**Description**
When the Active property is set to true on a TRender component, the component will be listed in the Print to File format options. The text that will show in the drop-down list that allows you to select the component will the same as that listed in the DisplayName property.

See also
[TRpRender Class], **Active**

### 6.48 **DLLFile**

**Declaration**

```delphi
property DLLFile: string;
```

**Default**

' ' (empty)

**Category** **Rave**

**Description**
This property sets the filename that will used if the LoadDesigner property is True. The end user files are either RavePack or RaveSolo DLL depending upon whether or not you are using packages. The end user DLL file can be renamed to better "fit" your project naming conventions.

**NOTE:**
This feature is only available with a Rave EUDL license. See the Nevrona website at http://www.nevrona.com for more information on obtaining an EUDL license.

See also
[TRvProject Class], **LoadDesigner**

**Example** (Delphi)

```delphi
RvProject.DLLFile := 'MyName.DLL';
```
Example (C++Builder)

```
RvProject->DLLFile = "MyName.DLL";
```

### 6.49 DriverName

**Declaration**

```
property DriverName: string;
```

**Category**

Printer

**Description**

This property will return the driver name for the currently selected printer.

**See also**

TBaseReport Class

**Example (Delphi)**

```
// Save current driver name
CurrPrintDriver := RvNDRWriter1.DriverName;
```

**Example (C++Builder)**

```
CurrPrintDriver = RvNDRWriter1->DriverName;
```

### 6.50 Duplex

**Declaration**

```
property Duplex: TDuplex;
```

**Default**

(will be equal to the duplex setting for the default printer)

**Category**

Printer

**Description**

This property will set the duplex mode for the current printer. Not all printers or drivers support duplex printing, use SupportDuplex to determine availability.

- **dupSimplex**: Simplex mode (Duplex mode NOT initialized)
- **dupHorizontal**: Duplex mode initialized - print Head to Toe
- **dupVertical**: Duplex mode initialized - print Head to Head

**See also**

TBaseReport Class, SupportDuplex

**Example (Delphi)**

```
if SupportDuplex then begin
  Duplex := dupVertical;
end; { if }
```

**Example (C++Builder)**

```
if (rp1->SupportDuplex()) {    rp1->Duplex = dupVertical;  }/ if
```

### 6.51 Engine

**Declaration**

```
property Engine: TRpComponent;
```
Default

nil

Category

Rave

Description

This property allows you to define a reporting engine to be used when printing Rave reports through the TRvProject component. If this property is not defined, a default TRvSystem component will be used. TRvNDRWriter and TRvSystem are all valid component classes that can be assigned to this property.

See also

TRvProject Class, Execute, ExecuteReport

Example (Delphi)

RvProject1.Engine := RvSystem1;

Example (C++Builder)

RvProject1->Engine = RvSystem1;

6.52 Extended

Declaration

property Extended: Boolean;

Default

false

Category

BarCode

Description

If this property is true then it will output Extended Code 39 format.

See also

TRpBarsBase Class, ExtendedText

Example (Delphi)

Extended := True;
Text := 'Test Data';

Example (C++Builder)

Extended = true;
Text = "Test Data";

6.53 ExtendedText

Declaration

property ExtendedText: string;

Category

BarCode

Description

When Extended is true, this property will contain the converted Code39 text that will be printed in the bar code.

See also

TRpBarsBase Class, Extended, Text
**Example (Delphi)**

```delphi
ShowMessage('The raw data of this Code 39 BarCode is ' + Code39Bar.ExtendedText);
```

**Example (C++Builder)**

```cpp
ShowMessage("The raw data of this Code 39 BarCode is " + Code39Bar->ExtendedText);
```

### 6.54 Field

**Declaration**

```delphi
gosubroutine Field: TMemoField;
```

**Category**

*Memo*

**Description**

This property will assign the contents of a *TMemoField* component to the memo buffer.

**See also**

*TMemoBuf Class, Pos, Size, TMemoField*

**Example (Delphi)**

```delphi
MemoBuf1.Field := MyMemoField;
```

**Example (C++Builder)**

```cpp
MemoBuf->Field = MyMemoField;
```

### 6.55 FieldAliasList

**Declaration**

```delphi
property FieldAliasList: TStrings;
```

**Default**

*(blank)*

**Category**

*Rave*

**Description**

With this property you can provide aliases or remove fields entirely in your application as far as the Rave designer is concerned. This can be used to provide easier to understand field names, remove unnecessary fields or to remove the need to read large blob fields out of reports that don't use them. The property is a simple string list and each line takes the form of "FieldName=FieldAlias". To remove a field from the list of fields that are sent to Rave, leave the FieldAlias blank. Fields that are not listed in the FieldAliasList will be passed to Rave as is (the default behavior). Field aliases can include blanks or other non-alphanumeric characters, but by doing so, the characters < and > will be automatically added around the field names for all field name references within Rave.

**See also**

*TRvCustomConnection Class*

### 6.56 FileName

**Declaration**

```delphi
property FileName: String;
```

**Default**

"" (empty)
Category **Control**

Description
Specifies the file name to create when the execute method is called. For the RenderText component, if you want to go directly out to a printer in text mode (much faster for dot-matrix printers than going through the Windows printer driver), then define FileName as PRN, LPT1 or LPT2.

See also
[TBaseReport Class](#)

**Example** (Delphi)
```
RvNDRWriter1.FileName := 'DOC1.DOC';
```

**Example** (C++Builder)
```
RvNDRWriter1->FileName = "DOC1.DOC";
```

### 6.57 FirstPage

**Declaration**
```
property FirstPage: integer;
```

**Default**
1

Category **Control**

Description
This property defines the first page of a range of pages to send to the printer. If the current page is outside this range, the property `PageInvalid` will be true.

See also
[TBaseReport Class](#), PageInvalid

**Example** (Delphi)
```
// print only pages 3 through 5
RvNDRWriter1.FirstPage := 3;
RvNDRWriter1.LastPage := 5;
```

**Example** (C++Builder)
```
RvNDRWriter1->FirstPage = 3;
RvNDRWriter1->LastPage = 5;
```

### 6.58 FontAlign

**Declaration**
```
property FontAlign: TFontAlign;
```

**Category** **Font**

Description
Returns or sets the current font alignment.
- `faTop` will align text at the top of the font located at FontTop
- `faBaseline` will align text at the baseline of the font located at FontBaseline
- `faBottom` will align text at the bottom of the font located at FontBottom

See also
[TBaseReport Class](#), Other FontXxxx properties, FontBaseline, FontBottom, FontTop, SetFont, ResetLineHeight
**Example (Delphi)**
```
FontAlign := faTop;
Print('This text is aligned at the top');
FontAlign := faBaseline;
```

**Example (C++Builder)**
```
rp1->FontAlign = faTop;
rp1->Print("This text is aligned at the top");
rp1->FontAlign = faBaseline;
```

### 6.59 FontBaseline

**Declaration**
```
property FontBaseline: double;
```

**Default**
see [ResetLineHeight](#)

**Category**
*Position*

**Description**
Returns or sets the baseline of the line font

**See also**
[TBaseReport Class, FontBottom, FontTop, LineBottom, LineMiddle, LineTop](#)

**Example (Delphi)**
```
FontBaseline := 1.8;
```

**Example (C++Builder)**
```
rp1->FontBaseline = 1.8;
```

### 6.60 FontBottom

**Declaration**
```
property FontBottom: double;
```

**Default**
see [ResetLineHeight](#)

**Category**
*Position*

**Description**
Returns or sets the bottom of the line font

**See also**
[TBaseReport Class, FontBaseline, FontTop, LineBottom, LineMiddle, LineTop](#)

**Example (Delphi)**
```
FontBottom := 2.0;
```

**Example (C++Builder)**
```
rp1->FontBottom = 2.0;
```

### 6.61 FontCharset

**Declaration**
```
property FontCharset: byte;
```
6.62 FontColor

Declaration
property FontColor: TColor;

Default
clBlack

Category
Font

Description
Returns or sets the font color.

See also
TBaseReport Class, Other FontXxxx properties, SetFont, TColor

Example (Delphi)
FontColor := clRed;
Print('This text is in red.');</p>

Example (C++Builder)
rpl->FontColor = clRed;
rpl->Print("This text is in red.");

6.63 FontHandle

Declaration
property FontHandle: HFont;

Category
Font

Description
This property will return the windows handle for the current printer font. This property will not normally be used but is provided for situations that require access to the printer font.

NOTE:
Canvas.Font.Handle will not equal FontHandle.
See also
TBaseReport Class

6.64 FontHeight

Declaration
   property FontHeight: double;

Default
   see ResetLineHeight

Category
   Font

Description
   Returns or sets the height of the line font.

NOTE:
   This applies to the line font only and not the current text font.

See also
   TBaseReport Class, Other FontXxxx properties, AscentHeight, DescentHeight, LineHeight

Example (Delphi)
   FontHeight := 0.25;

Example (C++Builder)
   rp1->FontHeight = 0.25;

6.65 FontName

Declaration
   property FontName: string;

Default
   'System'

Category
   Font

Description
   Returns or sets the current font name.

See also
   TBaseReport Class, Other FontXxxx properties, SetFont

Example (Delphi)
   FontName := 'Times New Roman';

Example (C++Builder)
   rp1->FontName = "Times New Roman";

6.66 FontPitch

Declaration
   property FontPitch: TFontPitch;
6.67 **FontRotation**

**Declaration**

```delphi
property FontRotation: integer;
```

**Default**

0

**Category**

Font

**Description**

Returns or sets the font rotation in degrees from 0 to 359. 0 is for normal text and the angles increase counter-clockwise. The text cursor will be updated according to the FontRotation.

**See also**

TBaseReport Class, Other FontXxxx properties

**Example (Delphi)**

```delphi
FontRotation := 45;
Print('This text is at 45 degrees');
FontRotation := 0;
Print('This is normal text');
```

**Example (C++Builder)**

```c++
rpl->FontRotation = 45;
rpl->Print("This text is at 45 degrees");
rpl->FontRotation = 0;
rpl->Print("This is normal text");
```

6.68 **Fonts**

**Declaration**

```delphi
property Fonts: TStrings;
```

**Default**

(list of fonts supported by the default printer)
Properties

Category
Printer

Description
This property will return a TStringList containing all of the fonts supported by the current printer.

See also
TBaseReport Class, FontName, SetFont, TStrings

Example (Delphi)
// Display the supported fonts in a TComboBox
ComboBox1.Items := RvNDRWriter1.Fonts;

Example (C++Builder)
ComboBox1->Items = RvNDRWriter1->Fonts;

6.69 FontSize

Declaration
property FontSize: double;

Default
10

Category
Font

Description
Returns or sets the point size of the current font.

See also
TBaseReport Class, Other FontXxxx properties, SetFont

Example (Delphi)
FontSize := 8;
Print('Small');
FontSize := 36;
Print('Large');

Example (C++Builder)
rpl->FontSize = 8;
rpl->Print("Small");
rpl->FontSize = 36;
rpl->Print("Large");

6.70 FontTop

Declaration
property FontTop: double;

Default
see ResetLineHeight

Category
Position

Description
Returns or sets the top of the line font

See also
TBaseReport Class, Other FontXxxx properties, LineBottom, LineMiddle, LineTop
Example (Delphi)

// Place the top of the text at 2.25"
FontTop := 2.25;

Example (C++Builder)

rpl->FontTop = 2.25;

6.71 FontWidth

Declaration

property FontWidth: double;

Default

0

Category

Font

Description

This is used to override the average character width for a font in units. To use normal character sizes, specify a value of 0.

See also

TBaseReport Class, FontSize

Example (Delphi)

// set average character width to 1/4 inch
FontWidth := 0.25;

Example (C++Builder)

rpl->FontWidth = 0.25;

6.72 FrameMode

Declaration

property FrameMode: TFrameMode;

Default

fmInside

Category

Graphics

Description

This property determines the technique used to draw the frames (borders) around graphical shapes such as rectangles and ellipses. This property will only have a noticeable effect with large pen widths.
Properties

### fmInside

![Diagram: The frame will be drawn inside the dimensions of the shape]

The frame will be drawn inside the dimensions of the shape.

### fmSplit

![Diagram: The frame will be drawn centered over the dimensions of the shape]

The frame will be drawn centered over the dimensions of the shape.

### fmOutside

![Diagram: The frame will be drawn outside the dimensions of the shape]

The frame will be drawn outside the dimensions of the shape.

**NOTE:**
If you are converting a report from ReportPrinter 2.0 or earlier that uses thick pens, you should set the frame mode to fmSplit which was the mode used by those older versions.

**See also**
- [TBaseReport Class](#), [Ellipse](#), [Rectangle](#)

**Example (Delphi)**

```delphi
FrameMode := fmOutside;
```

**Example (C++Builder)**

```cpp
rp1->FrameMode = fmOutside;
```

---

### 6.73 GridHoriz

**Declaration**

```delphi
property GridHoriz: double;
```

**Default**

0.0

**Category**

[Preview](#)

**Description**

This property will define the horizontal spacing, in units for a grid that will appear on the preview screen. A value of 0.0 will turn off the horizontal grid.

**See also**
- [TRvSystem Class](#), [GridPen](#), [GridVert](#)

**Example (Delphi)**

```delphi
GridHoriz := 0.25;
```

**Example (C++Builder)**

```cpp
GridHoriz = 0.25;
```

---

### 6.74 GridPen

**Declaration**

```delphi
property GridPen: TPen;
```
Default

(Standard Pen)

Category

Preview

Description

This property defines the pen used to draw the grid defined by GridVert and GridHoriz.

See also

TRvSystem Class, GridHoriz, GridVert, RulerType, TPen

Example (Delphi)

GridPen.Color := clAqua;

Example (C++Builder)

GridPen->Color = clAqua;

6.75 GridVert

Declaration

property GridVert: double;

Default

0.0

Category

Preview

Description

This property will define the vertical spacing, in units for a grid that will appear on the preview screen. A value of 0.0 will turn off the vertical grid.

See also

TBaseReport Class, GridHoriz, GridPen

Example (Delphi)

GridVert := 0.5;

Example (C++Builder)

GridVert = 0.5;

6.76 Height

Declaration

property Height: double;

Category

BarCode

Description

This is a read only property which contains the height of the entire bar code. If the 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

TRpBarsBase Class, BarHeight, PrintReadable
Example (Delphi)
TotalBarHeight := Height;
if TotalBarHeight > 1.0 then begin
  BarHeight := 1.0; {set total height to 1.0 inches}
end; { if }

Example (C++Builder)
TotalBarHeight = rp1->Height;
if (TotalBarHeight > 1.0) {
  BarHeight = 1.0; / set total height to 1.0 inches
}/ if

6.77 IgnoreFileSettings

Declaration
   property IgnoreFileSettings: Boolean

Default
   false

Category
   Printer

Description
   When this is set to true it will ignore the printer setup values (Paper Bin, Duplex, Collate, Copies) stored in
   the report file and will use whatever is currently set by the user. This allows a PrinterSetupDialog to be
called before the Execute method.

See also
   TRvRenderPrinter Class, ShowPrintDialog, ShowPrinterSetupDialog

Example (Delphi)
if RvRenderPrinter1.ShowPrinterSetupDialog then begin
  RvRenderPrinter1.IgnoreFileSettings := True;
  RvRenderPrinter1.Execute;
end; {if}

Example (C++Builder)
if (RvRenderPrinter1->ShowPrinterSetupDialog()) {
  RvRenderPrinter1->IgnoreFileSettings = true;
  RvRenderPrinter1->Execute();
}/ if

6.78 ImageQuality

Declaration
   property ImageQuality: TImageQualityRange read FImageQuality write FImageQuality

Default
   JPG'S image quality set to 90

Category
   Render, PDF

Description
   When sending images out to PDF, the bitmaps, metafiles, etc., are converted to JPG's in order to allow
   PDF to print them. By default the image quality for JPG's is set to 90. If you need to change the image
   quality, you can do this by setting the ImageQuality property. Valid values are 1 to 100 with 100 being the
   absolute best quality available.
See also
TRpRender Class, BufferDocument, MetafileDPI

6.79 Italic

Declaration
property Italic: Boolean;

Default
false

Category
Font

Description
This property returns or sets the italic attribute for the current font.

See also
TBaseReport Class, Bold, Strikeout, Underline

Example (Delphi)
Italic := true;
Print('Italic Text');
Italic := false;

Example (C++Builder)
rp1->Italic = true;
rp1->Print("Italic Text");
rp1->Italic = false;

6.80 Justify

Declaration
property Justify: TPrintJustify;

Default
pjLeft

Category
Memo

Description
This property sets the justification that PrintMemo will use when printing the memo buffer. Valid values are
pjBlock
pjCenter
pjLeft
pjRight

See also
TMemoBuf Class, PrintMemo

Example (Delphi)
MemoBuf.Justify := pjBlock; { Set block justification }

Example (C++Builder)
MemoBuf->Justify = pjBlock; / Set block justification

6.81 LastPage

Declaration
property LastPage: integer;
Default
9999

Category
Control

Description
This property defines the last page for a range of pages to send to the printer. If the current page is outside of this range, the property PagInvalid will be true.

See also
TBaseReport Class, PagInvalid

Example (Delphi)
// Print only pages 3 through 5
RVNDWriter1.FirstPage := 3;
RVNDWriter1.LastPage := 5;

Example (C++Builder)
RVNDWriter1->FirstPage = 3;
RVNDWriter1->LastPage = 5;

6.82 Left

Declaration
property Left: double;

Default
XPos

Category
BarCode

Description
Sets or returns the position for the left edge of the bar code. When a value is assigned to Left, the BarCodeJustify property is set to pjLeft as well.

See also
TRpBarsBase Class, BarCodeJustify, Center, Position, Right

Example (Delphi)
// start at 4.5 inches from left side
Left := 4.5;

Example (C++Builder)
Left = 4.5;

6.83 LeftWaste

Declaration
property LeftWaste: double;

Category
Printer

Description
This property returns the waste area on the left side of the page that the printer cannot print into. It is a good idea to make sure that the report's margins are greater than or equal to its waste areas.

See also
TBaseReport Class, BottomWaste, MarginLeft, RightWaste, TopWaste
Example (Delphi)
// Don't output in the printer waste regions
if MarginLeft < LeftWaste then begin
    MarginLeft := LeftWaste;
end; { if }
if MarginRight < RightWaste then begin
    MarginRight := RightWaste;
end; { if }
if MarginTop < TopWaste then begin
    MarginTop := TopWaste;
end; { if }
if MarginBottom < BottomWaste then begin
    MarginBottom := BottomWaste;
end; { if }

Example (C++Builder)
if (rp1->MarginLeft < rp1->LeftWaste) {
    rp1->MarginLeft = rp1->LeftWaste;
} if
if (rp1->MarginRight < rp1->RightWaste) {
    rp1->MarginRight = rp1->RightWaste;
} if
if (rp1->MarginTop < rp1->TopWaste) {
    rp1->MarginTop = rp1->TopWaste;
} if
if (rp1->MarginBottom < rp1->BottomWaste) {
    rp1->MarginBottom = rp1->BottomWaste;
} if

6.84 LineBottom

Declaration

property LineBottom: double;

Default

(Bottom of the current line)

Category

Position

Description

Returns or sets the bottom of the text line.

See also

TBaseReport Class, FontBaseline, FontBottom, FontTop, LineMiddle, LineTop

Example (Delphi)
// Place the text right on the bottom of the section
LineBottom := SectionBottom;

Example (C++Builder)
rp1->LineBottom = rp1->SectionBottom;

6.85 LineHeight

Declaration

property LineHeight: double;

Category

Position
Description
This property returns or sets the current height of a line. If a value is assigned to `LineHeight` then `LineHeightMethod` will be set to `lhmUser`.

See also
- `TBaseReport Class`, `LineHeightMethod`

Example (Delphi)
```delphi
// Save current line height to a temporary variable
CurrHeight := RvNDRWriter1.LineHeight
```

Example (C++Builder)
```c++
CurrHeight = RvNDRWriter1->LineHeight
```

6.86 **LineHeightMethod**

Declaration
```
property LineHeightMethod: TLineHeightMethod;
```

Default
- `lhmLinesPerInch`, `lhmFont` for `TRvSystem`

Category
- `Position`

Description
This property returns or sets the current method for calculating line heights. If equal to `lhmLinesPerInch`, then the `LinesPerInch` property determines the line height. If equal to `lhmFont`, then the current font determines the line height when a new line is generated. If equal to `lhmUser` the line height will not change unless the user changes `LineHeight` directly.

See also
- `TBaseReport Class`, `LinesPerInch`

Example (Delphi)
```delphi
RvNDRWriter1.LineHeightMethod := lhmFont;
```

Example (C++Builder)
```c++
RvNDRWriter1->LineHeightMethod = lhmFont;
```

6.87 **LineMiddle**

Declaration
```
property LineMiddle: double;
```

Default
- `(Middle of current line)`

Category
- `Position`

Description
This property returns or sets the middle of the current text line. It is useful for aligning the middle of the current line with graphics that might be placed around the text (e.g., bullets, etc.)

See also
- `TBaseReport Class`, `FontBaseline`, `FontBottom`, `FontTop`, `LineBottom`, `LineTop`

Example (Delphi)
```delphi
LineMiddle := 2.0;
```
Example (C++Builder)
rp1->LineMiddle = 2.0;

6.88 LineNum

Declaration

property LineNum: integer;

Default

1

Category

Position

Description

This property returns or sets the current line number. This property is highly dependent upon the current LineHeightMethod as well as the size of the current font if LineHeightMethod is equal to lhmFont. LineNum may not represent the actual line number if the report is jumping around the page instead of calling Prints and PrintLns.

See also

TBaseReport Class, LineHeight, LineHeightMethod

Example (Delphi)

with RVNDRWriter1 do  if Odd(LineNum) then begin    TabShade := 0;  end else begin    TabShade := 15;  end;  { if }end;  { with }

Example (C++Builder)

if ((rp1->LineNum 2) == 1) {    rp1->TabShade = 0;  }  else {    rp1->TabShade = 15;  }  else

6.89 LinesPerInch

Declaration

property LinesPerInch: integer;

Default

6

Category

Position

Description

This property will return or set the number of lines per inch if the LineHeightMethod property is equal to lhmLinesPerInch.

See also

TBaseReport Class, LineHeightMethod

Example (Delphi)

RVNDRWriter1.LineHeightMethod := lhmLinesPerInch;
Example (C++Builder)
RVNDRWriter1->LineHeightMethod = lhmLinesPerInch;

6.90 LineTop

Declaration
property LineTop: double;

Default
(Top of the current line)

Category
Position

Description
Returns or sets the top of the text line

See also
TRvBaseReport Class, FontBaseline, FontTop, LineBottom, LineMiddle

Example (Delphi)
// Place the top of the line at 4.0"
LineTop := 4.0;

Example (C++Builder)
LineTop = 4.0;

6.91 LoadDesigner

Declaration
property LoadDesigner: Boolean;

Default
false

Category
Rave

Description
This property determines if the end user designer will be loaded or not. If the LoadDesigner property is True then the filename in the DLLFile property will be loaded. The end user files are either RavePack or RaveSolo DLL depending upon whether you are using packages or not.

NOTE:
This feature is only available with a Rave BEX with EUDL license. See the Nevrona website at http://www.nevrona.com for more information on obtaining an EUDL license.

See also
TRvProject Class, DLLFile, Open

6.92 LocalFilter

Declaration
property LocalFilter: Boolean;

Default
False

Category
Rave

TRvQueryConnection and TRvTableConnection
Description
This property will determine whether filtering is done locally inside of the data connection component or whether it will rely on the filtering capabilities of the database. Local is provided to support filtering on fields that do not allow exact representation in string form (floating point / date-time fields).

See also
TRvCustomConnection Class

Example (Delphi)
RvCustomConnection1.LocalFilter := True;

Example (C++Builder)
RvCustomConnection1->LocalFilter = True;

6.93 LPI

Declaration
property LPI: double;

Default
6

Category
Misc

Components
TRvRenderTEXT

Description
Sets the Lines Per Inch for translation from vertical units to text lines.

See also
CPI, NewLine, TopBorder

Example (Delphi)
WITH RvRenderText1 do begin
  CPI := 16;
  LPI := 8;
  PrintLn('This text is 16 characters per inch');
  PrintLn('With 8 Lines per inch');
end; { with }

Example (C++Builder)
RvRenderText1->CPI = 16;
RvRenderText1->LPI = 8;
RvRenderText1->PrintLn("This text is 16 characters per inch");
RvRenderText1->PrintLn("With 8 Lines per inch");

6.94 MacroData

Declaration
property MacroData: TStrings;

Default
empty list

Category
Printing

Description
This property sets or returns the user-defined macro string in a list of strings for midUser01 to midUser20
See also
TBaseReport Class, Macro, TMacroID, TStrings

Example (Delphi)
// Add current user name for Macro(midUser01)
MacroData.Add(UserName);
RvRenderPrinter1.Execute;

Example (C++Builder)
rp1->MacroData->Add(UserName);
rp1->Execute();

6.95 MarginBottom

Declaration
property MarginBottom: double;

Default
0.0

Category
Position

Description
These properties return or set the current margin settings. Margins have no direct effect on printing other than providing values to reset the current section when a new page is generated or when ResetSection is called. Changing a margin setting will change the same section setting to the same measurement.

See also
TBaseReport Class, MarginLeft, MarginRight, MarginTop, section properties, ResetSection

Example (Delphi)
// This code shows how to set these properties. Also see PrintFooter
MarginLeft := 0.5;
MarginRight := 0.5;
MarginTop := 0.5;
MarginBottom := 1.0;

Example (C++Builder)
rp1->MarginLeft := 0.5;
rp1->MarginRight := 0.5;
rp1->MarginTop := 0.5;
rp1->MarginBottom := 1.0;

6.96 MarginLeft

Declaration
property MarginLeft: double;

Default
0.0

Category
Position

Description
These properties return or set the current margin settings. Margins have no direct effect on printing other than providing values to reset the current section when a new page is generated or when ResetSection is called. Changing a margin setting will change the same section setting to the same measurement.
See also
TBaseReport Class, MarginBottom, section properties, ResetSection

**Example (Delphi)**
// This code shows how to set these properties. Also see PrintFooter
MarginLeft := 0.5;

**Example (C++Builder)**
rpl->MarginLeft := 0.5;

### 6.97 MarginMethod

**Declaration**
```pascal
property MarginMethod: TMarginMethod;
```

**Default**
mmFixed

**Category**
Preview

**Description**
This property returns or sets the method used to draw the blank margin around the preview page. The setting *mmFixed* will keep the border the same size no matter what the value of *ZoomFactor*. The setting *mmScaled* will grow and shrink the border so that it maintains the same ratio as the rest of the page.

See also
TRvRenderPreview Class, MarginPercent

**Example (Delphi)**
RvRenderPreview1.MarginMethod := mmScaled;

**Example (C++Builder)**
RvRenderPreview1->MarginMethod = mmScaled;

### 6.98 MarginPercent

**Declaration**
```pascal
property MarginPercent: double;
```

**Default**
0.0

**Category**
Preview

**Description**
This property defines the percent of the page width that will appear as blank space around the preview page. A value of 0.0 would have no border. A value of 2.5 would create a border that is equal to 2.5% of the page width.

See also
TRvRenderPreview Class, MarginMethod

**Example (Delphi)**
// Set a 1 percent border
RvRenderPreview1.MarginPercent := 1.0;

**Example (C++Builder)**
RvRenderPreview1->MarginPercent = 1.0;
6.99 MarginRight

Declaration
   property MarginRight: double;

Default
   0.0

Category
   Position

Description
   These properties return or set the current margin settings. Margins have no direct effect on printing other than providing values to reset the current section when a new page is generated or when ResetSection is called. Changing a margin setting will change the same section setting to the same measurement.

See also
   TBaseReport Class, MarginBottom, MarginLeft, MarginTop, section properties, ResetSection

Example (Delphi)
   MarginRight := 0.5;

Example (C++Builder)
   rp1->MarginRight := 0.5;

6.100 MarginTop

Declaration
   property MarginTop: double;

Category
   Position

Description
   These properties return or set the current margin settings. Margins have no direct effect on printing other than providing values to reset the current section when a new page is generated or when ResetSection is called. Changing a margin setting will change the same section setting to the same measurement.

See also
   TBaseReport Class, MarginBottom, MarginLeft, MarginRight, section properties, ResetSection

Example (Delphi)
   MarginTop := 0.5;

Example (C++Builder)
   rp1->MarginTop := 0.5;

6.101 MaxCopies

Declaration
   property MaxCopies: longint;

Default
   (maximum number of copies supported by the default printer)

Category
   Printer

Description
   This property returns the maximum number of copies supported by the current printer.
See also
TBaseReport Class, Copies

**Example (Delphi)**
if MaxCopies = 1 then begin
  Copies := 1;
end; { if }

**Example (C++Builder)**
if (rp1->MaxCopies == 1) {
  rp1->Copies = 1;
} // if

### 6.102 MaxSize

**Declaration**

```plaintext
property MaxSize: longint;
```

**Default**

0

**Category**
Memo

**Description**
This property returns or sets the current size of the memo buffer. This is the size of available space and not the size of valid data (see **Size**). If a new value is assigned to **MaxSize**, the buffer will be adjusted to the smallest multiple of **BufferInc** that is greater than or equal to the desired new size.

See also
TMemoBuf Class, BufferInc, Size

**Example (Delphi)**
// Allocate at least 1000 characters
MemoBuf.MaxSize := 1000;

**Example (C++Builder)**
MemoBuf->Memo = 1000;

### 6.103 Memo

**Declaration**

```plaintext
property Memo: TMemo;
```

**Category**
Memo

**Description**
This property will assign the contents of a TMemo component to a memo buffer.

See also
TMemoBuf Class, Field, Text, TMemo component in Delphi help

**Example (Delphi)**
// Copy Memo1 into MemoBuf
MemoBuf. Memo := Memo1;

**Example (C++Builder)**
MemoBuf->Memo = Memo1;
6.104 MetafileDPI

Declaration

property MetafileDPI: Boolean; read FMetafileDPI write FMetafileDPI

Default
300

Category
Render PDF

Description
The MetafileDPI property can be used to increase or decrease the dots per inch used when saving the images in the PDF file. The higher the dots per inch the better quality the image will appear to have. The down side to a higher dots per inch is that the file size of the PDF will increase.

See also
TRpRender Class, ImageQuality

6.105 Monochrome

Declaration

property Monochrome: Boolean;

Default
false

Category
Preview

Description
This property defines whether the preview page is drawn in color or monochrome. A setting of true can drastically save memory, especially if the system is running in 8-bit or 24-bit color. Shadows will be disabled if Monochrome is true.

See also
TRvRenderPreview Class, ShadowDepth

Example (Delphi)
RvRenderPreview1.Monochrome := true;

Example (C++Builder)
RvRenderPreview1->Monochrome = true;

6.106 NoBufferLine

Declaration

property NoBufferLine: Boolean;

Default
false

Category
Graphics

Description
By default Rave buffers lines until the end of each page so that it can optimize the output for faster printing. Turn this option off if you need to have lines printed before other objects on a page.
See also
TBaseReport Class, LineTo, MoveTo

**Example (Delphi)**
// turn off line buffering
RvNDRWriter1.NoBufferLine := true;

**Example (C++Builder)**
RvNDRWriter1->NoBufferLine = true;

### 6.107 NoCRLF

**Declaration**

property NoCRLF: Boolean;

**Default**
false

**Category**
Memo

**Description**
This property will control whether PrintMemo finishes with a carriage-return linefeed (if false) or not (if true).

See also
TMemoBuf Class, PrintMemo

**Example (Delphi)**
// Don't do a NewLine after PrintMemo()
MemoBuf.NoCRLF := true;

**Example (C++Builder)**
MemoBuf->NoCRLF = true;

### 6.108 NoNewLine

**Declaration**

property NoNewLine: Boolean;

**Default**
false

**Category**
Memo

**Description**
Prevents the writing of an extra new line after the memo has been printed.

See also
TMemoBuf Class, PrintMemo

**Example (Delphi)**
MemoBuf.NowNewLine := true;

**Example (C++Builder)**
MemoBuf->NowNewLine = true;
6.109 NoNTColorFix

**Declaration**

```delphi
property NoNTColorFix: Boolean;
```

**Default**

`false`

**Category**

`Printer`

**Description**

Monochrome printers in Windows NT cannot print colors as shades of gray. Instead, any color other than black is printed as if it was white. Since this behaviour is often not desired when printing text, Rave will convert all text colors, except white, as black if the output is being sent to a monochrome printer on Windows NT. The NoNTColorFix property, if set to true, allows you to disable this color conversion but is generally not needed.

**See also**

`TBaseReport Class`, `FontColor`

**Example** (Delphi)

```delphi
// Disable NT color conversion
NoNTColorFix := true;
```

**Example** (C++Builder)

```cpp
NoNTColorFix = true;
```

6.110 NoPrinterPageHeight

**Declaration**

```delphi
property NoPrinterPageHeight: double;
```

**Default**

`11.0`

**Category**

`Printer`

**Description**

These properties define the page width and height for the print preview screen if no printers are defined for the current Windows system.

**See also**

`TBaseReport Class`, `NoPrinters`

**Example**

See `NoPrinters`

6.111 NoPrinterPageWidth

**Declaration**

```delphi
property NoPrinterPageWidth: double;
```

**Default**

`8.5`

**Category**

`Printer`
Description
These properties define the page width and height for the print preview screen if no printers are defined for the current Windows system.

See also
   TBaseReport Class, NoPrinters

Example
   See NoPrinters

6.112 OnCompress

Declaration
   property OnCompress: TCompressEvent;

Default
   empty

Category
   Render  PDF

Description
This property that can be assigned to an event. The event must be defined if you want to compress the page stream in the PDF file. You will also need to set the Use Compression property to true if you want the page stream compressed.

See also
   TRpRender Class, UseCompression

Example (Delphi)
   // Typically, the code defined inside the OnCompress event will be something similar to this:
   with TCompressionStream.Create(clMax, OutStream) do try
      CopyFrom(InStream, InStream.Size);
   finally
      Free
   end;  { with }

6.113 Orientation

Declaration
   property Orientation: TOrientation;

Default
   poPortrait

Category
   Printer

Description
This property will return or set the current page orientation to either poPortrait or poLandscape. Use poDefault to retain the setting defined by TPrinterSetupDialog.

See also
   TBaseReport Class

Example (Delphi)
   RVNDRWriter1.Orientation := poLandscape;

Example (C++Builder)
   RVNDRWriter1->Orientation = poLandscape;
6.114 OriginX

Declaration

    property OriginX: double;

Default

    0.0

Category

    Position

Description

These properties return or set the currently defined origin. Origins can be very useful for printing similar items that are at different locations of the page (Example (Delphi) labels).

See also

    TBaseReport Class, OriginY

Example (Delphi)

    RvNDRWriter1.OriginX := 2.0;

Example (C++Builder)

    RvNDRWriter1->OriginX = 2.0;

6.115 OriginY

Declaration

    property OriginY: double;

Default

    0.0

Category

    Position

Description

These properties return or set the currently defined origin. Origins can be very useful for printing similar items that are at different locations of the page (Example (Delphi) labels).

See also

    TBaseReport Class, OriginX

Example (Delphi)

    RvNDRWriter1.OriginY := 2.0;

Example (C++Builder)

    RvNDRWriter1->OriginY = 2.0;

6.116 OutputFileName

Declaration

    property OutputFileName: TFileName;

Default

    '' (empty)

Category

    Printer
Description
Specifies the file name that the report output should be sent to. This is a file with printer commands that can be later printed using a command from the DOS prompt like: "COPY /b TEST.DAT PRN"

See also
TRvSystem Class, OutputName

Example (Delphi)
RvSystem1.OutputFileName := 'TEST.DAT';

Example (C++Builder)
RvSystem1->OutputFileName = "TEST.DAT'';

6.117 OutputInvalid

Declaration
property OutputInvalid: Boolean;

Default
true

Category
Control

Description
Returns true if the current report destination is invalid. Will also return true if the report has been aborted or is finished executing. This can occur if the user has selected a page range that does not include the current page or the report has been aborted.

See also
TBaseReport Class, Abort, FirstPage, LastPage, Selection

6.118 OutputName

Declaration
property OutputName: string;

Default
'(empty)

Category
Printer

Description
This property defines an alternate output device for the current printer. The output device can be another port, 'LPT3:', or a file on the disk, 'C:\APP\PRINTER.DMP'. The contents of the file that is created will contain actual printer commands and can be copied to a printer at a later time with a DOS command. This can be useful for sending output to printers that are not hooked up to the current computer. To do this create the file, copy it to a computer hooked up to the printer and then use the copy command to send it to the printer port.

See also
TBaseReport Class, Port

Example (Delphi)
// COPY PRINTER.DMP LPT1 /B
RvNDWriter1.OutputName := 'C:\APP\PRINTER.DMP';

Example (C++Builder)
RvNDWriter1->OutputName = "C:\APP\PRINTER.DMP";
6.119 PageHeight

Declaration

property PageHeight: double;

Category

Printer

Description

This property returns the height of the currently selected paper size.

See also

TBaseReport Class, PageWidth

Example (Delphi)

// Save current page height
CurrPageHeight := RvNDRWriter1.PageHeight;

Example (C++Builder)

CurrPageHeight = RvNDRWriter1->PageHeight;

6.120 PageInc

Declaration

property PageInc: integer;

Default

1

Category

Preview

Description

This property will set or return the number of pages that the preview screen will be incremented or decremented by when NextPage or PrevPage is called.

See also

TRvRenderPreview Class, NextPage, PrevPage

Example (Delphi)

PageInc := 4;

Example (C++Builder)

PageInc = 4;

6.121 PageInvalid

Declaration

property PageInvalid: Boolean;

Category

Control

Description

This property will return whether the current page is valid for printing or not. Typically this property will be true if the current page is outside the range for FirstPage to LastPage.

See also

TBaseReport Class, FirstPage, LastPage
**Example (Delphi)**

if RvNDRWriter1.PageInvalid then begin
  { code to respond to an invalid page }
end; { if }

**Example (C++Builder)**

if (RvNDRWriter1.PageInvalid) {
  // code to respond to an invalid page
} if

### 6.122 Pages

**Declaration**

```delphi
property Pages: integer;
```

**Category**

Preview

**Description**

This property returns the total number of pages that exist inside the report file for a preview screen.

**See also**

TRvRenderPreview Class, Macro

**Example (Delphi)**

```delphi
Edit1.Text := IntToStr(RvRenderPreview1.Pages);
Form1.Invalidate;
```

**Example (C++Builder)**

```delphi
Edit1->Text = IntToStr(RvRenderPreview1->Pages);
Form1->Invalidate();
```

### 6.123 PageWidth

**Declaration**

```delphi
property PageWidth: double;
```

**Category**

Printer

**Description**

This property returns the width of the currently selected paper size.

**See also**

TBaseReport Class, PageHeight

**Example (Delphi)**

```delphi
// Save current page width
CurrPageWidth := RvNDRWriter1.PageWidth;
```

**Example (C++Builder)**

```delphi
CurrPageWidth = RvNDRWriter1->PageWidth;
```

### 6.124 Papers

**Declaration**

```delphi
property Papers: TStrings;
```

**Default**

(list of paper sizes supported by the default printer)
Category

Printer

Description

This property will return a TStringList of paper sizes that are supported by the current printer.

See also

TBaseReport Class, SelectPaper, SupportPaper, TStrings

Example (Delphi)

ListBox2.Items := RvNDRWriter1.Papers;

Example (C++Builder)

ListBox2->Items = RvNDRWriter1->Papers;

6.125 ParaJustify

Declaration

property ParaJustify: TTabJustify;

Default

tjNone

Category

RTF

Description

This property allows you to set the justification used for the current paragraph. Usually the justification is set by the first print command on a new paragraph (i.e. PrintCenter would set the paragraph to be center justified). Setting ParaJustify for other output components such as TRvNDRWriter or TRvRenderPrinter will have no effect.

See also

TBaseReport Class, NewPara

Example (Delphi)

With Sender as TBaseReport do begin
  ParaJustify := tjCenter;
  Print('This text is centered');
end; {with}

Example (C++Builder)

rpi->ParaJustify = tjCenter;
rpi->Print("This text is centered");

6.126 PIVar

Declaration

function PIVar(PIVarName: String): String;

Category

Printing

Description

This method allows you to initialize the value of a PIVar (Post Initialize Variable). Any PIVars of the same name that were previously printed will show this value. PIVars will use the value that is set after it is printed. 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. TRvSystem.SystemOptions.soUseFiler must be true if you are using PIVars in your report.
See also
TBaseReport Class, SetPIVar

**Example (Delphi)**

```delphi
with Sender as TBaseReport do begin
  Print('SubTotal:' + PIVar('SubTotal')); // Other print statements including new pages
  SetPIVar('SubTotal', FormatFloat(SubTotal));
end;
```

**Example (C++Builder)**

```cpp
rp1->Print("SubTotal:" + PIVar("SubTotal");
// Other print statements including new pages
rp1->SetPIVar("SubTotal", FormatFloat(SubTotal));
```

### 6.127 Port

**Declaration**

```delphi
property Port: string;
```

**Category**

Printer

**Description**

This property will return the port name for the currently selected printer.

See also
TBaseReport Class, PrinterIndex, OutputName

**Example (Delphi)**

```delphi
Edit1.Text := RvNDRWriter1.Port;
Form1.Invalidate;
```

**Example (C++Builder)**

```cpp
Edit1->Text = RvNDRWriter1->Port;
Form1->Invalidate();
```

### 6.128 Pos

**Declaration**

```delphi
property Pos: longint;
```

**Default**

0

**Category**

Memo

**Description**

This property will return or set the current position marker for the memo buffer. The first position is at index 0.

See also
TMemoBuf Class, Reset

**Example (Delphi)**

```delphi
// Save current memo buffer position
CurrMemoPos := MemoBuf1.Pos;
```

**Example (C++Builder)**

```cpp
CurrMemoPos = MemoBuf1->Pos;
```
6.129 Position

Declaration
property Position: double;

Category
BarCode

Description
This property sets or returns the positions of the bar code that is used in relation to the state of the BarCodeJustify property. This property along with BarCodeJustify is changed whenever the Left, Right or Center properties are changed.

See also
TRpBarsBase Class, BarCodeJustify, BarTop, Center, Left, Right

Example (Delphi)
// Bar Code will be centered at the SectionLeft + 3.0 point
BarCodeJustify := pjCenter;
Position := SectionLeft + 3.0;

Example (C++Builder)
BarCodeJustify = pjCenter;
Position = SectionLeft + 3.0;

6.130 PrintChecksum

Declaration
property PrintChecksum: Boolean

Default
false

Category
BarCode

Description
This property 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
TRpBarsBase Class, BarTop, UseChecksum

6.131 PrintEnd

Declaration
property PrintEnd: double;

Default
0.0

Category
Memo

Description
This property will return or set the rightmost position that the memo field will print in.
See also
TMemoBuf Class, PrintStart

Example (Delphi)
// Leave 1.5 inches for left margin
MemoBuf1.PrintEnd := 6.5;

Example (C++Builder)
MemoBuf1->PrintEnd = 6.5;

6.132 PrinterIndex

Declaration
property PrinterIndex: integer;

Default
-1

Category
Printer

Description
This property will return or set the currently selected printer as defined in the Printer.Printers string list. Set PrinterIndex to -1 to use the default printer.

See also
TBaseReport Class, SelectPrinter

Example (Delphi)
// Save current printer index
CurrIndex := RvNDRWriter1.PrinterIndex;

Example (C++Builder)
CurrIndex = RvNDRWriter1->PrinterIndex;

6.133 Printers

Declaration
property Printers: TStrings;

Default
(list of printers currently installed on the system)

Category
Printer

Description
This property will return a TStringList of printers that are currently installed on the user's computer.

See also
TBaseReport Class, SelectPrinter, TStrings

Example (Delphi)
ComboBox2.Items := Printers;

Example (C++Builder)
ComboBox2->Items := rp1->Printers;
6.134 Printing

Declaration

property Printing: Boolean;

Category

Control

Description

This property will be set to true after a call to Execute has been made and will remain true until the report has finished.

See also

TBaseReport Class, Execute

Example (Delphi)

if RvNDRWriter1.Printing then RvNDRWriter1.Abort;

Example (C++Builder)

if (RvNDRWriter1->Printing) RvNDRWriter1->Abort();

6.135 PrintReadable

Declaration

property PrintReadable: Boolean;

Default

true

Category

BarCode

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

TRpBarsBase Class, PrintTop, TextJustify

6.136 PrintStart

Declaration

property PrintStart: double;

Default

0.0

Category

Memo

Description

This property will return or set the leftmost position that the memo buffer will print in.

See also

TMemoBuf Class, PrintEnd
6.137 PrintTop

Declaration

property PrintTop: Boolean;

Default
false

Category
BarCode

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
TRpBarsBase Class, PrintReadable, TextJustify

Example (Delphi)

Code39.PrintTop := True;
Code39.Print;

Example (C++Builder)

Code39->PrintTop = true;
Code39->Print();

6.138 ProjectFile

Declaration

property ProjectFile: string;

Default
'' (empty)

Category
Rave

Description
This property defines the filename of the report project that will be loaded when the TRvProject component is opened. This parameter should point to a valid .RAV file.

See also
TRvProject Class, Active, Close, Open

6.139 Query

Declaration

property Query: TQuery;

Category
Rave
Description
Specifies the TQuery component that is connected to the TRvQueryConnection component.

See also
TRvQueryConnection Class

Example (Delphi)
CustOrdCXN.Query := CustOrdQuery;

Example (C++Builder)
CustOrdCXN->Query = CustOrdQuery;

6.140 RaveBlobDateTime

Declaration
property RaveBlobDateTime: TDateTime;

Category
Rave

Description
Returns the date and time that a report project was last loaded into the application form. This is not the date and time of the file that was loaded, but rather the date and time that the loading action was performed. If no report project is loaded, the value will be equal to 0.0.

See also
TRvProject Class, ClearRaveBlob, LoadRaveBlob, SaveRaveBlob

Example (Delphi)
Label1.Caption := DateTimeToStr(RvProject1.RaveBlobDateTime);

Example (C++Builder)
Label1->Caption = DateTimeToStr(RvProject1->RaveBlobDateTime);

6.141 ReadableHeight

Declaration
property ReadableHeight: double;

Category
BarCode

Description
Returns the height that the readable text adds to the bar code.

See also
TRpBarsBase Class, BarHeight, Height

6.142 ReportDateTime

Declaration
property ReportDateTime: TDateTime;

Default
(Date and time Execute or Start was called)

Category
Printing

Description
This property will set or return the date and time the report was started.
See also
   TBaseReport Class, Macro

**Example (Delphi)**
Edit1.Text := DateTimeToStr(ReportDateTime);

**Example (C++Builder)**
Edit1->Text = DateTimeToStr(rp1->ReportDateTime);

### 6.143 ReportDesc

**Declaration**
   property ReportDesc: string;

**Category**
   Rave

**Description**
A Rave report is defined by 3 items. The name property is the standard type name with no spaces or special characters. The full name is like a short title that can be more descriptive of the reports purpose. The description is more like a memo that would be the complete description about a report that could be displayed in a memobuf area for the user to select. This property will return the description of the currently selected report.

See also
   TRvProject Class, ReportFullName, ReportDescToMemo, ReportName, SelectReport

### 6.144 ReportDest

**Declaration**
   property ReportDest: TReportDest;

**Category**
   ReportSystem

**Description**
This property will be set to the actual destination of the report after the setup form has been exited. This can be useful for determining which selection the user has chosen (printer/preview/file) and assign that to other RvSystem components (in the DefaultDest property).

See also
   TRvSystem Class, DefaultDest

### 6.145 ReportFullName

**Declaration**
   property ReportFullName: string;

**Category**
   Rave

**Description**
A Rave report is defined by 3 items. The name property is the standard type name with no spaces or special characters. The full name is like a short title that can be more descriptive of the reports purpose. The description is more like a memo that would be the complete description about a report that could be displayed in a memobuf area for the user to select. This property will return the full name of the currently selected report.

See also
   TRvProject Class, ReportDesc, ReportName, SelectReport
6.146 ReportName

Declaration

```delphi
property ReportName: string;
```

Category
Rave

Description
A Rave report is defined by 3 items. The name property is the standard type name with no spaces or special characters. The full name is like a short title that can be more descriptive of the reports purpose. The description is more like a memo that would be the complete description about a report that could be displayed in a memobuf area for the user to select. This property will return the name of the currently selected report.

See also
TRvProject Class, ReportDesc, ReportFullName, SelectReport

6.147 RichEdit

Declaration

```delphi
property RichEdit: string
```

Category
Memo

Description
Imports the RTF contents stored in a TRichEdit component into a memo buffer.

NOTE:
This property does not exist in Delphi 1.0.

See also
TMemoBuf Class, RTFLoadFromStream, RTFText

Example (Delphi)

```delphi
MemoBuf1.RichEdit := RichEdit1;
```

Example (C++Builder)

```delphi
MemoBuf1->RichEdit = RichEdit1;
```

6.148 Right

Declaration

```delphi
property Right: double;
```

Category
BarCode

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
TRpBarsBase Class, BarCodeJustify, Center, Left, Position

Example (Delphi)

```delphi
BarCode1.Right := SectionRight;
```

Example (C++Builder)

```delphi
BarCode1->Right = rp1->SectionRight;
```
6.149 **RightWaste**

**Declaration**

```plaintext
property RightWaste: double;
```

**Category**

**Printer**

**Description**

This property returns the waste area on the right side of the page that the printer cannot print into. It is a good idea to make sure that the report's margins are greater than or equal to its waste areas.

**See also**

[TBaseReport Class], [BottomWaste], [LeftWaste], [MarginRight], [TopWaste]

**Example**

See [LeftWaste]

6.150 **RTFField**

**Declaration**

```plaintext
property RTFField: TMemoField
```

**Category**

**Memo**

**Description**

Imports a RTF string stored in a TMemoField component into a memo buffer.

**See also**

[TMemoBuf Class], [Field], [RTFText]

6.151 **RTFText**

**Declaration**

```plaintext
property RTFText: string
```

**Category**

**Memo**

**Description**

Imports an RTF string stored in a text variable into the memo buffer.

**See also**

[TMemoBuf Class], [RTFField]

6.152 **RulerType**

**Declaration**

```plaintext
property RulerType: TRulerType;
```

**Default**

```
rtNone
```

**Category**

**Preview**
Description
This will create a ruler around the preview screen that can be used to measure items during report development.

rtNone NO rulers will be visible
rtHorizCm A ruler in centimeters will be on the top of the page
rtVertCm A ruler in centimeters will be on the left side of the page
rtBothCm Rulers in centimeters will be on the top and left side of the page
rtHorizIn A ruler in inches will be on the top of the page
rtVertIn A ruler in inches will be on the left side of the page
rtBothIn Rulers in inches will be on the top and left side of the page

See also
TRvSystem Class, GridHoriz, GridPen, GridVert

6.153 RuntimeVisibility

Declaration
property RuntimeVisibility: Boolean;

Category
Rave

Description
This property determines the visibility of the data connection to an End User designer.

rtNone invisible to external programs at runtime
rtDeveloper visible only to developer version of Rave at runtime
rtEndUser visible to any version of Rave

NOTE:
If you are NOT distributing the end user report designer and are concerned about the visibility of your data to external application, you should set the RuntimeVisibility to rtNone before distributing your application.

See also
TRvCustomConnection Class, DevLock property on Rave Components

Example (Delphi)
RvCustomConnection1.RuntimeVisibility := rtNone;

Example (C++Builder)
RvCustomConnection1->RuntimeVisibility = rtNone;

6.154 ScaleX

Declaration
property ScaleX: double;

Default
100

Category
Control

Description
These properties return or set the current scaling percent to apply. A value of 100.0 results in normal size, while 200.0 will double the print size and 50.0 will half the print size. This can be used with OriginX and OriginY to print multiple pages per piece of paper.

See also
TBaseReport Class, OriginX, OriginY, ScaleY
Example (Delphi)
// Scale to fit 4 pages on one sheet of paper
RvNDRWriter1.ScaleX := 50.0;
RvNDRWriter1.ScaleY := 50.0;

Example (C++Builder)
RvNDRWriter1->ScaleX = 50.0;
RvNDRWriter1->ScaleY = 50.0;

6.155 ScaleY

Declaration
property ScaleY: double;

Default
100

Category
Control

Description
These properties return or set the current scaling percent to apply. A value of 100.0 results in normal size, while 200.0 will double the print size and 50.0 will half the print size. This can be used with OriginX and OriginY to print multiple pages per piece of paper.

See also
TBaseReport Class, OriginX, OriginY, ScaleX

Example (Delphi)
// Scale to fit 4 pages on one sheet of paper
RvNDRWriter1.ScaleX := 50.0;
RvNDRWriter1.ScaleY := 50.0;

Example (C++Builder)
RvNDRWriter1->ScaleX = 50.0;
RvNDRWriter1->ScaleY = 50.0;

6.156 ScrollBox

Declaration
property ScrollBox: TScrollBox;

Default
nil

Category
Preview

Description
This property defines the scroll box on the preview form that the report will be drawn in.

See also
TRvRenderPreview Class

Example (Delphi)
RvRenderPreview1.ScrollBox := Form1.ScrollBox1;

Example (C++Builder)
RvRenderPreview1->ScrollBox = Form1->ScrollBox1;
6.157 SectionBottom

Declaration

property SectionBottom: double;

Default
MarginBottom

Category
Position

Description
These properties return or set the current section of the paper to be printed on. Items that rely upon the current section settings are line starting points (Example (Delphi) after a CR call), setting columns, LinesLeft and ColumnLinesLeft. The section settings are reset to the margin values after each new page is generated. Changing a margin setting will change its corresponding section setting to the same measurement.

NOTE:
Section settings are different from margin setting in that the section values are always measurements from the upper or left side of the page while margins are measurements from the closest side of the page. (Example (Delphi) SectionRight := 8.0 would be the same as MarginRight := 0.5 for 8.5 inch wide paper.)

See also
TBaseReport Class, Margin properties, ResetSection, SectionLeft, SectionRight, SectionTop

Example (Delphi)
with RVNDWriter1 do begin
  SectionLeft := 1.0;
  SectionRight := 7.5;
  SectionTop := 1.5;
  SectionBottom := 1.0;
end; { with }

Example (C++Builder)
rpl->SectionLeft = 1.0;
rpl->SectionRight = 7.5;
rpl->SectionTop = 1.5;
rpl->SectionBottom = 1.0;

6.158 SectionLeft

Declaration

property SectionLeft: double;

Default
MarginLeft

Category
Position

Description
These properties return or set the current section of the paper to be printed on. Items that rely upon the current section settings are line starting points (Example (Delphi) after a CR call), setting columns, LinesLeft and ColumnLinesLeft. The section settings are reset to the margin values after each new page is generated. Changing a margin setting will change its corresponding section setting to the same measurement.
NOTE:
Section settings are different from margin setting in that the section values are always measurements
from the upper or left side of the page while margins are measurements from the closest side of the page.
(Example (Delphi) SectionRight := 8.0 would be the same as MarginRight := 0.5 for 8.5 inch wide paper.)

See also
TBaseReport Class, Margin properties, ResetSection, SectionBottom, SectionRight, SectionTop

Example
see SectionBottom

6.159 SectionRight

Declaration
property SectionRight: double;

Default
MarginRight

Category
Position

Description
These properties return or set the current section of the paper to be printed on. Items that rely upon the
current section settings are line starting points (Example (Delphi) after a CR call), setting columns,
LinesLeft and ColumnLinesLeft. The section settings are reset to the margin values after each new page
is generated. Changing a margin setting will change its corresponding section setting to the same
measurement.

NOTE:
Section settings are different from margin setting in that the section values are always measurements
from the upper or left side of the page while margins are measurements from the closest side of the page.
(Example (Delphi) SectionRight := 8.0 would be the same as MarginRight := 0.5 for 8.5 inch wide paper.)

See also
TBaseReport Class, Margin properties, ResetSection, SectionBottom, SectionLeft, SectionTop

Example
see SectionBottom

6.160 SectionTop

Declaration
property SectionTop: double;

Default
MarginTop

Category
Position

Description
These properties return or set the current section of the paper to be printed on. Items that rely upon the
current section settings are line starting points (Example (Delphi) after a CR call), setting columns,
LinesLeft and ColumnLinesLeft. The section settings are reset to the margin values after each new page
is generated. Changing a margin setting will change its corresponding section setting to the same
measurement.
NOTE:
Section settings are different from margin setting in that the section values are always measurements from the upper or left side of the page while margins are measurements from the closest side of the page. (Example (Delphi) SectionRight := 8.0 would be the same as MarginRight := 0.5 for 8.5 inch wide paper.)

See also
TBaseReport Class, Margin properties, ResetSection, SectionBottom, SectionLeft, SectionRight

Example
see SectionBottom

6.161 Selection

Declaration

property Selection: string;

Default
'' (empty)

Category
Control

Description
This property will override FirstPage and LastPage if not blank. Selection defines the valid pages in a print job and can contain separate page ranges, separated by commas or 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.
"e" or "even" pages
"o" or "odd" pages
"r" "reverse order" pages
"a" or "all"

See also
TBaseReport Class, FirstPage, LastPage, SystemOptions

Example (Delphi)

Selection := '1-11';  {Print pages 1 through 11}
Selection := '5-8,25';  {Print pages 5 through 8 and page 25}
Selection := '1,3,6-';  {Print pages 1,3,6 to end of job}
Selection := '1,e,9-11';  {Print all even pages and page 1, 9 through 11}
Selection := 'o';  {Print all odd pages}

Example (C++Builder)

Selection = "1-11";  / Print pages 1 through 11
Selection = "5-8,25";  / Print pages 5 through 8 and page 25
Selection = "1,3,6-";  / Print pages 1,3,6 to end of job
Selection = "1,e,9-11";  / Print all even pages and page 1, 9 through 11
Selection := "o";  / Print all odd pages

6.162 ServerMode

Declaration

property ServerMode: Boolean read FServerMode write FServerMode

Default
false

Category
Render
Description
This property specifies whether the HTML is being generated dynamically from the report server or is
being run locally. This affects things like whether the image files will be given a .tmp file type, which is the
case for servermode, or whether they are given the .jpg file type needed when running locally, which
enables the browser to deter the file type and display the image correctly.

See also
TRpRender Class, CacheDir

6.163 ShadowDepth

Declaration
property ShadowDepth: integer;

Default
0

Category
Preview

Description
This property will define the shadow depth of the preview page in pixels.

NOTE:
Shadows will not be drawn while the Monochrome property is true.

See also
TBaseReport Class, Monochrome

Example (Delphi)
ShadowDepth := 5;

Example (C++Builder)
ShadowDepth = 5;

6.164 Size

Declaration
property Size: longint;

Category
Memo

Description
This property will return the current size of the text in the memo buffer in bytes.

See also
TMemoBuf Class, MaxSize, Pos

Example (Delphi)
MemoBytes := MemoBuf1.Size;

Example (C++Builder)
MemoBytes = MemoBuf1->Size;

6.165 StatusFormat

Declaration
property StatusFormat: string;
Default
'Printing page'

Category
Misc

Description
This property defines the format for the text printed to StatusLabel during an UpdateStatus call. There are several special formatting character pairs that can be used within the string:

%c    current printing pass
%p    Current Page
%f    First Page
%l    Last Page
%d    Printer Device Name
%n    force a carriage return
%r    Printer Driver Name
%s    Total number of passes
%t    Printer Port
%0 through %9 Status Text Line (see StatusText)
%%%   % character

See also
TBaseReport Class, CurrentPass, StatusLabel, StatusText, TotalPasses, UpdateStatus

Example (Delphi)
RvNDRWriter1.StatusFormat := 'Generating page,'
RvNDRWriter1.StatusFormat := 'Printing page (Pass of )';

Example (C++Builder)
RvNDRWriter1->StatusFormat = "Generating page ";
RvNDRWriter1->StatusFormat = "Printing page (Pass of )";

6.166 StatusLabel

Declaration
property StatusLabel: TLabel;

Default
nil

Category
Misc

Description
This property defines the TLabel component that UpdateStatus will put the status text, StatusFormat, into.

See also
TBaseReport Class, StatusFormat, StatusText, UpdateStatus

Example (Delphi)
RvNDRWriter1.StatusLabel := StatusForm.Label1;

Example (C++Builder)
RvNDRWriter1->StatusLabel = StatusForm->Label1;

6.167 StatusText

Declaration
property StatusText: TStrings;
This property defines a string list of at most 10 strings that can replace the special formatting characters (%0 to %9) in StatusFormat.

See also
TBaseReport Class, StatusFormat, TStrings

Example (Delphi)
```
StatusText[1] := 'Inform user of report status';
UpdateStatus;
```

Example (C++Builder)
```
rp1->StatusText->Strings[1] = "Inform user of report status";
rp1->UpdateStatus();
```

### 6.168 StoreRAV

**Declaration**
```delphi
property StoreRAV: Boolean;
```

**Default**
false

**Category**
Rave

**Description**
This property will return whether a report project (RAV file) is stored in the executable or not. At
design-time, editing this property will bring up a dialog allowing you to load, save or remove a report
project from your application. The date and time that a report project was last loaded into is displayed in
the Object Inspector.

**NOTE:**
This is not the date and time of the file on disk, but the date and time that the load action was performed.
A warning will be displayed if a file, defined by ProjectFile, exists that is of a later date and time and you
will be prompted to use the version on the disk instead.

See also
TRvProject Class, ClearRaveBlob, LoadRaveBlob, ProjectFile, RaveBlobDateTime, SaveRaveBlob

### 6.169 Stream

**Declaration**
```delphi
property Stream: TStream;
```

**Default**
il

**Category**
Control

**Description**
This property returns or sets the stream used to either write to or read from the report file. A user created
stream can be assigned when StreamMode is equal to smUser but otherwise this property should not be
modified.
See also

TBaseReport Class, FileName, StreamMode

Example (Delphi)

```delphi
var   ReportStream: TMemoryStream;
begin
  ReportStream := TMemoryStream.Create;
  try
    with RvNDRWriter1 do begin
      ... smUser;
      Stream     := ReportStream;
      Execute;
    end;  { with }
  finally
    ReportStream.Free;
  end;  { tryf }
end;
```

Example (C++Builder)

```cpp
TMemoryStream* ReportStream = new TMemoryStream();
try {
  rp1->StreamMode = smUser;
  rp1->Stream = ReportStream;
  rp1->Execute();
} __finally {
  delete ReportStream;
} // tryf
```

6.170 StreamMode

Declaration

```delphi
property StreamMode: TStreamMode;
```

Default

smMemory

Category

Control

Description

This property defines how the stream for the report file is maintained.

- **smFile**
  This setting uses a TFileStream to store the report file and is very good for large reports, but may run a little slower.

- **smTemp**
  This will send the output to a temporary file in the \Windows\Temp directory. This filename used by smTempFile is created by the system and will be deleted when you exit the reporting system.

- **smMem**
  This setting uses a TMemoryStream and is good for small reports to run faster, but do not use this option for reports that may be large.

- **smUser**
  This does not create a stream, but uses the stream that has been assigned to the Stream property before the report was started. The programmer is responsible for creating and freeing the stream if smUser is used.

See also

TBaseReport Class, FileName, Stream

Example (Delphi)

```delphi
RvNDRWriter1.StreamMode := smMemory;
RvNDRWriter2.FileName   := 'TEMP.RpT';
RvNDRWriter2.StreamMode := smFile;
```
Example (C++Builder)
RvNDRWriter1->StreamMode = smMemory;
RvNDRWriter2->FileName = "TEMP.RPT";
RvNDRWriter2->StreamMode = smFile;

6.171 Strikeout

Declaration
property Strikeout: Boolean;

Default
false

Category
Font

Description
This property returns or sets the strikeout attribute for the current font.

See also
TBaseReport Class, Bold, Italic, Underline

Example (Delphi)
with RvNDRWriter1 do begin
  Strikeout := true;
  Print( 'Deleted Text' );
  Strikeout := false;
end; { with }

Example (C++Builder)
rp1->Strikeout = true;
rp1->Print( "Deleted Text" );
rp1->Strikeout = false;

6.172 Subscript

Declaration
property Subscript: Boolean;

Default
false

Category
Font

Description
Returns or sets the subscript setting for the current text font.

See also
TBaseReport Class, Superscript

Example (Delphi)
// Print a formula
Print('Y = Pi * X');
Subscript := true;
Print('a');
Subscript := false;
Example (C++Builder)
rp1->Print("Y = Pi * X");
rp1->Subscript = true;
rp1->Print("a");
rp1->Subscript = false;

6.173 Superscript

Declaration

property Superscript: Boolean;

Default
false

Category
Font

Description
Returns or sets the superscript setting for the current text font.

See also
TBaseReport Class, Subscript

Example (Delphi)
// Print a formula
Print('E = MC');
Superscript := true;
Print('2');
Superscript := false;

Example (C++Builder)
rp1->Print("E = MC");
rp1->Superscript = true;
rp1->Print("2");
rp1->Superscript = false;

6.174 SystemFiler

Declaration

property SystemFiler: TSystemFiler;

Category
ReportSystem

Description
All SystemFiler options operate in the same manner as the other components except for the stream mode of smMemory which does not require a filename and will use a TMemoryStream to contain a report.

See also
TRvSystem Class, Other System options

Example (Delphi)

Example (C++Builder)
RvSystem1->SystemFiler->AccuracyMethod = amAppearance;

6.175 SystemOptions

Declaration

property SystemOptions: TSystemOptions;
Category
ReportSystem

Description
The SystemOptions properties control the configuration of the TRvSystem component:
- `soUseFiler` will always send the report to a report file. This can be very useful if the Macro method has been used in the report.
- `soWaitForOK` determines whether the user has to press the OK button once the report has been generated for output.
- `soShowStatus` determines whether or not the status screen is displayed when the report is being generated.
- `soAllowPrintFromPreview` determines whether the user can print from the preview screen.
- `soAllowSaveFromPreview` determines whether the user can save from the preview screen.
- `soPreviewModal` determines if the preview screen will be modal.
- `soNoGenerate` will cause the RvSystem component to skip over the generation phase of the report and proceed straight to screen or the printer. This option should only be used with a StreamMode of smFile where the report file has been previously generated and needs only to be viewed or printed.

See also
TRvSystem Class, Other SystemXxxx options

Example (Delphi)
```delphi
// Disable the status screen
RvSystem1.SystemOptions := RvSystem1.SystemOptions - (soShowStatus);
```

Example (C++Builder)
```cpp
RvSystem1->SystemOptions = RvSystem1->SystemOptions >> soShowStatus;
```

6.176 SystemPreview

Declaration
```delphi
property SystemPreview: TSystemPreview;
```

Category
ReportSystem

Description
SystemPreview displays all the preview type options displayed in TRvRenderPreview. Following are the additional properties:
- `FormHeight` defines the height of the RvSystem report preview form.
- `FormState` defines the initial window status (normal, minimized or maximized) of the RvSystem report preview form.
- `FormWidth` defines the width of the RvSystem report preview form.

See also
TRvSystem Class, Other SystemXxxx options

Example (Delphi)
```delphi
RvSystem1.SystemPreview.FormState := wsMaximized;
```

Example (C++Builder)
```cpp
RvSystem1->SystemPreview->FormState = wsMaximized;
```
6.177 SystemPrinter

Declaration

property SystemPrinter: TSystemPrinter;

Category

ReportSystem

Description

SystemPrinter displays all the printer type options displayed in TRvRenderPrinter.

See also

TRvSystem Class, Other SystemXxxx options

Example (Delphi)

RvSystem1.SystemPrinter.MarginLeft := 0.5;

Example (C++Builder)

RvSystem1->SystemPrinter->MarginLeft = 0.5;

6.178 SystemSetups

Declaration

property SystemSetups: TSystemSetups;

Default

[ssAllowSetup, ssAllowCopies, ssAllowCollate, ssAllowDuplex, ssAllowDestPreview, ssAllowDestPrinter, ssAllowDestFile, ssAllowPrinterSetup]

Category

ReportSystem

Description

This property contains settings that define the behavior of the Printer Setup Dialog that TRvSystem uses. To see a description of each option see TSystemSetup.

See also

TRvSystem Class, TSystemSetup

Example (Delphi)

// Disable the setup screen

Example (C++Builder)

RvSystem1->SystemSetups = RvSystem1->SystemSetups >> ssAllowSetup;

6.179 TabColor

Declaration

property TabColor: TColor;

Default

clBlack

Category

Tabs

Description

This property defines the color that will be used to shade tab boxes created with SetTab. TabShade will define what percentage of TabColor is used.
See also
TBaseReport Class, SetTab, TabShade, TColor

6.180 TabJustify

Declaration
property TabJustify: TTabJustify;

Default
	jNone

Category
Tabs

Description
This property will override any tab justification that was defined with SetTab(). This can be useful for
column headings that are normally centered while the remaining data is justified according to the type of
data. tjNone will disable this feature while tjLeft, tjCenter, tjRight and tjBlock will set the justification
respectively.

See also
TBaseReport Class

Example (Delphi)
TabJustify := tjCenter;
PrintLn(#9'Name'#9'Number');
TabJustify := tjNone;

Example (C++Builder)
rp1->TabJustify = tjCenter;
rp1->PrintLn("\tName\tNumber");
rp1->TabJustify = tjNone;

6.181 Table

Declaration
property Table(MyPrinter: Trave);

Default
nil

Category
Rave

Description
Specifies the TTable component that is connected to the TRvTableConnection component.

See also
TRvTableConnection Class

Example (Delphi)
CustomerCXN.Table := CustomerQuery;

Example (C++Builder)
CustomerCXN->Table = CustomerQuery;

6.182 TabShade

Declaration
property TabShade: integer;
Default
0

Category
Tabs

Description
This property defines a default tab shading that will override the tab shading defined with SetTab but not override the setting of the ShadeOverride parameter of the Tab method. TabShade can be useful for printing barred rows of alternating shades by setting TabShade before each line is printed.

See also
TBaseReport Class, SetTab, Tab

Example (Delphi)
// alternate tab shading by even / odd line status
if Odd(LineNum) then begin
  TabShade := 0;
end else begin
  TabShade := 15;
end;

Example (C++Builder)
if ((rp1->LineNum 2) == 1) {
  TabShade = 0;
} else {
  TabShade = 15;
}

6.183 Text (TMemoBuf)

Declaration
property Text: string;

Default
" (empty)

Category
Memo

Description
This property will set the memo buffer to a string assigned to it. If this property is referenced, the first 255 characters (unless Delphi 2.0 is being used) of the memo buffer (or the size of the memo buffer, whichever is less) will be returned.

See also
TMemoBuf Class, SetData

Example (Delphi)
MemoBuf1.Text := 'New text assigned into MemoBuf1';

Example (C++Builder)
MemoBuf1->Text = "New text assigned into MemoBuf1";

6.184 Text (TRpBarsBase)

Declaration
property Text: string;
Category
   BarCode

Description
The text to be printed as a bar code.

NOTE:
Do not include the check character. The check character will be automatically calculated and printed according to the state of the UseChecksum property.

NOTE:
Any characters that are invalid for the bar code type will be deleted from the text property upon assignment.

See also
   TRpBarsBase Class, Print, PrintXY, TextJustify, UseChecksum

Example (Delphi)
// example of -- since "-" is not valid it will be stripped out
PostNetBC1.Text := '85283-3558';

Example (C++Builder)
PostNetBC1->Text = "85283-3558";

6.185 TextBKMode

Declaration
   property TextBKMode: TBKMode;

Default
   bkTransparent

Category
   Graphics

Description
This property will define the current text background mode as either bkTransparent, where text will print on top of graphics without erasing the background, or as bkOpaque, where text will print on top of graphics after the background is cleared.

NOTE:
Not all printer drivers support opaque text, especially PCL5 laserjet drivers. For these printers try setting graphics mode to Raster instead of HP/GL2 inside the printer setup window and opaque text printing may work.

See also
   TBaseReport Class, BKColor

Example (Delphi)
RvNDRWriter1.TextBKMode := bkOpaque;

Example (C++Builder)
RvNDRWriter1->TextBKMode = bkOpaque;

6.186 TextJustify

Declaration
   property TextJustify: TPrintJustify
Default
pjCenter

Category
BarCode

Description
Determines how the readable text is justified in relation to the bar code.
pjLeft Left justify the text portion
pjCenter Center justify the text portion
pjRight Right justify the text portion

See also
TRpBarsBase Class, PrintReadable, PrintTop, Text

6.187 Title

Declaration
property Title: string;

Default
'Rave Report'

Category
Misc

Description
This property defines the title for the current print job that will be displayed in the Windows Print Manager. (16 bit is limited to 31 characters).

See also
TBaseReport Class

Example (Delphi)
// This code causes the text "Sales Report" to show as the print job name in the print manager.
RvNDriver1.Title := 'Sales Report';

Example (C++Builder)
RvNDriver1->Title = "Sales Report";

6.188 TitlePreview

Declaration
property TitlePreview: TFormatString;

Default
'Report Preview'

Category
ReportSystem

Description
This property defines the caption that will be used for the RvSystem report preview form.

See also
TRvSystem Class, TitleSetup, TitleStatus

6.189 TitleSetup

Declaration
property TitleSetup: TFormatString;
Default
'Report Setup'

Category
ReportSystem

Description
This property defines the caption that will be used for the RvSystem report setup form.

See also
TRvSystem Class, TitlePreview, TitleStatus

6.190 TitleStatus

Declaration
property TitleStatus: TFormatString;

Default
'Report Status'

Category
ReportSystem

Description
This property defines the caption that will be used for the RvSystem report status form.

See also
TRvSystem Class, TitlePreview, TitleSetup

6.191 Top

Declaration
property Top: double;

Category
BarCode

Description
Sets or returns the position for the top edge of the bar code. The value for this property includes the readable text, if it is printed.

See also
TRpBarsBase Class, BarTop, PrintReadable, PrintTop

Example (Delphi)
// Print the bar code so the top is 3.5 inches down
BarCode1.Top := 3.5;

Example (C++Builder)
BarCode1->Top = 3.5;

6.192 TopWaste

Declaration
property TopWaste: double;

Category
Printer
Description
This property returns the waste area on the top side of the page that the printer cannot print into. It is a good idea to make sure that the report's margins are greater than or equal to its waste areas.

See also
TBaseReport Class, BottomWaste, LeftWaste, MarginTop, RightWaste

Example
See LeftWaste

6.193 TotalPasses

Declaration
property TotalPasses: Integer;

Category
Misc

Description
This is the value that will be returned when a %s is encountered in a StatusFormat string.

See also
TBaseReport Class, CurrentPass, StatusFormat, StatusLabel, StatusText, UpdateStatus

Example (Delphi)
RvNDRWriter1.StatusFormat := 'Printing page (Pass of )';

Example (C++Builder)
RvNDRWriter1->StatusFormat = "Printing page (Pass of )";

6.194 TransparentBitmaps

Declaration
property TransparentBitmaps: Boolean;

Default
false

Category
Graphics

Description
This property will control the mode that PrintBitmap and PrintBitmapRect use to draw bitmaps. A value of true will cause bitmaps to be combined (using the AND operator) with the current page contents while a value of false will replace the page contents with the bitmap.

See also
TBaseReport Class, PrintBitmap, PrintBitmapRect

Example (Delphi)
TransparentBitmaps := true;

Example (C++Builder)
TransparentBitmaps = true;

6.195 TruncateText

Declaration
function TruncateText(Value: String; Width: Double): String;
Category

Printing

Description

This property calculates the width of the string "Value" using the current font. If the text is wider than the Width parameter then it will be truncated by characters to fit.

See also

TBaseReport Class, PrintTab, SetFont

Example (Delphi)

RvNDRWriterlSetFont('Arial', 14);
TruncateText('This text is too long to fit within 2 inches', 2.0);

Example (C++Builder)

RvNDRWriterl->SetFont( "Arial", 14 );
TruncateText("This text is too long to fit within 2 inches", 2.0);

6.196 Underline

Declaration

property Underline: Boolean;

Default

false

Category

Font

Description

This property returns or sets the underline attribute for the current font.

See also

TBaseReport Class, Bold, Italic, Strikeout

Example (Delphi)

with RvNDRWriterl do begin
  Underline := true;
  Print( 'Underlined text' );
  Underline := false;
end; { with }

Example (C++Builder)

rp1->Underline = true;
rp1->Print( "Underlined text" );
rp1->Underline = false;

6.197 Units

Declaration

property Units: TPrintUnits;

Default

unInCh

Category

Units

Description

This property sets the current units mode to one of the following values: unInCh, unMM, unCM, unPoint and unUser. If the setting is unUser then the units factor is determined by the value in UnitsFactor.
6.198 **UnitsFactor**

**Declaration**

```plaintext
property UnitsFactor: double;
```

**Default**

1.0

**Category**

*Units*

**Description**

This property returns or sets 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*)

**Example** (Delphi)

```plaintext
RvNDRWriter1.Units := unInch;
```

**Example** (C++Builder)

```plaintext
RvNDRWriter1->Units = unInch;
```

6.199 **UseBreakingSpaces**

**Declaration**

```plaintext
property UseBreakingSpaces: Boolean read FUseBreakingSpaces write FUseBreakingSpaces
```

**Default**

false

**Category**

*Render*

**Description**

This property specifies whether the HTML output will allow text spaces to be written out as plain text or with \&nbsp;.

**See also**

*TRpRender Class*

6.200 **UseChecksum**

**Declaration**

```plaintext
property UseChecksum: Boolean
```
Default
false (Code128 := true)

Category
BarCode

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

See also
TRpBarsBase Class, BarHeight, BarWidth, PrintReadable, Text, Width

6.201 UseCompression

Declaration
property UseCompression: Boolean read FCompression write FCompression

Default
false

Category
Render PDF

Description
This property determines whether you want to compress the page stream when sending the report out to PDF. The code that actually provides the compression must be defined in the OnCompress event.

See also
TRpRender Class, OnCompress

6.202 UseSetRange

Declaration
property UseSetRange: Boolean;

Default
false

Category
Rave

Description
This property will determine whether filters are handled by the TTable.Filter property or the TTable.SetRange method.

See also
TRvTableConnection Class

6.203 Version

Declaration
property Version: String;

Category
Misc

Description
Returns the current release version of Rave.

See also
TRpComponent Class
6.204 WideFactor

Declaration
property WideFactor: double

Default
3.0

Category
BarCode

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

See also
TRpBarsBase Class, BarHeight, BarWidth, Width

Example (Delphi)
// set wide to narrow bar ratio to be 2.5
WideFactor := 2.5;

Example (C++Builder)
WideFactor = 2.5;

6.205 Width

Declaration
property Width: double;

Category
BarCode

Description
This property will return the calculated width of the entire bar code for the current value of Text.

See also
TRpBarsBase Class, BarWidth, Text, WideFactor

Example (Delphi)
// get width of bar code for ABC123
var BarCodeWidth: double;
BarCode1.Text := 'ABC123';
BarCodeWidth := BarCode1.Width;

Example (C++Builder)
double BarCodeWidth;
BarCode1->Text = "ABC123";
BarCodeWidth = BarCode1->Width;

6.206 XDPI

Declaration
property XDPI: integer;

Category
Printer

Description
This property returns the horizontal dots per inch for the current printer.
See also

TBaseReport Class

Example (Delphi)

```
CurrXDPI := RvNDRWriter1.XDPI;
```

Example (C++Builder)

```
CurrXDPI = RvNDRWriter1->XDPI;
```

### 6.207 XPos

**Declaration**

```
property XPos: double;
```

**Default**

0.0

**Category**

Position

**Description**

This property sets or returns the horizontal text cursor position.

See also

TBaseReport Class, CursorXPos, CursorYPos, YPos

Example (Delphi)

```
XPos := 0.45;
YPos := 0.95;
Print('Text at ( 0.45, 0.95 )');
```

Example (C++Builder)

```
rp1->XPos = 0.45;
rp1->YPos = 0.95;
rp1->Print("Text at ( 0.45, 0.95 )");
```

### 6.208 YDPI

**Declaration**

```
property YDPI: integer;
```

**Category**

Printer

**Description**

This property returns the vertical dots per inch for the current printer.

See also

TBaseReport Class, All other units conversion functions

Example (Delphi)

```
CurrYDPI := RvNDRWriter1.YDPI;
```

Example (C++Builder)

```
CurrYDPI = RvNDRWriter1->YDPI;
```

### 6.209 YPos

**Declaration**

```
property YPos: double;
```
Default
0.0

Category
Position

Description
This property sets or returns the vertical text cursor position.

See also
TBaseReport Class, CursorXPos, CursorYPos, XPos

Example (Delphi)
XPos := 0.45;
YPos := 0.95;
Print('Text at ( 0.45, 0.95 )');

Example (C++Builder)
rp1->XPos = 0.45;
rp1->YPos = 0.95;
rp1->Print("Text at ( 0.45, 0.95 )");

6.210 ZoomFactor

Declaration
property ZoomFactor: double;

Default
100.0

Category
Preview

Description
This property defines the current zoom percent. A value of 100.0 is normal size, 200.0 is double normal size and 50.0 is half size.

See also
TRvRenderPreview Class, ZoomIn, ZoomOut

Example
This code updates the text in a field where the ZoomFactor can be edited by the user. It would be important to keep these well synchronized if more than one event can change this property.

Example (Delphi)
var   S1: string[10];
begin  Str(RvRenderPreview1.ZoomFactor:1:1,S1);
      ZoomEdit.Text := S1;
      RvRenderPreview1.RedrawPage;
end;

Example (C++Builder)
AnsiString S1;
S1 = FloatToStrF(RvRenderPreview1->ZoomFactor, ffGeneral,1,1);
ZoomEdit->Text = S1;
RvRenderPreview1->RedrawPage();
6.211 ZoomInc

**Declaration**

```delphi
property ZoomInc: integer;
```

**Default**

10

**Category**

Preview

**Description**

This property defines the amount that `ZoomIn` and `ZoomOut` modifies `ZoomFactor`.

**See also**

`TRvRenderPreview Class`, `ZoomFactor`, `ZoomIn`, `ZoomOut`

**Example** (Delphi)

```delphi
// This code causes the `ZoomFactor` property to be incremented by 10 when `ZoomIn` and `ZoomOut` are called.
RvRenderPreview1.ZoomInc := 10;
```

**Example** (C++Builder)

```cpp
RvRenderPreview1->ZoomInc = 10;
```

6.212 ZoomPageFactor

**Declaration**

```delphi
property ZoomPageFactor: double;
```

**Category**

Preview

**Description**

This property will return the zoom factor that will zoom the current page so that the entire page is visible. This value can then be assigned to `ZoomFactor`. You should consider the extra width used by a shadow if you have assigned a value to the `ShadowDepth` preview property.

**See also**

`TRvRenderPreview Class`, `ShadowDepth`, `ZoomFactor`, `ZoomPageWidthFactor`

**Example** (Delphi)

```delphi
// use an `OnPreviewShow` event with the following
with Sender As TRvRenderPreview do begin
    ZoomFactor := ZoomPageFactor - (ShadowDepth + 5) / 10;
end; { with }
```

**Example** (C++Builder)

```cpp
TRvRenderPreview* fp = dynamic_cast<TRvRenderPreview*>(Sender);
fp->ZoomFactor = fp->ZoomPageFactor - (fp->ShadowDepth + 5) / 10;
```

6.213 ZoomPageWidthFactor

**Declaration**

```delphi
property ZoomPageWidthFactor: double;
```

**Category**

Preview
Description
This property will return the zoom factor that will zoom the current page so that the entire page width is visible. This value can then be assigned to `ZoomFactor`. You should consider the extra width used by a shadow if you have assigned a value to the `ShadowDepth` preview property.

See also
- `TRvRenderPreview Class`, `ShadowDepth`, `ZoomFactor`, `ZoomPageFactor`

Example (Delphi)
```
// use an OnPreviewShow event with the following
with Sender As TRvRenderPreview do begin
  ZoomFactor := ZoomPageWidthFactor - (ShadowDepth + 3) / 10;
end; { with }
```

Example (C++Builder)
```
TRvRenderPreview* fp = dynamic_cast<TRvRenderPreview*>(Sender);
fp->ZoomFactor = fp->ZoomPageWidthFactor - (fp->ShadowDepth + 3) / 10;
```
Types

Chapter VII
7 Types

Typed constants, unlike true constants, can hold values of array, record, procedural, and pointer types. Typed constants cannot occur in constant expressions.

7.1 TAccuracyMethod

Declaration

\[ \text{TAccuracyMethod} = (\text{amPositioning}, \text{amAppearance}); \]

Category

Control

Description

\( \text{amPositioning} \) This setting will cause the string to be written one character at a time

\( \text{amAppearance} \) This setting will cause the whole string to be written at one time

See also

TBaseReport Class, AccuracyMethod

Example

See AccuracyMethod

7.2 TBKMode

Declaration

\[ \text{TBKMode} = (\text{bkTransparent}, \text{bkOpaque}); \]

Category

Graphics

Description

\( \text{bkTransparent} \) This setting will write the text without erasing the background

\( \text{bkOpaque} \) This setting will write the text after the background has been cleared

See also

TBaseReport Class, TextBKMode

Example

See TextBKMode

7.3 TFontAlign

Declaration

\[ \text{TFontAlign} = (\text{faBaseline}, \text{faTop}, \text{faBottom}); \]

Category

Font

Description

\( \text{faBaseline} \) This setting will align the font at the baseline of the font

\( \text{faTop} \) This setting will align the font at the top of the line

\( \text{faBottom} \) This setting will align the font at the bottom of the line

See also

TBaseReport Class, FontAlign

Example

see FontAlign
7.4 TLineHeightMethod

Declaration

TLineHeightMethod = (lhmLinesPerInch, lhmFont);

Category
Position

Description

<table>
<thead>
<tr>
<th></th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>lhmLinesPerInch</td>
<td>This setting will cause the number of lines to be fit per inch</td>
</tr>
<tr>
<td>lhmFont</td>
<td>This setting will cause the line to adjust to the font size</td>
</tr>
<tr>
<td>lhmUser</td>
<td>This setting will allow the user to define LineHeight directly</td>
</tr>
</tbody>
</table>

See also
TBaseReport Class, LineHeightMethod, LineHeight

Example
See LineHeightMethod

7.5 TMacroID

Declaration

TMacroID = (midCurrDateShort, midCurrDateLong, midCurrDateUS, midCurrDateInter, midCurrTimeShort, midCurrTimeLong, midCurrTimeAMPM, midCurrTime24, midFirstPage, midLastPage, midTotalPages, midCurrentPage, midPrinterName, midDriverName, midPortName, midUser01..midUser20);

Category
Printing

Description

<table>
<thead>
<tr>
<th></th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>midCurrDateShort</td>
<td>Returns the short date format</td>
</tr>
<tr>
<td>midCurrDateLong</td>
<td>Returns the long date format</td>
</tr>
<tr>
<td>midCurrDateUS</td>
<td>Returns the date as MM/DD/YY</td>
</tr>
<tr>
<td>midCurrDateInter</td>
<td>Returns the date as DD/MM/YY</td>
</tr>
<tr>
<td>midCurrTimeShort</td>
<td>Returns the short time format</td>
</tr>
<tr>
<td>midCurrTimeLong</td>
<td>Returns the long time format</td>
</tr>
<tr>
<td>midCurrTimeAMPM</td>
<td>Returns the time in am/pm format</td>
</tr>
<tr>
<td>midCurrTime24</td>
<td>Returns the time in 24 hour format</td>
</tr>
<tr>
<td>midFirstPage</td>
<td>Returns the first page number</td>
</tr>
<tr>
<td>midLastPage</td>
<td>Returns the last page number</td>
</tr>
<tr>
<td>midTotalPages</td>
<td>Returns the total number of pages</td>
</tr>
<tr>
<td>midCurrentPage</td>
<td>Returns the current page number</td>
</tr>
<tr>
<td>midPrinterName</td>
<td>Returns the printer name</td>
</tr>
<tr>
<td>midDriverName</td>
<td>Returns the driver name</td>
</tr>
<tr>
<td>midPortName</td>
<td>Returns the port name</td>
</tr>
<tr>
<td>midUser01 through midUser20</td>
<td>Returns the n'th entry from MacroData</td>
</tr>
</tbody>
</table>

See also
TBaseReport Class, Macro, MacroData

Example
See Macro

7.6 TMarginMethod

Declaration

TMarginMethod = (mmScaled, mmFixed);
### 7.7 TOrientation

**Declaration**

```pascal
TOrientation = (poPortrait, poLandscape, poDefault);
```

**Category**
Control

**Description**

- **poPortrait**: Portrait mode
- **poLandscape**: Landscape mode
- **poDefault**: Default mode on the current printer

**See also**

 TBBaseReport Class, Orientation

**Example**

see Orientation example

### 7.8 TPrintJustify

**Declaration**

```pascal
TPrintJustify = (pjCenter, pjLeft, pjRight, pjBlock);
```

**Category**
Printing

**Description**

- **pjCenter**: Center justify
- **pjLeft**: Justify to the left
- **pjRight**: Justify to the right
- **pjBlock**: Block (full) justify

**See also**

 TBBaseReport Class, Justify, PrintFooter, PrintHeader, SetTab

**Example**

See SetTab

### 7.9 TPrintUnits

**Declaration**

```pascal
TPrintUnits = (unInch, unMM, unCM, unPoint, unUser);
```

**Category**
Units
### Types

#### Description

- **unInch**: This setting will set the units to inches
- **unMM**: This setting will set the units to millimeters
- **unCM**: This setting will set the units to centimeters
- **unPoint**: This setting will set the units to pixels
- **unUser**: This setting will set the units to a scale provided by the user

See also

- **TBaseReport Class, Units**

#### Example

see **Units**

7.10 **TReportDest**

Declaration

```pascal
TReportDest = (rdPreview, rdPrinter, rdFile);
```

Category

**ReportSystem**

Description

- **rdPreview**: This setting will send the report to the preview screen
- **rdPrinter**: This setting will send the report to the printer
- **rdFile**: This setting will send the report to a file

See also

- **TRvSystem Class, DefaultDest**

Example

see **DefaultDest**

7.11 **TStreamMode**

Declaration

```pascal
TStreamMode = (smMemory, smTempFile, smFile, smUser);
```

Category

**Control**

Description

- **smMemo**: This setting will use a memory stream for input and output
- **smFile**: This setting will use a file for input and output
- **smTemp**: This setting will send the output to a temporary file in the \Windows\Temp directory. This filename used by smTempFile is created by the TRvSystem component and will be deleted when it is finished. If this stream mode is used with a custom preview system utilizing TRvNDRWriter, TRvRenderPrinter and TRvRenderPreview components, the generated FileName property from the TRvNDRWriter component must be transferred to the TRvRenderPrinter and TRvRenderPreview components output
- **smUser**: This setting will use stream defined by user for input and output

See also

- **TBaseReport Class, Stream, StreamMode**

Example

See **StreamMode**

7.12 **TSystemOption**

Declaration

```pascal
TSystemOption = (soUseFiler, soWaitForOK, soShowStatus,
```
soAllowPrintFromPreview, soPreviewModal);

Category
ReportSystem

Description
see SystemOptions

See also
TRvSystem Class, SystemOptions

Example
see SystemOptions

7.13 TSystemOptions

Declaration
TSystemOptions = Set of TSystemOption;

Category
ReportSystem

Description
see SystemOptions

See also
TRvSystem Class, SystemOptions

Example
see SystemOptions

7.14 TSystemSetup

Declaration
TSystemSetup = (ssAllowSetup, ssAllowCopies, ssAllowCollate, ssAllowDuplex, ssAllowDestPreview, ssAllowDestPrinter, ssAllowDestFile, ssAllowPrinterSetup);

Category
ReportSystem

Description

ssAllowSetup If false, the setup screen will not be displayed
ssAllowCopies If false, the user will not be able to change the copies
ssAllowCollate If false, the user will not be able to change the collation mode
ssAllowDuplex If false, the user will not be able to change the duplex mode
ssAllowDestPreview If false, the user will not be able to select the preview screen as the report destination
ssAllowDestPrinter If false, the user will not be able to select the printer as the report destination
ssAllowDestFile If false, the user will not be able to select a disk file as the report destination
ssAllowPrinterSetup If false, the user will not be able to select the printer setup dialog

See also
TRvSystem Class, SystemSetups

Example
see SystemSetups

7.15 TSystemSetups

Declaration
TSystemSetups = Set of TSystemSetup;
### TSystemSetup

**Category**  
ReportSystem

**Description**  
see *TSystemSetup*

**See also**  
TRvSystem Class, SystemSetups, TSystemSetup

**Example**  
see SystemSetups

### 7.16 TTabJustify

**Declaration**  
```
TTabJustify = (tjCenter, tjLeft, tjRight, tjBlock, tjNone);
```

**Category**  
Tabs

**Description**
- **tjCenter**: This setting will center justify tabs  
- **tjLeft**: This setting will left justify tabs  
- **tjRight**: This setting will right justify tabs  
- **tjBlock**: This setting will block justify tabs  
- **tjNone**: This setting will disable justification override

**See also**  
TBaseReport Class, TabJustify

**Example**  
see TabJustify
8 Archived

The following components have been archived and are no longer being actively developed or supported on new platforms such as CLX, Linux or .NET. They will remain part of the BEX windows version of Rave for backwards compatibility. However, their functionality is better supported by other areas in Rave and are not recommended for use in new reporting projects. These archived components will NOT be recognized by the BE versions of Rave Reports (bundled with a Borland product).

* TFilePreview - Superseded by TRvRenderPreview
* TFilePrinter - Superseded by TRvRenderPrinter
* TReportPrinter - Superseded by TRvRenderPrinter
* TRpHTMLFiler - Superseded by TRvRenderHTML
* TRTFFiler - Superseded by TRvRenderRTF
* TTextFiler - Superseded by TRvRenderText

The following components are also being archived and are superseded by the visual reporting components for table style reports. These archived components should not be used in new reporting projects.

* TDetailShell - Superseded by visual reporting components (Region/Bands)
* TLabelShell - Superseded by visual reporting components (Region/Bands using settings of Region.Columns and Region.ColumnWidth)
* TMasterShell - Superseded by visual reporting components (Region/Bands)
* TReportShell - Superseded by visual reporting components (Region/Bands)
* TDbTablePrinter - Superseded by visual reporting components (Region/Bands)
* TTablePrinter - Superseded by visual reporting components (Region/Bands using custom data connection for non-database data)

The archived events, methods and properties are listed following this section.
8.1 Components

This will be a list of all archived components (those only available in the BEX version).

8.1.1 TDbTablePrinter

Unit

![Diagram]

Hierarchy

TBaseShell

TTablePrinter

TDbTablePrinter

Description

This component has the capability to generate table style listings with little to no code required. You can even link multiple table printers together for master-detail relationships to multiple levels. The table printers get their flexibility from two other components, TDbTableColumn and TTableSection. TDbTableColumns are responsible for all the properties and events for a single column of data while the TTableSections are responsible for all headers and footers.

TDbTablePrinter Events

OnGetNextRow, OnInitMaster, OnInitTable, OnValidateRow

TDbTablePrinter Methods

Create, Default, Execute

TDbTablePrinter Properties

DataSet, DetailKey, DetailTablePrinter, Engine, Font, MasterKey, MasterTablePrinter, Pen, ReprintHeader, StartPos, TableColumn, TableColumns, TableSection, TextBKMode

8.1.2 TDetailShell

Unit

![Diagram]

Hierarchy

TBaseShell

TDetailShell

Description

This component is the simplest of the framework components, so we will start with that. TDetailShell is good for table listing style reports where you still want to code the report, but in a more structured and easier to maintain format. There are 2 main parts to the shell components, the sections and the events.

TDetailShell Events


TDetailShell Methods

Execute, PrintBodyFooter, PrintBodyHeader, PrintRow

TDetailShell Properties

Engine, IsNewPage, IsReprint, SectionBodyFooter, SectionBodyHeader, SectionRow, StartNewPage
8.1.3 TLabelShell

Unit

RLabel

Hierarchy

TBaseShell

TLabelShell

Description
This shell component is a specialized one that will help with label style reports. There are many predefined formats or you can create a completely custom one. TLabelShell gets its power from OriginX and OriginY. Using these properties to change the upper left hand corner of the page to the upper left hand corner of each label, your reporting code will use the exact same positions for each label on the page.

TLabelShell Events


TLabelShell Methods

Execute

TLabelShell Properties

Border, Col, DrawExtents, DrawPen, DrawPreviewOnly, LabelBrand, LabelHeight, LabelShape, LabelWidth, NumAcross, NumDown, PrintByRow, Row, SpacingHeight, SpacingLeft, SpacingTop, SpacingWidth

8.1.4 TMasterShell

Unit

RShell

Hierarchy

TBaseShell

TDetailShell

TMasterShell

Description
This component is the next shell component in Rave we'll discuss and provides a good balance between simplicity and functionality. It adds the concept of groups and detail sections to the structure that was presented for TDetailShell.

TMasterShell Events


TMasterShell Methods

Execute, PrintBodyFooter, PrintBodyHeader, PrintDetail, PrintGroupFooter, PrintGroupHeader, PrintRow

TMasterShell Properties

DetailReport, Engine, IsNewPage, IsReprint, Reprint, SectionBodyFooter, SectionBodyHeader, SectionGroupFooter, SectionGroupHeader, SectionRow, StartNewPage
8.1.5 TReportShell

Unit

RpShell

Hierarchy

TBaseShell
| TDetailShell
| TMasterShell
| TReportShell

Description
This shell component is very similar to a TMasterShell component, but adds report and page headers and footers.

TReportShell Events

TReportShell Methods
Execute, PrintBodyFooter, PrintBodyHeader, PrintDetail, PrintGroupFooter, PrintGroupHeader, PrintPageFooter, PrintPageHeader, PrintReportFooter, PrintReportHeader, PrintRow

TReportShell Properties
DetailReport, Engine, IsNewPage, IsReprint, Reprint, SectionBodyFooter, SectionBodyHeader, SectionGroupFooter, SectionGroupHeader, SectionPageFooter, SectionPageHeader, SectionReportFooter, SectionReportHeader, SectionRow, StartNewPage

8.1.6 TTablePrinter

Unit

RpTable

Hierarchy

TBaseShell
| TTablePrinter
Description
TTTablePrinter and TTableColumn are the non-database counterparts to TDbTablePrinter and TDbTableColumn. One difference between the two sets of components is the obvious lack of database type properties in TTablePrinter and TTableColumn. The other difference is that the event handlers will be used quite a bit more extensively to provide the data to the table printer component from whatever data source it is coming from. The most common event handlers that will be overridden for TTablePrinter are OnInitTable and GetNextRow, and for TTableColumn they are OnAddTotal, OnRowSetup and OnRowHeight. For more information, look at the demo projects included with Rave for a more detailed example of how to use TTablePrinter.

TTablePrinter Events
OnInitTable, GetNextRow, OnInitMaster, OnInitTable, OnInitValidateRow

TTablePrinter Methods
Default, Execute

TTablePrinter Properties
DetailTablePrinter, Engine, Font, MasterTablePrinter, Pen, ReprintHeader, StartPos, TableColumn, TableColumns, TableSection, TextBKMode
8.2 Events

This will be a list of all archived events (those only available in the BEX version).

8.2.1 OnAddTotal event

Declaration

```
procedure OnAddTotal(TTableColumn: TTableColumn);
```

Category

TablePrinter

Components

TDbTableColumn, TTableColumn

Description

This event is called to update the totals, GrandTotal, PageTotal and SubTotal for a table column.

See also

GrandTotal, PageTotal, SubTotal

8.2.2 OnBodyAfter

Declaration

```
procedure OnBodyAfter(ReportPrinter: TBaseReport;
                      ReportShell: TDetailShell);
```

Category

Shell

Components

TDetailShell, TMasterShell, TReportShell

Description

This is where you de-initialize resources for use in the body of the report.

See also

OnBodyBefore, OnBodyFooter, OnBodyHeader

8.2.3 OnBodyBefore

Declaration

```
procedure OnBodyBefore(ReportPrinter: TBaseReport;
                       ReportShell: TDetailShell);
```

Category

Shell

Components

TDetailShell, TMasterShell, TReportShell

Description

This is where you initialize resources for use in the body of the report.

See also

OnBodyAfter, OnBodyFooter, OnBodyHeader

8.2.4 OnBodyFooter

Declaration

```
procedure OnBodyFooter(ReportPrinter: TBaseReport;
                       ReportShell: TDetailShell);
```
8.2.5 OnBodyHeader

Declaration

```pascal
procedure OnBodyHeader(ReportPrinter: TBaseReport;
                        ReportShell: TDetailShell);
```

Default

nil

Category

Shell

Components

TDetailShell, TMasterShell, TReportShell

Description

This event is called to print the body footer for a shell report.

See also

PrintBodyFooter, PrintBodyHeader, SectionBodyFooter, SectionBodyHeader

8.2.6 OnDetailAfter

Declaration

```pascal
procedure OnDetailAfter(ReportPrinter: TBaseReport;
                        ReportShell: TDetailShell);
```

Default

nil

Category

Shell

Components

TDetailShell, TMasterShell, TReportShell

Description

This event is called after the detail section of a shell report is printed.

See also

DetailReport, PrintDetail

8.2.7 OnDetailBefore

Declaration

```pascal
procedure OnDetailBefore(ReportPrinter: TBaseReport;
                         ReportShell: TDetailShell);
```
Default
nil

Category
Shell

Components
TDetailShell, TMasterShell, TReportShell

Description
This event is called before the detail section of a shell report is printed.

See also
DetailReport, PrintDetail

8.2.8 OnEndOfSection

Declaration
procedure OnEndOfSection(Sender: TObject);

Category
Shell

Components
TDetailShell, TMasterShell, TReportShell

Description
This event is called after the detail section of a shell report is printed.

See also
DetailReport, PrintDetail

8.2.9 OnGroupAfter

Declaration
procedure OnGroupAfter(ReportPrinter: TBaseReport;
ReportShell: TDetailShell);

Default
nil

Category
Shell

Components
TMasterShell, TReportShell

Description
This is where you de-initialize resources used in the group.

See also
OnGroupAfterLast, OnGroupBeforeFirst

8.2.10 OnGroupAfterLast

Declaration
procedure OnGroupAfterLast(ReportPrinter: TBaseReport;
ReportShell: TDetailShell);

Default
nil
Category
  Shell

Components
  TMasterShell, TReportShell

Description
  This event is called after the last group.

See also
  OnGroupAfter, OnGroupBefore

8.2.11 OnGroupBefore

Declaration
  procedure OnGroupBefore(ReportPrinter: TBaseReport;
                          ReportShell: TDetailShell);

Default
  nil

Category
  Shell

Components
  TMasterShell, TReportShell

Description
  This is where you initialize resources for use in the group.

See also
  OnGroupAfterLast, OnGroupBeforeFirst

8.2.12 OnGroupBeforeFirst

Declaration
  procedure OnGroupBeforeFirst(ReportPrinter: TBaseReport;
                             ReportShell: TDetailShell);

Default
  nil

Category
  Shell

Components
  TMasterShell, TReportShell

Description
  This event is called before the first group.

See also
  OnGroupAfter, OnGroupBefore

8.2.13 OnGroupFooter

Declaration
  procedure OnGroupFooter(ReportPrinter: TBaseReport;
                          ReportShell: TDetailShell);

Default
  nil
Category
Shell

Components
TMasterShell, TReportShell

Description
This event is called to print the group footer for a shell report.

See also
PrintBodyFooter, PrintBodyHeader, SectionGroupFooter, SectionGroupHeader

8.2.14 OnGroupFooter

Declaration
procedure OnGroupFooter(ReportPrinter: TBaseReport;
                        ReportShell: TDetailShell);

Default
nil

Category
Shell

Components
TMasterShell, TReportShell

Description
This event is called to print the group header for a shell report.

See also
PrintBodyFooter, PrintBodyHeader, SectionGroupFooter, SectionGroupHeader

8.2.15 OnLabelAfter event

Declaration
procedure OnLabelAfter(ReportPrinter: TBaseReport;
                        LabelShell: TLabelShell;
                        var Valid: Boolean);

Category
Label

Components
TLabelShell

Description
This event is where you would de-initialize any resources for a label.

See also
OnLabelBefore, OnLabelPrint

8.2.16 OnLabelBefore

Declaration
procedure OnLabelBefore(ReportPrinter: TBaseReport;
                        LabelShell: TLabelShell;
                        var Valid: Boolean);

Category
Label
Components
**TLabelShell**

Description
This event is where you would initialize any resources for a label.

See also
*OnLabelAfter, OnLabelPrint*

### 8.2.17 OnLabelPrint event

**Declaration**
```pascal
procedure OnLabelPrint(ReportPrinter: TBaseReport;
LabelShell: TLabelShell;
var Valid: Boolean);
```

**Category**
**Label**

**Components**
**TLabelShell**

Description
This event is called to print the contents of each label.

See also
*Col, Row, TextWidth*

### 8.2.18 OnOverFlow event

**Declaration**
```pascal
procedure OnOverFlow(TTableColumn: TTableColumn);
```

**Category**
**TablePrinter**

**Components**
**TDbTableColumn, TTableColumn**

Description
This event is called if text is too wide to print in the current column and OverflowMethod for the TTableColumn component is set to omUser. This allows you to modify the text however you wish so that it will fit.

See also
*OverflowMethod*

### 8.2.19 OnPageAfter

**Declaration**
```pascal
procedure OnPageAfter(ReportPrinter: TBaseReport;
LabelShell: TLabelShell);
```

**Default**
nil

**Category**
**Shell**

**Components**
**TLabelShell, TReportShell**
Description
This event is called after each page of a shell report.

8.2.20 OnPageBefore
Declaration
procedure OnPageBefore(ReportPrinter: TBaseReport;
                        LabelShell: TLabelShell);
Default
nil
Category
Shell
Components
TLabelShell, TReportShell
Description
This event is called before each page of a shell report.

8.2.21 OnPageFooter
Declaration
procedure OnPageFooter(ReportPrinter: TBaseReport;
                        ReportShell: TDetailShell);
Default
nil
Category
Shell
Components
TReportShell
Description
This event is called to print the page footer for a shell report.
See also
PrintPageFooter, PrintPageHeader, SectionPageFooter, SectionPageHeader

8.2.22 OnPageHeader
Declaration
procedure OnPageHeader(ReportPrinter: TBaseReport;
                        ReportShell: TDetailShell);
Default
nil
Category
Shell
Components
TReportShell
Description
This event is called to print the page header for a shell report.
See also
PrintPageFooter, PrintPageHeader, SectionPageFooter, SectionPageHeader
8.2.23 **OnReportAfter**

**Declaration**

```pascal
procedure OnReportAfter: TReportEvent;
```

**Default**

`nil`

**Category**

`Shell`

**Components**

` TLabelShell, TRReportShell`

**Description**

This event is called after a shell report.

8.2.24 **OnReportBefore**

**Declaration**

```pascal
```

**Default**

`nil`

**Category**

`Shell`

**Components**

` TLabelShell, TRReportShell`

**Description**

This event is called before a shell report.

8.2.25 **OnReportFooter**

**Declaration**

```pascal
procedure OnReportFooter: TReportEvent;
```

**Default**

`nil`

**Category**

`Shell`

**Components**

` TRReportShell`

**Description**

This is where you put the output that goes into the report footer.

**See also**

`PrintReportFooter, PrintReportHeader, SectionReportFooter, SectionReportHeader`

8.2.26 **OnReportHeader**

**Declaration**

```pascal
procedure OnReportHeader: TReportEvent;
```

**Default**

`nil`
Category
Shell

Components
TReportShell

Description
This is where you put the output that goes into the report header.

See also
PrintReportFooter, PrintReportHeader, SectionReportFooter, SectionReportHeader

8.2.27 OnRowAfter

Declaration
procedure OnRowAfter: TReportValidEvent;

Default
nil

Category
Shell

Components
TDetailShell, TMasterShell, TReportShell

Description
This event is where you de-initialize output that goes into a row. A value of false for Valid results in no more rows being assumed.

See also
PrintRow, SectionRow

8.2.28 OnRowBefore

Declaration
procedure OnRowBefore: TReportValidEvent;

Default
nil

Category
Shell

Components
TDetailShell, TMasterShell, TReportShell

Description
This event is where you initialize output that goes into a row. A value of false for Valid results in no more rows being assumed.

See also
PrintRow, SectionRow

8.2.29 OnRowPrint

Declaration
procedure OnRowPrint: TReportValidEvent;

Default
nil
Category
Shell

Components
TDetailShell, TMasterShell, TReportShell

Description
This event is where you put the output that goes into a row. A value of false for Valid results in no more rows being assumed.

See also
PrintRow, SectionRow
8.3 Methods

This will be a list of all archived methods (those only available in the BEX version).

8.3.1 Default

Declaration

procedure Default(var Valid: Boolean);

Category

TablePrinter

Components

TTablePrinter, TDbTablePrinter

Description

This method will call the original code that would have executed if the OnGetNextRow, OnInitMaster, OnInitTable and OnValidateRow events are not overridden.

See also

OnGetNextRow, OnInitMaster, OnInitTable, OnValidateRow

8.3.2 PrintBodyFooter

Declaration

procedure PrintBodyFooter;

Category

Shell

Components

TDetailShell, TMasterShell, TReportShell

Description

This method will reprint the body footer by calling OnBodyFooter events with IsReprint set to true.

See also

OnBodyFooter, OnBodyHeader

Example (Delphi)

MasterShell1.PrintBodyFooter;

Example (C++ Builder)

MasterShell1->PrintBodyFooter();

8.3.3 PrintBodyHeader

Declaration

procedure PrintBodyHeader;

Category

Shell

Components

TDetailShell, TMasterShell, TReportShell

Description

This method will reprint the body header by calling OnBodyHeader events with IsReprint set to true.

See also

OnBodyFooter, OnBodyHeader
8.3.4 PrintDetail

Declaration

procedure PrintDetail;

Category

Shell

Components

TMasterShell, TReportShell

Description

This method will reprint the detail section of the group by calling OnDetailBefore, DetailReport.Execute and then OnDetailAfter with IsReprint set to true.

See also

DetailReport, OnDetailAfter, OnDetailBefore

Example (Delphi)

ReportShell1.PrintDetail;

Example (C++ Builder)

ReportShell1->PrintDetail();

8.3.5 PrintGroupFooter

Declaration

procedure PrintGroupFooter;

Category

Shell

Components

TMasterShell, TReportShell

Description

This method will reprint the group footer by calling the OnGroupFooter events with IsReprint set to true.

See also

OnGroupFooter, OnGroupHeader

Example (Delphi)

MasterShell1.PrintGroupFooter;

Example (C++ Builder)

MasterShell1->PrintGroupFooter();

8.3.6 PrintGroupHeader

Declaration

procedure PrintGroupHeader;

Category

Shell

Components

TMasterShell, TReportShell
### 8.3.7 PrintPageFooter

**Declaration**

```delphi
procedure PrintPageFooter;
```

**Category**

Shell

**Components**

TReportShell

**Description**

This method will reprint the page footer by calling the `OnPageFooter` event with `IsReprint` set to true.

**See also**

OnPageFooter, OnPageHeader

**Example** (Delphi)

```delphi
ReportShell1.PrintPageFooter;
```

**Example** (C++ Builder)

```cpp
ReportShell1->PrintPageFooter();
```

### 8.3.8 PrintPageHeader

**Declaration**

```delphi
procedure PrintPageHeader;
```

**Category**

Shell

**Components**

TReportShell

**Description**

This method will reprint the page header by calling the `OnPageHeader` event with `IsReprint` set to true.

**See also**

OnPageFooter, OnPageHeader

**Example** (Delphi)

```delphi
ReportShell1.PrintPageHeader;
```

**Example** (C++ Builder)

```cpp
ReportShell1->PrintPageHeader();
```

### 8.3.9 PrintReportFooter

**Declaration**

```delphi
procedure PrintReportFooter;
```
Category
Shell

Components
TReportShell

Description
This method will reprint the report footer by calling the OnReportFooter event with IsReprint set to true.

See also
OnReportFooter, OnReportHeader

Example (Delphi)
ReportShell1.PrintReportFooter;

Example (C++ Builder)
ReportShell1->PrintReportFooter();

8.3.10 PrintReportHeader

Declaration
procedure PrintReportHeader;

Category
Shell

Components
TReportShell

Description
This method will reprint the report header by calling the OnReportHeader event with IsReprint set to true.

See also
OnReportFooter, OnReportHeader

Example (Delphi)
ReportShell1.PrintReportHeader;

Example (C++ Builder)
ReportShell1->PrintReportHeader();

8.3.11 PrintRow

Declaration
procedure PrintRow;

Category
Shell

Components
TDetailShell, TMasterShell, TReportShell

Description
This method will reprint the row section of the group by calling OnRowBefore and OnRowPrint.

See also
OnRowAfter, OnRowBefore, OnRowPrint

Example (Delphi)
DetailShell1.PrintRow;
Example (C++ Builder)
DetailShell1->PrintRow();

8.3.12 SetupSection

Declaration
function SetupSection(BaseReport: TBaseReport): Boolean;

Category
ReportSection

Components
TReportSection

Description
This method will adjust the section to match the settings defined. If there is not enough height available for
the section then a new page will be generated. The tabs and font will also be initialized if TabIndex or
FontIndex are non-zero. This method is called automatically by the shell and table printer components.

See also
Enabled

Example (Delphi)
ReportSection.SetupSection(Sender as TBaseReport);

Example (C++ Builder)
ReportSection->SetupSection( dynamic_cast<TBaseReport*>(Sender));

8.4 Properties

This will be a list of all archived properties (those only available in the BEX version).

8.4.1 AsFloat

Declaration
property AsFloat: Double;

Default
0.0

Category
TablePrinter

Components
TDbTableColumn, TTableColumn

Description
Returns or sets the current contents of a table column as a floating point value. If the contents can not be
converted to a floating point value, 0.0 will be returned.

See also
AsInteger

Example (Delphi)
WITH TableColumn do begin
  PageTotal := PageTotal + AsFloat;
end; { with }

Example (C++Builder)
TableColumn->PageTotal := TableColumn->PageTotal + TableColumn->AsFloat;
8.4.2 AsInteger

**Declaration**

```
property AsInteger: Integer;
```

**Default**

0

**Category**

TablePrinter

**Components**

TDbTableColumn, TTableColumn

**Description**

Returns or sets the current contents of a table column as an integer value. If the contents cannot be converted to an integer value, 0 will be returned.

**See also**

AsFloat

**Example (Delphi)**

```delphi
WITH TableColumn do begin
  GrandTotal := GrandTotal + AsInteger;
end;
```

**Example (C++Builder)**

```
TableColumn->GrandTotal := TableColumn->GrandTotal + TableColumn->AsInteger;
```

8.4.3 Border

**Declaration**

```
property Border: Double;
```

**Default**

0.0

**Category**

Label

**Components**

TLABELShell

**Description**

This property defines the border, in units, around the edges of each label that the section will be initialized to before entering the OnLabelPrint event.

**See also**

SectionBottom, SectionLeft, SectionRight, SectionTop

**Example (Delphi)**

```
// Make sure there is a 0.1" border around the label
LabelShell1.Border := 0.1;
```

**Example (C++Builder)**

```
LabelShell1->Border = 0.1;
```

8.4.4 Bottom

**Declaration**

```
property Bottom: double;
```
Default
0.0

Category
ReportSection

Description
This property defines the placement for the bottom of the section. All values are in units and are relative to a specific position on the page, such as the page or margin edge, which is defined by the relative XxxxxMethod property (such as LeftMethod).

See also
Left, Right, Top, BottomMethod

Example (Delphi)
ReportSection.Bottom := 0.5;

Example (C++Builder)
ReportSection->Bottom := 0.5;

8.4.5 BottomMethod

Declaration
property BottomMethod: TDistanceMethod;

Default
dmMargin (except TopMethod is dmSpecial)

Category
ReportSection

Description
These properties will define where each side of the section will be measured.
- **dmPage**: will measure from the closest page edge
- **dmMargin**: will measure from the closest margin edge
- **dmSpecial** for the SectionTop will measure from the current cursor position
- **dmSpecial** for SectionRight and SectionBottom will measure from the left and top section positions respectively.

See also
Left, LeftMethod, RightMethod, TopMethod

Example (Delphi)
ReportSection.LeftMethod := pjMargin;

Example (C++Builder)
ReportSection->LeftMethod = pjMargin;

8.4.6 BoxLines

Declaration
property BoxLines: TBoxLines;

Default
biAll

Category
TablePrinter

Components
TDbTableColumn, TTableColumn, TTableSection
Description
This property returns or sets the row box settings for each table column or table section component.
These constants are listed below:

- **blAll**: Lines drawn on all sides
- **blNone**: No lines drawn
- **blBottom**: Line drawn on bottom only
- **blLeft**: Line drawn on left side only
- **blLeftBottom** and bottom
- **blLeftRight** and right
- **blLeftTop** and top
- **blRight**: Line drawn on right side only
- **blRightBottom** and bottom
- **blRightTop** and top
- **blTop**: Line drawn on top only
- **blTopBottom** Lines drawn on top and bottom
- **blNoTop**
- **blNoBottom**
- **blNoLeft**
- **blNoRight**

See also
- **TBoxLines**

Example (Delphi)

```delphi
With TableColumn do begin
  BoxLines := blNoLeft;
end; { with }
```

Example (C++Builder)

```cpp
TableColumn->BoxLines = blNoLeft;
```

8.4.7 Col

**Declaration**

```
property Col: integer;
```

**Default**

none

**Category**

**Label**

**Components**

**TLabelShell**

**Description**

This property will return the current label column that is being printed. This property will only return valid values while the TLabelShell component is executing.

See also
- **Row**

Example (Delphi)

```delphi
Print('On Column ' + IntToStr(Col));
```

Example (C++Builder)

```cpp
rp1->Print("On Column " + IntToStr(Col));
```

8.4.8 Description

**Declaration**

```
property Description: TComponentName;
```
Default
none

Category
TablePrinter

Components
TDbTableColumn, TTableColumn, TTableSection

Description
This property returns or sets the description of a table section or table column component. This property is normally only used at design time to identify an item.

See also
PrintDetail

Example (Delphi)
Description := 'Customer Name';

Example (C++Builder)
Description = 'Customer Name';

8.4.9 DetailReport

Declaration
property DetailReport: TDetailShell;

Default
nil

Category
Shell

Components
TMasterShell, TReportShell

Description
This property defines the shell component that will be called to print the detail section of the report. TReportShell components cannot be used as a detail report of another shell component. The Execute method of DetailReport will be called between the OnDetailBefore and OnDetailAfter events.

See also
PrintDetail

Example (Delphi)
MasterShell1.DetailReport := DetailShell1;

Example (C++Builder)
MasterShell1->DetailReport = DetailShell1;

8.4.10 DetailTablePrinter

Declaration
property DetailTablePrinter: TBaseShell;

Default
nil

Category
TablePrinter
Components
   TDbTablePrinter, TTablePrinter

Description
   This property defines the table printer component that will be called to print the detail section.

See also
   PrintDetail

Example (Delphi)
   DBTablePrinter1.DetailTablePrinter := DBTablePrinter2;

Example (C++Builder)
   DBTablePrinter1->DetailTablePrinter = DBTablePrinter2;

8.4.11 DisplayFormat

Declaration
   property DisplayFormat: string;

Default
   (empty)

Category
   TablePrinter

Components
   TDbTableColumn, TTableColumn

Description
   This property defines the formatting string that will be used when converting numeric or date/time data to text. If this property is blank then default formatting will occur.

See also
   FormatFloat, FormatDateTime

8.4.12 DrawExtents

Declaration
   property DrawExtents: Boolean;

Default
   false

Category
   Label

Components
   TLabelShell

Description
   This property determines whether the sides of the label are drawn. This can be useful for determining the placement of text without having to actually print on labels.

See also
   DrawPen, DrawPreviewOnly

Example (Delphi)
   LabelShell1.DrawExtents := true;
Example (C++Builder)
LabelShell1->DrawExtents = true;

8.4.13 DrawPen

Declaration
property DrawPen: TPen;

Default
stock pen

Category
Label

Components
TLabelShell

Description
This property defines the pen used to draw the sides of the label.

See also
DrawExtents, DrawPreviewOnly, TPen

Example (Delphi)
LabelShell1.DrawPen.Color := clBlue;

Example (C++Builder)
LabelShell1->DrawPen->Color = clBlue;

8.4.14 DrawPreviewOnly

Declaration
property DrawPreviewOnly: Boolean;

Default
true

Category
Label

Components
TLabelShell

Description
This property will determine whether the label sides drawn by DrawPen appear in the print preview screen only or on both the printer and preview screen.

See also
DrawExtents, DrawPen

Example (Delphi)
LabelShell1.DrawPreviewOnly := false;

Example (C++Builder)
LabelShell1->DrawPreviewOnly = false;

8.4.15 Enabled

Declaration
property Enabled: Boolean;

Default
true
Category
   ReportSection

Components
   TReportSection

Description
   This property will enable or disable the section settings during a call to SetupSection. If this property is set to false then no settings for the section will be used.

See also
   SetupSection

Example (Delphi)
   ReportSection.Enabled := false;

Example (C++Builder)
   ReportSection->Enabled = false;

8.4.16 Font

Declaration
   property Font: TFont;

Default
   System font

Category
   TablePrinter

Components
   TDbTableColumn, TTableColumn, TTablePrinter, TTableSection

Description
   This property defines or returns the font that will be used to draw the contents of a table section or table column.

See also
   Other FontXxxx properties

Example (Delphi)
   TableColumn.Font.Color := clRed;

Example (C++Builder)
   TableColumn->Font->Color = clRed;

8.4.17 FontIndex

Declaration
   property FontIndex: integer;

Default
   0

Category
   ReportSection

Components
   TReportSection
Description
This property defines the saved font position that will be initialized during a call to SetupSection. A value of 0 will not cause any font settings to be changed.

See also
SaveFont

Example (Delphi)
// Use the font settings save in position 1
FontIndex := 1;

Example (C++Builder)
ReportShell1->SectionGroupFooter->FontIndex = 1;

8.4.18 Height

Declaration
property Height: double;

Default
0.0

Category
ReportSection

Components
TReportSection

Description
This property defines the minimum height for a section. If the value is 0 then no minimum height will be required. If the value is greater than zero and there is not enough room in the section a new page will be generated.

See also
HeightMethod, Reprint

Example (Delphi)
ReportSection.HeightMethod := hmUnits;
ReportSection.Height := 0.5;

Example (C++Builder)
ReportSection->HeightMethod = hmUnits;
ReportSection->Height = 0.5;

8.4.19 HeightMethod

Declaration
property HeightMethod: THeightMethod;

Default
hmLines

Category
ReportSection

Components
TReportSection

Description
This property defines the units that Height is measured in. hmUnits will define the height in terms of units while hmLines will define the height in terms of a number of lines (defined by LineHeight).
See also

*Height*

**Example**

see *Height*(ReportSection)

### 8.4.20 IsNewPage

**Declaration**

property IsNewPage: Boolean;

**Category**

Shell

**Components**

TDetailShell, TMasterShell, TReportShell

**Description**

This property will be true immediately after a new page is generated and will remain true until after the first row is printed. This can be useful for determining if a title bar for a table needs to be printed.

See also

*PrintDetail*

**Example** (Delphi)

IF IsNewPage then begin
  ReprintTitleBar;
end;  { if }
PrintNormalData;

**Example** (C++Builder)

if (ReportShell->IsNewPage) {
  ReprintTitleBar();
}  // if
PrintNormalData();

### 8.4.21 IsReprint

**Declaration**

property IsReprint: Boolean;

**Category**

Shell

**Components**

TDetailShell, TMasterShell, TReportShell

**Description**

This property will be true inside the printing events if they were called to print again. This can happen as a result of the Reprint property or by a call to any of the Shell PrintXxxx methods (such as PrintBodyHeader).

See also

*Reprint, all Shell PrintXxxx methods*

**Example** (Delphi)

RvRenderPrinter.Print('Normal Data');
IF IsReprint then begin
  ReportPrinter.Print(' - continued...');
end;  { if }
ReportPrinter.NewLine;
Example (C++Builder)
   RvRenderPrinter->Print("Normal Data");
   if (ReportShell->IsReprint) {
      RvRenderPrinter->Print(" - continued...");
   } // if
   RvRenderPrinter->NewLine();

8.4.22 LabelBrand

Declaration
   property LabelBrand: TLabelBrand;

Default
   lbAV5160

Category
   Label

Components
   TLabelShell

Description
   This property can be used to define the brand of labels that you are using. There are many model
   numbers for the Avery© label line already supported or you can type in your own selections using the
   lbCustom value.

See also
   CPI, NewLine

Example (Delphi)
   LabelShell11.LabelBrand := lbAV5267;

Example (C++Builder)
   LabelShell11->LabelBrand = lbAV5267;

8.4.23 LabelHeight

Declaration
   property LabelHeight: double;

Default
   1.0

Category
   Label

Components
   TLabelShell

Description
   This property defines the height of the label in units.

See also
   LabelWidth, NumAcross, NumDown

Example (Delphi)
   LabelShell11.LabelHeight := 0.75;

Example (C++Builder)
   LabelShell11->LabelHeight = 0.75;
8.4.24 LabelShape

Declaration

property LabelShape: TLabelShape;

Default

IsRoundRect

Category

Label

Components

TLabelShell

Description

Specifies the shape of the label that appears in the preview screen.

- IsRect represents a rectangle with square corners
- IsRound represents an elliptical or circular shape
- IsRoundRect represents a rectangle with the corners rounder off

8.4.25 LabelWidth

Declaration

property LabelWidth: double;

Default

2.63

Category

Label

Components

TLabelShell

Description

This property defines the width of the label in units.

See also

LabelHeight, NumAcross, NumDown

Example (Delphi)

LabelShell1.LabelWidth := 2.5;

Example (C++Builder)

LabelShell1->LabelWidth = 2.5;

8.4.26 Left

Declaration

property Left: double;

Default

0.0

Category

ReportSection

Components

TReportSection
8.4.27 LeftMethod

**Declaration**
property LeftMethod: TDistanceMethod;

**Default**
dmMargin (except TopMethod is dmSpecial)

**Category**
ReportSection

**Description**
This property defines where the left side of the section will be measured.
- *dmPage* will measure from the closest page edge
- *dmMargin* will measure from the closest margin edge
- *dmSpecial* for the *SectionTop* will measure from the current cursor position
- *dmSpecial* for *SectionRight* and *SectionBottom* will measure from the left and top section positions respectively.

**See also**
BottomMethod, LeftMethod, RightMethod, TopMethod

**Example (Delphi)**
ReportSection.LeftMethod := pjMargin;

**Example (C++Builder)**
ReportSection->LeftMethod = pjMargin;

8.4.28 MinHeight

**Declaration**
property MinHeight: double;

**Default**
0.0

**Category**
ReportSection

**Components**
TReportSection

**Description**
This property defines the minimum height for a section. If the value is 0 then no minimum height will be required. If the value is greater than zero and there is not enough room in the section a new page will be generated.
See also
- HeightMethod, Reprint

**Example (Delphi)**

```delphi
ReportSection.HeightMethod := hmUnits;
ReportSection.MinHeight := 0.5;
```

**Example (C++Builder)**

```cpp
ReportSection->HeightMethod = hmUnits;
ReportSection->MinHeight = 0.5;
```

### 8.4.29 NumAcross

**Declaration**

```
property NumAcross: integer;
```

**Default**

3

**Category**

- Label

**Components**

- TLabelShell

**Description**

This property defines the number of labels across each page.

See also
- LabelHeight, LabelWidth, NumDown

**Example (Delphi)**

```delphi
LabelShell1.NumAcross := 2;
```

**Example (C++Builder)**

```cpp
LabelShell1->NumAcross = 2;
```

### 8.4.30 NumDown

**Declaration**

```
property NumDown: integer;
```

**Default**

10

**Category**

- Label

**Components**

- TLabelShell

**Description**

This property defines the number of labels down each page.

See also
- LabelHeight, LabelWidth, NumAcross

**Example (Delphi)**

```delphi
LabelShell1.NumDown := 7;
```

**Example (C++Builder)**

```cpp
LabelShell1->NumDown = 7;
```
8.4.31 PrintByRow

Declaration

property PrintByRow: Boolean;

Default
true

Category
Label

Components
TLabelShell

Description
This property determines whether the TLabelShell component processes the labels of the page by rows or columns. If PrintByRow is true then all labels on the top row would be printed first. If PrintByRow is false then all labels on the left column would be printed first.

See also
Col, Row

Example (Delphi)
LabelShell1.PrintByRow := false;

Example (C++Builder)
LabelShell1->PrintByRow = false;

8.4.32 Reprint

Declaration

property Reprint: Boolean;

Default
false

Category
Shell

Components
TMasterShell, TReportShell

Description
This property defines whether the current row will be called to reprint if the detail section wraps to a second page. This can be useful to reprint the master record for detail records on another page.

See also
IsReprint

Example (Delphi)
MasterShell1.Reprint := true;

Example (C++Builder)
MasterShell1->Reprint = true;

8.4.33 Right

Declaration

property Right: double;

Default
0.0
Rave Reports Developer Reference

Category
  ReportSection

Components
  TReportSection

Description
These properties define the placement for each side of the section. All values are in units and are relative to a specific position on the page, such as the page or margin edge, which is defined by the relative XxxxxMethod property (such as LeftMethod).

See also
  BottomMethod, LeftMethod, RightMethod, TopMethod

Example (Delphi)
  ReportSection.Bottom := 0.5;

Example (C++Builder)
  ReportSection->Bottom = 0.5;

8.4.34 RightMethod

Declaration
  property RightMethod: TDistanceMethod;

Default
  dmMargin (except TopMethod is dmSpecial)

Category
  ReportSection

Description
This property defines where the left side of the section will be measured.
  

| dmPage       | will measure from the closest page edge |
| dmMargin     | will measure from the closest margin edge |
| dmSpecial    | for the SectionTop will measure from the current cursor position |
| dmSpecial    | for SectionRight and SectionBottom will measure from the left and top section positions respectively. |

See also
  BottomMethod, Left, LeftMethod, TopMethod

Example (Delphi)
  ReportSection.LeftMethod := pjMargin;

Example (C++Builder)
  ReportSection->LeftMethod = pjMargin;

8.4.35 Row

Declaration
  property Row: Boolean;

Default
  (current row of the label being printed)

Category
  Label

Components
  TLabelShell
Description
This property will return the current row of the label being printed.

See also
Col

8.4.36 SectionBodyFooter

Declaration
property SectionBodyFooter: TReportSection;

Default
Standard section values

Category
Shell

Components
TMasterShell, TReportShell

Description
Brings up the section editor, which you can use to define the section settings for the body footer.

See also
OnBodyFooter, OnBodyHeader

Example (Delphi)
SectionBodyFooter.Lines := 1;

Example (C++Builder)
SectionBodyFooter->Lines = 1;

8.4.37 SectionBodyHeader

Declaration
property SectionBodyHeader: TReportSection;

Default
Standard section values

Category
Shell

Components
TMasterShell, TReportShell

Description
Brings up the section editor, which you can use to define the section settings for the body header.

See also
OnBodyFooter, OnBodyHeader

Example (Delphi)
SectionBodyHeader.Lines := 1;

Example (C++Builder)
SectionBodyHeader->Lines = 1;

8.4.38 SectionGroupFooter

Declaration
property SectionGroupFooter: TReportSection;
Default
  Standard section values

Category
  Shell

Components
  TMasterShell, TReportShell

Description
  Brings up the section editor, which you can use to define the section settings for the group footer.

See also
  OnGroupFooter, OnGroupHeader

Example (Delphi)
  SectionGroupFooter.FontIndex := 4;

Example (C++Builder)
  SectionGroupFooter->FontIndex = 4;

8.4.39 SectionGroupFooter

Declaration
  property SectionGroupFooter: TReportSection;

Default
  Standard section values

Category
  Shell

Components
  TMasterShell, TReportShell

Description
  Brings up the section editor, which you can use to define the section settings for the group header.

See also
  OnGroupFooter, OnGroupHeader

Example (Delphi)
  SectionGroupHeader.FontIndex := 5;

Example (C++Builder)
  SectionGroupHeader->FontIndex = 5;

8.4.40 SectionPageFooter

Declaration
  property SectionPageFooter: TReportSection;

Default
  Standard section values

Category
  Shell

Components
  TReportShell
**Description**

Brings up the section editor, which you can use to define the section settings for the page footer.

See also

*OnPageFooter, OnPageHeader*

**Example (Delphi)**

```delphi
SectionPageFooter.BottomMethod := dmPage;
SectionPageFooter.Bottom := 0.25;
```

**Example (C++Builder)**

```cpp
SectionPageFooter->BottomMethod = dmPage;
SectionPageFooter->Bottom = 0.25;
```

### 8.4.41 SectionPageFooter

**Declaration**

```delphi
gtype SectionPageFooter: TReportSection;
```

**Default**

Standard section values

**Category**

*Shell*

**Components**

*TReportShell*

**Description**

Brings up the section editor, which you can use to define the section settings for the page header.

See also

*OnPageFooter, OnPageHeader*

**Example (Delphi)**

```delphi
SectionPageHeader.TopMethod := dmPage;
SectionPageHeader.Top := 0.25;
```

**Example (C++Builder)**

```cpp
SectionPageHeader->TopMethod = dmPage;
SectionPageHeader->Top = 0.25;
```

### 8.4.42 SectionReportFooter

**Declaration**

```delphi
gtype SectionReportFooter: TReportSection;
```

**Default**

Standard section values

**Category**

*Shell*

**Components**

*TReportShell*

**Description**

Brings up the section editor, which you can use to define the section settings for the report footer.

See also

*OnReportFooter, OnReportHeader*
Example (Delphi)

```
SectionReportHeader.TopMethod := dmMargin;
SectionReportHeader.Top := 0.25;
```

Example (C++Builder)

```
SectionReportHeader->TopMethod = dmMargin;
SectionReportHeader->Top = 0.25;
```

8.4.43 SectionReportHeader

Declaration

```
property SectionReportHeader: TReportSection;
```

Default

Standard section values

Category

Shell

Components

TReportShell

Description

Brings up the section editor, which you can use to define the section settings for the report header.

See also

OnReportFooter, OnReportHeader

Example (Delphi)

```
SectionReportHeader.TopMethod := dmMargin;
SectionReportHeader.Top := 0.25;
```

Example (C++Builder)

```
SectionReportHeader->TopMethod = dmMargin;
SectionReportHeader->Top = 0.25;
```

8.4.44 SectionRow

Declaration

```
property SectionRow: TReportSection;
```

Default

Standard section values

Category

Shell

Components

TDetailShell, TMasterShell, TReportShell

Description

Brings up the section editor, which you can use to define the section settings for each row.

See also

OnRowPrint

Example (Delphi)

```
SectionRow.TabIndex := 2;
```

Example (C++Builder)

```
SectionRow->TabIndex = 2;
```
8.4.45 SkipNum

Declaration

    property SkipNum: integer;

Default

    0

Category

    Label

Components

    TLabelShell

Description

When SkipNum is set to a non-zero value, then TLabelShell will skip that many labels before it start printing. This can be useful if you want to prompt your users for the number of labels that have already been printed on the first sheet. This will automatically be reset to 0 after the label report has been executed.

See also

    StartCol, StartRow

Example (Delphi)

    DbTablePrinter2.SkipNum := 7;

Example (C++ Builder)

    DbTablePrinter2->SkipNum = 7;

8.4.46 SpacingHeight

Declaration

    property SpacingHeight: double;

Default

    1.0

Category

    Label

Components

    TLabelShell

Description

This property returns or sets the label height plus the vertical spacing between two adjacent labels.

See also

    SpacingLeft, SpacingTop, SpacingWidth

Example (Delphi)

    LabelShell11.SpacingHeight := 1.2;

Example (C++ Builder)

    LabelShell11->SpacingHeight = 1.2;

8.4.47 SpacingLeft

Declaration

    property SpacingLeft: double;
Rave Reports Developer Reference

**8.4.48 SpacingTop**

**Declaration**

```delphi
property SpacingTop: double;
```

**Default**

0.5

**See also**

SpacingHeight, SpacingLeft, SpacingWidth

**Example (Delphi)**

```delphi
LabelShell1.SpacingTop := 0.75;
```

**Example (C++Builder)**

```c++
LabelShell1->SpacingTop = 0.75;
```

---

**8.4.49 SpacingWidth**

**Declaration**

```delphi
property SpacingWidth: double;
```

**Default**

2.75

**See also**

SpacingHeight, SpacingLeft, SpacingWidth

**Example (Delphi)**

```delphi
LabelShell1.SpacingWidth := 0.75;
```

**Example (C++Builder)**

```c++
LabelShell1->SpacingWidth = 0.75;
```
Description
This property returns or sets the label width plus the horizontal spacing between two adjacent labels.

See also
SpacingHeight, SpacingLeft, SpacingTop, TextWidth

Example (Delphi)
LabelShell1.SpacingWidth := 3.00;

Example (C++Builder)
LabelShell1->SpacingWidth = 3.00;

8.4.50 StartCol

Declaration
property StartCol: integer;

Default
0

Category
Label

Components
TLabelShell

Description
Use both StartCol and StartRow to instruct TLabelShell to start printing at a particular row and column. This will automatically be reset to 0 after the label report has been executed.

See also
SkipNum, StartRow

Example (Delphi)
DbTablePrinter2.StartCol := 1;
DbTablePrinter2.StartRow := 2;

Example (C++Builder)
DbTablePrinter2->StartCol = 1;
DbTablePrinter2->StartRow = 2;

8.4.51 StartNewPage

Declaration
property StartNewPage: Boolean;

Default
false

Category
Shell

Components
TDetailShell, TMasterShell, TReportShell

Description
This property, if true, will force a page break before each new group prints. This can be useful for forms that contain one record per page.

See also
NewPage
Example (Delphi)
ReportShell1.StartNewPage := true;

Example (C++Builder)
ReportShell1->StartNewPage = true;

8.4.52 StartRow
Declaration
property StartRow: integer;

Default
0

Category
Label

Components
TLlabelShell

Description
Use both StartCol and StartRow to instruct TLabelShell to start printing at a particular row and column. This will automatically be reset to 0 after the label report has been executed.

See also
SkipNum, StartCol

Example (Delphi)
DbTablePrinter2.StartCol := 1;
DbTablePrinter2.StartRow := 2;

Example (C++Builder)
DbTablePrinter2->StartCol = 1;
DbTablePrinter2->StartRow = 2;

8.4.53 TabIndex
Declaration
property TabIndex: integer;

Default
0

Category
ReportSection

Components
TReportSection

Description
These properties defines the saved tabs position that will be initialized during a call to SetupSection. A value of 0 will not cause any tab settings to be changed.

See also
SaveTabs

Example (Delphi)
TabIndex := 4; { Use the tab settings save in position 4 }

Example (C++Builder)
TabIndex = 4; // Use the tab settings save in position 4
8.4.54 Top

Declaration

property Top: double;

Default

0.0

Category

ReportSection

Components

TReportSection

Description

These properties define the placement for each side of the section. All values are in units and are relative to a specific position on the page, such as the page or margin edge, which is defined by the relative XxxxxMethod property (such as LeftMethod).

See also

BottomMethod, LeftMethod, RightMethod, TopMethod

Example (Delphi)

ReportSection.Bottom := 0.5;

Example (C++Builder)

ReportSection->Bottom = 0.5;

8.4.55 TopMethod

Declaration

property TopMethod: TDistanceMethod;

Default

dmSpecial

Category

ReportSection

Description

This property defines where the left side of the section will be measured.

- **dmPage** will measure from the closest page edge
- **dmMargin** will measure from the closest margin edge
- **dmSpecial** for the *SectionTop* will measure from the current cursor position
- **dmSpecial** for *SectionRight* and *SectionBottom* will measure from the left and top section positions respectively.

See also

BottomMethod, Left, LeftMethod, RightMethod

Example (Delphi)

ReportSection.LeftMethod := pjMargin;

Example (C++Builder)

ReportSection->LeftMethod = pjMargin;

8.4.56 Width

Declaration

property Width: Double;
Default
1.0

Category
TablePrinter

Components
TDbTableColumn, TTableColumn, TTableSection

Description
This property defines the width, in units, of the table section or table column.

See also
StartPos

Example (Delphi)
TableColumn.Width := 2.5;
Example (C++Builder)
TableColumn->Width := 2.5;
8.5 Types

This will be a list of all archived Types (those only available in the BEX version).

8.5.1 TBoxLines

Declaration

\[
\text{TBoxLines} = \{\text{blNone, blLeft, blRight, blLeftRight, blTop, blLeftTop, blRightTop, blNoBottom, blBottom, blLeftBottom, blRightBottom, blNoTop, blTopBottom, blNoRight, blNoLeft, blAll}\};
\]

Category

TablePrinter

Description

- blAll: Lines drawn on all sides
- blNone: No lines drawn
- blBottom: Line drawn on bottom only
- blLeft: Line drawn on left side only
- blLeftBottom: and bottom
- blLeftRight: and right
- blLeftTop: and top
- blRight: Line drawn on right side only
- blRightBottom: and bottom
- blRightTop: and top
- blTop: Line drawn on top only
- blTopBottom: Lines drawn on top and bottom
- blNoTop: All lines except indicated are drawn
- blNoBottom
- blNoLeft
- blNoRight

See also

BoxLines

Example

See BoxLines

8.5.2 TPrintJustifyVert

Declaration

\[
\text{TPrintJustifyVert} = \{\text{pjTop, pjMiddle, pjBottom}\};
\]

Category

Printing

Description

- pjTop: Justify at the top of the row box
- pjMiddle: Justify by the middle of the row box
- pjBottom: Justify by the bottom of the row box

See also

TBaseReport Class, Justify, PrintFooter, PrintHeader, SetTab

Example

See SetTab
8.6 RpDev function

Declaration

function RpDev: TRpDevice;

Category

Printer

Components

RpDevice unit

Description

This function will return the current RpDevice object that is managing printing for Rave.

See also

RpDevice.PAS (BEX only)
By Category

Chapter IX
9 By Category

This is a list of events (54), methods (204) and properties (120) by functional areas or groups.

**BarCode** 35
- BarBottom property (read/write) double
- BarCodeJustify property (read/write)
- BarCodeRotation property (read/write)
- BarHeight property (read/write)
- BarTop property (read/write) double
- BarWidth property (read/write) double
- BaseReport property (read/write)
- Bottom property (read/write) double
- Center property (read/write) double
- CheckSum property (read only) Boolean
- CodePage property (read/write)
- Create method
- Extended property (read/write) Boolean
- ExtendedText property (read only) string
- Height property (read only) double
- IsValidChar method Boolean
- Left property (read/write) double

**Memo** 46
- Position property (read/write) double
- Print method
- PrintCheckSum property (read/write) Boolean
- PrintFimA method
- PrintFimB method
- PrintFimC method
- PrintJustify property (read/write) Boolean
- PrintReadable property (read/write) Boolean
- PrintTop property (read/write) Boolean
- PrintXY method
- ReadableHeight property (read only) double
- Right property (read/write) double
- Text property (read/write) string
- TextJustify property (read/write)
- Top property (read/write) double
- UseCheckSum property (read/write) Boolean
- WideFactor property (read/write) double
- Width property (read only) double

**Control** 41
- BEX BaseReport property (read/write)

**ReportSection** BEX 14
- BEX Position property (read/write) double
- BEX Print method
- BEX PrintCheckSum property (read/write) Boolean
- BEX PrintFimA method
- BEX PrintFimB method
- BEX PrintFimC method
- BEX PrintJustify property (read/write) Boolean
- BEX PrintReadable property (read/write) Boolean
- BEX PrintTop property (read/write) Boolean
- BEX PrintXY method
- BEX ReadableHeight property (read only) double
- BEX Right property (read/write) double
- BEX Text property (read/write) string
- BEX TextJustify property (read/write)
- BEX Top property (read/write) double
- BEX UseCheckSum property (read/write) Boolean
- BEX WideFactor property (read/write) double
- BEX Width property (read only) double

**Column** 12
- BEX ClearColumns method
- BEX ColumnEnd property (read only) double
- BEX ColumnLinesLeft method
- BEX ColumnNum property (read/write) integer
- BEX Columns property (read only) integer
- BEX ColumnStart property (read only) integer
- BEX ColumnWidth property (read only) double

**Misc** 15

**Position** 44

**ReportSystem** 16

**RTF** 4

**Shell** BEX 48

**TablePrinter** BEX 66

**Tabs** 21

**Column** 24
- ClearColumns method
- ColumnEnd property (read only) double
- ColumnLinesLeft method
- ColumnNum property (read/write) integer
- Columns property (read only) integer
- ColumnStart property (read only) integer
- ColumnWidth property (read only) double

**Font** 24
- NewColumn method
- NewLine method
- NewPara method

**Graphics** 40

**ReportSection** BEX 14

**Label** BEX 23

**Rave** 68

**Render** 26

**Units** 15
### 9.3 Category Control

- **Abort** method
- **AbortPage** method
- **AccuracyMethod** property (read/write/pub)
- **AllowAll** method
- **AllowBeforePrint** method
- **AllowPrinterOnly** method
- **BaseReport** property (read/write)
- **CurrentPage** property (read only) integer
- **EndLink** method
- **Engine** property (read/write/pub)
- **Execute** method (2)
- **ExecuteCustom** method
- **FileName** property (read/write/pub) string
- **Finish** method
- **FirstPage** property (read/write/pub) integer
- **LastPage** property (read/write/pub) integer
- **MakeLink** method
- **NewColumn** method
- **NewLine** method
- **NewPage** method
- **NewPara** method

### 9.4 Category Font

- **AssignFont** method
- **Bold** property (read/write) Boolean
- **CreateFont** method
- **(FontAlign** property (read/write)
- **FontCharset** property (read/write) byte
- **FontColor** property (read/write)
- **FontHandle** property (read only)
- **FontHeight** property (read/write)
- **FontName** property (read/write) string
- **FontPitch** property (read/write)
- **FontRotation** property (read/write) integer
- **FontSize** property (read/write) double
- **FontWidth** property (read/write) double

- **Italic** property (read/write) Boolean
- **PopFont** method Boolean
- **PushFont** method Boolean
- **SetFontAlign** method Boolean
- **SaveFont** method Boolean
- **SetFont** method
- **Strikeout** property (read/write) Boolean
- **Subscript** property (read/write) Boolean
- **Superscript** property (read/write) Boolean
- **TFontAlign** type
- **UnderLine** property (read/write) Boolean
9.5 Category Graphics

Arc method
BKColor property (read/write)
BrushCopy method
CalcGraphicHeight method double
CalcGraphicWidth method double
Chord method
CopyRect method
CreateBrush method
CreatePen method
CreatePoint method
CreateRect method

Draw method
DrawFocusRect method
Ellipse method
FillRect method
FillFill method
FrameMode property (read/write)
FrameRect method

GraphicFieldToBitmap method
LineTo method
MoveTo method
NoBufferLine property (read/write) Boolean
Pie method
Polygon method
Polyline method
PrintBitmap method
PrintBitmapRect method
PrintImageRect method
Rect method
Rectangle method
RegisterGraphic method
ReUseGraphic method
RoundRect method

SetBrush method
SetPen method
ShadeToColor method
StretchDraw method

TextBKMode property (read/write/pub)
TextRect method
TBKMode type
UnRegisterGraphic method

9.6 Category Label

Border property (read/write/pub) double
Col property (read only) integer

DrawExtents property (read/write/pub) Boolean
DrawPen property (read/write/pub)
DrawPreviewOnly property (read/write/pub) Boolean

FrameMode property (read/write/pub)
FrameRect method

GraphicFieldToBitmap method
LineTo method
MoveTo method
NoBufferLine property (read/write) Boolean
Pie method
Polygon method
Polyline method
PrintBitmap method
PrintBitmapRect method
PrintImageRect method
Rect method
Rectangle method
RegisterGraphic method
ReUseGraphic method
RoundRect method

SetBrush method
SetPen method
ShadeToColor method
StretchDraw method

TextBKMode property (read/write/pub)
TextRect method
TBKMode type
UnRegisterGraphic method

OnLabelAfter event (read/write/pub)
OnLabelBefore event (read/write/pub)
OnLabelPrint event (read/write/pub)
PrintByRow property (read/write/pub) Boolean
Row property (read only) integer

SkipNum property (read/write/pub) integer
SpacingHeight property (read/write/pub) double
SpacingLeft property (read/write/pub) double
SpacingTop property (read/write/pub) double
SpacingWidth property (read/write/pub) double
StartCol property (read/write/pub) integer
StartRow property (read/write/pub) integer
9.7 Category Memo

- **Append** method
- **AppendMemoBuf** method
- **Buffer** property (read only)
- **BufferInc** property (read/write) longint
- **ConstraintHeightLeft** method double
- **Delete** method
- **Empty** method Boolean
- **Field** property (write only)
- **FreeSaved** method
- **GetMemoLine** method string
- **GetNextLine** method
- **Insert** method
- **InsertMemoBuf** method
- **Justify** property (read/write)
- **LoadFromStream** method
- **MaxSize** property (read/write) longint
- **Memo** property (read/write)
- **MemoHeightLeft** method double
- **MemoLines** method longint
- **MemoLinesLeft** method longint
- **NoCRLF** property (read/write) Boolean
- **NoNewLine** property (read/write) Boolean
- **Pos** property (read/write) longint
- **PrintEnd** property (read/write) double
- **PrintHeight** method
- **PrintLines** method
- **PrintMemo** method
- **PrintStart** property (read/write) double
- **ReplaceAll** method
- **Reset** method
- **RestoreBuffer** method
- **RestoreState** method
- **RichEdit** property (write only) string
- **RTFField** property (write only)
- **RTFLoadFromFile** method
- **RTFLoadFromStream** method
- **RTFText** property (write only) string
- **SaveBuffer** method
- **SaveState** method
- **SaveToStream** method
- **SearchFirst** method Boolean
- **SearchNext** method Boolean
- **SetData** method
- **SetRTF** method (archived)
- **Size** property (read only) longint
- **Text** property Memo (read/write) string

9.8 Category Misc -

- **CPI** property (read/write/pub) double (Archived)
- **Create** method
- **CurrentPass** property (read/write/pub) integer
- **Destroy** method
- **LPI** property (read/write) double (Archived)
- **Macro** method
- **RestoreDataSet** property (read/write/pub) 5.1.1
- **StatusLabel** property (read/write/pub) string
- **StatusText** property (read/write/pub)
- **Title** property (read/write/pub) string
- **TotalPasses** property (read/write)
- **UpdateStatus** method
- **Version** property
- **Visible** property (read/write/pub) 5.1.1
9.9 Category Position

- AdjustLine method
- AscentHeight property (read only) double
- CR method
- CursorXPos property (read only) longint
- CursorYPos property (read only) longint
- DescentHeight property (read only) double
- FontBaseline property (read/write) double
- FontBottom property (read/write) double
- FontHeight property (read/write) double
- FontTop property (read/write) double
- GotoFooter method
- GotoHeader method
- GotoXY method
- Home method
- LF method
- LineBottom property (read/write) double
- LineHeight property (read only) double
- LineHeightMethod property (read/write/pub)
- LineMiddle property (read/write) double
- LineNum property (read/write) integer
- LinesLeft method integer
- LinesPerInch property (read/write/pub)
- LineTop property (read/write) double

9.10 Category Preview

- Clear method
- GridHoriz property (read/write/pub) double
- GridPen property (read/write/pub)
- GridVert property (read/write/pub) double
- MarginMethod property (read/write/pub)
- MarginPercent property (read/write/pub) double
- Monochrome property (read/write/pub) Boolean
- NextPage method
- OnPageChange event (read/write/pub)
- OnPreviewSetup event (read/write/pub)
- OnPreviewShow event (read/write/pub)
- OnZoomChange event (read/write/pub)
- Paging property (read/write/pub) integer
- Pages property (read only) integer
- PrevPage method
- PrintPage method
- RedrawPage method
- RulerType property (read/write/pub)
- ScrollBox property (read/write/pub)
- ShadowDepth property (read/write/pub) integer
- TMargin method
- TMarginMethod type
- ZoomFactor property (read/write/pub) double
- ZoomIn method
- ZoomInc property (read/write/pub) integer
- ZoomOut method
- ZoomPageFactor property (read only) double
- ZoomPageWidthFactor property (read only) double
### 9.11 Category Printer

<table>
<thead>
<tr>
<th>Property/Method</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Bins</strong> property (read only)</td>
</tr>
<tr>
<td><strong>BottomWaste</strong> property (read only) double</td>
</tr>
<tr>
<td><strong>Canvas</strong> property (read only)</td>
</tr>
<tr>
<td><strong>Collate</strong> property (read/write) Boolean</td>
</tr>
<tr>
<td><strong>Copies</strong> property (read/write/pub) integer</td>
</tr>
<tr>
<td><strong>DeviceName</strong> property (read only) string</td>
</tr>
<tr>
<td><strong>DevMode</strong> property (read/write)</td>
</tr>
<tr>
<td><strong>DriverName</strong> property (read only) string</td>
</tr>
<tr>
<td><strong>Duplex</strong> property (read/write)</td>
</tr>
<tr>
<td><strong>Fonts</strong> property (read only)</td>
</tr>
<tr>
<td><strong>IgnoreFileSettings</strong> property (read/write) Boolean</td>
</tr>
<tr>
<td><strong>LeftWaste</strong> property (read only) double</td>
</tr>
<tr>
<td><strong>MaxCopies</strong> property (read/write/pub) longint</td>
</tr>
<tr>
<td><strong>NoNTColorFix</strong> property (read/write) Boolean</td>
</tr>
<tr>
<td><strong>NoPrinterPageHeight</strong> property (read/write) double</td>
</tr>
<tr>
<td><strong>NoPrinterPageWidth</strong> property (read/write) double</td>
</tr>
<tr>
<td><strong>NoPrinters</strong> method Boolean</td>
</tr>
<tr>
<td><strong>Orientation</strong> property (read/write/pub)</td>
</tr>
<tr>
<td><strong>OutputFileName</strong> property (read/write)</td>
</tr>
<tr>
<td><strong>OutputName</strong> string property (read/write) string</td>
</tr>
<tr>
<td><strong>PageHeight</strong> property (read only) double</td>
</tr>
<tr>
<td><strong>PageWidth</strong> property (read only) double</td>
</tr>
<tr>
<td><strong>Papers</strong> property (read only)</td>
</tr>
<tr>
<td><strong>Port</strong> property (read only) string</td>
</tr>
<tr>
<td><strong>PrintData</strong> method</td>
</tr>
<tr>
<td><strong>PrintDataStream</strong> method</td>
</tr>
<tr>
<td><strong>PrinterIndex</strong> property (read/write) integer</td>
</tr>
<tr>
<td><strong>Printers</strong> property (read only)</td>
</tr>
<tr>
<td><strong>RecoverPrinter</strong> method</td>
</tr>
<tr>
<td><strong>ReleasePrinter</strong> method</td>
</tr>
<tr>
<td><strong>ResetPrinter</strong> method</td>
</tr>
<tr>
<td><strong>RightWaste</strong> property (read only) double</td>
</tr>
<tr>
<td><strong>RPDev</strong> function</td>
</tr>
<tr>
<td><strong>RPName</strong> property (read only)</td>
</tr>
<tr>
<td><strong>SelectBin</strong> method</td>
</tr>
<tr>
<td><strong>SelectPaper</strong> method Boolean</td>
</tr>
<tr>
<td><strong>SelectPrinter</strong> method Boolean</td>
</tr>
<tr>
<td><strong>SetPaperSize</strong> method</td>
</tr>
<tr>
<td><strong>SetPrintDialog</strong> method</td>
</tr>
<tr>
<td><strong>SetPrintSetupDialog</strong> method</td>
</tr>
<tr>
<td><strong>SupportBin</strong> method Boolean</td>
</tr>
<tr>
<td><strong>SupportCollate</strong> method Boolean</td>
</tr>
<tr>
<td><strong>SupportDuplex</strong> method Boolean</td>
</tr>
<tr>
<td><strong>SupportOrientation</strong> method Boolean</td>
</tr>
<tr>
<td><strong>SupportPaper</strong> method Boolean</td>
</tr>
<tr>
<td><strong>TopWaste</strong> property (read only) double</td>
</tr>
<tr>
<td><strong>XDPI</strong> property (read only) integer</td>
</tr>
<tr>
<td><strong>YDPI</strong> property (read only) integer</td>
</tr>
</tbody>
</table>

### 9.12 Category Printing

<table>
<thead>
<tr>
<th>Property/Method</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Macro</strong> method string</td>
</tr>
<tr>
<td><strong>MacroData</strong> property (read/write)</td>
</tr>
<tr>
<td><strong>MakeLink</strong> method</td>
</tr>
<tr>
<td><strong>PIVar</strong> method string</td>
</tr>
<tr>
<td><strong>Print</strong> method</td>
</tr>
<tr>
<td><strong>PrintBlock</strong> method</td>
</tr>
<tr>
<td><strong>PrintCenter</strong> method</td>
</tr>
<tr>
<td><strong>PrintCharJustify</strong> method</td>
</tr>
<tr>
<td><strong>PrintFooter</strong> method</td>
</tr>
<tr>
<td><strong>PrintHeader</strong> method</td>
</tr>
<tr>
<td><strong>PrintLeft</strong> method</td>
</tr>
<tr>
<td><strong>PrintLn</strong> method</td>
</tr>
<tr>
<td><strong>PrintRight</strong> method</td>
</tr>
<tr>
<td><strong>PrintTab</strong> method</td>
</tr>
<tr>
<td><strong>PrintXY</strong> method</td>
</tr>
<tr>
<td><strong>ReportDateTime</strong> property (read/write)</td>
</tr>
<tr>
<td><strong>SetPIVar</strong> method</td>
</tr>
<tr>
<td><strong>TMacroID</strong> type</td>
</tr>
<tr>
<td><strong>TPrintJustify</strong> type</td>
</tr>
<tr>
<td><strong>TPrintJustifyVert</strong> type (archived BEX only)</td>
</tr>
<tr>
<td><strong>TruncateText</strong> property (read/write/pub) string</td>
</tr>
</tbody>
</table>
## 9.13 Category Rave -

<table>
<thead>
<tr>
<th>Property/Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Active</strong></td>
<td>property (read/write) Boolean</td>
</tr>
<tr>
<td><strong>ClearRaveBlob</strong></td>
<td>method</td>
</tr>
<tr>
<td><strong>Close</strong></td>
<td>method</td>
</tr>
<tr>
<td><strong>DataSet</strong></td>
<td>property (read/write)</td>
</tr>
<tr>
<td><strong>Design</strong></td>
<td>method</td>
</tr>
<tr>
<td><strong>DesignReport</strong></td>
<td>method</td>
</tr>
<tr>
<td><strong>DisableDataSource</strong></td>
<td>property (read/write/pub) 5.1.1</td>
</tr>
<tr>
<td><strong>DLLFile</strong></td>
<td>property (read/write/pub)</td>
</tr>
<tr>
<td><strong>Engine</strong></td>
<td>property</td>
</tr>
<tr>
<td><strong>Execute</strong></td>
<td>method</td>
</tr>
<tr>
<td><strong>ExecuteReport</strong></td>
<td>method</td>
</tr>
<tr>
<td><strong>FieldAliasList</strong></td>
<td>property</td>
</tr>
<tr>
<td><strong>GetParam</strong></td>
<td>method</td>
</tr>
<tr>
<td><strong>GetReportCategoryList</strong></td>
<td>method</td>
</tr>
<tr>
<td><strong>GetReportList</strong></td>
<td>method</td>
</tr>
<tr>
<td><strong>LoadDesigner</strong></td>
<td>property</td>
</tr>
<tr>
<td><strong>LoadFromFile</strong></td>
<td>method</td>
</tr>
<tr>
<td><strong>LoadFromStream</strong></td>
<td>method</td>
</tr>
<tr>
<td><strong>LoadRaveBlob</strong></td>
<td>method</td>
</tr>
<tr>
<td><strong>LocalFilter</strong></td>
<td>property Boolean</td>
</tr>
<tr>
<td><strong>OnAfterClose</strong></td>
<td>event (read/write/pub)</td>
</tr>
<tr>
<td><strong>OnAfterOpen</strong></td>
<td>event (read/write/pub)</td>
</tr>
<tr>
<td><strong>OnBeforeClose</strong></td>
<td>event (read/write/pub)</td>
</tr>
<tr>
<td><strong>OnBeforeOpen</strong></td>
<td>event (read/write/pub)</td>
</tr>
<tr>
<td><strong>OnCreate</strong></td>
<td>event</td>
</tr>
<tr>
<td><strong>OnDesignerSave</strong></td>
<td>event (read/write/pub)</td>
</tr>
<tr>
<td><strong>OnDesignerSaveAs</strong></td>
<td>event (read/write/pub)</td>
</tr>
<tr>
<td><strong>OnDesignerShow</strong></td>
<td>event</td>
</tr>
<tr>
<td><strong>OnDestroy</strong></td>
<td>event</td>
</tr>
<tr>
<td><strong>OnEOF</strong></td>
<td>event</td>
</tr>
<tr>
<td><strong>OnFirst</strong></td>
<td>event</td>
</tr>
<tr>
<td><strong>OnGetCols</strong></td>
<td>event</td>
</tr>
<tr>
<td><strong>OnGetRow</strong></td>
<td>event</td>
</tr>
<tr>
<td><strong>OnGetSorts</strong></td>
<td>event</td>
</tr>
<tr>
<td><strong>OnNext</strong></td>
<td>event</td>
</tr>
<tr>
<td><strong>OnOpen</strong></td>
<td>event</td>
</tr>
<tr>
<td><strong>OnRestore</strong></td>
<td>event</td>
</tr>
<tr>
<td><strong>OnSetFilter</strong></td>
<td>event</td>
</tr>
<tr>
<td><strong>OnSetSort</strong></td>
<td>event</td>
</tr>
<tr>
<td><strong>OnValidateRow</strong></td>
<td>event</td>
</tr>
<tr>
<td><strong>Open</strong></td>
<td>method</td>
</tr>
<tr>
<td><strong>ProjectFile</strong></td>
<td>property (read/write/pub)</td>
</tr>
<tr>
<td><strong>Query</strong></td>
<td>property (read/write/pub)</td>
</tr>
<tr>
<td><strong>RaveBlobDateTime</strong></td>
<td>property (read/write/pub)</td>
</tr>
<tr>
<td><strong>ReportDesc</strong></td>
<td>property</td>
</tr>
<tr>
<td><strong>ReportDescToMemo</strong></td>
<td>method</td>
</tr>
<tr>
<td><strong>ReportFullName</strong></td>
<td>property</td>
</tr>
<tr>
<td><strong>ReportName</strong></td>
<td>property</td>
</tr>
<tr>
<td><strong>RestoreDataSet</strong></td>
<td>property (read/write/pub) 5.1.1</td>
</tr>
<tr>
<td><strong>RuntimeVisibility</strong></td>
<td>property</td>
</tr>
<tr>
<td><strong>Save</strong></td>
<td>method</td>
</tr>
<tr>
<td><strong>SaveRaveBlob</strong></td>
<td>method</td>
</tr>
<tr>
<td><strong>SaveToFile</strong></td>
<td>method</td>
</tr>
<tr>
<td><strong>SaveToStream</strong></td>
<td>method</td>
</tr>
<tr>
<td><strong>SelectReport</strong></td>
<td>method</td>
</tr>
<tr>
<td><strong>SetParam</strong></td>
<td>method</td>
</tr>
<tr>
<td><strong>StoreRAV</strong></td>
<td>property (read only/special/pub)</td>
</tr>
<tr>
<td><strong>OnAfterOpen</strong></td>
<td>event (read/write/pub)</td>
</tr>
<tr>
<td><strong>UseSetRange</strong></td>
<td>property (read/write/pub)</td>
</tr>
<tr>
<td><strong>Table</strong></td>
<td>property (read/write/pub)</td>
</tr>
<tr>
<td><strong>WriteBCDData</strong></td>
<td>method</td>
</tr>
<tr>
<td><strong>WriteBlobData</strong></td>
<td>method</td>
</tr>
<tr>
<td><strong>WriteBoolData</strong></td>
<td>method</td>
</tr>
<tr>
<td><strong>WriteCurrData</strong></td>
<td>method</td>
</tr>
<tr>
<td><strong>WriteDateTime</strong></td>
<td>method</td>
</tr>
<tr>
<td><strong>WriteFloatData</strong></td>
<td>method</td>
</tr>
<tr>
<td><strong>WriteIntData</strong></td>
<td>method</td>
</tr>
<tr>
<td><strong>WriteNullData</strong></td>
<td>method</td>
</tr>
<tr>
<td><strong>WriteStrData</strong></td>
<td>method</td>
</tr>
</tbody>
</table>
9.14 Category Render -

- **Active** property (read/write) ALL
- **BufferDocument** property (read/write/pub) PDF 6.0.2
- **CacheDir** property (read/write/pub) HTML PDF
- **CPI** property (read/write/pub) TEXT 5.1.4
- **DisplayName** property (read/write/pub) ALL
- **DocInfo Author** property (read/write/pub) PDF 5.1.4
- **DocInfo Creator** property (read/write/pub) PDF 5.1.4
- **DocInfo DocInfo Author** property (read/write/pub) PDF 5.1.4
- **DocInfo DocInfo Creator** property (read/write/pub) PDF 5.1.4
- **DocInfo DocInfo KeyWords** property (read/write/pub) PDF
- **DocInfo DocInfo Producer** property (read/write/pub) PDF
- **DocInfo DocInfo Subject** property (read/write/pub) PDF
- **DocInfo DocInfo Title** property (read/write/pub) PDF
- **EmbedFonts** property (read/write/pub) PDF 5.1.4
- **FileExtension** property (read/write/pub) ALL 5.1.4
- **FontEncoding** property (read/write/pub) PDF
- **FormFeed** property (read/write/pub) TEXT 5.1.4
- **ImageEncoding** property (read/write/pub) RTF 5.1.4
- **ImageOutput** property (read/write/pub) RTF
- **ImageQuality** property (read/write/pub) PDF
- **LeftBorder** property (read/write/pub) TEXT 5.1.4
- **LPI** property (read/write/pub) TEXT 5.1.4
- **MetafileDPI** property (read/write/pub) PDF
- **OnCompress** event (read/write) PDF
- **OnDecodeImage** event (read/write/pub) HTML PDF RTF
- **ServerMode** property (read/write/pub) HTML PDF
- **TopBorder** property (read/write/pub) TEXT 5.1.4
- **UseCompression** property (read/write/pub) PDF

9.15 Category ReportSection

- **Bottom** property (read/write/pub) double
- **BottomMethod** property (read/write/pub)
- **Enabled** property (read/write/pub) Boolean
- **FontIndex** property (read/write/pub) integer
- **Height** property (read/write/pub) double
- **HeightMethod** property (read/write/pub)
- **Left** property (read/write/pub) double
- **LeftMethod** property (read/write/pub)
- **MinHeight** property (read/write/pub) double
- **Right** property (read/write/pub) double
- **RightMethod** property (read/write/pub)
- **SetupSection** method
- **TabIndex** property (read/write/pub) integer
- **Top** property (read/write/pub) double
- **TopMethod** property (read/write/pub)

9.16 Category ReportSystem

- **DefaultDest** property (read/write/pub)
- **OverrideDest** event (read/write)
- **OverrideSetup** event (read/write)
- **OverrideStatus** event (read/write)
- **ReportDest** property (read only)
- **SystemFiler** property (read/write/pub)
- **SystemOptions** property (read/write/pub)
- **SystemPreview** property (read/write/pub)
- **SystemPrinter** property (read/write/pub)
- **SystemSetups** property (read/write/pub)
- **TitlePreview** property (read/write/pub)
- **TitleSetup** property (read/write/pub)
- **TitleStatus** property (read/write/pub)
- **TRReportDest** type
- **TSReportDest** type
- **TSystemOption(s)** type
- **TSystemSetup(s)** type

9.17 Category RTF

- **NewPara** method
- **ParaJustify** property
- **SetRTF** method (archived)
- **SoftLine** method (archived)
Category Shell

<table>
<thead>
<tr>
<th>Property/Method</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DetailReport</strong> property (read/write/pub)</td>
</tr>
<tr>
<td><strong>IsNewPage</strong> property (read only) Boolean</td>
</tr>
<tr>
<td><strong>IsReprint</strong> property (read only) Boolean</td>
</tr>
<tr>
<td>OnBodyAfter event</td>
</tr>
<tr>
<td>OnBodyBefore event</td>
</tr>
<tr>
<td>OnBodyFooter event (read/write/pub)</td>
</tr>
<tr>
<td>OnBodyHeader event (read/write/pub)</td>
</tr>
<tr>
<td>OnDetailAfter event (read/write/pub)</td>
</tr>
<tr>
<td>OnDetailBefore event (read/write/pub)</td>
</tr>
<tr>
<td>OnEndOfSection event (read/write/pub)</td>
</tr>
<tr>
<td>OnGroupAfter event (read/write/pub)</td>
</tr>
<tr>
<td>OnGroupAfterLast event (read/write/pub)</td>
</tr>
<tr>
<td>OnGroupBefore event (read/write/pub)</td>
</tr>
<tr>
<td>OnGroupBeforeFirst event (read/write/pub)</td>
</tr>
<tr>
<td>OnGroupFooter event (read/write/pub)</td>
</tr>
<tr>
<td>OnGroupHeader event (read/write/pub)</td>
</tr>
<tr>
<td>OnPageAfter event (read/write/pub)</td>
</tr>
<tr>
<td>OnPageBefore event (read/write/pub)</td>
</tr>
<tr>
<td>OnPageFooter event (read/write/pub)</td>
</tr>
<tr>
<td>OnPageHeader event (read/write/pub)</td>
</tr>
<tr>
<td>OnReportAfter event (read/write/pub)</td>
</tr>
<tr>
<td>OnReportBefore event (read/write/pub)</td>
</tr>
<tr>
<td>OnReportFooter event (read/write/pub)</td>
</tr>
<tr>
<td>OnReportHeader event (read/write/pub)</td>
</tr>
<tr>
<td>OnRowAfter event (read/write/pub)</td>
</tr>
<tr>
<td>OnRowBefore event (read/write/pub)</td>
</tr>
<tr>
<td>OnRowPrint event (read/write/pub)</td>
</tr>
<tr>
<td>PrintBodyFooter method</td>
</tr>
<tr>
<td>PrintBodyHeader method</td>
</tr>
<tr>
<td>PrintDetail method</td>
</tr>
<tr>
<td>PrintGroupFooter method</td>
</tr>
<tr>
<td>PrintGroupHeader method</td>
</tr>
<tr>
<td>PrintPageFooter method</td>
</tr>
<tr>
<td>PrintPageHeader method</td>
</tr>
<tr>
<td>PrintReportFooter method</td>
</tr>
<tr>
<td>PrintReportHeader method</td>
</tr>
<tr>
<td>PrintRow method</td>
</tr>
<tr>
<td>Reprint property (read/write/pub) Boolean</td>
</tr>
<tr>
<td>SectionBodyFooter property (read/write/pub)</td>
</tr>
<tr>
<td>SectionBodyHeader property (read/write/pub)</td>
</tr>
<tr>
<td>SectionGroupFooter property (read/write/pub)</td>
</tr>
<tr>
<td>SectionGroupHeader property (read/write/pub)</td>
</tr>
<tr>
<td>SectionPageFooter property (read/write/pub)</td>
</tr>
<tr>
<td>SectionPageHeader property (read/write/pub)</td>
</tr>
<tr>
<td>SectionReportFooter property (read/write/pub)</td>
</tr>
<tr>
<td>SectionReportHeader property (read/write/pub)</td>
</tr>
<tr>
<td>SectionRow property (read/write/pub)</td>
</tr>
<tr>
<td>StartNewPage property (read/write/pub) Boolean</td>
</tr>
</tbody>
</table>
### 9.19  Category TablePrinter -

<table>
<thead>
<tr>
<th>Property/Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>AsFloat</code></td>
<td>property (read/write) double</td>
</tr>
<tr>
<td><code>AsInteger</code></td>
<td>property (read/write) double</td>
</tr>
<tr>
<td><code>BoxLines</code></td>
<td>property (read/write/pub)</td>
</tr>
<tr>
<td><code>Default</code></td>
<td>method</td>
</tr>
<tr>
<td><code>Description</code></td>
<td>property (read/write/pub)</td>
</tr>
<tr>
<td><code>DetailTablePrinter</code></td>
<td>property (read/write/pub)</td>
</tr>
<tr>
<td><code>DisplayFormat</code></td>
<td>property (read/write/pub) string</td>
</tr>
<tr>
<td><code>Font</code></td>
<td>property (read/write/pub)</td>
</tr>
<tr>
<td><code>GrandTotal</code></td>
<td>property (read/write) double</td>
</tr>
<tr>
<td><code>Heading</code></td>
<td>property (read/write/pub)</td>
</tr>
<tr>
<td><code>Justify</code></td>
<td>property (read/write/pub)</td>
</tr>
<tr>
<td><code>JustifyVert</code></td>
<td>property (read/write/pub)</td>
</tr>
<tr>
<td><code>Margin</code></td>
<td>property (read/write) double</td>
</tr>
<tr>
<td><code>Margin100</code></td>
<td>property (read/write) double</td>
</tr>
<tr>
<td><code>MasterTablePrinter</code></td>
<td>property (read/write)</td>
</tr>
<tr>
<td><code>MemoBuf</code></td>
<td>property (read/write)</td>
</tr>
<tr>
<td><code>OnAddTotal</code></td>
<td>event (read/write/pub)</td>
</tr>
<tr>
<td><code>OnAfter</code></td>
<td>event (read/write/pub)</td>
</tr>
<tr>
<td><code>OnFooterAfter</code></td>
<td>event (read/write/pub)</td>
</tr>
<tr>
<td><code>OnFooterBefore</code></td>
<td>event (read/write/pub)</td>
</tr>
<tr>
<td><code>OnFooterPrint</code></td>
<td>event (read/write/pub)</td>
</tr>
<tr>
<td><code>OnFooterSetup</code></td>
<td>event (read/write/pub)</td>
</tr>
<tr>
<td><code>OnFooterAfter</code></td>
<td>event (read/write/pub)</td>
</tr>
<tr>
<td><code>OnFooterBefore</code></td>
<td>event (read/write/pub)</td>
</tr>
<tr>
<td><code>OnFooterPrint</code></td>
<td>event (read/write/pub)</td>
</tr>
<tr>
<td><code>OnFooterSetup</code></td>
<td>event (read/write/pub)</td>
</tr>
<tr>
<td><code>OnGetNextRow</code></td>
<td>event (read/write/pub)</td>
</tr>
<tr>
<td><code>OnHeaderAfter</code></td>
<td>event (read/write/pub)</td>
</tr>
<tr>
<td><code>OnHeaderBefore</code></td>
<td>event (read/write/pub)</td>
</tr>
<tr>
<td><code>OnHeaderHeight</code></td>
<td>event (read/write/pub)</td>
</tr>
<tr>
<td><code>OnHeaderPrint</code></td>
<td>event (read/write/pub)</td>
</tr>
<tr>
<td><code>OnInitMaster (2)</code></td>
<td>event (read/write/pub)</td>
</tr>
<tr>
<td><code>OnInitTable</code></td>
<td>event (read/write/pub)</td>
</tr>
<tr>
<td><code>OnOverFlow</code></td>
<td>event (read/write/pub)</td>
</tr>
<tr>
<td><code>OnRowBefore</code></td>
<td>event (read/write/pub)</td>
</tr>
<tr>
<td><code>OnRowHeight</code></td>
<td>event (read/write/pub)</td>
</tr>
<tr>
<td><code>OnRowPrint (TTable)</code></td>
<td>event (read/write/pub)</td>
</tr>
<tr>
<td><code>OnRowSetup</code></td>
<td>event (read/write/pub)</td>
</tr>
<tr>
<td><code>OnValidateRow</code></td>
<td>event (read/write/pub)</td>
</tr>
<tr>
<td><code>OutputType</code></td>
<td>property (read/write)</td>
</tr>
<tr>
<td><code>OverflowMethod</code></td>
<td>property (read/write/pub)</td>
</tr>
<tr>
<td><code>OverflowReplace</code></td>
<td>property (read/write/pub)</td>
</tr>
<tr>
<td><code>PageTotal</code></td>
<td>property (read/write) double</td>
</tr>
<tr>
<td><code>Pen</code></td>
<td>property (read/write/pub)</td>
</tr>
<tr>
<td><code>PrintBox</code></td>
<td>method</td>
</tr>
<tr>
<td><code>PrintDefault</code></td>
<td>method</td>
</tr>
<tr>
<td><code>ReportPrinter</code></td>
<td>property (read/write)</td>
</tr>
<tr>
<td><code>ReprintHeader</code></td>
<td>property (read/write/pub) Boolean</td>
</tr>
<tr>
<td><code>Section</code></td>
<td>property (read/write/pub)</td>
</tr>
<tr>
<td><code>SectionType</code></td>
<td>property (read only)</td>
</tr>
<tr>
<td><code>ShadeColor</code></td>
<td>property (read/write/pub)</td>
</tr>
<tr>
<td><code>ShadePercent</code></td>
<td>property (read/write/pub) byte</td>
</tr>
<tr>
<td><code>SplitRow</code></td>
<td>property (read/write/pub) Boolean</td>
</tr>
<tr>
<td><code>SubTotal</code></td>
<td>property (read/write) double</td>
</tr>
<tr>
<td><code>TableColumn</code></td>
<td>property (read only)</td>
</tr>
<tr>
<td><code>TableColumns</code></td>
<td>property (read only) integer</td>
</tr>
<tr>
<td><code>TableItem</code></td>
<td>property (read/write)</td>
</tr>
<tr>
<td><code>TablePrinter</code></td>
<td>property (read/write)</td>
</tr>
<tr>
<td><code>TableSection</code></td>
<td>property (read only)</td>
</tr>
<tr>
<td><code>Tab</code></td>
<td>method</td>
</tr>
<tr>
<td><code>TabColor</code></td>
<td>property (read/write)</td>
</tr>
<tr>
<td><code>TabEnd</code></td>
<td>method double</td>
</tr>
<tr>
<td><code>TabJustify</code></td>
<td>property (read/write)</td>
</tr>
<tr>
<td><code>TabShade</code></td>
<td>property (read/write/pub) integer</td>
</tr>
<tr>
<td><code>TabStart</code></td>
<td>method double</td>
</tr>
<tr>
<td><code>TabWidth</code></td>
<td>method double</td>
</tr>
<tr>
<td><code>TBoxLines</code></td>
<td>type</td>
</tr>
<tr>
<td><code>TDuplex</code></td>
<td>type</td>
</tr>
<tr>
<td><code>Text</code></td>
<td>property (read/write/pub)</td>
</tr>
<tr>
<td><code>Totals</code></td>
<td>property (read/write/pub) Boolean</td>
</tr>
<tr>
<td><code>UseParentFont</code></td>
<td>property (read/write/pub) Boolean</td>
</tr>
<tr>
<td><code>UseParentPen</code></td>
<td>property (read/write/pub) Boolean</td>
</tr>
</tbody>
</table>

### 9.20  Category Tabs

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>BoxLineColor</code></td>
<td>property (read/write)</td>
</tr>
<tr>
<td><code>BoxLineType</code></td>
<td>constants</td>
</tr>
<tr>
<td><code>ClearAllTabs</code></td>
<td>method</td>
</tr>
<tr>
<td><code>ClearTabs</code></td>
<td>method</td>
</tr>
<tr>
<td><code>FinishTabBox</code></td>
<td>method</td>
</tr>
<tr>
<td><code>GetTab</code></td>
<td>method</td>
</tr>
<tr>
<td><code>ParaJustify</code></td>
<td>property (read/write)</td>
</tr>
<tr>
<td><code>PopTabs</code></td>
<td>method Boolean</td>
</tr>
<tr>
<td><code>PushTabs</code></td>
<td>method Boolean</td>
</tr>
<tr>
<td><code>ResetTabs</code></td>
<td>method</td>
</tr>
<tr>
<td><code>RestoreTabs</code></td>
<td>method Boolean</td>
</tr>
<tr>
<td><code>SaveTabs</code></td>
<td>method Boolean</td>
</tr>
<tr>
<td><code>SetTab</code></td>
<td>method</td>
</tr>
<tr>
<td><code>Tab</code></td>
<td>method</td>
</tr>
<tr>
<td><code>TabColor</code></td>
<td>property (read/write)</td>
</tr>
<tr>
<td><code>TabEnd</code></td>
<td>method double</td>
</tr>
<tr>
<td><code>TabJustify</code></td>
<td>property (read/write)</td>
</tr>
<tr>
<td><code>TabShade</code></td>
<td>property (read/write/pub) integer</td>
</tr>
<tr>
<td><code>TabStart</code></td>
<td>method double</td>
</tr>
<tr>
<td><code>TabWidth</code></td>
<td>method double</td>
</tr>
<tr>
<td><code>TTabJustify</code></td>
<td>type</td>
</tr>
</tbody>
</table>
9.21 Category Units

- **TPrintUnits** type
- **Units** property (read/write/pub)
- **UnitsFactor** property (read/write/pub) double
- **XD2I** method double
- **XD2U** method double
- **XI2D** method longint
- **XI2U** method double
- **XU2D** method longint
- **XU2I** method double
- **YD2I** method double
- **YD2U** method double
- **YI2D** method longint
- **YI2U** method double
- **YU2D** method longint
- **YU2I** method double
Format Codes
10 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:

10.1 Alphanumeric Items

The following is a list of the different alphanumeric format codes and what they accomplish for each output type.

Examples:

<table>
<thead>
<tr>
<th>Format String</th>
<th>123456.78</th>
<th>-123.0</th>
<th>0.5</th>
<th>0.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>#,##0.00</td>
<td>123,456.78</td>
<td>-123.00</td>
<td>0.50</td>
<td>0.00</td>
</tr>
<tr>
<td>.#</td>
<td>123456.8</td>
<td>-123</td>
<td>0.5</td>
<td>0</td>
</tr>
<tr>
<td>$,0.00</td>
<td>$123,456.78</td>
<td>$-123.00</td>
<td>$0.50</td>
<td>$0.00</td>
</tr>
<tr>
<td>0.00;(0.00);-‘</td>
<td>123456.78</td>
<td>(123.00)</td>
<td>0.50</td>
<td>-----</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Specifier</th>
<th>Represents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>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.</td>
</tr>
<tr>
<td>#</td>
<td>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.</td>
</tr>
<tr>
<td>.</td>
<td>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 a the decimal separator in the output string is determined by the Number Format of the International section in the Windows Control Panel.</td>
</tr>
<tr>
<td>,</td>
<td>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.</td>
</tr>
<tr>
<td>E+</td>
<td>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.</td>
</tr>
<tr>
<td>'xx'/'&quot;xx&quot;</td>
<td>Characters enclosed in single or double quotes are output as-is, and do not affect formatting.</td>
</tr>
<tr>
<td>;</td>
<td>Separates sections for positive, negative, and zero numbers in the format string.</td>
</tr>
</tbody>
</table>

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 place holders ('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 place holders 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.

10.2 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
<table>
<thead>
<tr>
<th>Specifier</th>
<th>Displays</th>
</tr>
</thead>
<tbody>
<tr>
<td>c</td>
<td>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.</td>
</tr>
<tr>
<td>d</td>
<td>Displays the day as a number without a leading zero (1-31).</td>
</tr>
<tr>
<td>dd</td>
<td>Displays the day as a number with a leading zero (01-31).</td>
</tr>
<tr>
<td>ddd</td>
<td>Displays the day as an abbreviation (Sun-Sat) using the strings given by the ShortDayNames global variable.</td>
</tr>
<tr>
<td>ddddd</td>
<td>Displays the day as a full name (Sunday-Saturday) using the strings given by the LongDayNames global variable.</td>
</tr>
<tr>
<td>m</td>
<td>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.</td>
</tr>
<tr>
<td>mm</td>
<td>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.</td>
</tr>
<tr>
<td>mmm</td>
<td>Displays the month as an abbreviation (Jan-Dec) using the strings given by the ShortMonthNames global variable.</td>
</tr>
<tr>
<td>mmmm</td>
<td>Displays the month as a full name (January-December) using the strings given by the LongMonthNames global variable.</td>
</tr>
<tr>
<td>yy</td>
<td>Displays the year as a two-digit number (00-99).</td>
</tr>
<tr>
<td>yyyy</td>
<td>Displays the year as a four-digit number (0000-9999).</td>
</tr>
<tr>
<td>h</td>
<td>Displays the hour without a leading zero (0-23).</td>
</tr>
<tr>
<td>hh</td>
<td>Displays the hour with a leading zero (00-23).</td>
</tr>
<tr>
<td>n</td>
<td>Displays the minute without a leading zero (0-59).</td>
</tr>
<tr>
<td>nn</td>
<td>Displays the minute with a leading zero (00-59).</td>
</tr>
<tr>
<td>s</td>
<td>Displays the second without a leading zero (0-59).</td>
</tr>
<tr>
<td>ss</td>
<td>Displays the second with a leading zero (00-59).</td>
</tr>
<tr>
<td>t</td>
<td>Displays time using format given by the ShortTimeFormat global variable.</td>
</tr>
<tr>
<td>tt</td>
<td>Displays time using format given by the LongTimeFormat global variable.</td>
</tr>
<tr>
<td>am/pm</td>
<td>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.</td>
</tr>
<tr>
<td>a/p</td>
<td>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.</td>
</tr>
<tr>
<td>ampm</td>
<td>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.</td>
</tr>
<tr>
<td>/</td>
<td>Displays date separator character given by DateSeparator global variable.</td>
</tr>
<tr>
<td>:</td>
<td>Displays time separator character given by TimeSeparator global variable.</td>
</tr>
<tr>
<td>'xx'/'&quot;xx&quot;</td>
<td>Characters enclosed in single or double quotes are displayed as-is, and do not affect formatting.</td>
</tr>
</tbody>
</table>
Index

- A -

Archived 224
AsFloat 245
AsInteger 246
Border 246
Bottom 246
BottomMethod 247
BoxLines 247
Col 248
Components 225
Default 241
Description 248
DetailReport 249
DetailTablePrinter 249
DisplayFormat 250
DrawExtents 250
DrawPen 251
DrawPreviewOnly 251
Enabled 251
Events 230
Font 252
FontIndex 252
Height 253
HeightMethod 253
IsNewPage 254
IsReprint 254
LabelBrand 255
LabelHeight 255
LabelShape 256
LabelWidth 256
Left 256
LeftMethod 257
LPI 164
Methods 241
MinHeight 257
NumAcross 258
NumDown 258
OnAddTotal 230
OnBodyAfter 230
OnBodyBefore 230
OnBodyFooter 230
OnBodyHeader 231
OnDetailAfter 231
OnDetailBefore 231
OnEndOfSection 232
OnGroupAfter 232
OnGroupAfterLast 232
OnGroupBefore 233
OnGroupBeforeFirst 233
OnGroupFooter 233
OnGroupHeader 234
OnLabelAfter 234
OnLabelBefore 234
OnLabelPrint 235
OnOverFlow 235
OnPageAfter 235
OnPageBefore 236
OnPageFooter 236
OnPageHeader 236
OnReportAfter 237
OnReportBefore 237
OnReportFooter 237
OnReportHeader 237
OnRowAfter 238
OnRowBefore 238
OnRowPrint 238
PrintBodyFooter 241
PrintBodyHeader 241
PrintByRow 259
PrintDetail 242
PrintGroupFooter 242
PrintGroupHeader 242
PrintPageFooter 243
PrintPageHeader 243
PrintReportFooter 243
PrintReportHeader 244
PrintRow 244
properties 245
Reprint 259
Right 259
RightMethod 260
Row 260
RpDev 274
SectionBodyFooter 261
SectionBodyHeader 261
SectionGroupFooter 261
SectionGroupHeader 262
SectionPageFooter 262
SectionPageHeader 263
SectionReportFooter 263
SectionReportHeader 264
SectionRow 264
SetupSection 245
SkipNum 265
SpacingHeight 265
SpacingLeft 265
SpacingTop 266
SpacingWidth 266
StartCol 267
StartNewPage 267
StartRow 268
Rave Reports Developer Reference

Archived 224
TabIndex 268
TBoxLines 273
TDbTablePrinter 225
TDetailShell 225
TLabelShell 226
TMasterShell 226
Top 269
TopMethod 269
TPrintJustifyVert 273
TReportShell 227
TTablePrinter 227
Types 273
Width 269

- B -

BarCode 276
BEX only
Archived 224
HTML 283
PDF 283
RTF 283
TDbTablePrinter 225
TDetailShell 225
TLabelShell 226
TMasterShell 226
TReportShell 227
TTablePrinter 227

- C -

Categories 276
Category
BarCode 276
Column 276
Control 277
Font 277
Graphics 278
Label 278
Memo 279
Misc 279
Position 280
Preview 280
Printer 281
Printing 281
Rave 282
Render 283
ReportSection 283
ReportSystem 283
RTF 283
Shell 284
TablePrinter 285

Tabs 285
Units 286
Classes 4
TBaseReport 4
TCanvasReport 5
TDbMemoBuf 6
TMemoBuf 6
TRpBarsBase 7
TRpBaseComponent 7
TRpComponent 7
TRpRender 8
TRpRenderCanvas 8
TRpRenderStream 9

Column 276
Components 12
TRvCustomConnection 12
TRvDataSetConnection 13
TRvNDRWriter 14
TRvProject 15
TRvQueryConnection 16
TRvRenderBitmap 17
TRvRenderHTML 17
TRvRenderJPEG 18
TRvRenderMetafile 18
TRvRenderPDF 19
TRvRenderPreview 20
TRvRenderPrinter 21
TRvRenderRTF 23
TRvRenderText 23
TRvSystem 24
TRvTableConnection 25

Constants
amAppearance 126, 216
amPositioning 126, 216
bkOpaque 202, 216
bkTransparent 202, 216
BOXLINENONE 134
BOXLINENOBOTTOM 134
BOXLINENOLEFT 134
BOXLINELLEFT 134
BOXLINENOBOTTOM 134
BOXLINENOLEFT 134
BOXLINENONE 134
BOXLINENORIGHT 134
BOXLINENOTOP 134
BOXLINERIGHT 134
BOXLINETOP 134
BOXLINETOPBOTTOM 134
cpCodeA 136
cpCodeB 136
cpCodeC 136
dupHorizontal 145
dupSimplex 145
Constants

dupVertical 145
faBaseline 148, 216
faBottom 148, 216
faTop 148, 216
fmInsSide 154
fmOutside 154
fmSplit 154
FormHeight 198
FormState 198
FormWidth 198
IhmFont 161, 217
IhmLinesPerInch 161, 217
IhmUser 217
midCurrDateInter 217
midCurrDateLong 217
midCurrDateShort 217
midCurrDateUS 217
midCurrentPage 217
midCurrTime24 217
midCurrTimeAMPM 217
midCurrTimeLong 217
midCurrTimeShort 217
midDriverName 217
midFirstPage 217
midLastPage 217
midPortName 217
midPrintName 217
midTotalPages 217
midUser0 217
midUser20 217
mmFixed 217
mmScaled 217
NewJustify 107
NewLines 107
NewMargin 107
NewPos 107
NewShade 107
pjBlock 158, 218
pjCenter 128, 158, 202, 218
pjLeft 128, 158, 202, 218
pjRight 128, 158, 202, 218
poDefault 172, 218
poLandscape 172, 218
poPortrait 172, 218
rdFile 219
rdPreview 219
rdPrinter 219
Rot0 129
Rot180 129
Rot270 129
Rot90 129
rtBothCm 186
rtBothIn 186
rtDeveloper 187
rtEndDate 187
rtHorizCm 186
rtHorizIn 186
rtNone 186, 187
rtVertCm 186
rtVertIn 186
smFile 195, 219
smMemory 195, 219
smTempFile 195, 219
smUser 195, 219
soAllowPrintFromPreview 197
soNoGenerate 197
soPreviewModal 197
soShowStatus 197
soUseFiler 197
soWaitForOK 197
ssAllowCollate 220
ssAllowCopies 220
ssAllowDestFile 220
ssAllowDestPreview 220
ssAllowDestPrinter 220
ssAllowDuplex 220
ssAllowPrinterSetup 220
ssAllowSetup 220
tjBlock 200, 221
tjCenter 200, 221
tjLeft 200, 221
tjNone 200, 221
tjRight 200, 221
unCM 206, 218
unInch 206, 218
unMM 206, 218
unPoint 206, 218
unUser 206, 218
Control 277

- E -

Events 28
OnAfterClose 28
OnAfterOpen 28
OnAfterPrint 28
OnBeforeClose 29
OnBeforeOpen 29
OnBeforePrint 29
OnCreate 30
OnDecodImage 30
OnDesignerSave 31
OnDesignerSaveAs 31
OnDesignerShow 31
OnDestroy 32
Events
OnEOF 32
OnFirst 32
OnGetCols 33
OnGetRow 33
OnGetSorts 33
OnNewColumn 33
OnNewPage 34
OnNext 35
OnOpen 35
OnPageChange 35
OnPreviewSetup 36
OnPreviewShow 37
OnPrint 37
OnPrintFooter 37
OnPrintHeader 38
OnPrintPage 39
OnRestore 39
OnSetFilter 40
OnSetSort 40
OnValidateRow 40
OnZoomChange 40
OverridePreview 41
OverrideSetup 41
OverrideStatus 42

- F -
Font 277
Format
  Alphanumeric 288
  Date 289
  Time 289

- G -
Graphics 278
  GridPen 155
  GridVert 156

- H -
Height 156

- I -
IgnoreFileSettings 157
  ImageQuality 157
  Introduction 2
  Italic 158

- J -
  Justify 158

- L -
  Label 278
  LastPage 158
  Left 159
  LeftWaste 159
  LineBottom 160
  LineHeight 160
  LineHeightMethod 161
  LineMiddle 161
  LineNum 162
  LinesPerInch 162
  LineTop 163
  LoadDesigner 163
  LocalFilter 163

- M -
  MacroData 164
  MarginBottom 165
  MarginLeft 165
  MarginMethod 166
  MarginPercent 166
  MarginRight 167
  MarginTop 167
  MaxCopies 167
  MaxSize 168
  Memo 168, 279
  MetafileDPI 169

Methods 44
  Abort 44
  AbortPage 44
  AdjustLine 44
  AllowAll 45
  AllowPreviewOnly 45
  AllowPrinterOnly 46
  Append 46
  AppendMemoBuf 46
  Arc 47
  AssignFont 47
  BrushCopy 47
  CalcGraphicHeight 48
  CalcGraphicWidth 48
  Chord 49
  Clear 49
  ClearAllTabs 49
  ClearColumns 50
  ClearRaveBlob 50
<table>
<thead>
<tr>
<th>Method</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ClearTabs</td>
<td>51</td>
</tr>
<tr>
<td>Close</td>
<td>51</td>
</tr>
<tr>
<td>ConstraintHeightLeft</td>
<td>51</td>
</tr>
<tr>
<td>CopyRect</td>
<td>52</td>
</tr>
<tr>
<td>CR</td>
<td>52</td>
</tr>
<tr>
<td>Create</td>
<td>52, 53</td>
</tr>
<tr>
<td>CreateBrush</td>
<td>54</td>
</tr>
<tr>
<td>CreateFont</td>
<td>54</td>
</tr>
<tr>
<td>CreatePen</td>
<td>55</td>
</tr>
<tr>
<td>CreatePoint</td>
<td>55</td>
</tr>
<tr>
<td>CreateRect</td>
<td>56</td>
</tr>
<tr>
<td>Delete</td>
<td>56</td>
</tr>
<tr>
<td>Design</td>
<td>56</td>
</tr>
<tr>
<td>DesignReport</td>
<td>57</td>
</tr>
<tr>
<td>Destroy</td>
<td>57</td>
</tr>
<tr>
<td>Draw</td>
<td>57</td>
</tr>
<tr>
<td>DrawFocusRect</td>
<td>58</td>
</tr>
<tr>
<td>Ellipse</td>
<td>58</td>
</tr>
<tr>
<td>Empty</td>
<td>59</td>
</tr>
<tr>
<td>EndLink</td>
<td>71</td>
</tr>
<tr>
<td>Execute</td>
<td>59, 60</td>
</tr>
<tr>
<td>ExecuteCustom</td>
<td>60</td>
</tr>
<tr>
<td>ExecuteReport</td>
<td>60</td>
</tr>
<tr>
<td>FillRect</td>
<td>61</td>
</tr>
<tr>
<td>Finish</td>
<td>61</td>
</tr>
<tr>
<td>FinishTabBox</td>
<td>61</td>
</tr>
<tr>
<td>FloodFill</td>
<td>62</td>
</tr>
<tr>
<td>FrameRect</td>
<td>62</td>
</tr>
<tr>
<td>FreeSaved</td>
<td>63</td>
</tr>
<tr>
<td>GetMemoLine</td>
<td>63</td>
</tr>
<tr>
<td>GetNextLine</td>
<td>63</td>
</tr>
<tr>
<td>GetParam</td>
<td>64</td>
</tr>
<tr>
<td>GetReportCategoryList</td>
<td>64</td>
</tr>
<tr>
<td>GetReportList</td>
<td>65</td>
</tr>
<tr>
<td>GetTab</td>
<td>65</td>
</tr>
<tr>
<td>GotoFooter</td>
<td>65</td>
</tr>
<tr>
<td>GotoHeader</td>
<td>65</td>
</tr>
<tr>
<td>GotoXY</td>
<td>66</td>
</tr>
<tr>
<td>GraphicFieldToBitmap</td>
<td>66</td>
</tr>
<tr>
<td>Home</td>
<td>67</td>
</tr>
<tr>
<td>Insert</td>
<td>67</td>
</tr>
<tr>
<td>InsertMemoBuf</td>
<td>67</td>
</tr>
<tr>
<td>IsValidChar</td>
<td>68</td>
</tr>
<tr>
<td>LF</td>
<td>68</td>
</tr>
<tr>
<td>LinesLeft</td>
<td>68</td>
</tr>
<tr>
<td>LineTo</td>
<td>69</td>
</tr>
<tr>
<td>LoadFromFile</td>
<td>69, 70</td>
</tr>
<tr>
<td>LoadFromStream</td>
<td>70</td>
</tr>
<tr>
<td>LoadRaveBlob</td>
<td>71</td>
</tr>
<tr>
<td>Macro</td>
<td>71</td>
</tr>
<tr>
<td>MakeLink</td>
<td>71</td>
</tr>
<tr>
<td>MemoHeightLeft</td>
<td>72</td>
</tr>
<tr>
<td>MemoLines</td>
<td>72</td>
</tr>
<tr>
<td>MemoLinesLeft</td>
<td>73</td>
</tr>
<tr>
<td>MoveTo</td>
<td>73</td>
</tr>
<tr>
<td>NewColumn</td>
<td>73</td>
</tr>
<tr>
<td>NewLine</td>
<td>74</td>
</tr>
<tr>
<td>NewPage</td>
<td>74</td>
</tr>
<tr>
<td>NewPara</td>
<td>74</td>
</tr>
<tr>
<td>NextPage</td>
<td>75</td>
</tr>
<tr>
<td>NoPrinters</td>
<td>75</td>
</tr>
<tr>
<td>Open</td>
<td>76</td>
</tr>
<tr>
<td>Pie</td>
<td>76</td>
</tr>
<tr>
<td>Polygon</td>
<td>76</td>
</tr>
<tr>
<td>Polyline</td>
<td>77</td>
</tr>
<tr>
<td>PopFont</td>
<td>77</td>
</tr>
<tr>
<td>PopPos</td>
<td>78</td>
</tr>
<tr>
<td>PopTabs</td>
<td>78</td>
</tr>
<tr>
<td>PrevPage</td>
<td>79</td>
</tr>
<tr>
<td>Print</td>
<td>79</td>
</tr>
<tr>
<td>PrintBitmap</td>
<td>80</td>
</tr>
<tr>
<td>PrintBitmapRect</td>
<td>80</td>
</tr>
<tr>
<td>PrintBlock</td>
<td>80</td>
</tr>
<tr>
<td>PrintCenter</td>
<td>81</td>
</tr>
<tr>
<td>PrintCharJustify</td>
<td>81</td>
</tr>
<tr>
<td>PrintData</td>
<td>81</td>
</tr>
<tr>
<td>PrintDataStream</td>
<td>82</td>
</tr>
<tr>
<td>PrintFimA</td>
<td>82</td>
</tr>
<tr>
<td>PrintFimB</td>
<td>83</td>
</tr>
<tr>
<td>PrintFimC</td>
<td>83</td>
</tr>
<tr>
<td>PrintFooter</td>
<td>83</td>
</tr>
<tr>
<td>PrintHeader</td>
<td>84</td>
</tr>
<tr>
<td>PrintHeight</td>
<td>84</td>
</tr>
<tr>
<td>PrintImageRect</td>
<td>84</td>
</tr>
<tr>
<td>PrintJustify</td>
<td>85</td>
</tr>
<tr>
<td>PrintLeft</td>
<td>86</td>
</tr>
<tr>
<td>PrintLines</td>
<td>86</td>
</tr>
<tr>
<td>PrintLn</td>
<td>86</td>
</tr>
<tr>
<td>PrintMemo</td>
<td>87</td>
</tr>
<tr>
<td>PrintPage</td>
<td>87</td>
</tr>
<tr>
<td>PrintRight</td>
<td>88</td>
</tr>
<tr>
<td>PrintTab</td>
<td>88</td>
</tr>
<tr>
<td>PrintXY</td>
<td>88, 89</td>
</tr>
<tr>
<td>PushFont</td>
<td>89</td>
</tr>
<tr>
<td>PushPos</td>
<td>89</td>
</tr>
<tr>
<td>PushTabs</td>
<td>90</td>
</tr>
<tr>
<td>RecoverPrinter</td>
<td>90</td>
</tr>
<tr>
<td>Rectangle</td>
<td>90</td>
</tr>
<tr>
<td>RedrawPage</td>
<td>91</td>
</tr>
<tr>
<td>RegisterGraphic</td>
<td>91</td>
</tr>
<tr>
<td>ReleasePrinter</td>
<td>92</td>
</tr>
<tr>
<td>ReplaceAll</td>
<td>92</td>
</tr>
<tr>
<td>ReportDescToMemo</td>
<td>93</td>
</tr>
<tr>
<td>Reset</td>
<td>93</td>
</tr>
<tr>
<td>ResetLineHeight</td>
<td>94</td>
</tr>
</tbody>
</table>
Methods 44
ResetPrinter 94
ResetSection 94
ResetTabs 95
RestoreBuffer 95
RestoreFont 95
RestorePos 96
RestoreState 96
RestoreTabs 96
ReuseGraphic 97
RoundRect 97
RTFLoadFromFile 97
RTFLoadFromStream 98
Save 98
SaveBuffer 98
SaveFont 98
SavePos 99
SaveRaveBlob 99
SaveState 100
SaveTabs 49, 100
SaveToFile 100
SaveToStream 100, 101
SearchFirst 101
SearchNext 102
SelectBin 102
SelectPaper 102
SelectPrinter 103
SelectReport 103
SetBrush 103
SetColumns 104
SetColumnWidth 104
SetData 105
SetFont 105
SetPageSize 105
SetParam 106
SetPen 106
SetPIVar 107
SetRTF 107
SetTab 107
SetTopOfPage 108
ShadeToColor 109
ShowPrintDialog 109
ShowPrinterSetupDialog 109
SoftLine 110
Start 110
StartLink 71
StretchDraw 110
SupportBin 111
SupportCollate 111
SupportDuplex 111
SupportOrientation 111
SupportPaper 112
Tab 112
TabEnd 112
TabStart 113
TabWidth 113
TextRect 114
TextWidth 114
UnregisterGraphic 115
UpdateStatus 115
WriteBCDData 115
WriteBlobData 116
WriteBoolData 116
WriteCurrData 117
WriteDateTime 117
WriteFloatData 117
WriteIntData 118
WriteNullData 118
WriteStrData 119
XD2I 119
XD2U 119
XI2D 120
XI2U 120
XU2D 120
XU2I 121
YD2I 121
YD2U 121
YI2D 122
YI2U 122
YU2D 122
YU2I 123
ZoomIn 123
ZoomOut 124
Misc 279
Monochrome 169

- N -
NoBufferLine 169
NoCRLF 170
NoNewLine 170
NoNTColorFix 171
NoPrinterPageHeight 171
NoPrinterPageWidth 171

- O -
OnCompress 172
Orientation 172
OriginX 173
OriginY 173
OutputFileName 173
OutputInvalid 174
OutputName 174
Index

- P -

PageHeight 175
PageInc 175
PageInvalid 175
Pages 176
PageWidth 176
Papers 176
ParaJustify 177
PiVar 177
Port 178
Pos 178
Position 179, 280
Preview 280
PrintChecksum 179
PrintEnd 179
Printer 281
PrinterIndex 180
Printers 180
Printing 181, 281
PrintReadable 181
PrintStart 181
PrintTop 182
ProjectFile 182
Properties 126
    Aborted 126
    AccuracyMethod 126
    Active 127
    AscentHeight 127
    BarBottom 128
    BarCodeJustify 128
    BarCodeRotation 129
    BarHeight 129
    BarsBase 131
    BarTop 129
    BarWidth 130
    BaseReport 130, 131
    Bins 131
    BKColor 132
    Bold 132
    Bottom 133
    BottomWaste 133
    BoxLineColor 133
    BoxLineXxx 134
    Buffer 134
    BufferInc 134
    CacheDir 135
    Canvas 135
    Center 135
    CheckSum 136
    CodePage 136
    Collate 137

- Q -

Query 182
- R -
Rave 282
RaveBlobDateTime 183
ReadableHeight 183
Reference 276
Render 283
ReportDateTime 183
ReportDesc 184
ReportDest 184
ReportFullName 184
ReportName 185
ReportSection 283
ReportSystem 283
RichEdit 185
Right 185
RightWaste 186
RTF 283
RTFField 186
RTFText 186
RulerType 186
RuntimeVisibility 187

- S -
ScaleX 187
ScaleY 188
ScrollBox 188
SectionBottom 189
SectionLeft 189
SectionRight 190
SectionTop 190
Selection 191
ServerMode 191
ShadowDepth 192
Shell 284
Size 192
StatusFormat 192
StatusLabel 193
StatusText 193
StoreRAV 194
Stream 194
StreamMode 195
Strikeout 196
Subscript 196
Superscript 197
Support 2
SystemFiler 197
SystemOptions 197
SystemPreview 198
SystemPrinter 199
SystemSetups 199

- T -
TabColor 199
TabJustify 200
Table 200
TablePrinter 285
Tabs 285
TabShade 200
Text 201
TextBKMode 202
TextJustify 202
Title 203
TitlePreview 203
TitleSetup 203
TitleStatus 204
Top 204
TopWaste 204
TotalPasses 205
TransparentBitmaps 205
TruncateText 205
Types 216
TAccuracyMethod 216
TBKMode 216
TFontAlign 216
TLineHeightMethod 217
TMacroID 217
TMarginMethod 217
TOrientation 218
TPrintJustify 218
TPrintUnits 218
TReportDest 219
TStreamMode 219
TSystemOption 219
TSystemOptions 220
TSystemSetup 220
TSystemSetups 220
TTabJustify 221

- U -
Underline 206
Units 206, 286
UnitsFactor 207
UseChecksum 207
UseCompression 208
UseSetRange 208

- V -
Version 208
- W -
  WideFactor  209
  Width  209

- X -
  XDPI  209
  XPos  210

- Y -
  YDPI  210
  YPos  210

- Z -
  ZoomFactor  211
  ZoomInc  212
  ZoomPageFactor  212
  ZoomPageWidthFactor  212