# **Rave Reports**

**Borland Edition 5.0** 

## **Developers Guide**

**Tutorial and Reference** 

This manual and all material accompanying it is Copyright (C) 1995-2002, Nevrona Designs, All Rights Reserved (rev BE50a)

## **Table of Contents**

Getting Started	3
Single User License Agreement	
Limited Warranty	
Technical Support	4
Chapter 1 - Introduction to Rave	
Rave Toolbar	
Output Components	
Rave Classes	
Chapter 2 - Quick Start with Rave	7
Welcome to RAVE!	
Structure of Rave	7
Chapter 3 - TRvSystem Component	9
Description	
Properties	
Events	
Chapter 4 - Rave Data Connection	
Data Bridge	
Naming the Data Connection	
Controlling the Visibility of the connection	
Custom Data Connections	
Chapter 5 - Customizing Data Connections	
Using Events to Customize your Data Connection	
OnGetCols Event	
OnOpen event	
OnFirst Event	
OnNext event	
OnEOF event	
OnGetRow event	15
OnValidateRow event	16
OnRestore event	16
Chapter 6 - TRvNDRWriter component	17
Description	
Properties and Events	
Chapter 7 - Rendering components	
TRvRenderPreview Description	
TRvRenderPreview Properties	
TRvRenderPreview Events	
TRvRenderPrinter Description	
TRvRenderPrinter Properties and Events	
TRvRenderPDF Description	
TRvRenderHTML Description	
TRvRenderRTF Description	
TRvRenderText Description	20

Chapter 8 - TRvProject Component	21
	21
Engine Property	21
Using TRvProject	21
Appendix A - Formatting	23
•••	23
Date / Time items	24
Appendix B - Keyboard / Mouse Shortcuts	27
Page Designer or Project Tree	27
Page Designer Only	27
Project Tree Only	27
Appendix C - Property Descriptions	29

## Getting Started Technical Information

## **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.

## **Limited Warranty**

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

#### 2. Disclaimer of Warranties - Nevrona Designs disclaims all other warranties, either

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

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

4. **Governing Law** - This Agreement will be construed and governed in accordance with laws of the State of Arizona.

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

## **Technical Support**

Technical support, product updates, addons and other information relating to Rave Reports can be found at the Nevrona Designs web site. Please visit one of the following web pages for more information:

Technical Support - http://www.nevrona.com/support

Addons, Tips and Tricks and other information - http://www.nevrona.com/rave

Updates - http://www.nevrona.com/rave/download.html

# **Tutorial 1 Introduction to Rave**

## **Rave Toolbar**

There are 2 different types of objects in Rave, Output Components and Report Classes. The Output Components are responsible for sending the report to a variety of destinations. The Report Classes are non-component classes that manage other reporting tasks.



Outp	ut Components	
RV Sveter	TRvSystem	Incorporates a standard printer and preview system into one easy to use component.
₽¥ ●	TRvNDRWriter	Generates an NDR stream or file (a proprietary format) from report execution.
RV Etil	TRvRenderPreview	Displays a preview dialog for an NDR stream or file.
R	TRvRenderPrinter	Sends an NDR stream or file to the printer.
PDF	TRvRenderPDF	Converts an NDR stream or file to PDF format.
HTML	TRvRenderHTML	Converts an NDR stream or file to HTML format.
RTF	TRvRenderRTF	Converts an NDR stream or file to RTF format.
TEXT	TRvRenderText	Converts an NDR stream or file to Text format.

#### 4 -- 4 - ----

\_

## **Rave Classes**

RV Rave	TRvProject	Provides a connection to a report project that was created with the Rave visual designer. Use this component to get a listing of all available reports or to execute a specific report.
RV C	TRvCustomConnection	Connects custom data (generated through events) to DirectDataViews created with the Rave visual designer.
RVP C	TRvDataSetConnection	Connects TDataSet data (e.g. TClientDataSet, 3rd party TDataSet descendents) to DirectDataViews created with the Rave visual designer.
RV#	TRvTableConnection	Connects TTable components to DirectDataViews created with the Rave visual designer.
RV <sup>th</sup> Q	TRvQueryConnection	Connects TQuery components to DirectDataViews created with the Rave visual designer.

# Tutorial 2 Quick Start with Rave

## Welcome to RAVE!

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

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

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

## Structure of Rave

1) Data connection components - If you noticed in the above example, Rave uses data from your application. The standard data connection components, TRvCustomConnection, TrvDataSetConnection and TRvTableConnection provide a bridge between the data in your application and the Rave visual components. The 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. More detailed usage of data connection components is explained in much more detail in

a later tutorial.

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

# Tutorial 3 TRvSystem Component

### Description

The TRvSystem component is a very powerful component that integrates the functionality of the 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**

SYSTEM

*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.

Object Inspecto	or 🗵
RvSystem1	TRvSystem 💽
Properties Ever	nts
DefaultDest	rdPreview
Name	RvSystem1
⊞ SystemFiler	(TSystemFiler)
E SystemOptions	[soShowStatus,sov
E SystemPreview	(TSystemPreview)
E SystemPrinter	(TSystemPrinter)
E SystemSetups	[ssAllowSetup,ssA
Tag	0
TitlePreview	Report Preview
TitleSetup	Output Options
TitleStatus	Report Status
Version	4.10 (VCL6)
All shown	

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.

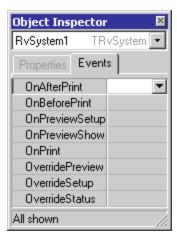
*SystemPreview* displays all of the preview type options found in TRvRenderPreview. *SystemPrinter* displays all of the printer type options found in TRvNDRWriter.

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.

### 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 \RAVE4\SOURCE will also show how to interface with TRvSystem and can be used as starting points for customized versions of the different forms.



# Tutorial 4 Rave Data Connection

## Data Bridge

As was mentioned previously, data connections provide a bridge between the data in your Delphi or C++Builder application and a Rave report. The first thing you have to do is choose the type of Rave data connection that you will need. This is determined by the type of database components that you are using. Use the table below to help you decide the best data connection component to use:

Data Connection Component	Best used with	How to connect
TRvCustomConnection	Memory arrays, non-database files or non-TDataSet database components	Define events such as OnFirst, OnNext, OnEOF, OnGetCols and OnGetRow.
TRvDataSetConnection	3rd Party TDataSet descendent database components	Initialize the DataSet property to the TDataSet component
TRvTableConnection	TTable, replacement TTable or its descendents	Initialize the Table property to the TTable component

## Naming the Data Connection

The Name property of the data connection component is used as the name of the data connection itself. It is important to use unique names for your data connection components since there can be no duplication across your application. It is also good pratice to include a unique application identifier in your data connection names since data connections are visible from other applications. For example, if your application is called Wizbang Object Wizard you could prepend the letters WOW in front of each data connection name to help insure your data connection names are unique for your application.

## Controlling the Visibility of the connection

The Visible property of the data connection component is used to define whether the data connection is visible to other applications or the end-user version of the Rave visual designer. The default value of False means that the data connection will only be visible to a Rave report being printed from within the same application as the data connection component or from the programmer version of the Rave visual designer. A value of True for the Visible property means that the data connection will be visible to any Rave report being run from within any application or from the end-user or programmer version of the Rave visual designer. While data connections of this type are available to the end-user designer, they will not be displayed when then end-user attempts to create a new data view unless they belong to the application that started the Rave visual designer. If you want to make data connections from other applications available to your end-users it is best to create a report project with the programmer version of the Rave visual designer with the data views already defined for these external data connections.

## **Custom Data Connections**

In the future, there will be other custom data connection components available written specifically for other database components. See the file CUSTOMDB.TXT in the source directory for more information on availability and installation.

Developers Guide

## Tutorial 5 Customizing Data Connections

## Using Events to Customize your Data Connection



Through the events in the data connection components, you can customize how the data is sent to your Rave reports. For non-database data using the TRvCustomConnection component, you will need to provide all access to your data through these events. For

database data connection components such as TRvDataSetConnection, you will normally only want to override the OnValidateRow event. The data connection events are as follows:

Event	Description
OnEOF	Called when Rave wants to determine if it is at the end of the data. A true value should be returned only if there are no rows or if a call to the OnNext event has moved past the last row.
OnFirst	Called when Rave wants the data cursor to be moved to the first row of the data. With Rave's advanced buffering system, this will usually only be called once at the beginning of a data session.
OnGetCols	Called when Rave wants to retrieve the meta-data of the data. This includes the field name, type, character size, full name and description. See below for more details.
OnGetRow	Called when Rave wants to retrieve the data for the current row. See below for more details.
OnGetSorts	Called when Rave wants to retrieve the available sorting methods.
OnNext	Called when Rave wants to move the data cursor to the next row.
OnOpen	Called when Rave wants to initialize the data session. The current state should be saved for the later call to OnRestore.
OnRestore	Called when Rave wants to restore the data session to its state before it was opened.
OnSetFilter	Called when Rave wants to filter the data for items such as Master-Detail reports.
OnSetSort	Called when Rave wants to sort the data. See below for more details.
OnValidateRow	Called for each row to allow custom filtering of the data. For a custom data connection, this event is normally not needed since filtering the data through the OnNext event is more efficient. However, this event can be very useful for other, more automated data connections such as TRvDataSetConnection. See below for more details.
	other, more automated data connections such as TRvDataSetConnection

#### 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 allieviate 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).

## **OnGetCols Event**

The OnGetCols event is called when Rave want to retrieve the meta-data information of the data. Inside this event you will want to call the Connection.WriteField method for each field (column) of your data. The definition of WriteField is as follows:

procedure WriteField(Name: string;

DataType: TRpDataType; Width: integer; FullName: string; Description: string);

Name is the short name of the field and should only contain alphanumeric characters. DataType is the type of data that this field represents and should be one of the following values: dtString, dtInteger, dtBoolean, dtFloat, dtCurrency, dtBCD, dtDate, dtTime, dtDateTime, dtBlob, dtMemo or dtGraphic. Width is the relative character width of the field. Full name is a more descriptive name of the field and can include spaces and other non-alphanumeric characters. If FullName is blank then the short name is used as the field's full name. Description is a full description of the field and is typically edited with memo component so it can contain multiple lines. Use the description property to define how the field is to be used and provide any other information about the field's data.

Example:

procedure TDataForm.CustomCXNGetCols(Connection: TRvCustomConnection); begin

With Connection do begin

```
WriteField('Index',dtInteger,8,'Index Field','Description 1');
WriteField('Name',dtString,30,'Name Field','Description 2');
WriteField('Amount',dtFloat,20,'Amount Field','Description 3');
end; { with }
```

## **OnOpen event**

The OnOpen event is called to initialize a data session. In this event you can open up data files, initialize variables and save the current state of the data for the OnRestore event which will be called to terminate the data session.

Example:

```
procedure TDataForm.CustomCXNOpen(Connection: TRvCustomConnection);
begin
AssignFile(DataFile,'DATAFILE.DAT');
Reset(DataFile,1);
end;
```

## **OnFirst Event**

The OnFirst event is called to position the data cursor to the first row of the data.

Example:

```
procedure TDataForm.CustomCXNFirst(Connection: TRvCustomConnection);
begin
Seek(DataFile,0);
BlockRead(DataFile,DataRecord,SizeOf(DataRecord),DataRead);
end:
```

## **OnNext event**

The OnNext event is called to move the data cursor to the next row of data.

Example:

procedure TDataForm.CustomCXNNext(Connection: TRvCustomConnection); begin

BlockRead(DataFile,DataRecord,SizeOf(DataRecord),DataRead); end;

## **OnEOF** event

The OnEOF event is called to return whether the data cursor is beyond the EOF or not. A true value should be returned only if there are no rows or if a call to the OnNext event has moved past the last row.

Example:

```
procedure TMainForm.CustomCXNEOF(Connection: TRvCustomConnection; var EOF: Boolean);
```

begin

```
EOF := DataRead < SizeOf(DataRecord);
end;
```

### **OnGetRow event**

The OnGetRow event is called to retrieve the data for the current row. There are several methods used to write the data to a special buffer used by Rave. The order and types of the fields written must match exactly the field definitions provided in the OnGetCols event.

The following is a list of the methods provided by the Connection object for writing data out to the data buffer.

```
procedure WriteStrData(FormatData: string;
               NativeData: string); { dtString }
procedure WriteIntData(FormatData: string;
               NativeData: integer); { dtInteger }
procedure WriteBoolData(FormatData: string;
                NativeData: boolean); { dtBoolean }
procedure WriteFloatData(FormatData: string;
                 NativeData: extended); { dtFloat }
procedure WriteCurrData(FormatData: string;
                NativeData: currency): { dtCurrency }
procedure WriteBCDData(FormatData: string;
               NativeData: currency); { dtBCD }
procedure WriteDateTimeData(FormatData: string;
                   NativeData: TDateTime); { dtDate, dtTime and
                                    dtDateTime }
procedure WriteBlobData(var Buffer;
                   Len: longint); { dtBlob, dtMemo and
                              dtGraphic }
```

There is also a special method called WriteNullData (no parameters) that can be called for any field that contains uninitialized or null data. The FormatData parameter is used to pass a preformatted string of the data for this field. The NativeData parameter is intended to pass the unformatted or raw data of the field. If special formatting is defined in the Rave report then the formatting will be applied to NativeData. If no special formatting is defined in the Rave report then the FormatData value will be used for printing.

Example:

```
procedure TDataForm.CustomCXNGetRow(Connection: TRvCustomConnection);
begin
With Connection do begin
WriteIntData(",DataRecord.IntField);
WriteStrData(",DataRecord.StrField);
WriteFloatData(",DataRecord.FloatField);
end; { with }
```

end;

### **OnValidateRow event**

The OnValidateRow event is called for each row and allows you to control whether the current row will be included in the report or not. This is usually the only event that will be defined for non-custom data connections.

Example:

```
procedure TDataForm.CustomCXNValidateRow(Connection: TRvCustomConnection; var ValidRow: Boolean);
```

begin

```
ValidRow := DataRecord.FloatField >= 0.0;
end:
```

### **OnRestore event**

The OnRestore event is called to terminate and restore a data session to its previous state. In this event you can close data files, free resources and restore the state of the data to its state that it was before the OnOpen event was called.

Example:

procedure TDataForm.CustomCXNRestore(Connection: TRvCustomConnection); begin CloseFile(DataFile); end;

# Tutorial 6 TRvNDRWriter component

### Description

RV **•**  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.

bject Inspector RvNDRWriter1	TRvNDRWriter 💽
Properties Events	
AccuracyMethod	amAppearance
Copies	1
FileName	
FirstPage	1
LastPage	9999
LineHeightMethod	IhmLinesPerInch
LinesPerInch	6
MarginBottom	0
MarginLeft	0
MarginRight	0
MarginTop	0
Name	RvNDRWriter1
Orientation	poPortrait
ScaleX	100
ScaleY	100
StatusFormat	Printing page %p
StatusLabel	
StatusText	(TStrings)
StreamMode	smMemory
TabShade	0
Tag	0
TextBKMode	bkTransparent
Title	Rave Report
Units	unInch
UnitsFactor	1
Version	4.10 (VCL6)
II shown	

Object Inspect	or 🛛 🛛
RvNDRWriter1	TRvNDRWriter 💌
Properties Eve	ents
OnAfterPrint	
OnBeforePrint	
OnPrint	
StatusLabel	•
All shown	

## **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.

Developers Guide

# **Tutorial 7 Rendering components**

## **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.

### **TRvRenderPreview 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.

## TRvRenderPreview 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.

## TRvRenderPrinter 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.

## TRvRenderPrinter 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 descendent 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.

## TRvRenderPDF Description



TRvRenderPDF will allow the generation of PDF (Adobe Acrobat) documents from reports. Fonts can be embedded into the PDF document by setting *EmbedFonts* to true. The desired quality of images included in the PDF file can be set with a percentage

quality in the ImageQuality property. Compressed PDF documents can be generated by setting UseCompression to true, including ZLib in the uses and defining an OnCompress event such as:

```
With TCompressionStream.Create(clMax, OutStream) do try
CopyFrom(InStream, InStream.Size);
finally
Free;
end; { with }
```

The easiest way to include rendering capability in an application is to drop a render component on a form which will automatically register that format with the standard setup and preview dialogs of TRvSystem. If a more automatic means of rendering is desired, the *Render* method can be called by passing in either an NDR TStream object or a file name of an NDR file as the single parameter.

## **TRvRenderHTML** Description



TRvRenderHTML will convert an NDR stream or file into HTML pages. Text, graphic, line and rectangle objects objects are supported. The resulting output is in HTML 4.0 format and is designed to match the printed output as closely as possible.

## **TRvRenderRTF** Description



TRvRenderRTF will convert an NDR stream or file into RTF format. The resulting RTF document is designed to appear as close as possible to the original printed report. Elements in the document are included in separate "frames" to support accurate no on the page.

positioning on the page.

## TRvRenderText Description

TRvRenderText will convert and NDR stream or file to text format. Only text items in the report are supported, all other objects such as graphics or lines will be ignored. The *CPI* property allows the characters per inch to be defined and the *LPI* property allows the lines per inch to be defined to match the final output device. Note that the final output of the text

lines per inch to be defined to match the final output device. Note that the final output of the text file to a printer may not match the settings of *CPI* or *LPI* since there are no special formatting commands inserted into the text file.

# Chapter 8 TRvProject Component

### Overview

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.

## **Engine Property**

The Engine property of TRvProject allows you to define an alternate output engine to be used. This allows you to define custom setup and preview screens through the TRvSystem component or to generate NDR streams or files through the TRvNDRWriter component.

## Using TRvProject

The following is a basic sequence of steps that you would perform while using the TRvProject component within your application:

- 1) Call RvProject.Open; to open the report project defined in the ProjectFile property.
- Call RvProject.GetReportList(ListBox1.Items,true); to load a list of report names into ListBox1.
- 3) Whenever the user click on ListBox1 (ListBox1.OnClick), call RvProject.SelectReport( ListBox1.Items[ListBox1.ItemIndex],true); and then RvProject.ReportDescToMemo( Memo1); to select the current report and to copy the description of the report into Memo1.
- 4) Call RvProject.Execute; to print the currently selected report.
- 5) Call RvProject.Design; to start then end-user designer for the currently selected report (available with End User Designer License (EUDL) only).
- 6) Call RvProject.Close; to close the report project and free the memory used by it.

This is just a basic overview of a typical application and how it would call the TRvProject methods and properties. There are several other properties and methods defined in the main reference that give you even more capabilities.

To conserve space in your application, Rave only preloads the Graphics, Standard and Reporting components. Barcode components as well as any other custom components must be registered and compiled into the application explicitly if they are used in any Rave reports. The following is a list of the steps to include the barcode components into an application:

1) On the form that contains the TRvProject component for the application, add the uniy RvCsBars to the uses clause.

2) Define the TRvProject.OnCreate event and call the RaveRegister method of the RvCsBars unit as follows:

procedure TReportForm.RvProjectCreate(Sender: TObject); begin

RvCsBars.RaveRegister;

end;

The above two steps would be required of any custom component that is used in a Rave report. If you do not do the correct steps you will get an error such as "Class TRavePostNetBarcode not found" when opening the reporting project.

## Appendix A Formatting

Below is a list of different format codes and what they will accomplish for each output type.

### AlphaNumeric Items

Description:	DisplayFormat f format specifier		0 0		ng. The follow	wing
Examples:	Format String	123456.78	-123.0	0.5	0.0	

 r onnac oanng	120100110	12010	0.0	0.0
#,##0.00	123,456.78	-123.00	0.50	0.00
#.#	123456.8	-123	.5	0
\$,0.00	\$123,456.78	\$-123.00	\$0.50	\$0.00
0.00;(0.00);'-'	123456.78	(123.00)	0.50	

#### Specifier Represents

- 0 Digit place holder. If value being formatted has a digit where the '0' appears, then the digit is copied to the output string. Otherwise, a '0' is in the output string.
- Digit place holder. If value being formatted has a digit where the '#' appears, then the digit is copied to the output string. Otherwise, nothing appears in that position.
   Decimal point. The first '.' character in the format string determines the location of the decimal separator in the formatted value. The actual character used as a the decimal separator in the output string is determined by the Number Format of the International section in the Windows Control Panel.
- , Thousand separator. If the format string contains a ',' characters, the output will have thousand separators inserted between each group of three digits to the left of the decimal point. The actual character used as a the thousand separator in the output is determined by the Number Format of the International section in the Windows Control Panel.
- E+ Scientific notation. If any of the strings 'E+', 'E-', 'e+', or 'e-' are contained in the format string, the number is formatted using scientific notation. A group of up to four '0' characters can immediately follow the 'E+', 'E-', 'e+', or 'e-' to determine the minimum number of digits in the exponent. The 'E+' and 'e+' formats cause a plus sign to be output for positive exponents and a minus sign to be output for negative exponents. The 'E-' and 'e-' formats output a sign character only for negative exponents.
- 'xx'/"xx" Characters enclosed in single or double quotes are output as-is, and do not affect formatting.
- ; Separates sections for positive, negative, and zero numbers in the format string.

The locations of the leftmost '0' before the decimal point in the format string and the rightmost '0' after the decimal point in the format string determine the range of digits that are always present in the output string.

The number being formatted is always rounded to as many decimal places as there are digit placeholders ('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 placeholders 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.
	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.
	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.

## 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 1998 d mmm yy => 21 Sep 98

Specifier	Displays
С	Displays date using format given by ShortDateFormat global variable, followed by time using format given by LongTimeFormat global variable. The time is not displayed if
	fractional part of the DateTime value is zero.
d	Displays the day as a number without a leading zero (1-31).
dd	Displays the day as a number with a leading zero (01-31).
ddd	Displays the day as an abbreviation (Sun-Sat) using the strings given by the ShortDayNames global variable.
dddd	Displays the day as a full name (Sunday-Saturday) using the strings given by the LongDayNames global variable.
ddddd	Displays the date using the format given by the ShortDateFormat global variable.
ddddd	Displays the date using the format given by the LongDateFormat global variable.
m	Displays the month as a number without a leading zero (1-12). If the m specifier
	immediately follows an h or hh specifier, the minute rather than the month is displayed.
mm	Displays the month as a number with a leading zero (01-12). If the mm specifier
	immediately follows an h or hh specifier, the minute rather than the month is displayed.
mmm	Displays the month as an abbreviation (Jan-Dec) using the strings given by the ShortMonthNames global variable.
mmmm	Displays the month as a full name (January-December) using the strings given by the
	LongMonthNames global variable.
уу	Displays the year as a two-digit number (00-99).
уууу	Displays the year as a four-digit number (0000-9999).
h	Displays the hour without a leading zero (0-23).
hh	Displays the hour with a leading zero (00-23).
n	Displays the minute without a leading zero (0-59).
nn	Displays the minute with a leading zero (00-59).

- s Displays the second without a leading zero (0-59).
- ss Displays the second with a leading zero (00-59).
- t Displays the time using the format given by the ShortTimeFormat global variable.
- tt Displays the time using the format given by the LongTimeFormat global variable.
- am/pm Uses the 12-hour clock for the preceding h or hh specifier, and displays 'am' for any hour before noon, and 'pm' for any hour after noon. The am/pm specifier can use lower, upper, or mixed case, and the result is displayed accordingly.
- a/p Uses the 12-hour clock for the preceding h or hh specifier, and displays 'a' for any hour before noon, and 'p' for any hour after noon. The a/p specifier can use lower, upper, or mixed case, and the result is displayed accordingly.
- ampm Uses the 12-hour clock for the preceding h or hh specifier, and displays the contents of the TimeAMString global variable for any hour before noon, and the contents of the TimePMString global variable for any hour after noon.
- "/" Displays the date separator character given by the DateSeparator global variable.
- : Displays the time separator character given by the TimeSeparator global variable.
- 'xx'/"xx" Characters enclosed in single or double quotes are displayed as-is, and do not affect formatting.

Developers Guide

## Appendix B Keyboard / Mouse Shortcuts

Below is a list of different Keyboard / Mouse combinations that can be used as a shortcut. See Preferences - Shortcuts for assigning keyboard keys to your own shortcuts.

## Page Designer or Project Tree

Click	on a component selects that component
Right Click	shows context menu for that component
Shift Alt Click	adds all components of the same type as the component clicked on to the selection list of the current page designer
Shift Ctrl Click	adds all children of clicked component to selection list
Shift Click	on a component toggles the selection for that component. This can be used to select multiple components.

## Page Designer Only

Click	in blank area of Page Designer removes selection of all components
Ctrl + Arrow Keys	taps (moves) selected components in direction of arrow key
Ctrl C / Ctrl Ins	copies selection to clipboard
Ctrl Click	centers the design window to location clicked
Ctrl F4	unloads current global page
Ctrl V / Shift Ins	paste clipboard to page designer
Ctrl X / Shift Del	cuts selection to clipboards
Delete	deletes currently selected component(s)
Escape	changes selection to parent of current component
F9	executes the current report
F11	toggles between page designer and property panel
Shift + Arrow Keys	changes size of selected components (Up = decrement height, Down = increment height, Left = decrement width, Right = increment width)

### **Project Tree Only**

Alt Drag	DataField component to page designer - creates text component
Alt Drag	selected component to container component in Project Tree - makes the destination component (must be a container component like sections or regions) the parent of all selected components
Ctrl Drag	DataField component to page designer - creates DataText component
Ctrl Drag	component to page designer - creates a mirror of component
Double Click	on Global Page node - loads selected page into page designer
Double Click	on Report node - actives selected report

Developers Guide

# Appendix C Property Descriptions

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

## Abort method

#### Declaration

procedure Abort;

#### Category

Control

#### Component/Class

TBaseReport

#### 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

Aborted, Execute

#### Example

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

#### C++Builder

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

## Aborted property (read only)

#### Declaration

property Aborted: boolean;

#### Category

Control

#### Component/Class

TBaseReport

#### Description

This property will be set to true after a call to *Abort* has been made.

### See also

Abort

#### Example

#### . Delphi

```
RvNDRWriter1.Execute;
if RvNDRWriter1.Aborted then begin
  StatusFormat := #13 + 'Report Canceled!';
end else begin
  StatusFormat := #13 + 'Report Completed!';
end; { else }
UpdateStatus;
```

#### C++Builder

```
rp->Execute();
if (rp->Aborted) {
  rp->StatusFormat = "\nReport Canceled!";
} else {
  rp->StatusFormat = "\nReport Completed!";
}
rp->UpdateStatus();
```

## AbortPage method

#### Declaration

procedure AbortPage;

Category

Control

Component/Class TBaseReport

#### Description

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

See also

<u>Abort</u>

#### Example

```
Delphi
RvNDRWriter1.AbortPage;
C++Builder
rp->AbortPage();
```

## AccuracyMethod property (read/write/published)

#### Declaration

property AccuracyMethod: TAccuracyMethod;

#### Default

amAppearance {TRvNDRWriter}, amPositioning {TRvSystem}

#### Category

Control

#### Component/Class

TBaseReport

#### Description

This property controls how text is written to the report file. If *AccuracyMethod* is equal to *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 *amAppearance* then the text string is written out as a complete string in the normal fashion. The problem with *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.

#### Example

Delphi
RvNDRWriter1.AccuracyMethod := amAppearance;
C++Builder
RvNDRWriter1->AccuracyMethod = amAppearance;

## Active property (read/write)

#### Declaration

property Active: Boolean;

#### Default

false

Category Rave

Component/Class

#### 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

Close, OnAfterClose, OnAfterOpen, OnBeforeClose, OnBeforeOpen, Open

#### Example

Same as RaveProject1.Open;

```
Delphi
RvProject1.Active := True; { Same as RvProject1.Open; }
C++Builder
RvProject1->Active = true;
```

## Active property (read/write)

#### Declaration

property Active: boolean read FActive write FActive

Default

true

Category Render

Component/Class

TRpRender

#### 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

**DisplayName** 

## AdjustLine method

#### Declaration

procedure AdjustLine;

#### Category

Position

#### Component/Class

TBaseReport

#### 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

**ResetLineHeight** 

#### Example

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

#### C++Builder

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

## AllowAll method

#### Declaration

procedure AllowAll;

#### Category

Control

### Component/Class

TBaseReport

#### Description

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

#### See also

AllowPreviewOnly, AllowPrinterOnly

#### Example

Draw a line on the preview screen only

#### Delphi

```
AllowPreviewOnly;
MoveTo(1.5,1.5);
LineTo(6.5,1.5);
AllowAll;
```

#### C++Builder

rp->AllowPreviewOnly(); rp->MoveTo(1.5,1.5); rp->LineTo(6.5,1.5); rp->AllowAll();

### AllowPreviewOnly method

#### Declaration

procedure AllowPreviewOnly;

#### Category

Control

#### Component/Class

TBaseReport

#### 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

AllowAll, AllowPrinterOnly

#### Example

See AllowAll

### AllowPrinterOnly method

#### Declaration

procedure AllowPrinterOnly;

#### Category

Control

#### Component/Class

TBaseReport

#### 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

AllowAll, AllowPreviewOnly

#### Example

See AllowAll

### Append method

Declaration

procedure Append(Text: string);

Category

Memo

Component/Class TMemoBuf

#### Description

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

See also Insert

Example

#### Delphi

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

### AppendMemoBuf method

#### Declaration procedure AppendMemoBuf(MemoBuf: TMemoBuf); Category

Memo

#### Component/Class TMemoBuf

#### Description

Will append MemoBuf to the current memo buffer.

See also

InsertMemoBuf

```
Delphi
MemoBuf1.AppendMemoBuf(MemoBuf2);
C++Builder
MemoBuf1->AppendMemoBuf(MemoBuf2);
```

### Arc method

#### Declaration

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

#### Category

Graphics

#### Component/Class

TBaseReport

#### 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

Ellipse, Pie

#### Example

Delphi
RvNDRWriter1.Arc(1.0,1.0,3.0,3.0,3.0,2.0,0.0,0.0);
C++Builder
RvNDRWriter1->Arc(1.0,1.0,3.0,3.0,3.0,2.0,0.0,0.0);

### AscentHeight property (read only)

#### Declaration

property AscentHeight: double;

#### Category

Position

#### Component/Class

. TBaseReport

#### Description

Returns the height of the line font above the baseline. **NOTE:** This applies to the line font only and not to the current textfont.

#### See also

DescentHeight, FontHeight, LineHeight

### AssignFont method

#### Declaration

procedure AssignFont(Font: TFont);

#### Category

Font

#### Component/Class TBaseReport

#### Description

Selects current font to the TFont object from list.

### See also

SetFont

#### Example

```
Delphi
RvNDRWriter1.AssignFont( FontDialog1.Font );
C++Builder
RvNDRWriter1->AssignFont( FontDialog1->Font );
```

### BarBottom property (read/write)

#### Declaration

property BarBottom: double;

#### Default

pjLeft

#### Category BarCode

#### Component/Class TRpBarsBase

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

BarTop, Bottom, PrintReadable, PrintTop

#### Example

See Create { bar code }

### BarCodeJustify property (read/write)

#### Declaration

property BarCodeJustify: TPrintJustify

#### Default

pjLeft

#### Category

BarCode

#### Component/Class TRpBarsBase

#### Description

This determines where the bar code is printed relative to the Position property.

pjLeft99	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

Center, Left, Position, Right

#### Example

equivalent to Center := 2.5;

#### Delphi

Position := 2.5; BarCodeJustify := pjCenter;

#### C++Builder

rpl->Position = 2.5; rpl->BarCodeJustify = pjCenter;

### BarCodeRotation property (read/write)

#### Declaration

property BarCodeRotation: TBarCodeRotation

#### Default

Rot0

#### Category

BarCode

#### Component/Class TRpBarsBase

#### 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.

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

#### See also

<u>Left</u>, <u>Top</u>

#### Example

print Bar Code upside down

#### Delphi

BarCodeRotation := Rot180; C++Builder rpl->BarCodeRotation = Rot180;

### BarHeight property (read/write)

#### Declaration

property BarHeight: double;

#### Default

0.5 ( PostNet 0.125 )

#### Category

BarCode

#### Component/Class TRpBarsBase

Description

Sets or returns the value for the tallest bar.

#### See also

<u>BarWidth</u>

#### Example

Bars will be 3/10 inch tall

## Delphi BarHeight := 0.3; C++Builder

rpl->BarHeight = 0.3;

### BarTop property (read/write)

#### Declaration

property BarTop: double;

### Default

0

#### Category

BarCode

#### Component/Class TRpBarsBase

#### 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

BarBottom, PrintReadable, PrintTop, Top

#### Example

Delphi
BarCode1.BarTop := 0.5;
C++Builder
BarCode1->BarTop = 0.5;

### BarWidth property (read/write)

#### Declaration

property BarWidth: double

#### Default

0.01 (PostNet 0.020)

#### Category

BarCode

Component/Class TRpBarsBase

Description Sets or returns the value of the narrow bar width.

See also

BarHeight, Width

#### Example

set narrow bar width to 2/100 ths

#### **Delphi** BarWidth := 0.02;

C++Builder

rpl->BarWidth = 0.02;

### BaseReport property (read/write)

#### Declaration

property BaseReport: TBaseReport

#### Default

nil

### Category

Memo

### Component/Class

TMemoBuf

#### 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

MemoHeightLeft, MemoLinesLeft, PrintHeight, PrintLines

```
Delphi
MemoBuf.BaseReport := Sender as TBaseReport;
C++Builder
MemoBuf->BaseReport = dynamic_cast<TBaseReport*>(Sender);
```

### BaseReport property (read/write)

#### Declaration

property BaseReport: TBaseReport

#### Default

nil

Category

BarCode

Component/Class TRpBarsBase

#### 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

Create\_TRpBarsBase

```
Delphi
Barcodel.BaseReport := ( Sender as TBaseReport );
C++Builder
Barcodel->BaseReport = dynamic_cast<TBaseReport*>(Sender);
```

### BaseReport property (read/write)

#### Declaration

property BaseReport: TBaseReport

#### Default

nil

Category

Control

#### Component/Class TRvSystem

#### 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

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 }
```

```
rp1->BaseReport->Print("This is a test");
```

### Bins property (read only)

#### Declaration

property Bins: TStrings;

#### Default

(the list of bins for the default printer)

#### Category

Printer

#### Component/Class TBaseReport

#### Description

This property will return a TStringList containing all of the valid printer bins for the current printer.

#### See also

SelectBin, SupportBin, TStrings

#### Example

Display the printer bins in a list box

Delphi

ListBox1.Items := RvNDRWriter1.Bins; C++Builder ListBox1->Items = RvNDRWriter1->Bins;

### BKColor property (read/write)

#### Declaration

property BKColor: TColor;

#### Default

clWhite

#### Category

Graphics

### Component/Class

TBaseReport

#### Description

This property returns or sets the current background color for text output.

See also

TColor, TextBKMode

```
Delphi
RvNDRWriter1.BKColor := clWhite;
C++Builder
RvNDRWriter1->BKColor = clWhite;
```

### Bold property (read/write)

#### Declaration

property Bold: boolean;

#### Default

false

### Category

Font

Component/Class TBaseReport

#### Description

This property returns or sets the bold attribute for the current font

#### See also

Italic, Strikeout, Underline

#### Example

# Delphi with RvNDRWriter1 do begin Bold := true; Print( 'Bold Text' ); Bold := false; end; { with }

#### C++Builder

rpl->Bold = true; rpl->Print( "Bold Text" ); rpl->Bold = false;

### Bottom property (read/write)

#### Declaration

property Bottom: double;

#### Category BarCode

Component/Class TRpBarsBase

#### 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

BarBottom, PrintReadable, PrintTop

### BottomWaste property (read only)

#### Declaration

property BottomWaste: double;

#### Category

Printer

#### Component/Class

TBaseReport

#### 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

LeftWaste, MarginBottom, RightWaste, TopWaste

#### Example

See LeftWaste

### BoxLineColor property (read/write)

#### Declaration

property BoxLineColor: TColor;

Default

clBlack

#### Category Tabs

Component/Class TBaseReport

#### Description

This property will define the color used to draw the sides of tab boxes defined with SetTab.

#### See also

SetTab, Tab, TabColor, TColor

```
Delphi
RvNDRWriter1.BoxLineColor := clGreen;
C++Builder
RvNDRWriter1->BoxLineColor = clGreen;
```

### BrushCopy method

#### Declaration

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

#### Category

Graphics

#### **Component/Class TBaseReport**

#### 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

CreateRect, TColor, TRect

#### Example

#### Delphi

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

### Buffer property (read only)

#### Declaration

property Buffer: ^Array[ 0..MaxBufSize ] of Char;

#### Category

Memo

#### **Component/Class**

**TMemoBuf** 

#### Description

This property is a pointer to memory buffer used by TMemoBuf. NOTE: Not normally necessary to access this property.

#### See also

LoadFromFile, SetData, Text

### BufferInc property (read/write)

#### Declaration

property BufferInc: longint;

#### Default

256

Category Memo

Component/Class TMemoBuf

#### 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

<u>MaxSize</u>

#### Example

```
Delphi
MemoBuf.BufferInc := 128;
C++Builder
MemoBuf->BufferInc = 128;
```

### CacheDir property

#### Declaration

property CacheDir: String read FCacheDir write FCacheDir

#### Category

Render

#### Component/Class

**T**RpRender

#### 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

<u>ServerMode</u>

### CalcGraphicHeight method

#### Declaration

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

#### Category

Graphics

#### Component/Class

TBaseReport

#### 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

CalcGraphicWidth, PrintBitmap, PrintBitmapRect, StretchDraw

#### Example

#### 

### CalcGraphicWidth method

#### Declaration

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

#### Category

Graphics

#### Component/Class

TBaseReport

#### 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

CalcGraphicHeight, PrintBitmap, PrintBitmapRect, StretchDraw

#### Example

### Canvas property (read only)

#### Declaration

property Canvas: TCanvas;

#### Category

Printer

#### Component/Class

TBaseReport

#### 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

<u>RpDev</u>, <u>TCanvas</u>

#### Example

Save the current canvas

#### Delphi

RvNDRWriter1.Canvas.Pen := SavePen; C++Builder RvNDRWriter1->Canvas->Pen = SavePen;

### Center property (read/write)

#### Declaration

property Center: double;

#### Default

relative to Left and Right properties

#### Category

BarCode

#### Component/Class TRpBarsBase

#### 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

BarCodeJustify, Left, Position, Right

```
Delphi
Barcodel.Center := (SectionLeft + SectionRight) / 2.0;
C++Builder
Barcodel->Center = (rpl->SectionLeft + rpl->SectionRight)/2.0;
```

### CheckSum property (read only)

#### Declaration

property CheckSum: boolean;

#### Category

BarCode

#### Component/Class

TRpBarsBase

#### 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

<u>UseChecksum</u>

### Chord method

#### Declaration

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

#### Category

Graphics

#### Component/Class

TBaseReport

#### 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

Ellipse

#### Example

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

### Clear method

#### Declaration

procedure Clear;

#### Category

Preview

#### Component/Class

**TRvRenderPreview** 

#### 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

<u>ScrollBox</u>

#### Example

Clear the preview screen

#### Delphi

```
RvRenderPreview1.Clear;
C++Builder
RvRenderPreview1->Clear();
```

### **ClearAllTabs** method

#### Declaration

procedure ClearAllTabs;

#### Category

Tabs

### Component/Class

TBaseReport

#### 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

ClearTabs, SaveTabs

#### Example

Clear all tabs, including saved tabs

#### Delphi

ClearAllTabs; C++Builder rp1->ClearAllTabs();

### ClearColumns method

#### Declaration

procedure ClearColumns;

#### Category

Column

Component/Class TBaseReport

#### Description

This method removes all current column settings.

#### See also

SetColumns, SetColumnWidth

#### Example

Delphi
RvNDRWriter1.ClearColumns;
C++Builder
RvNDRWriter1->ClearColumns();

### ClearRaveBlob method

#### Declaration

procedure ClearRaveBlob;

#### Category

Rave

#### Component/Class

TRvProject

#### 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

LoadRaveBlob, RaveBlobDateTime, SaveRaveBlob, StoreRAV

```
Delphi
RvProject1.ClearRaveBlob;
C++Builder
RvProject1->ClearRaveBlob();
```

### ClearTabs method

#### Declaration

procedure ClearTabs;

Category

Tabs

Component/Class TBaseReport

#### Description

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

#### See also

ResetTabs, SetTab

#### Example

```
Delphi
RvNDRWriter1.ClearTabs;
C++Builder
RvNDRWriter1->ClearTabs();
```

### **Close** method

#### Declaration

procedure Close;

#### Category

Rave

### Component/Class

TRvProject

#### 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

Active, OnAfterClose, OnAfterOpen, OnBeforeClose, OnBeforeOpen, Open

```
Delphi
RvProject1.Close;
C++Builder
RvProject1->Close();
```

### CodePage property (read/write)

#### Declaration

property CodePage: TCodePage128;

#### Default

cpCodeA

Category BarCode

Component/Class TRpBarsBase

#### Description

Specifies whether Code A, Code B or Code C is being used.cpCodeAsets 128 output to Code AcpCodeBsets 128 output to Code BcpCodeCsets 128 output to Code C

#### Example

set 128 code output to C

#### Delphi

CodePage := cpCodeC; Text := '125692';

#### C++Builder

Barcode1->CodePage = cpCodeC; Barcode1->Text = "125692";

### Collate property (read/write)

#### Declaration

property Collate: boolean

#### Default

(will be equal to the collation setting for the default printer)

#### Category

Printer

#### Component/Class TBaseReport

#### 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

SupportCollate

#### Example

```
Delphi
if SupportCollate then begin
  Collate := true;
end; { if }
```

```
if (rp1->SupportCollate()) {
  rp1->Collate = true;
}
```

### ColumnEnd property (read only)

#### Declaration

property ColumnEnd: double;

#### Category

Column

#### Component/Class

TBaseReport

#### 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

ColumnNum, SetColumns, SetColumnWidth

#### Example

Print memo buffer

#### Delphi

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

```
rp->SetColumns(3,0.25);
MemoBuf->PrintStart = rp->ColumnStart;
MemoBuf->PrintEnd = rp->ColumnEnd;
rp->PrintMemo(MemoBuf, rp->ColumnLinesLeft(), false);
```

### ColumnLinesLeft property

#### Declaration

function ColumnLinesLeft: integer;

#### Category

Column

#### Component/Class

TBaseReport

#### 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

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; { while }
```

### ColumnNum property (read/write)

#### Declaration

property ColumnNum: integer;

#### Default

1

Category

Column

Component/Class TBaseReport

#### Description

This property will return or set the current column number that the text cursor is on.

#### See also

Columns, SetColumns, SetColumnWidth

#### Example

Delphi
CurrColNum := RvNDRWriter1.ColumnNum;
C++Builder
CurrColNum = RvNDRWriter1->ColumnNum;

### Columns property (read only)

#### Declaration

property Columns: integer;

#### Category

Column

#### Component/Class

TBaseReport

#### Description

This property returns the number of columns that are available from the last call to SetColumns or SetColumnWidth.

#### See also

ColumnNum, SetColumns, SetColumnWidth

#### Example

Delphi

CurrColumns := RvNDRWriter1.Columns; C++Builder CurrColumns = RvNDRWriter1->Columns;

### ColumnStart property (read only)

#### Declaration

property ColumnStart: double;

#### Category

Column

#### Component/Class

TBaseReport

#### 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

ColumnNum, SetColumns, SetColumnWidth

#### Example

```
Delphi
CurrColStart := RvNDRWriter1.ColumnStart;
C++Builder
CurrColStart := RvNDRWriter1->ColumnStart;
```

### ColumnWidth property (read only)

#### Declaration

property ColumnWidth: double;

#### Category

Column

#### Component/Class TBaseReport

Description

This property returns the width of the current column.

#### See also

SetColumns, SetColumnWidth

```
Delphi
CurrColWidth := RvNDRWriter1.ColumnWidth;
C++Builder
CurrColWidth := RvNDRWriter1->ColumnWidth;
```

### ConstraintHeightLeft method

#### Declaration

function ConstraintHeightLeft(Constraint: double): double;

#### Category

Memo

#### Component/Class

TMemoBuf

#### 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

MemoHeightLeft, PrintEnd, PrintMemo, PrintStart, TMemoBuf

#### Example

#### Delphi

MemoBuf.BaseReport := Sender as TBaseReport; HeightLeft := MemoBuf.ConstraintHeightLeft(5.0);

#### C++Builder

MemoBuf->BaseReport = rp; HeightLeft = MemoBuf->ConstraintHeightLeft(5.0);

### Copies property (read/write/published)

#### Declaration

property Copies: integer;

#### Default

1

#### Category

Printer

#### Component/Class TBaseReport

#### 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

<u>MaxCopies</u>

#### Example

Print three copies

```
Delphi
RvNDRWriter1.Copies := 3;
C++Builder
RvNDRWriter1->Copies = 3;
```

### CopyRect method

#### Declaration

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

#### Category

Graphics

#### Component/Class TBaseReport

#### 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

CreateRect, TCanvas, TRect

```
Delphi
RvNDRWriter1.CopyRect( DstRect, DstCanvas, SrcRect);
C++Builder
RvNDRWriter1->CopyRect( DstRect, DstCanvas, SrcRect);
```

### CR method

#### Declaration

procedure CR;

#### Category

Position

#### Component/Class

TBaseReport

#### 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

ColumnStart, LE, 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 }
```

```
rp->SectionLeft = 3.0;
rp->Println("This text is 3 inches from left");
rp->SectionLeft = 1.0;
rp->CR();
```

### Create method

#### Declaration

constructor Create(AOwner: TComponent);

#### Category

Misc

#### Component/Class

TBaseReport

#### Description

This constructor should be called to create an instance of a component. This constructor should not normally be called if the component is placed visually on a form.

#### See also

Destroy

#### Example

Dynamically create a Rave component

#### Delphi

```
var
    MyReportPrinter: TRvNDRWriter;
begin
 MyReportPrinter := TRvNDRWriter.Create(self);
 with MyReportPrinter do try
   MarginTop := 1.0;
   MarginBottom := 1.5;
   MarginRight := 1.0;
   MarginLeft := 1.0;
   OnPrint := MyOnPrintMethod;
   Execute;
  finally
   Free;
                { This will call the Destroy method }
 end; { with }
end;
```

```
TRvNDRWriter* rpl;
rpl = new TRvNDRWriter(this);
try {
  rpl->MarginTop = 1.0;
  rpl->MarginBottom = 1.5;
  rpl->MarginRight = 1.0;
  rpl->MarginLeft = 1.0;
  rpl->OnPrint = MyOnPrintMethod;
  rpl->Execute();
}
__finally {
  delete rpl;
}/ tryf
```

### Create method

#### Declaration

constructor Create( BaseRpt: TBaseReport );

#### Category

BarCode

#### Component/Class

TRpBarsBase

#### 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

BaseReport (bar code)

#### Example

```
Delphi
BarCodel := TRpBarsPostNet.Create(Sender as TBaseReport);
with BarCodel do begin
   BarHeight := 0.125;
   BarWidth := 0.020;
   UseChecksum := True;
   Text := '85283-3558'; {'-' will be stripped}
   Left := MarginLeft + 1.0;
   Print;
end; {if}
BarCodel.Free;
```

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

### CreateBrush method

#### Declaration

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

#### Category

Graphics

#### Component/Class TBaseReport

#### 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

SetBrush, TBrush, TBrushStyle, TColor

#### Example

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

```
TBrush* MyBrush;
MyBrush = rp->CreateBrush(clRed, bsSolid, NULL);
MyBrush->Free();
```

### CreateFont method

#### Declaration

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

#### Category

Font

#### Component/Class

TBaseReport

#### 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

RestoreFont, SaveFont, SetFont, TFont

#### Example

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

#### C++Builder

TFont\* MyFont; MyFont = rp->CreateFont("Times New Roman",8.00);

### CreatePen method

#### Declaration

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

#### Category

Graphics

#### Component/Class TBaseReport

#### 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

SetPen, TColor, TPen, TPenMode, TPenStyle

#### Example

```
Delphi
MyPen := CreatePen(clBlack,psSolid,1,pmBlack);
C++Builder
MyPen = rp->CreatePen(clBlack,psSolid,1,pmBlack);
```

### CreatePoint method

#### Declaration

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

#### Category

Graphics

#### Component/Class TBaseReport

Description

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

#### See also

TPoint

<u>......</u>

#### Example Delphi

```
MyPoint := CreatePoint(1.00,6.00);
C++Builder
MyPoint = rp->CreatePoint(1.00,6.00);
```

# CreateRect method

## Declaration

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

### Category

Graphics

#### Component/Class **TBaseReport**

# Description

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

# See also

CopyRect, TextRect, TRect

# Example

```
Delphi
MyRect := CreateRect(1.00,6.00,3.00,8.00);
C++Builder
MyRect = rp->CreateRect(1.00,6.00,3.00,8.00);
```

# CurrentPage property (read only)

```
Declaration
   property CurrentPage: integer;
Category
   Control
Component/Class
   TBaseReport
Description
   This property returns the current page number.
Example
   Delphi
   with RvRenderPreview1 do begin
     PageEdit.Text := IntToStr(CurrentPage);
     PageLabel.Caption := 'Page ' +
       IntToStr(CurrentPage-FirstPage+1) +
       ' of ' + IntToStr(Pages);
   end; { with }
   C++Builder
   PageEdit->Text = IntToStr( RvRenderPreview1->CurrentPage);
   PageLabel->Caption = AnsiString("Page ") +
                 IntToStr(RvRenderPreview1->CurrentPage -
                 RvRenderPreview1->FirstPage+1) +
                 AnsiString(" of ") +
```

IntToStr( RvRenderPreview1->Pages);

# CurrentPass property (read/write)

## Declaration

property CurrentPass: Integer;

## Category

Misc

# Component/Class

TBaseReport

# 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

StatusFormat, StatusLabel, StatusText, TotalPasses, UpdateStatus

# Example

```
Delphi
RvNDRWriter1.StatusFormat := 'Printing page (Pass of )';
C++Builder
RvNDRWriter1->StatusFormat = "Printing page (Pass of )";
```

# CursorXPos property (read only)

# Declaration

property CursorXPos: longint;

## Category

Position

### Component/Class TBaseReport

# Description

This property returns the horizontal text cursor position in printer units (dots).

# See also

CursorYPos, XPos, YPos

# Example

Delphi
CurrentXDots := RvNDRWriter1.CursorXPos;
C++Builder
CurrentXDots = RvNDRWriter1->CursorXPos;

# CursorYPos property (read only)

# Declaration

property CursorYPos: longint;

# Category

Position

### Component/Class TBaseReport

# Description

This property returns the vertical text cursor position in printer units (dots).

# See also

CursorXPos, XPos, YPos

# Example

```
Delphi
CurrentYDots := RvNDRWriter1.CursorYPos;
C++Builder
CurrentYDots = RvNDRWriter1->CursorYPos;
```

# DataSet property (read/write/published)

# Declaration property DataSet: TDataSet; Default nil Category Rave Component/Class TRvDataSetConnection Description Specifies the *dataset* to use with the current TRvDataSetConnection component. Example Delphi CustomerCXN.DataSet := CustomerTable;

### C++Builder

CustomerCXN->DataSet = CustomerTable;

# DefaultDest property (read/write/published)

### Declaration

property DefaultDest: TReportDest;

### Default

rdPreview

Category

ReportSystem

Component/Class TRvSystem

# 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

ReportDest, TReportDest

# Example

```
Delphi
RvSystem1.DefaultDest := rdPrinter;
C++Builder
RvSystem1->DefaultDest = rdPrinter;
```

# Delete method

### Declaration

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

# Category

Memo

# Component/Class

TMemoBuf

### Description

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

### See also

Insert

### Example

Delete 5 characters at current position

### Delphi

MemoBuf.Delete(MemoBuf.Pos,5); C++Builder MemoBuf->Delete(MemoBuf->Pos,5);

# DescentHeight property (read only)

### Declaration

property DescentHeight: double;

### Category

Position

# Component/Class

TBaseReport

### 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

AscentHeight, FontHeight, LineHeight

# Design method

### Declaration

procedure Design;

# Category

Rave

# Component/Class

TRvProject

## 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

DesignReport, Execute, ExecuteReport, SelectReport

# Example

Delphi
RvProject1.Design;
C++Builder
RvProject1->Design();

# DesignReport method

# Declaration

procedure DesignReport(ReportName: string);

### Category

Rave

# Component/Class

TRvProject

## 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 it's 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

Design, Execute, ExecuteReport

### Example

Delphi
RvProject1.DesignReport('CustomerListing');
C++Builder
RvProject1->DesignReport("Customer Listing");

# **Destroy** method

### Category

Misc

# Component/Class

TBaseReport

### Description

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

## See also

Create

### Example

see Create

# DeviceName property (read only)

### Declaration

property DeviceName: string;

Category

Printer

Component/Class TBaseReport

### Description

This property will return the device name for the currently selected printer.

See also

**PrinterIndex** 

Example

Save current device name

### Delphi

CurrDeviceName := RvNDRWriter1.DeviceName; C++Builder CurrDeviceName = RvNDRWriter1->DeviceName;

# DevMode property (read/write)

### Declaration

property DevMode: PDevMode;

### Category

Printer

### Component/Class TBaseReport

## 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

TDevMode structure in Windows API help.

## Example

Save current printer device mode and set the print resolution to low

#### Delphi

CurrDevMode := RvNDRWriter1.DevMode; RvNDRWriter1.DevMode^.dmPrintQuality := DMRES\_LOW;

```
PDevMode CurrDevMode = RvNDRWriter1->DevMode;
RvNDRWriter1->DevMode->dmPrintQuality = DMRES_LOW;
```

# DisplayName property (read/write)

## Declaration

property DisplayName: string read FDisplayName write SetDisplayName;

### Category

Render

## Component/Class

TRpRender

### 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

<u>Active</u>

# DLLFile property (read/write/published)

### Declaration

property DLLFile: string;

### Default

'' (empty)

Category

Rave

# Component/Class

TRvProject

### 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

LoadDesigner

```
Delphi
RvProject.DLLFile := 'MyName.DLL';
C++Builder
RvProject->DLLFile = "MyName.DLL";
```

# DrawFocusRect method

# Declaration

procedure DrawFocusRect(const Rect: TRect);

## Category

Graphics

# Component/Class

TBaseReport

# Description

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

# See also

CreateRect, TRect

# Example

Delphi
RvNDRWriter1.DrawFocusRect(CreateRect(1.0,1.0,2.0,3.0));
C++Builder
RvNDRWriter1->DrawFocusRect(rp->CreateRect(1.0,1.0,2.0,3.0));

# Draw method

# Declaration

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

## Category

Graphics

## Component/Class

TBaseReport

## 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

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;
```

```
TGraphic* MyLogo;
MyLogo = new TMetafile();
try {
    MyLogo->LoadFromFile("MYLOGO.WMF");
    RvNDRWriter1->Draw(1.0,2.0,MyLogo);
    }
    ___finally {
        delete MyLogo;
    }/ tryf
```

# DriverName property (read only)

# Declaration

property DriverName: string;

### Category

Printer

### Component/Class TBaseReport

гразеттеро

# Description

This property will return the driver name for the currently selected printer.

# Example

Save current driver name

# Delphi

```
CurrPrintDriver := RvNDRWriter1.DriverName;
C++Builder
CurrPrintDriver = RvNDRWriter1->DriverName;
```

# Duplex property (read/write)

# Declaration

property Duplex: TDuplex;

# Default

(will be equal to the duplex setting for the default printer)

## Category

Printer

### Component/Class TBaseReport

# 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

SupportDuplex

# Example

### Delphi

if SupportDuplex then begin
 Duplex := dupVertical;
end; { if }

```
if (rp->SupportDuplex()) {
    rp->Duplex = dupVertical;
}/ if
```

# Ellipse method

# Declaration

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

# Category

Graphics

### Component/Class TBaseReport

Пазстер

# Description

This method draws an ellipse bounded by the rectangle defined by (X1,Y1) and (X2,Y2).

# See also

<u>Arc, Pie</u>

# Example

```
Delphi
Ellipse(5.375,1.25,7.375,2.75);
C++Builder
rp->Ellipse(5.375,1.25,7.375,2.75);
```

# Empty method

# Declaration

function Empty: boolean;

# Category

Memo

# Component/Class

TMemoBuf

# 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

<u>Pos</u>, <u>Size</u>

# Example

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

```
if (!MemoBuf1->Empty()) {
    rp->PrintMemo(MemoBuf1,0,false);
}/ if
```

# Engine property (read/write/published)

### Declaration

property Engine: TRpComponent;

### Default

nil

Category Rave

Component/Class TRvProject

### 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

Execute, ExecuteReport

### Example

```
Delphi
RvProject1.Engine := RvSystem1;
C++Builder
RvProject1->Engine = RvSystem1;
```

# Execute method

#### Declaration

procedure Execute;

#### Category

Control

# Component/Class

TBaseReport

### 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

Abort, Printing, All printing event handlers

```
Delphi
RvNDRWriter1.Execute;
C++Builder
RvNDRWriter1->Execute();
```

# Execute method

### Declaration

procedure Execute;

## Category

Rave

# Component/Class

TRvProject

### 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

ExecuteReport, SelectReport

### Example

Delphi
RvProject1.Execute;
C++Builder
RvProject1->Execute();

# ExecuteCustom method

### Declaration

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

### Category

Control

#### Component/Class

. TRvRenderPreview

#### 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 *NewLast page*, if non-zero, will only print the report file for that page range.

#### See also

Copies, Execute

Example

Print 2 copies of only the first four pages

#### Delphi

```
RvRenderPrinter1.ExecuteCustom( 1, 4, 2);
C++Builder
RvRenderPrinter1->ExecuteCustom( 1, 4, 2);
```

# ExecuteReport method

## Declaration

procedure ExecuteReport(ReportName: string);

### Category

Rave

## Component/Class

TRvProject

### 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

Execute

### Example

```
Delphi
RvProject1.ExecuteReport('CustomerListing');
C++Builder
RvProject1->ExecuteReport("CustomerListing");
```

# Extended property (read/write)

### Declaration

property Extended: boolean;

### Default

false

# Category

BarCode

Component/Class TRpBarsBase

### Description

If this property is true then it will output Extended Code 39 format.

See also

### ExtendedText

### Example

```
Delphi
Extended := True;
Text := 'Test Data';
```

#### C++Builder

Extended = true; Text = "Test Data";

# ExtendedText property (read only)

### Declaration

property ExtendedText: string;

### Category

BarCode

# Component/Class

TRpBarsBase

### Description

When Extended is true, this property will contain the converted Code39 text that will be printed in the bar code.

### See also

Extended, Text

# Example

```
Delphi
ShowMessage('The raw data of this Code 39 BarCode is ' +
    Code39Bar.ExtendedText);
```

### C++Builder

```
ShowMessage("The raw data of this Code 39 BarCode is " +
   Code39Bar->ExtendedText);
```

# Field property

### Declaration

property Field: TMemoField;

# Category

Memo

#### Component/Class TMemoBuf

### Description

This property will assign the contents of a TMemoField component to the memo buffer.

### See also

Pos, Size, TMemoField

```
Delphi
MemoBuf1.Field := MyMemoField;
C++Builder
MemoBuf->Field = MyMemoField;
```

# FieldAliasList property (read/write/published)

Default

(blank)

## Category

Rave

### Component/Class

TRvCustomConnection

### 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.

# FileName property (read/write/published)

### Declaration

property FileName: String;

### Default

'' (empty)

# Component/Class

TBaseReport

### Description

Specifies the file name to create when the execute method is called.

```
Delphi
RvNDRWriter1.FileName := 'DOC1.DOC';
C++Builder
RvNDRWriter1->FileName = "DOC1.DOC";
```

# FillRect method

# Declaration

procedure FillRect(const Rect: TRect);

# Category

Graphics

### Component/Class TBaseReport

# Description

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

# See also

CreateRect, TRect

# Example

```
Delphi
FillRect( CreateRect( 1.0, 1.0, 2.0, 3.0 ) );
C++Builder
rp->FillRect(rp->CreateRect(1.0, 1.0, 2.0, 3.0));
```

# Finish method

# Declaration

procedure Finish;

# Category

Control

# Component/Class

TBaseReport

# 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

<u>Start</u>

```
Delphi
RvRenderPreview1.Finish;
C++Builder
RvRenderPreview1->Finish();
```

# FinishTabBox method

### Declaration

procedure FinishTabBox(Width: integer);

### Category

Tabs

### Component/Class

TBaseReport

### 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

<u>SetTab</u>

# Example

```
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);
```

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

# FirstPage property (read/write/published)

### Declaration

property FirstPage: integer;

#### Default

1

Category

Control

Component/Class TBaseReport

### 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

PageInvalid

### Example

print only pages 3 through 5

#### Delphi

```
RvNDRWriter1.FirstPage := 3;
RvNDRWriter1.LastPage := 5;
```

#### C++Builder

```
RvNDRWriter1->FirstPage = 3;
RvNDRWriter1->LastPage = 5;
```

# FloodFill method

### Declaration

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

#### Category

Graphics

#### Component/Class TBaseReport

#### 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, Color, is encountered. *FillStyle* defines the method of fill used. (*fsBorder* will fill until the color, *Color*, is encountered and *fsSurface* will fill while the color, *Color*, is still encountered.)

# See also

PageInvalid, TColor

```
Delphi
FloodFill(2.0,3.0,clRed,fsBorder);
C++Builder
FloodFill(2.0,3.0,clRed,fsBorder);
```

# FontAlign property (read/write)

# Declaration

property FontAlign: TFontAlign;

### Category

Font

# Component/Class

TBaseReport

### 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

Other FontXxxx properties, <u>FontBaseline</u>, <u>FontBottom</u>, <u>FontTop</u>, <u>SetFont</u>, ResetLineHeight should link to font category

# Example

#### Delphi

FontAlign := faTop; Print('This text is aligned at the top'); FontAlign := faBaseline;

```
rp->FontAlign = faTop;
rp->Print("This text is aligned at the top");
rp->FontAlign = faBaseline;
```

# FontBaseline property (read/write)

Declaration property FontBaseline: double; Default see ResetLineHeight Category Position Component/Class TBaseReport Description Returns or sets the baseline of the line font

See also

FontBottom, FontTop, LineBottom, LineMiddle, LineTop

# Example

Delphi
FontBaseline := 1.8;
C++Builder
rp->FontBaseline = 1.8;

# FontBottom property (read/write)

# Declaration

property FontBottom: double;

### Default

see ResetLineHeight

# Category

Position

Component/Class TBaseReport

## Description

Returns or sets the bottom of the line font

### See also

FontBaseline, FontTop, LineBottom, LineMiddle, LineTop

# Example

Delphi
FontBottom := 2.0;
C++Builder
rp->FontBottom = 2.0;

# FontCharset property (read/write)

### Declaration

property FontCharset: byte;

### Default

DEFAULT\_CHARSET

# Category

Font

#### Component/Class TBaseReport

### Description

Allows you to change the character set of the current font. Other values can be found in the Windows API help under LOGFONT

# Example

Delphi
SetFont( 'Wingdings', 10 );
FontCharSet := SYMBOL\_CHARSET;

### C++Builder

```
rp->SetFont( "Wingdings", 10 );
rp->FontCharSet = SYMBOL_CHARSET;
```

# FontColor property (read/write)

### Declaration

property FontColor: TColor;

## Default

clBlack

# Category

Font

#### Component/Class TBaseReport

#### Description

Returns or sets the font color.

### See also

Other FontXxxx properties, SetFont, TColor

### Example

Delphi
FontColor := clRed;
Print('This text is in red.');

```
rp->FontColor = clRed;
rp->Print("This text is in red.");
```

# FontHandle property (read only)

# Declaration

property FontHandle: HFont;

## Category

Font

# Component/Class

TBaseReport

# 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*.

# FontHeight property (read/write)

# Declaration

property FontHeight: double;

# Default

see ResetLineHeight

# Category

Font

#### Component/Class TBaseReport

# 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

Other FontXxxx properties, <u>AscentHeight</u>, <u>DescentHeight</u>, <u>LineHeight</u>

# Example

Delphi
FontHeight := 0.25;
C++Builder
rp->FontHeight = 0.25;

# FontName property (read/write)

# Declaration

property FontName: string;

## Default

'System'

#### Category Font

Component/Class TBaseReport

# Description

Returns or sets the current font name.

# See also

Other FontXxxx properties, SetFont

# Example

Delphi
FontName := 'Times New Roman';
C++Builder
rp->FontName = "Times New Roman";

# FontPitch property (read/write)

# Declaration

property FontPitch: TFontPitch;

# Default

fpDefault

#### Category Font

Component/Class TBaseReport

# Description

Returns or sets the pitch setting for the current font. The normal setting of *fpDefault* will use the font's normal pitch. *fpFixed* will attempt to convert the font to a fixed-width font and *fpVariable* will attempt to convert the font to a variable-width font. Setting a font to a pitch other than what it was designed for may have no effect or may cause another font to be substituted in its place.

### See also

Other FontXxxx properties, SetFont

```
Delphi
FontPitch := fpVariable;
C++Builder
rp->FontPitch = fpVariable;
```

# FontRotation property (read/write)

### Declaration

property FontRotation: integer;

### Default

0

# Category

Font

### Component/Class TBaseReport

### 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

Other FontXxxx properties

# Example

```
Delphi
FontRotation := 45;
Print('This text is at 45 degrees');
FontRotation := 0;
Print('This is normal text');
```

```
rp->FontRotation = 45;
rp->Print("This text is at 45 degrees");
rp->FontRotation = 0;
rp->Print("This is normal text");
```

# Fonts property (read only)

# Declaration

property Fonts: TStrings;

# Default

(list of fonts supported by the default printer)

# Category

Printer

### Component/Class TBaseReport

# Description

This property will return a TStringList containing all of the fonts supported by the current printer.

# See also

FontName, SetFont, TStrings

# Example

Display the supported fonts in a TComboBox

### Delphi

Combobox1.Items := RvNDRWriter1.Fonts; C++Builder ComboBox1->Items = RvNDRWriter1->Fonts;

# FontSize property (read/write)

# Declaration

property FontSize: double;

### Default

10

# Category

Font

### Component/Class TBaseReport

Description

Returns or sets the point size of the current font.

# See also

Other FontXxxx properties, SetFont

# Example

```
Delphi
FontSize := 8;
Print('Small');
FontSize := 36;
Print('Large');
```

```
rp->FontSize = 8;
rp->Print("Small");
rp->FontSize = 36;
rp->Print("Large");
```

# FontTop property (read/write)

## Declaration

property FontTop: double;

## Default

see ResetLineHeight

# Category

Position

Component/Class TBaseReport

# Description

Returns or sets the top of the line font

# See also

Other FontXxxx properties, LineBottom, LineMiddle, LineTop

# Example

Place the top of the text at 2.25"

# Delphi

FontTop := 2.25; C++Builder rp->FontTop = 2.25;

# FontWidth property (read/write)

# Declaration

property FontWidth: double;

### Default

0

# Category

Font

#### Component/Class TBaseReport

# 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

<u>FontSize</u>

# Example

set average character width to 1/4 inch

# **Delphi** FontWidth := 0.25;

**C++Builder** rp->FontWidth = 0.25;

# FrameMode property (read/write)

## Declaration

property FrameMode: TFrameMode;

### Default

fmInside

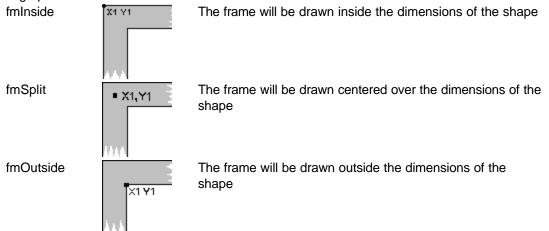
### Category

Graphics

#### Component/Class TBaseReport

# 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.



**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

Ellipse, Rectangle

### Example

Delphi
FrameMode := fmOutside;
C++Builder
rp->FrameMode = fmOutside;

# FrameRect method

# Declaration

procedure FrameRect(const Rect: TRect);

## Category

Graphics

# Component/Class

TBaseReport

# Description

This method draws the rectangle *Rect* using the current brush to draw the border of the rectangle. *FrameRect* does not fill the rectangle with the current brush.

# See also

CreateRect, TRect

# Example

```
Delphi
RvNDRWriter1.FrameRect( CreateRect( 1.0,1.0, 2.0,3.0 ) );
C++Builder
RvNDRWriter1->FrameRect( rp->CreateRect(1.0,1.0,2.0,3.0) );
```

# FreeSaved method

### Declaration

procedure FreeSaved;

## Category

Memo

#### Component/Class TMemoBuf

# 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

RestoreBuffer, SaveBuffer

```
Delphi
MemoBuf1.FreeSaved;
C++Builder
MemoBuf1->FreeSaved();
```

# GetMemoLine method

# Declaration

function GetMemoLine( MemoBuf: TMemoBuf; var EOL: boolean): string;

# Category

Memo

# Component/Class

TBaseReport

# 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

MemoLines, PrintMemo, TMemoBuf

# Example

```
Delphi
Println(GetMemoLine(MemoBuf, EOL));
C++Builder
rp->Println(rp->GetMemoLine(MemoBuf, EOL));
```

# GetNextLine method

# Declaration

function GetNextLine(var EOL: boolean): string;

### Category

Memo

### Component/Class

TBaseReport

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

### See also

MemoLines, PrintMemo, TMemoBuf

```
Delphi
Println(GetNextLine(EOL));
C++Builder
rp->Println(rp->GetNextLine(EOL));
```

# GetReportCategoryList method

# Declaration

procedure GetReportCategoryList(ReportList: TStrings; Categories: string); FullName: boolean);

# Category

Rave

# Component/Class

TRvProject

# 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

SelectReport

# GetReportList method

# Declaration

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

# Category

Rave

# Component/Class

**T**RvProject

### 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

SelectReport

# GetTab method

# Declaration

function GetTab(Index: integer): PTab;

# Category

Tabs

# Component/Class

TBaseReport

# 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

TabIndex

# GotoFooter method

### Declaration

procedure GotoFooter;

# Category

Position

Component/Class TBaseReport

### Description

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

# See also

MarginBottom, PrintFooter, SectionBottom

# Example

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

### C++Builder

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

# GotoHeader method

### Declaration

procedure GotoHeader;

Category

Position

Component/Class TBaseReport

### Description

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

### See also

MarginTop, PrintHeader, SectionTop

### Example

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

### C++Builder

```
RvNDRWriter1->GotoHeader();
RvNDRWriter1->Print("Line just below SectionTop");
```

# GotoXY method

#### Declaration

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

### Category

Position

#### Component/Class TBaseReport

# Description

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

### See also

XPos, YPos

# Example

This code shows how to position the output at specific coordinates.

#### Delphi

```
GotoXY(1.0,8.5);
Print('Text at 1.0,8.5');
```

```
rp->GotoXY(1.0,8.5);
rp->Print("Text at 1.0,8.5");
```

# GraphicFieldToBitmap method

#### Declaration

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

#### Category

Graphics

#### Component/Class TBaseReport

#### 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

PrintBitmap, PrintBitmapRect, TGraphicField

#### Example

Convert and print a TGraphicField

#### Delphi

```
Bitmap := TBitmap.Create;
GraphicFieldToBitmap(TablelGraphic,Bitmap);
PrintBitmapRect(5.375,3.5,7.375,5.5,Bitmap);
Bitmap.Free;
```

```
Bitmap := new Graphic::TBitmap();
rp->GraphicFieldToBitmap(Table1Graphic,Bitmap);
rp->PrintBitmapRect(5.375,3.5,7.375,5.5,Bitmap);
delete Bitmap;
```

## GridHoriz property (read/write/published)

#### Declaration

property GridHoriz: double;

#### Default

0.0

Category Preview

Component/Class TRvSystem

#### 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

GridPen, GridVert

#### Example

Delphi
GridHoriz := 0.25;
C++Builder
GridHoriz = 0.25;

# GridPen property (read/write/published)

#### Declaration

property GridPen: TPen;

#### Default

(Standard Pen)

#### Category Preview

Component/Class

TRVSyster

#### Description

This property defines the pen used to draw the grid defined by GridVert and GridHoriz.

#### See also

GridHoriz, GridVert, RulerType, TPen

```
Delphi
GridPen.Color := clAqua;
C++Builder
GridPen->Color = clAqua;
```

# GridVert property (read/write/published)

#### Declaration

property GridVert: double;

#### Default

0.0

Category

Preview

Component/Class TBaseReport

#### 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

GridHoriz, GridPen

#### Example

Delphi
GridVert := 0.5;
C++Builder
GridVert = 0.5;

# Height property (read only)

#### Declaration

property Height: double;

#### Category

BarCode

#### Component/Class

TRpBarsBase

#### 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

BarHeight, PrintReadable

#### Example

#### Delphi

```
TotalBarHeight := Height;
if TotalBarHeight > 1.0 then begin
  BarHeight := 1.0; {set total height to 1.0 inches}
end; { if}
```

```
TotalBarHeight = rp->Height;
if (TotalBarHeight > 1.0) {
  BarHeight = 1.0; / set total height to 1.0 inches
}/ if
```

# Home method

#### Declaration

procedure Home;

#### Category

Position

#### Component/Class

TBaseReport

#### Description

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

### Example

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

```
rp->SetFont("Arial",10);
rp->Home();
Print("Text in the Home position");
```

## IgnoreFileSettings property (read/write)

#### Declaration

property IgnoreFileSettings: boolean

#### Default

false

Category Misc

- -

Component/Class TRvRenderPrinter

#### 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

ShowPrintDialog, ShowPrinterSetupDialog

#### Example

#### Delphi

```
if RvRenderPrinter1.ShowPrinterSetupDialog then begin
    RvRenderPrinter1.IgnoreFileSettings := True;
    RvRenderPrinter1.Execute;
end; {if}
```

#### C++Builder

```
if (RvRenderPrinter1->ShowPrinterSetupDialog()) {
    RvRenderPrinter1->IgnoreFileSettings = true;
    RvRenderPrinter1->Execute();
}/ if
```

## ImageQuality property (read/write)

#### Declaration

property ImageQuality: TImageQualityRange read FImageQuality write FImageQuality

#### Default

JPG'S image quality set to 90

#### Category

Render

#### Component/Class

TRpRender

#### 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

<u>MetafileDPI</u>

# InsertMemoBuf method

#### Declaration

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

#### Category

Memo

#### Component/Class TMemoBuf

Description

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

#### See also

AppendMemoBuf

#### Example

```
Delphi
MemoBuf1.InsertMemoBuf(10,MemoBuf2);
C++Builder
MemoBuf1->InsertMemoBuf(10,MemoBuf2);
```

## Insert method

#### Declaration

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

#### Category

Memo

#### Component/Class TMemoBuf

#### Description

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

#### See also

Append

```
Delphi
MemoBuf.Insert(0,'This text will now be first');
C++Builder
MemoBuf->Insert(0,"This text will now be first");
```

# IsValidChar method

#### Declaration

function IsValidChar( Ch: char ): boolean;

#### Category

BarCode

#### Component/Class TRpBarsBase

#### Description

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

#### Example

following will return false because 2of5 only support numbers

#### Delphi

Code2of5.IsValidCar('A')
C++Builder
Code2of5->IsValidCar('A')

## Italic property (read/write)

#### Declaration

property Italic: boolean;

#### Default

false

#### Category

Font

#### Component/Class

TBaseReport

#### Description

This property returns or sets the italic attribute for the current font.

#### See also

Bold, Strikeout, Underline

#### Example

Delphi

```
Italic := true;
Print('Italic Text');
Italic := false;
```

#### C++Builder

rp->Italic = true; rp->Print("Italic Text"); rp->Italic = false;

# Justify property (read/write)

#### Declaration

property Justify: TPrintJustify;

#### Default

pjLeft

## Category

Memo

#### Component/Class TMemoBuf

#### Description

This property sets the justification that *PrintMemo* will use when printing the memo buffer. Valid values are

pjBlock pjCenter pjLeft pjRight

#### See also

PrintMemo

```
Delphi
MemoBuf.Justify := pjBlock; { Set block justification }
C++Builder
MemoBuf->Justify = pjBlock; / Set block justification
```

# LastPage property (read/write/published)

#### Declaration

property LastPage: integer;

#### Default

9999

Category

Control

Component/Class TBaseReport

#### 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 *PageInvalid* will be true.

See also

PageInvalid

#### Example

Print only pages 3 through 5

#### Delphi

RvNDRWriter1.FirstPage := 3; RvNDRWriter1.LastPage := 5;

#### C++Builder

RvNDRWriter1->FirstPage = 3; RvNDRWriter1->LastPage = 5;

# Left property (read/write)

#### Declaration

property Left: double;

#### Default

XPos

#### Category

BarCode

#### Component/Class TRpBarsBase

#### 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

BarCodeJustify, Center, Position, Right

#### Example

start at 4.5 inches from left side

**Delphi** Left := 4.5; **C++Builder** Left = 4.5;

## LeftWaste property (read only)

#### Declaration

property LeftWaste: double;

#### Category

Printer

#### Component/Class

TBaseReport

#### 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

BottomWaste, MarginLeft, RightWaste, TopWaste

#### Example

Don't output in the printer waste regions

#### Delphi

```
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 }</pre>
```

```
if (rp->MarginLeft < rp->LeftWaste) {
  rp->MarginLeft = rp->LeftWaste;
}/ if
if (rp->MarginRight < rp->RightWaste) {
  rp->MarginRight = rp->RightWaste;
}/ if
if (rp->MarginTop < rp->TopWaste) {
  rp->MarginTop = rp->TopWaste;
}/ if
if (rp->MarginBottom < rp->BottomWaste) {
  rp->MarginBottom = rp->BottomWaste;
}/ if
```

## LF method

#### Declaration

procedure LF;

#### Category

Position

#### Component/Class

TBaseReport

#### 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

CR, LineHeight, LineNum, NewLine, SectionBottom, SectionTop

#### Example

```
Delphi
RvNDRWriter1.LF;
C++Builder
RvNDRWriter1->LF();
```

## LineBottom property (read/write)

#### Declaration

property LineBottom: double;

#### Default

(Bottom of the current line)

#### Category

Position

#### Component/Class TBaseReport

See also

FontBaseline, FontBottom, FontTop, LineMiddle, LineTop

#### Example

Place the text right on the bottom of the section

#### Delphi

```
LineBottom := SectionBottom;
C++Builder
rp->LineBottom = rp->SectionBottom;
```

# LineHeight property (read/write)

#### Declaration

property LineHeight: double;

#### Category

Position

#### Component/Class

TBaseReport

#### 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 *IhmUser*.

#### See also

LineHeightMethod

#### Example

Save current line height to a temporary variable

#### Delphi

CurrHeight := RvNDRWriter1.LineHeight C++Builder CurrHeight = RvNDRWriter1->LineHeight

# LineHeightMethod property (read/write/published)

#### Declaration

property LineHeightMethod: TLineHeightMethod;

#### Default

IhmLinesPerInch, IhmFont for TRvSystem

#### Category

Position

#### Component/Class

. TBaseReport

#### 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

LinesPerInch

```
Delphi
RvNDRWriter1.LineHeightMethod := lhmFont;
C++Builder
RvNDRWriter1->LineHeightMethod = lhmFont;
```

# LineMiddle property (read/write)

#### Declaration

property LineMiddle: double;

#### Default

(Middle of current line)

#### Category

Position

#### Component/Class TBaseReport

#### 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

FontBaseline, FontBottom, FontTop, LineBottom, LineTop

Example

Delphi LineMiddle := 2.0; C++Builder rp->LineMiddle = 2.0;

# LineNum property (read/write)

#### Declaration

property LineNum: integer;

#### Default

1

Category

Position

Component/Class TBaseReport

#### 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 *IhmFont*. *LineNum* may not represent the actual line number if the report is jumping around the page instead of calling *Prints* and *Printlns*.

#### See also

LineHeight, LineHeightMethod

#### Example

```
Delphi
with RvNDRWriter1 do
    if Odd(LineNum) then begin
        TabShade := 0;
    end else begin
        TabShade := 15;
    end; { if }
end; { with }
```

```
if ((rp->LineNum 2) == 1) {
    rp->TabShade = 0;
    }
    else {
        rp->TabShade = 15;
    }/ else
```

# LinesLeft method

#### Declaration

function LinesLeft: integer;

#### Category

Position

#### Component/Class

TBaseReport

#### Description

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

#### See also

ColumnLinesLeft, SectionBottom

#### Example

```
Delphi
if RvNDRWriter1.LinesLeft < 3 then begin
    RvNDRWriter1.NewPage;
end; { if }</pre>
```

#### C++Builder

```
if (RvNDRWriter1->LinesLeft() < 3) {
  RvNDRWriter1->NewPage();
  }/ if
```

## LinesPerInch property (read/write/published)

#### Declaration

property LinesPerInch: integer;

Default

6

Category

Position

Component/Class TBaseReport

#### Description

This property will return or set the number of lines per inch if the *LineHeightMethod* property is equal to *IhmLinesPerInch*.

#### See also

**LineHeightMethod** 

```
Delphi
RvNDRWriter1.LineHeightMethod := lhmLinesPerInch;
C++Builder
RvNDRWriter1->LineHeightMethod = lhmLinesPerInch;
```

# LineTo method

#### Declaration

procedure LineTo(X,Y: double);

#### Category

Graphics

#### Component/Class

TBaseReport

#### 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

<u>MoveTo</u>

#### 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}

rp->MoveTo(	1.0,	1.0	);
rp->LineTo(	3.0,	3.0	);
rp->MoveTo(	1.0,	3.0	);
rp->LineTo(	3.0,	1.0	);

# LineTop property (read/write)

#### Declaration

property LineTop: double;

#### Default

(Top of the current line)

#### Category

Position

Component/Class TBaseReport

Description Returns or sets the top of the text line

#### See also

FontBaseline, FontTop, LineBottom, LineMiddle

#### Example

Place the top of the line at 4.0"

#### Delphi

LineTop := 4.0; C++Builder LineTop = 4.0;

## LoadDesigner property (read/write)

#### Declaration

property LoadDesigner: boolean;

#### Default

false

#### Category

Rave

#### Component/Class

TRvProject

#### 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 EUDL license. See the Nevrona website at http://www.nevrona.com for more information on obtaining an EUDL license.

See also

DLLFile, Open

# LoadFromFile method

#### Declaration

function LoadFromFile( FileName: String);

#### Category

Memo

#### Component/Class TMemoBuf

#### Description

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

#### See also

LoadFromStream, RTFLoadFromFile, SaveToStream

#### Example

```
Delphi
MemoBufl.LoadFromFile('Letter.Txt');
C++Builder
MemoBufl->LoadFromFile("Letter.Txt");
```

# LoadFromFile method

#### Declaration

procedure LoadFromFile(FileName: string);

#### Category

Rave

#### Component/Class TRvProject

#### Description

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

#### See also

LoadFromStream, SaveToFile, SaveToStream

```
Delphi
RvProject1.LoadFromFile('Project1.Rav');
C++Builder
RvProject1->LoadFromFile("Project1.Rav");
```

# LoadFromStream method

#### Declaration

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

#### Category

Memo

#### Component/Class TMemoBuf

#### Description

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

#### See also

SaveToStream

#### Example

```
Delphi
MemoBuf1.LoadFromStream( MyStream, StreamSize );
C++Builder
MemoBuf1->LoadFromStream( MyStream, StreamSize );
```

# LoadFromStream method

```
Declaration
```

procedure LoadFromStream(Stream: TStream);

#### Category

Rave

#### Component/Class TRvProject

TIXVETUJECI

## Description

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

#### See also

LoadFromFile, SaveToFile, SaveToStream

#### Example

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

## LoadRaveBlob method

#### Declaration

procedure LoadRaveBlob(Stream: TStream);

#### Category

Rave

#### Component/Class

TRvProject

#### 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

ClearRaveBlob, RaveBlobDateTime, SaveRaveBlob, StoreRAV

#### Example

```
Delphi
RvProject1.LoadRaveBlob( MyStream );
C++Builder
RvProject1->LoadRaveBlob( MyStream );
```

## LocalFilter property (read/write/published)

#### Declaration

property LocalFilter: Boolean;

#### Default

TRvQueryConnection and TRvTableConnection

#### Category

Rave

False

#### Component/Class

TRvCustomConnection

#### Description

This property will determine whether filtering is done locally inside of the data connection component or whether it will rely on the filtering capabilites 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).

```
Delphi
RvCustomConnection1.LocalFilter := True;
C++Builder
RvCustomConnection1->LocalFilter + True;
```

# Macro method

#### Declaration

function Macro(MacroID: TMacroID): string;

#### Default

6

## Category

Misc

#### Component/Class TBaseReport

#### 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 MacroIDs see the type definition of TMacroID.

#### See also

MacroData, TMacroID

#### Example

Print the current page and total pages

#### Delphi

```
PrintRight(Macro(midCurrentPage) + ' of ' +
Macro(midTotalPages), 8.0);
```

```
rp->PrintRight(rp->Macro(midCurrentPage) + " of " +
rp->Macro(midTotalPages), 8.0);
```

# MacroData property (read/write)

#### Declaration

property MacroData: TStrings;

#### Default

empty list

#### Category

Printing

#### Component/Class TBaseReport

#### Description

This property sets or returns the user-defined macro string in a list of strings for midUser01 to midUser20

#### See also

Macro, TMacroID, TStrings

#### Example

Add current user name for Macro(midUser01)

#### Delphi

MacroData.Add(UserName); RvRenderPrinter1.Execute;

```
rp->MacroData->Add(UserName);
RvRenderPrinter1->Execute();
```

## MarginBottom property (read/write/published)

#### Declaration

property MarginBottom: double;

#### Default

0.0

Category Position

1 0311011

Component/Class TBaseReport

#### 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

MarginLeft, MarginRight, MarginTop, section properties, ResetSection

#### Example

This code shows how to set these properties. Also see PrintFooter

#### Delphi

```
MarginLeft := 0.5;
MarginRight := 0.5;
MarginTop := 0.5;
MarginBottom := 1.0;
```

```
rp->MarginLeft := 0.5;
rp->MarginRight := 0.5;
rp->MarginTop := 0.5;
rp->MarginBottom := 1.0;
```

# MarginLeft property (read/write/published)

#### Declaration

property MarginLeft: double;

#### Default

0.0

Category Position

Component/Class TBaseReport

#### 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

MarginBottom, section properties, ResetSection

#### Example

This code shows how to set these properties. Also see PrintFooter

Delphi
MarginLeft := 0.5;
C++Builder
rp->MarginLeft := 0.5;

## MarginMethod property (read/write/published)

#### Declaration

property MarginMethod: TMarginMethod;

#### Default

mmFixed

Category

Preview

#### Component/Class

TRvRenderPreview

#### 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

**MarginPercent** 

#### Example

```
Imple
Delphi
RvRenderPreview1.MarginMethod := mmScaled;
C++Builder
RvRenderPreview1->MarginMethod = mmScaled;
```

## MarginPercent property (read/write/published)

#### Declaration

property MarginPercent: double;

#### Default

0.0

#### Category

Preview

#### Component/Class

TRvRenderPreview

#### 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

MarginMethod

#### Example

Set a 1bord/er

#### Delphi

```
RvRenderPreview1.MarginPercent := 1.0;
C++Builder
RvRenderPreview1->MarginPercent = 1.0;
```

## MarginRight property (read/write/published)

#### Declaration

property MarginRight: double;

#### Default

0.0

Category Position

Component/Class

TBaseReport

#### 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

MarginBottom, MarginLeft, MarginTop, section properties, ResetSection

#### Example

```
Delphi
MarginRight := 0.5;
C++Builder
rp->MarginRight := 0.5;
```

# MarginTop property (read/write/published)

#### Declaration

property MarginTop: double;

#### Category

Position

#### Component/Class

TBaseReport

#### 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

MarginBottom, MarginLeft, MarginRight, section properties, ResetSection

#### Example

Delphi
MarginTop := 0.5;
C++Builder
rp->MarginTop := 0.5;

# MaxCopies property (read/write/published)

#### Declaration

property MaxCopies: longint;

#### Default

(maximum number of copies supported by the default printer)

#### Category

Printer

#### Component/Class TBaseReport

## Description

This property returns the maximum number of copies supported by the current printer.

#### See also

<u>Copies</u>

#### Example

```
Delphi
if MaxCopies = 1 then begin
  Copies := 1;
end; { if }
```

```
if (rp->MaxCopies == 1) {
    rp->Copies = 1;
    }/ if
```

## MaxSize property (read/write)

#### Declaration

property MaxSize: longint;

#### Default

0

Category Memo

Component/Class

#### TMemoBuf

#### 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

BufferInc, Size

#### Example

Allocate at least 1000 characters

```
Delphi
MemoBuf.MaxSize := 1000;
C++Builder
MemoBuf->Memo = Memol;
```

## Memo property (read/write)

#### Declaration

property Memo: TMemo;

Category Memo

Component/Class TMemoBuf

#### Description

This property will assign the contents of a TMemo component to a memo buffer.

#### See also

Field, Text, TMemo component in Delphi help

#### Example

Copy Memo1 into MemoBuf

#### Delphi

```
MemoBuf.Memo := Memol;
C++Builder
MemoBuf->Memo = Memol;
```

## MemoHeightLeft method

#### Declaration

function MemoHeightLeft: double;

Category

Memo

Component/Class

TMemoBuf

#### 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

ConstraintHeightLeft, MemoLinesLeft, PrintEnd, PrintMemo, PrintStart, TMemoBuf

#### Example

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

#### C++Builder

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

## MemoLinesLeft method

#### Declaration

function MemoLinesLeft: longint;

#### Category

Memo

#### Component/Class

TMemoBuf

#### 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

PrintEnd, PrintMemo, PrintStart, MemoHeightLeft, TMemoBuf

#### Example

Delphi
MemoBuf.BaseReport := Sender as TBaseReport;
LinesLeft := MemoBuf.MemoLinesLeft;

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

## MemoLines method

#### Declaration

function MemoLines(MemoBuf: TMemoBuf): longint;

#### Category

Memo

#### Component/Class

TBaseReport

#### 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

PrintEnd, PrintMemo, PrintStart, TMemoBuf

#### Example

Save number of lines needed to print memo

#### Delphi

```
LinesLeft := RvNDRWriterl.MemoLines(MyMemo);
C++Builder
LinesLeft = RvNDRWriterl->MemoLines(MyMemo);
```

## MetafileDPI property (read/write)

#### Declaration

property MetafileDPI: boolean; read FMetafileDPI write FMetafileDPI

#### Default

300

#### Category

Render

#### Component/Class

TRpRender

#### 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

ImageQuality

# Monochrome property (read/write/published)

#### Declaration

property Monochrome: boolean;

#### Default

false

Category Preview

TICVICV

Component/Class TRvRenderPreview

#### 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

ShadowDepth

#### Example

```
Delphi
RvRenderPreview1.Monochrome := true;
C++Builder
RvRenderPreview1->Monochrome = true;
```

# MoveTo method

#### Declaration

procedure MoveTo(X,Y: double);

#### Category

Graphics

Component/Class TBaseReport

#### Description

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

#### See also

LineTo

```
Delphi
RvNDRWriter1.MoveTo( NewX, NewY );
C++Builder
RvNDRWriter1->MoveTo( NewX, NewY );
```

# NewColumn method

#### Declaration

procedure NewColumn;

#### Category

Control

#### Component/Class

TBaseReport

#### 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

SetColumns, SetColumnWidth

#### Example

Delphi
RvNDRWriter1.NewColumn;
C++Builder
RvNDRWriter1->NewColumn();

# NewLine method

#### Declaration

procedure NewLine;

#### Category

Position

#### Component/Class TBaseReport

#### Description

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

#### See also

ColumnStart, CR, LE, ResetTabs

#### Example

Delphi
RvNDRWriter1.NewLine;
C++Builder
RvNDRWriter1->NewLine();

## NewPage method

#### Declaration

procedure NewPage;

#### Category

Control

#### Component/Class

TBaseReport

#### 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

AbortPage, OnNewPage, OnPrintHeader, OnPrintFooter

#### Example

Delphi
RvNDRWriter1.NewPage;
C++Builder
RvNDRWriter1->NewPage();

# NextPage method

#### Declaration

procedure NextPage;

#### Category

Preview

#### Component/Class

**T**RvRenderPreview

#### 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

CurrentPage, PrevPage, OnPageChange

```
Delphi
RvRenderPreview1.NextPage;
C++Builder
RvRenderPreview1->NextPage();
```

# NoBufferLine property (read/write)

#### Declaration

property NoBufferLine: boolean;

#### Default

false

Category

Graphics

Component/Class TBaseReport

#### 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

<u>LineTo, MoveTo</u>

#### Example

turn off line buffering

#### Delphi

```
RvNDRWriter1.NoBufferLine := true;
C++Builder
RvNDRWriter1->NoBufferLine = true;
```

## NoCRLF property (read/write)

#### Declaration

property NoCRLF: boolean;

Default

false

Category Memo

Component/Class

TMemoBuf

#### Description

This property will control whether *PrintMemo* finishes with a carriage-return linefeed (if false) or not (if true).

#### See also

<u>PrintMemo</u>

#### Example

Don't do a NewLine after PrintMemo()

#### Delphi

MemoBuf.NoCRLF := true; C++Builder MemoBuf->NoCRLF = true;

# NoNewLine property (read/write)

# Declaration

property NoNewLine: boolean;

#### Default

false

Category Memo

Component/Class TMemoBuf

# Description

Prevents the writing of an extra new line after the memo has been printed.

# See also

PrintMemo

# Example

Delphi
MemoBuf.NowNewLine := true;
C++Builder
MemoBuf->NowNewLine = true;

# NoNTColorFix property (read/write)

# Declaration

property NoNTColorFix: boolean;

# Default

false

# Category

Printer

#### Component/Class TBaseReport

# 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

**FontColor** 

# Example

Disable NT color conversion

# Delphi

```
NoNTColorFix := true;
C++Builder
NoNTColorFix = true;
```

# NoPrinterPageHeight property (read/write)

# Declaration

property NoPrinterPageHeight: double;

# Default

11.0

Category Printer

Component/Class TBaseReport

# 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

**NoPrinters** 

# Example

See NoPrinters

# NoPrinterPageWidth property (read/write)

# Declaration

property NoPrinterPageWidth: double;

# Default

8.5

# Category

Printer

# Component/Class TBaseReport

# 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

**NoPrinters** 

# Example

See NoPrinters

# NoPrinters method

# Declaration

function NoPrinters: boolean;

#### Category

Printer

# Component/Class

TBaseReport

# 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

NoPrinterPageHeight, NoPrinterPageWidth

# Example

Set up for landscape paper

#### Delphi

```
if NoPrinters then begin
   NoPrinterPageHeight := 8.5;
   NoPrinterPageWidth := 11.0;
end; { if }
```

#### C++Builder

```
if (rp->NoPrinters()) {
   rp->NoPrinterPageHeight = 8.5;
   rp->NoPrinterPageWidth = 11.0;
}/ if
```

# OnAfterClose event (read/write/published)

Declaration procedure OnAfterClose( Sender: Tobject );

Category Rave

Component/Class TRvProject

Description

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

See also

Active, Close, OnAfterOpen, OnBeforeClose, OnBeforeOpen, Open

# OnAfterOpen event (read/write/published)

# Declaration

procedure OnAfterOpen( Sender: Tobject );

# Category

Rave

#### Component/Class TRvProject

I I VF I UJECI

# Description

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

# See also

Active, Close, OnAfterClose, OnBeforeClose, OnBeforeOpen, Open

# **OnAfterPrint** event (read/write/published)

# Declaration

procedure OnAfterPrint(Sender: TObject)

# Category

Control

# Component/Class

TBaseReport

# 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

Execute, OnBeforePrint

# Example

```
Delphi
procedure TReportForm.AfterPrintReport5(Sender: TObject);
begin { AfterPrintReport5 }
   CustomerTable.Close;
end; { AfterPrintReport5 }
```

```
void __fastcall TReportForm:: AfterPrintReport5 (TObject *Sender)
{
   CustomerTable->Close();
}
```

# **OnBeforeClose** event (read/write/published)

# Declaration

procedure OnBeforeClose( Sender: Tobject );

# Category

Rave

#### Component/Class TRvProject

# Description

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

# See also

Active, Close, OnAfterClose, OnAfterOpen, OnBeforeOpen, Open

# **OnBeforeOpen** event (read/write/published)

# Declaration

procedure OnBeforeOpen( Sender: Tobject );

# Category

Rave

# Component/Class

TRvProject

# Description

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

# See also

Active, Close, OnAfterClose, OnAfterOpen, OnBeforeClose, Open

# **OnBeforePrint** event (read/write/published)

# Declaration

procedure OnBeforePrint(Sender: TObject);

#### Category

Control

#### Component/Class

TBaseReport

#### 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

Execute, OnAfterPrint

# Example

#### Delphi

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

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

# OnCompress property (read/write)

# Declaration

property OnCompress: TCompressEvent;

Default

' ' empty

Category Render

Component/Class TRpRender

# 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

**UseCompression** 

# Example

Typcially, the code defined inside the OnCompress event will be something similar to this:

# Delphi

```
with TCompressionStream.Create(clMax, OutStream) do try
   CopyFrom(InStream, InStream.Size);
finally
   Free
end; { with }
```

# OnCreate event (read/write/published)

# Declaration

procedure OnCreate(Sender: TObject);

Category

Rave

# Component/Class

TRvProject

# 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

<u>OnDestroy</u>

# OnDecodeImage event (read/write/published)

#### Declaration

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

#### Category

Graphics

# Component/Class

TBaseReport

# Description

This event is called when Rave needs to convert image data (created from the PrinitImageRect 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

PrintlmageRect

# Example

```
Delphi
var
  Image: TJPEGImage;
 Format: word;
 Data: THandle;
 Palette: HPalette;
if ImageType = 'JPG' then begin
                               / Create a TJPEGImage class
  Image := TJPEGImage.Create;
  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 }
```

```
if (ImageType == "JPG") {
    Image = new TJPEImage(); / 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);
}/ if
```

# OnDesignerSave event (read/write/published)

# Declaration

procedure OnDesignerSave(Sender: TObject);

# Category

Rave

# Component/Class

TRvProject

# 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

OnDesignerSaveAs, OnDesignerShow, SaveToStream

# OnDesignerSaveAs event (read/write/published)

# Declaration

procedure OnDesignerSaveAs(Sender: TObject);

# Category

Rave

# Component/Class

TRvProject

# 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

OnDesignerSave, OnDesignerShow, SaveToStream

# **OnDesignerShow** event (read/write/published)

# Declaration

procedure OnDesignerShow(Sender: TObject);

# Category

Rave

# Component/Class

**TRvProject** 

# 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

**OnDesignerSave** 

# **OnDestroy** event (read/write/published)

# Declaration

procedure OnDestroy(MyPrinter: TRave);

# Category

Rave

#### **Component/Class TRvProject**

# 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

**OnCreate** 

# **OnEOF** event (read/write/published)

# Declaration

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

# Category

Rave

# Component/Class

TRvCustomConnection

# 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

OnFirst, OnNext

# **OnFirst** event (read/write/published)

# Declaration

procedure OnFirst(Connection: TRvCustomConnection);

Category Rave

Nave

Component/Class

TRvCustomConnection

# 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

OnEOE, OnNext

# OnGetCols event (read/write/published)

# Declaration

procedure OnGetCols(Connection: TRvCustomConnection);

# Category

Rave

# Component/Class

TRvCustomConnection

# 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

<u>OnGetRow</u>

# OnGetRow event (read/write/published)

# Declaration

procedure OnGetRow(Connection: TRvCustomConnection);

# Category

Rave

# Component/Class

TRvCustomConnection

# 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

OnFirst, OnNext

# OnGetSorts event (read/write/published)

# Declaration

procedure OnGetSorts(Connection: TRvCustomConnection);

# Category

Rave

# **Component/Class**

TRvCustomConnection

# 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

<u>OnSetSort</u>

# OnNewColumn event (read/write/published)

#### Declaration

procedure OnNewColumn(Sender: TObject);

#### Category

Control

# Component/Class

TBaseReport

#### 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

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;
```

```
void __fastcall TReportForm:: OnNewColumnReport10 (TObject *Sender)
{
    TBaseReport* rp = dynamic_cast<TBaseReport*>(Sender);
    rp->Underline = true;
    rp->Println("Column Titles");
    rp->Underline = false;
}
```

# OnNewPage event (read/write/published)

# Declaration

procedure OnNewPage(Sender: TObject);

#### Category

Control

#### Component/Class

TBaseReport

#### Description

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

#### See also

NewPage, SelectBin

# Example

```
Delphi
procedure TRpForm.RvNDRWriterlNewPage(Sender: TObject);
begin
  with Sender as TBaseReport do begin
    PrintBitmapRect(0.5,0.5,1.20,1.20,Logo);
    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;
```

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

# **OnNext** event (read/write/published)

# Declaration

procedure OnNext(Connection: TRvCustomConnection);

# Category

Rave

# Component/Class

TRvCustomConnection

# 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

OnEOE, OnFirst

# OnOpen event (read/write/published)

#### Declaration

procedure OnOpen(Connection: TRvCustomConnection);

Category

Rave

# Component/Class

. TRvCustomConnection

#### 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

**OnRestore** 

# OnPageChange event (read/write/published)

# Declaration

procedure OnPageChange(Sender: TObject);

#### Category

Preview

#### Component/Class

**TRvRenderPreview** 

#### 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

NextPage, PrevPage, PrintPage

# Example

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

```
void __fastcall TPreForm::RvRenderPreviewlPageChange(TObject
*Sender)
{
   TBaseReport* rp = dynamic_cast<TBaseReport*>(Sender);
   PageEdit->Text = IntToStr(rp->CurrentPage);
   PageLabel->Caption = "Page " + IntToStr(rp->CurrentPage -
      rp->FirstPage + 1) + " of " + IntToStr(RvRenderPreviewl->Pages);
}
```

# OnPreviewSetup event (read/write/published)

# Declaration

procedure OnPreviewSetup( Sender: TObject );

# Category

Preview

# Component/Class

TRvSystem

# 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

**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 }
```

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

# OnPreviewShow event (read/write/published)

# Declaration

procedure OnPreviewShow( Sender: TObject );

# Category

Preview

# Component/Class

TRvSystem

# 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

**OnPreviewSetup** 

# Example

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

# C++Builder

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

# OnPrint event (read/write/published)

# Declaration

procedure OnPrint(Sender: TObject);

# Category

Control

# Component/Class

TBaseReport

# 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

Execute, NewPage, OnPrintPage

# **OnPrintFooter** event (read/write/published)

# Declaration

procedure OnPrintFooter(Sender: TObject);

#### Category

Control

#### Component/Class

TBaseReport

#### 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

GotoFooter, PrintFooter, OnPrintHeader

# Example

```
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 }
```

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

# **OnPrintHeader** event (read/write/published)

#### Declaration

procedure OnPrintHeader(Sender: TObject);

#### Category

Control

#### Component/Class

TBaseReport

#### 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

GotoHeader, OnPrintFooter, PrintHeader

# Example

```
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 }
```

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

# OnPrintPage event (read/write/published)

# Declaration

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

# Category

Control

# Component/Class

TBaseReport

#### 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

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 }
```

# OnRestore event (read/write/published)

#### Declaration

procedure OnRestore(Connection: TRvCustomConnection);

#### Category

Rave

#### Component/Class

**TRvCustomConnection** 

#### 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

<u>OnOpen</u>

# **OnSetFilter** event (read/write/published)

#### Declaration

procedure OnSetFilter(Connection: TRvCustomConnection);

# Category

Rave

#### Component/Class

TRvCustomConnection

#### 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

<u>OnSetSort</u>

# OnSetSort event (read/write/published)

# Declaration

procedure OnSetSort(Connection: TRvCustomConnection);

# Category

Rave

# Component/Class

TRvCustomConnection

# 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

**OnSetFilter** 

# OnValidateRow event (read/write/published)

# Declaration

procedure OnValidateRow(Connection: TRvCustomConnection; var ValidRow: boolean);

# Category

Rave

# Component/Class

TRvCustomConnection

# 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

<u>OnSetFilter</u>

# OnZoomChange event (read/write/published)

# Declaration

procedure OnZoomChange(Sender: TObject);

# Category

Preview

# Component/Class

**TRvRenderPreview** 

# 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

RedrawPage, ZoomIn, ZoomOut

# Example

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

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

# Open method

# Declaration

procedure Open;

# Category

Rave

# Component/Class

TRvProject

# Description

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

# See also

Close, LoadDesigner, OnAfterOpen, OnBeforeOpen, ProjectFile, Save

# Orientation property (read/write/published)

# Declaration

property Orientation: TOrientation;

# Default

poPortrait

Category Printer

Component/Class TBaseReport

# 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*.

# Example

```
Delphi
RvNDRWriter1.Orientation := poLandscape;
C++Builder
RvNDRWriter1->Orientation = poLandscape;
```

# OriginX property (read/write)

# Declaration

property OriginX: double;

# Default

0.0

Category

Position

Component/Class TBaseReport

# 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

<u>OriginY</u>

# Example

```
Delphi
RvNDRWriter1.OriginX := 2.0;
C++Builder
RvNDRWriter1->OriginX = 2.0;
```

# OriginY property (read/write)

# Declaration

property OriginY: double;

# Default

0.0

# Category

Position

# Component/Class

TBaseReport

# 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

<u>OriginX</u>

# Example

```
Delphi
RvNDRWriterl.OriginY := 2.0;
C++Builder
RvNDRWriterl->OriginY = 2.0;
```

# OutputFileName property (read/write)

#### Declaration

property OutputFileName: TFileName;

#### Default

'' (empty)

Category Printer

Component/Class TRvSystem

#### 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

<u>OutputName</u>

# Example

```
Delphi
RvSystem1.OutputFileName := 'TEST.DAT';
C++Builder
RvSystem1->OutputFileName = "TEST.DAT";
```

# OutputInvalid property (read only)

# Declaration

property OutputInvalid: boolean;

#### Default

true

# Category

Control

# Component/Class

TBaseReport

# 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

Abort, FirstPage, LastPage, Selection

# OutputName property (read/write)

# Declaration

property OutputName: string;

#### Default

'' (empty)

Category Printer

Component/Class TBaseReport

# 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

Port

# Example

COPY PRINTER.DMP LPT1 /B

#### Delphi

```
RvNDRWriter1.OutputName := 'C:\APP\PRINTER.DMP';
C++Builder
RvNDRWriter1->OutputName = "C:\APP\PRINTER.DMP";
```

# OverridePreview event (read/write)

# Declaration

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

# Category

ReportSystem

# Component/Class

TRvSystem

# 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

<u>OverridePreviewProc</u>

# OverrideSetup event (read/write)

# Declaration

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

# Category

ReportSystem

Component/Class TRvSystem

# 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

OverrideSetupProc

# OverrideStatus event (read/write)

# Declaration

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

# Category

ReportSystem

# Component/Class

TRvSystem

# 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

OverrideStatusProc

# PageHeight property (read only)

# Declaration

property PageHeight: double;

Category

Printer

Component/Class TBaseReport

# Description

This property returns the height of the currently selected paper size.

See also

PageWidth

Example

Save current page height

# Delphi

CurrPageHeight := RvNDRWriter1.PageHeight; C++Builder CurrPageHeight = RvNDRWriter1->PageHeight;

# PageInc property (read/write/published)

# Declaration

property PageInc: integer;

# Default

1

Category Preview

Component/Class

TRvRenderPreview

# 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

NextPage, PrevPage

# Example

```
Delphi
PageInc := 4;
C++Builder
PageInc = 4;
```

# PageInvalid property (read only)

# Declaration

property PageInvalid: boolean;

#### Category

Control

# Component/Class

TBaseReport

#### 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

FirstPage, LastPage

# Example

```
Delphi
if RvNDRWriter1.PageInvalid then begin
  { code to respond to an invalid page }
end; { if }
```

# C++Builder

```
if (RvNDRWriter1.PageInvalid) {
    / code to respond to an invalid page
}/ if
```

# Pages property (read only)

#### Declaration

property Pages: integer;

# Category

Preview

#### Component/Class

**TRvRenderPreview** 

#### Description

This property returns the total number of pages that exist inside the report file for a preview screen.

# See also

Macro

# Example

```
Delphi
Edit1.Text := IntToStr(RvRenderPreview1.Pages);
Form1.Invalidate;
```

```
Editl->Text = IntToStr(RvRenderPreviewl->Pages);
Forml->Invalidate();
```

# PageWidth property (read only)

# Declaration

property PageWidth: double;

Category

Printer

Component/Class TBaseReport

# Description

This property returns the width of the currently selected paper size.

See also

PageHeight

Example

Save current page width

# Delphi

CurrPageWidth := RvNDRWriter1.PageWidth; C++Builder CurrPageWidth = RvNDRWriter1->PageWidth;

# Papers property (read only)

# Declaration

property Papers: TStrings;

Default

(list of paper sizes supported by the default printer)

Category

Printer

# Component/Class

TBaseReport

# Description

This property will return a TStringList of paper sizes that are supported by the current printer.

See also

SelectPaper, SupportPaper, TStrings

# Example

```
Delphi
ListBox2.Items := RvNDRWriter1.Papers;
C++Builder
ListBox2->Items = RvNDRWriter1->Papers;
```

# Pie method

# Declaration

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

# Category

Graphics

# Component/Class

TBaseReport

# 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

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(3.25,1.0,5.25,3.0,0.0,0.0,3.25,7.0);
SetBrush(clBlack, bsBDiagonal, nil);
Pie(3.25,1.0,5.25,3.0,3.25,7.0,5.25,2.0);
```

```
rp->SetBrush(clBlack, bsHorizontal, NULL);
rp->Pie(3.25,1.0,5.25,3.0,5.25,2.0,0.0,0.0);
rp->SetBrush(clBlack, bsVertical, NULL);
rp->Pie(3.25,1.0,5.25,3.0,0.0,0.0,3.25,7.0);
rp->SetBrush(clBlack, bsBDiagonal, NULL);
rp->Pie(3.25,1.0,5.25,3.0,3.25,7.0,5.25,2.0);
```

# PIVar property

# Declaration

function PIVar(PIVarName: String): String;

# Category

Printing

# Component/Class

TBaseReport

# 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

<u>SetPIVar</u>

# Example

#### Delphi

```
with Sender as TBaseReport do begin
    Print('SubTotal:' + PIVar('SubTotal'));
/ Other print statements including new pages
    SetPIVar('SubTotal',FormatFloat(SubTotal));
end; {with}
```

```
rp->Print("SubTotal:" + PIVar("SubTotal"));
/ Other print statements including new pages
rp->SetPIVar("SubTotal",FormatFloat(SubTotal));
```

# Polygon method

# Declaration

procedure Polygon(const Points: array of TPoint);

# Category

Graphics

# Component/Class

TBaseReport

# Description

This method will draw a polygon using the current pen defined by the points contained in the open array *Points*. It also closes the shape between the first and last points and fills it using the current brush.

# Example

#### . Delphi

```
POINT points[3];
points[0] = rp->CreatePoint(1.0,2.0);
points[1] = rp->CreatePoint(2.0,3.0);
points[2] = rp->CreatePoint(5.0,2.0);
RvNDRWriter1->Polygon(points,2);
```

# Polyline method

# Declaration

procedure Polyline(const Points: array of TPoint);

# Category

Graphics

# Component/Class

TBaseReport

# Description

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

#### See also

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);
```

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

## PopFont method

### Declaration

function PopFont: boolean;

### Category

Font

### Component/Class

TBaseReport

### 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

PushFont

### Example

```
Delphi
PushFont;
SetFont('Arial',10);
Println('This is in Arial');
PopFont;
```

```
rp->PushFont();
rp->SetFont("Arial",10);
rp->Println("This is in Arial");
rp->PopFont();
```

## PopPos method

### Declaration

function PopPos: boolean;

### Category

Position

### Component/Class

TBaseReport

### 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

<u>PushPos</u>

### Example

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

### C++Builder

```
rp->PushPos();
rp->PrintXY(4,1.5,"Name");
rp->PopPos();
```

## PopTabs method

### Declaration

function PopTabs: boolean;

Category Tabs

## Component/Class

TBaseReport

### 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

<u>PushTabs</u>

## Port property (read only)

### Declaration

property Port: string;

### Category

Printer

### Component/Class TBaseReport

Твазентере

### Description

This property will return the port name for the currently selected printer.

### See also

PrinterIndex, OutputName

### Example

```
Delphi
Edit1.Text := RvNDRWriter1.Port;
Form1.Invalidate;
```

### C++Builder

```
Editl->Text = RvNDRWriterl->Port;
Forml->Invalidate();
```

## Position property (read/write)

### Declaration

property Position: double;

### Category

BarCode

### Component/Class

TRpBarsBase

### 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

BarCodeJustify, BarTop, Center, Left, Right

### Example

Bar Code will be centered at the SectionLeft + 3.0 point

### Delphi

```
BarCodeJustify := pjCenter;
Position := SectionLeft + 3.0;
```

```
BarCodeJustify = pjCenter;
Position = SectionLeft + 3.0;
```

## Pos property (read/write)

### Declaration

property Pos: longint;

### Default

0

## Category

Memo

### Component/Class TMemoBuf

### Description

This property will return or set the current position marker for the memo buffer. The first position is at index 0.

### See also

<u>Reset</u>

### Example

Save current memo buffer position

### Delphi

CurrMemoPos := MemoBufl.Pos; C++Builder CurrMemoPos = MemoBufl->Pos;

## PrevPage method

### Declaration

procedure PrevPage;

### Category

Preview

### Component/Class

**T**RvRenderPreview

### 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

CurrentPage, NextPage, OnPageChange

### Example

Delphi
RvRenderPreview1.PrevPage;
C++Builder
RvRenderPreview1->PrevPage();

## Print method

### Declaration

procedure Print(Text: string);

### Category

Printing

### Component/Class

TBaseReport

### 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

all other print functions

### Example

```
Delphi
RvNDRWriter1.Print('Hello World!');
C++Builder
RvNDRWriter1->Print("Hello World!");
```

## Print method

### Declaration

procedure Print;

### Category

BarCode

### Component/Class

TRpBarsBase

### 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.

### Example

GotoXY, PrintReadable, PrintTop, PrintXY, Text

### Delphi

BarCode1.Text := '12345';
BarCode1.Print;

### C++Builder

BarCodel->Text = "12345"; BarCodel->Print();

## PrintBitmap method

### Declaration

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

### Category

Graphics

### Component/Class

TBaseReport

### 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

PrintBitmapRect

### Example

Print MyBitmap in upper left corner four times its size

Delphi

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

## PrintBitmapRect method

### Declaration

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

### Category

Graphics

### Component/Class TBaseReport

### 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

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;
```

### C++Builder

```
TBitmap* Bitmap = new TBitmap();
Bitmap.LoadFromFile("RpDEMO.BMP");
rp->PrintBitmapRect(5.375,3.5,7.375,5.5,Bitmap);
delete Bitmap;
```

## PrintBlock method

### Declaration

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

### Category

Printing

### Component/Class

TBaseReport

### 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

All other print functions

```
Delphi
PrintBlock('This is block justified text',0.5,4.0);
C++Builder
Rp->PrintBlock("This is block justified text",0.5,4.0);
```

## PrintCenter method

### Declaration

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

### Category

Printing

### Component/Class

TBaseReport

### Description

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

### See also

all other print functions

### Example

```
Delphi
PrintCenter('Text centered at 2.0', 2.0);
C++Builder
rp->PrintCenter("Text centered at 2.0", 2.0);
```

## PrintCharJustify method

### Declaration

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

### Category

Printing

### Component/Class TBaseReport

### 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

PrintLeft, PrintRight

### Example

Print the number justified by the decimal point

### Delphi

```
PrintCharJustify(NumStr,'.',4.25);
C++Builder
rp->PrintCharJustify(NumStr,".",4.25);
```

## PrintChecksum property (read/write)

### Declaration

property PrintChecksum: boolean

### Default

false

Category BarCode

Component/Class TRpBarsBase

### 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

BarTop, UseChecksum

## PrintData method

### Declaration

procedure PrintData(Value: string);

### Category

Printer

### Component/Class

TBaseReport

### 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

All other print functions, PrintDataStream

```
Delphi
RvNDRWriter1.PrintData( SpecialCodes );
C++Builder
RvNDRWriter1->PrintData( SpecialCodes );
```

## PrintDataStream method

### Declaration

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

### Category

Printer

### Component/Class

TBaseReport

### 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

PrintData

### Example

### Delphi

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

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

## PrintEnd property (read/write)

### Declaration

property PrintEnd: double;

### Default

0.0

Category Memo

Component/Class TMemoBuf

### Description

This property will return or set the rightmost position that the memo field will print in.

### See also

PrintStart

### Example

Leave 1.5 inches for left margin

### Delphi

MemoBufl.PrintEnd := 6.5; C++Builder MemoBufl->PrintEnd = 6.5;

## PrinterIndex property (read/write)

### Declaration

property PrinterIndex: integer;

### Default

-1

### Category Printer

Printer

## Component/Class

TBaseReport

### 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

<u>SelectPrinter</u>

### Example

Save current printer index

### Delphi

```
CurrIndex := RvNDRWriter1.PrinterIndex;
C++Builder
CurrIndex = RvNDRWriter1->PrinterIndex;
```

## Printers property (read only)

### Declaration

property Printers: TStrings;

### Default

(list of printers currently installed on the system)

### Category

Printer

### Component/Class TBaseReport

### Description

This property will return a TStringList of printers that are currently installed on the user's computer.

### See also

SelectPrinter, TStrings

### Example

```
Delphi
ComboBox2.Items := Printers;
C++Builder
ComboBox2->Items := rp->Printers;
```

## PrintFimA method

### Declaration

procedure PrintFimA( X,Y: double );

### Category

BarCode

### Component/Class TRpBarsBase

Description

This method prints a PostNet FIM A at the given X, Y location.

See also

PrintFimB, PrintFimC

### Example

Delphi
PostNetBC1.PrintFimA(3.5,0.5);
C++Builder
PostNetBC1->PrintFimA(3.5,0.5);

## PrintFimB method

### Declaration

procedure PrintFimB( X,Y: double );

### Category

BarCode

Component/Class TRpBarsBase

### Description

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

See also

PrintFimA, PrintFimC

### Example

Delphi
PostNetBC1.PrintFimB(3.5,0.5);
C++Builder
PostNetBC1->PrintFimB(3.5,0.5);

## PrintFimC method

```
Declaration
    procedure PrintFimC( X,Y: double );
Category
    BarCode
Component/Class
    TRpBarsBase
Description
    This method prints a PostNet FIM C at the given X, Y location.
See also
    PrintFimA, PrintFimB
Example
    Delphi
    PostNetBC1.PrintFimC(3.5,0.5);
    C++Builder
    PostNetBC1->PrintFimC(3.5,0.5);
```

## PrintFooter method

### Declaration

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

### Category

Printing

### Component/Class

TBaseReport

### Description

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

### See also

All other print functions, GotoFooter

### Example

```
Delphi
PrintFooter('Date 01/20/95', pjRight);
C++Builder
PrintFooter("Date 01/20/95", pjRight);
```

## PrintHeader method

### Declaration

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

### Category

Printing

### Component/Class TBaseReport

### Description

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

### See also

All other print functions, GotoHeader

```
Delphi
PrintHeader( 'Report Header Text', pjCenter);
C++Builder
PrintHeader( "Report Header Text", pjCenter);
```

## PrintHeight method

### Declaration

procedure PrintHeight(Height:double; PrintTabs: boolean);

### Category

Memo

### Component/Class

TMemoBuf

### 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:** <u>You must initialize</u> the TMemoBuf.BaseReport before calling this method.

See also

BaseReport, TMemoBuf, MemoHeightLeft

## PrintImageRect method

### Declaration

procedure PrintImageRect(X1,Y1,X2,Y2: double; ImageStream: Tstream; ImageType: string);

### Category

Graphics

### Component/Class

TBaseReport

### 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

CalcGraphicHeight, CalcGraphicWidth, OnDecodeImage, PrintBitmap, StretchDraw

### Example

```
Delphi
with Sender as TBaseReport do begin
Stream := TMemoryStream.Create;
Image := TJPEGImage.Create;
try
Image.LoadFromFile('image1.jpg');
Image.SaveToStream(Stream);
Stream.Position := 0;
PrintImageRect(1,1,3,3,Stream,'JPG');
finally
Image.Free;
Stream.Free
end; {tryf}
end; {with}
```

```
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;
  rp->PrintImageRect(1,1,3,3,Stream, "JPG");
}
finally {
  delete Image;
  delete Stream;
}; {tryf}
```

## Printing property (read only)

### Declaration

property Printing: boolean;

### Category

Control

### Component/Class

TBaseReport

### 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

**Execute** 

### Example

### Delphi if RvNDRWriter1.Printing then RvNDRWriter1.Abort; C++Builder

if (RvNDRWriter1->Printing) RvNDRWriter1->Abort();

## PrintJustify method

### Declaration

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

### Category

Printing

## Component/Class

TBaseReport

### 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

PrintBlock, PrintCenter, PrintLeft, PrintRight

### Example

Delphi

### C++Builder

## PrintLeft method

### Declaration

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

### Category

Printing

### Component/Class

TBaseReport

### Description

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

### See also

All other print functions

```
Delphi
RvNDRWriter1.PrintLeft( 'Text left at 4.0', 4.0);
C++Builder
RvNDRWriter1->PrintLeft( "Text left at 4.0", 4.0);
```

## PrintLines method

### Declaration

procedure PrintLines(Lines: longint; PrintTabs: boolean);

### Category

Memo

### Component/Class

TMemoBuf

### 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

BaseReport, MemoLinesLeft, TMemoBuf

## PrintLn method

### Declaration

procedure PrintLn(Text: string);

### Category

Printing

### Component/Class

TBaseReport

### 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

All other print functions, <u>NewLine</u>

### Example

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

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

## PrintMemo method

### Declaration

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

### Category

Memo

### Component/Class

TBaseReport

### 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

MemoLines, TMemoBuf

### Example

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

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

## PrintPage method

### Declaration

procedure PrintPage(PageNum: word);

Category

Preview

Component/Class

TRvRenderPreview

### 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

OnPageChange, RedrawPage

### Example

```
Delphi
RvRenderPreview1.PrintPage( 2);
C++Builder
RvRenderPreview1->PrintPage( 2);
```

## PrintReadable property (read/write)

### Declaration

property PrintReadable: boolean;

### Default

true

### Category

BarCode

### Component/Class TRpBarsBase

Description

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

**NOTE:** For UPC bar codes, text is always printed.

### See also

PrintTop, TextJustify

## PrintRight method

### Declaration

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

### Category

Printing

Component/Class TBaseReport

### Description

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

### See also

all other print functions

### Example

```
Delphi
RvNDRWriter1.PrintRight('Right justified at 3.0',3.0 );
C++Builder
RvNDRWriter1->PrintRight("Right justified at 3.0",3.0 );
```

## PrintStart property (read/write)

### Declaration

property PrintStart: double;

### Default

0.0

### Category Memo

Component/Class TMemoBuf

пиетнов

### Description

This property will return or set the leftmost position that the memo buffer will print in.

See also

<u>PrintEnd</u>

### Example

Leave 1.5 inches for right margin

### Delphi

MemoBuf1.PrintStart := 1.5; C++Builder MemoBuf1->PrintStart = 1.5;

## PrintTab method

### Declaration

procedure PrintTab(Text: string);

### Category

Printing

### Component/Class

TBaseReport

### 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

Print, Println, Tab

### Example

```
Delphi
PrintTab(FieldByName('Name'));
C++Builder
PrintTab(FieldByName("Name"));
```

## PrintTop property (read/write)

### Declaration

property PrintTop: boolean;

### Default

false

### Category

BarCode

### Component/Class TRpBarsBase

Description

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

### See also

PrintReadable, TextJustify

### Example

```
Delphi
Code39.PrintTop := True;
Code39.Print;
```

```
Code39->PrintTop = true;
Code39->Print();
```

## PrintXY method

### Declaration

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

### Category

Printing

## Component/Class

TBaseReport

### 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

All other print functions, GotoXY

### Example

```
Delphi
RvNDRWriter1.PrintXY( 1.0, 2.0, 'Text above (1.0, 2.0)');
C++Builder
RvNDRWriter1->PrintXY( 1.0, 2.0, "Text above (1.0, 2.0)");
```

## PrintXY method

### Declaration

procedure PrintXY( X,Y: double );

### Category

BarCode

### Component/Class TRpBarsBase

### 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

Print, PrintReadable, PrintTop, Text

### Example

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

### C++Builder

Code2of5->Text = "12345"; Code2of5->PrintXY( 1.0, 2.0 );

## ProjectFile property (read/write/published)

### Declaration

property ProjectFile: string;

### Default

'' (empty)

Category Rave

Component/Class

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

Active, Close, Open

## PushFont method

Declaration

function PushFont: boolean;

Category Font

Component/Class TBaseReport

### Description

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

See also

PopFont

### Example

see PopFont

## PushPos method

### Declaration

function PushPos: boolean;

### Category

Position

Component/Class TBaseReport

### Description

This method will push the current text cursor position onto an internal stack for later retrieval by *PopPos*.

See also

<u>PopPos</u>

### Example

see PopPos

## PushTabs method

### Declaration

function PushTabs: boolean;

### Category

Tabs

## Component/Class

TBaseReport

### Description

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

### See also

<u>PopTabs</u>

## Query property (read/write/published)

### Declaration

property Query: TQuery;

### Category

Rave

### Component/Class

TRvQueryConnection

### Description

Specifies the TQuery component that is connected to the TRvQueryConnection component.

### Example

```
Delphi
CustOrdCXN.Query := CustOrdQuery;
C++Builder
CustOrdCXN->Query = CustOrdQuery;
```

## RaveBlobDateTime property (read/write/published)

### Declaration

property RaveBlobDateTime: TDateTime;

### Category

Rave

### Component/Class

**T**RvProject

### 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

ClearRaveBlob, LoadRaveBlob, SaveRaveBlob

```
Delphi
Label1.Caption := DateTimeToStr(RvProject1.RaveBlobDateTime);
C++Builder
Label1->Caption = DateTimeToStr(RvProject1->RaveBlobDateTime);
```

## ReadableHeight property (read only)

### Declaration

property ReadableHeight: double;

Category

BarCode

Component/Class TRpBarsBase

Description

Returns the height that the readable text adds to the bar code.

See also BarHeight, Height

## RecoverPrinter method

### Declaration

procedure RecoverPrinter;

Category

Printer

Component/Class

TBaseReport

### Description

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

See also

ReleasePrinter

### Example

See ReleasePrinter

## Rectangle method

### Declaration

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

Category

Graphics

Component/Class

TBaseReport

### 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

RoundRect

### Example

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

## RedrawPage method

Declaration

procedure RedrawPage;

Category

Preview

Component/Class TRvRenderPreview

Description

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

See also

PrintPage

```
Delphi
RvRenderPreview1.RedrawPage;
C++Builder
RvRenderPreview1->RedrawPage();
```

## RegisterGraphic method

### Declaration

procedure RegisterGraphic( index: integer);

### Category

Graphics

### Component/Class

TBaseReport

### 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

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 }
finaly
    Bitmap.Free;
end; { with }
```

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

## ReleasePrinter method

### Declaration

procedure ReleasePrinter;

### Category

Printer

### Component/Class

TBaseReport

### 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

**RecoverPrinter** 

### Example

### Delphi

RvNDRWriter1.ReleasePrinter; PrinterSetupDialog1.Execute; RvNDRWriter1.RecoverPrinter;

### C++Builder

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

## ReplaceAll method

### Declaration

procedure ReplaceAll(SearchText: string; ReplaceText: string; CaseMatters: boolean);

Category

Memo

### Component/Class

TMemoBuf

### 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

SearchFirst, SearchNext

### Example

```
Delphi
MemoBuf.ReplaceAll('ame, Name, false);
MemoBuf.ReplaceAll('ddress, Address, false);
```

### C++Builder

```
MemoBuf->ReplaceAll("ame, Name, false);
MemoBuf->ReplaceAll("ddress, Address, false);
```

## ReportDateTime property (read/write)

### Declaration

property ReportDateTime: TDateTime;

### Default

(Date and time Execute or Start was called)

### Category

Printing

### Component/Class

**TBaseReport** 

### Description

This property will set or return the date and time the report was started.

See also Macro

```
Delphi
```

```
Editl.Text := DateTimeToStr(ReportDateTime);
C++Builder
Editl->Text = DateTimeToStr(rp->ReportDateTime);
```

## ReportDesc property (read only)

### Declaration

property ReportDesc: string;

### Category

Rave

### Component/Class

TRvProject

### 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 memobul area for the user to select. This property will return the description of the currently selected report.

### See also

ReportFullName, ReportDescToMemo, ReportName, SelectReport

## ReportDescToMemo method

### Declaration

procedure ReportDescToMemo(Memo: TCustomMemo);

### Category

Rave

### Component/Class

TRvProject

### Description

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

### See also

ReportDesc, SelectReport

## ReportDest property (read only)

### Declaration

property ReportDest: TReportDest;

### Category

ReportSystem

### Component/Class

TRvSystem

### 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

<u>DefaultDest</u>

## ReportFullName property (read only)

### Declaration

property ReportFullName: string;

### Category

Rave

### Component/Class

TRvProject

### 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 memobul area for the user to select. This property will return the full name of the currently selected report.

### See also

ReportDesc, ReportName, SelectReport

## ReportName property (read only)

### Declaration

property ReportName: string;

### Category

Rave

### Component/Class

TRvProject

### 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 memobul area for the user to select. This property will return the name of the currently selected report.

### See also

ReportDesc, ReportFullName, SelectReport

## Reset method

Declaration

procedure Reset;

### Category

Control

### Component/Class TBaseReport

TDasertept

### Description

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

### See also

<u>ResetPrinter</u>

### Example

Delphi
RvNDRWriter1.Reset;
C++Builder
RvNDRWriter1->Reset();

## Reset method

### Declaration

procedure Reset;

### Category

Memo

### Component/Class TMemoBuf

### 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

Pos

### Example

```
Delphi
MemoBufl.Reset;
C++Builder
MemoBufl->Reset();
```

## ResetLineHeight method

### Declaration

procedure ResetLineHeight;

### Category

Position

### Component/Class

TBaseReport

### 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

LineHeight, LineHeightMethod

```
Delphi
RvNDRWriter1.ResetLineHeight;
C++Builder
RvNDRWriter1->ResetLineHeight();
```

## ResetPrinter method

## Declaration

procedure ResetPrinter;

## Category

Printer

## Component/Class

TBaseReport

## 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

<u>DevMode</u>

## Example

```
Delphi
RvNDRWriter1.ResetPrinter;
C++Builder
RvNDRWriter1->ResetPrinter();
```

## ResetSection method

## Declaration

procedure ResetSection;

## Category

Position

## Component/Class

TBaseReport

## Description

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

## See also

All Margin and Section properties

```
Delphi
RvNDRWriter1.ResetSection;
C++Builder
RvNDRWriter1->ResetSection();
```

# ResetTabs method

## Declaration

procedure ResetTabs;

## Category

Tabs

#### Component/Class TBaseReport

## Description

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

See also

ClearTabs, SetTab

## Example

Delphi
RvNDRWriter1.ResetTabs;
C++Builder
RvNDRWriter1->ResetTabs();

## RestoreBuffer method

Declaration

procedure RestoreBuffer;

Category

Memo

#### Component/Class TMemoBuf

## Description

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

## See also

SaveBuffer

## RestoreFont method

## Declaration

function RestoreFont(Index: integer): boolean;

## Category

Font

## Component/Class

TBaseReport

## 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

SaveFont

## Example

Restore the font saved in position 10

#### Delphi

RestoreFont(10); C++Builder rp->RestoreFont(10);

## RestorePos method

## Declaration

function RestorePos(Index: byte): boolean;

## Category

Position

## Component/Class

TBaseReport

## Description

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

## See also

SavePos

```
Delphi
RvNDRWriter1.RestorePos(1);
C++Builder
RvNDRWriter1->RestorePos(1);
```

## RestoreState method

## Declaration

procedure RestoreState;

## Category

Memo

## Component/Class

TMemoBuf

## Description

This method restores the cursor position and other state information of the memobuffer back to what it was when SaveState was called.

NOTE: This does not effect the contents of the memo buffer.

## See also

Pos, RestoreBuffer, SaveState

## RestoreTabs method

## Declaration

function RestoreTabs(Index: integer): boolean;

Category Tabs

# Component/Class

TBaseReport

## Description

This method will restore the tab settings, saved by a previous *SaveTabs* call, using an *Index* from 1 to 10.

The result of this function will be true if the call was successful.

## See also

RestoreTabs, SetTab

## Example

Restore the tab settings in position 3

## Delphi

RestoreTabs(3); C++Builder RestoreTabs(3);

## ReuseGraphic method

## Declaration

procedure ReuseGraphic;

## Category

Graphics

## Component/Class

TBaseReport

## 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

RegisterGraphic, UnregisterGraphic

## Example

See RegisterGraphic

# RichEdit property

## Declaration

property RichEdit: string

## Category

Memo

#### Component/Class TMemoBuf

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

RTFLoadFromStream, RTFText

## Example

Delphi
MemoBuf1.RichEdit := RichEdit1;
C++Builder
MemoBuf1->RichEdit = RichEdit1;

# Right property (read/write)

## Declaration

property Right: double;

## Category

BarCode

## Component/Class

TRpBarsBase

## 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

BarCodeJustify, Center, Left, Position

## Example

```
Delphi
BarCodel.Right := SectionRight;
C++Builder
BarCodel->Right = rp->SectionRight;
```

# RightWaste property (read only)

## Declaration

property RightWaste: double;

## Category

Printer

#### Component/Class TBaseReport

## 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

BottomWaste, LeftWaste, MarginRight, TopWaste

## Example

See LeftWaste

## RoundRect method

## Declaration

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

## Category

Graphics

## Component/Class

TBaseReport

## 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

Ellipse, Rectangle

## Example

Delphi
RoundRect(1.125,3.5,3.125,5.0,0.25,0.25);
C++Builder
rp->RoundRect(1.125,3.5,3.125,5.0,0.25,0.25);

# RTFField property

#### Declaration

property RTFField: TMemoField

## Category

Memo .

#### Component/Class TMemoBuf

## Description

Imports a RTF string stored in a TMemoField component into a memo buffer.

## See also

Field, RTFText

# RTFLoadFromFile method

## Declaration

procedure RTFLoadFromFile( FileName: String);

## Category

Memo

#### Component/Class TMemoBuf

## Description

Load an RTF text file into the memo buffer.

## See also

LoadFromFile, RTFLoadFromStream

## Example

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

# RTFLoadFromStream method

## Declaration

procedure RTFLoadFromStream( stream: Tstream; BufSize: longint);

## Category

Memo

## Component/Class

TMemoBuf

## Description

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

## See also

LoadFromFile, RTFLoadFromFile

## RTFText property

## Declaration

property RTFText: string

## Category

Memo

#### Component/Class TMemoBuf

Description

Imports an RTF string stored in a text variable into the memo buffer.

## See also

**RTFField** 

# RulerType property (read/write/published)

## Declaration

property RulerType: TRulerType;

## Default

rtNone

Category

Preview

#### Component/Class TRvSystem

## 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

GridHoriz, GridPen, GridVert

## RuntimeVisibility property (read/write/published)

## Declaration

property RuntimeVisibility: Boolean;

## Category

Rave

## Component/Class

TRvCustomConnection

## 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:	visibile 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

DevLock property on Rave Compoents

## Example

```
Delphi
RvCustomConnection1.RuntimeVisibility := rtNone;
C++Builder
RvCustomConnection1->RuntimeVisibility = rtNone;
```

## Save method

## Declaration

procedure Save;

## Category

Rave

## Component/Class

TRvProject

## Description

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

See also

Close, Open, ProjectFile

# SaveBuffer method

### Declaration

procedure SaveBuffer;

## Category

Memo

## Component/Class

TMemoBuf

## 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

## FreeSaved, RestoreBuffer

## Example

Save original contents

#### Delphi

```
MemoBuf.SaveBuffer;
C++Builder
MemoBuf->SaveBuffer();
```

# SaveFont method

## Declaration

function SaveFont(Index: integer): boolean;

## Category

Font

## Component/Class

**TBaseReport** 

## 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

**RestoreFont** 

## Example

Save the current font settings in position 2

## Delphi

SaveFont(2);
C++Builder
rp->SaveFont(2);

# SavePos method

## Declaration

function SavePos(Index: byte): boolean;

## Category

Position

#### Component/Class TBaseReport

твазекерог

## 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

<u>RestorePos</u>

## Example

Delphi
RvNDRWriter1.SavePos(1);
C++Builder
RvNDRWriter1->SavePos(1);

## SaveRaveBlob method

## Declaration

function SaveRaveBlob(Stream: TStream);

## Category

Rave

### Component/Class TRvProject

TRVProjec

## 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

ClearRaveBlob, LoadRaveBlob, RaveBlobDateTime, StoreRAV

```
Delphi
RvProject1.SaveRaveBlob( MyStream );
C++Builder
RvProject1->SaveRaveBlob( MyStream );
```

# SaveState method

#### Declaration

procedure SaveState;

## Category

Memo

## Component/Class

TMemoBuf

## 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

Pos, RestoreState, SaveBuffer

## SaveTabs method

#### Declaration

function SaveTabs(Index: integer): boolean;

Category

Tabs

Component/Class

TBaseReport

### 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

RestoreTabs, SetTab

## Example

Save the current tab settings in position 5

Delphi

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

# SaveToFile method

## Declaration

function SaveToFile(FileName: String);

## Category

Rave

#### Component/Class TRvProject

Description

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

## See also

LoadFromStream, Save, SaveToStream

## Example

```
Delphi
RvProject1.SaveToFile('Project1.Rav');
C++Builder
RvProject1->SaveToFile("Project1.Rav");
```

## SaveToStream method

## Declaration

procedure SaveToStream(Stream: TStream);

Category Memo

## Component/Class TMemoBuf

## Description

This method will save the memo buffer to the stream.

## See also

LoadFromStream

```
Delphi
MemoBufl.SaveToStream( MyStream );
C++Builder
MemoBufl->SaveToStream( MyStream );
```

# SaveToStream method

## Declaration

procedure SaveToStream(Stream: TStream);

#### Category

Rave

#### Component/Class TRvProject

III VEIUJECI

## Description

This method will save the report project to Stream.

#### See also

LoadFromFile, LoadFromStream, Save, SaveToFile

## Example

Delphi
RvProject1.SaveToStream(RaveStream);
C++Builder
RvProject1->SaveToStream(RaveStream);

## ScaleX property (read/write/published)

## Declaration

property ScaleX: double;

## Default

100

## Category

Control

## Component/Class

TBaseReport

## 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

OriginX, OriginY, ScaleY

## Example

Scale to fit 4 pages on one sheet of paper

#### Delphi

RvNDRWriter1.ScaleX := 50.0; RvNDRWriter1.ScaleY := 50.0;

### C++Builder

RvNDRWriter1->ScaleX = 50.0; RvNDRWriter1->ScaleY = 50.0;

## ScaleY property (read/write/published)

## Declaration

property ScaleY: double;

#### Default

100

Category

Control

Component/Class TBaseReport

## 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

OriginX, OriginY, ScaleX

## Example

Scale to fit 4 pages on one sheet of paper

#### Delphi

RvNDRWriter1.ScaleX := 50.0; RvNDRWriter1.ScaleY := 50.0;

#### C++Builder

RvNDRWriter1->ScaleX = 50.0; RvNDRWriter1->ScaleY = 50.0;

# ScrollBox property (read/write/published)

#### Declaration

property ScrollBox: TScrollBox;

Default

nil

## Category

Preview

## Component/Class

TRvRenderPreview

#### Description

This property defines the scroll box on the preview form that the report will be drawn in.

```
Delphi
RvRenderPreview1.ScrollBox := Form1.ScrollBox1;
C++Builder
RvRenderPreview1->ScrollBox = Form1->ScrollBox1;
```

## SearchFirst method

## Declaration

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

## Category

Memo

## Component/Class

TMemoBuf

## 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

Pos, SearchNext

## Example

Store the number of occurrences of 'APPLE' in apples

## Delphi

```
Apples := 0;
Found := MemoBuf.SearchFirst('APPLE', false);
while Found do begin
    Inc(Apples);
    Found := MemoBuf.SearchNext;
end; { while }
```

## C++Builder

```
Apples := 0;
Found := MemoBuf->SearchFirst("APPLE", false);
while (Found == true) {
   Apples++;
   Found = MemoBuf->SearchNext();
}/ while
```

# SearchNext method

## Declaration

function SearchNext: boolean;

## Category

Memo

## Component/Class TMemoBuf

## 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

Pos, SearchFirst

## Example

See SearchFirst

## SectionBottom property (read/write)

### Declaration

property SectionBottom: double;

#### Default

MarginBottom

### Category

Position

#### Component/Class TBaseReport

## 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

Margin properties, ResetSection, SectionLeft, SectionRight, SectionTop

## Example

```
Delphi
with RvNDRWriter1 do begin
  SectionLeft := 1.0;
  SectionRight := 7.5;
  SectionTop := 1.5;
  SectionBottom := 1.0;
end; { with }
```

## C++Builder

```
rp->SectionLeft = 1.0;
rp->SectionRight = 7.5;
rp->SectionTop = 1.5;
rp->SectionBottom = 1.0;
```

## SectionLeft property (read/write)

## Declaration

property SectionLeft: double;

## Default

MarginLeft

## Category

Position

#### Component/Class TBaseReport

## 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

Margin properties, <u>ResetSection</u>, <u>SectionBottom</u>, <u>SectionRight</u>, <u>SectionTop</u>

## Example

see SectionBottom

## SectionRight property (read/write)

## Declaration

property SectionRight: double;

### Default

MarginRight

## Category

Position

#### Component/Class TBaseReport

## 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

Margin properties, <u>ResetSection</u>, <u>SectionBottom</u>, <u>SectionLeft</u>, <u>SectionTop</u>

## Example

see SectionBottom

## SectionTop property (read/write)

## Declaration

property SectionTop: double;

#### Default

MarginTop

## Category

Position

#### Component/Class TBaseReport

## 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

Margin properties, <u>ResetSection</u>, <u>SectionBottom</u>, <u>SectionLeft</u>, <u>SectionRight</u>

## Example

see SectionBottom

# SelectBin method

#### Declaration

function SelectBin(BinName: string): boolean;

#### Category

Printer

## Component/Class

TBaseReport

## 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

Bins, OnNewPage, SupportBin

```
Delphi
SelectBin('UPPER');
C++Builder
SelectBin("UPPER");
```

# Selection property (read/write)

## Declaration

property Selection: string;

#### Default

'' (empty)

## Category

Control

#### Component/Class TBaseReport

## 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

FirstPage, LastPage, SystemOptions

## Example

#### Delphi

#### C++Builder

## SelectPaper method

## Declaration

function SelectPaper(PaperName: string): boolean;

## Category

Printer

## Component/Class

TBaseReport

## 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

Papers, SupportPaper

## Example

```
Delphi
SelectPaper('LEGAL');
C++Builder
SelectPaper("LEGAL");
```

## SelectPrinter method

## Declaration

function SelectPrinter(SubStr: string): boolean;

## Category

Printer

#### Component/Class TBaseReport

## Description

This method will set the current printer to the first printer in Printers that contains the substring SubStr in its name. If no printer is found then the current printer is not changed and a false value is returned.

## See also

PrinterIndex

```
Delphi
SelectPrinter('Laser');
C++Builder
SelectPrinter("Laser");
```

## SelectReport method

## Declaration

function SelectReport(ReportName: string; FullName: boolean): boolean;

## Category

Rave

## Component/Class

TRvProject

## 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

GetReportList, ReportFullName, ReportName

## ServerMode property (read/write)

## Declaration

property ServerMode: Boolean read FServerMode write FServerMode

## Default

false

## Category

Render

## Component/Class

TRpRender

## 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

**CacheDir** 

## SetBrush method

## Declaration

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

## Category

Graphics

## Component/Class

TBaseReport

## 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

CreateBrush, TBrushStyle, TColor

```
Delphi
RvNDRWriter1.SetBrush(clBlack, bsClear, nil);
C++Builder
RvNDRWriter1->SetBrush(clBlack, bsClear, NULL);
```

## SetColumns method

## Declaration

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

## Category

Column

## Component/Class

TBaseReport

## 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

ColumnWidth, SectionLeft, SectionRight, SetColumnWidth

## Example

This code shows how to create 4 columns and send output to them. Also see PrintMemo. { with 0.5" between each }

#### Delphi

```
SetColumns(4,0.5);
while ColumnLinesLeft > 0 do begin
    Println(IntToStr(LinesLeft) + '/' +
    IntToStr(ColumnLinesLeft) + '/' +
    IntToStr(LineNum) + '/' +
    IntToStr(ColumnNum));
end; { while }
```

## C++Builder

## SetColumnWidth method

## Declaration

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

## Category

Column

## Component/Class

TBaseReport

## 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

Columns, SectionLeft, SectionRight, SetColumns

## Example

Create columns 2 inches wide and a half of an inch apart

```
Delphi
```

RvNDRWriter1.SetColumnWidth( 2.0, 0.5 ); C++Builder RvNDRWriter1->SetColumnWidth( 2.0, 0.5 );

## SetData method

## Declaration

procedure SetData(var Buffer; BufSize: longint);

## Category

Memo

## Component/Class

TMemoBuf

## 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

Text

## Example

Assign a PChar to a memo buffer

```
Delphi
```

```
MemoBuf.SetData(PCharVar<sup>*</sup>, StrLen(PCharVar));
C++Builder
```

# SetFont method

## Declaration

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

#### Category

Font

## Component/Class

TBaseReport

#### 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

AssignFont, CreateFont, FontCharSet

#### Example

```
Delphi
RvNDRWriter1.SetFont( 'Arial', 10 );
C++Builder
RvNDRWriter1->SetFont( "Arial", 10 );
```

## SetPaperSize method

#### Declaration

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

## Category

Printer

## Component/Class

TBaseReport

## 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.

## Example

Set papersize to 10" wide by 12" high then set papersize to 8.5 wide by 14" high

#### Delphi

```
RvNDRWriter1.SetPaperSize(0,10,12);
RvNDRWriter1.SetPaperSize(DMPAPER_LEGAL,0,0);
```

#### C++Builder

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

## SetParam method

## Declaration

procedure SetParam(ParamName: string; ParamValue: string);

## Category

Rave

## Component/Class

TRvProject

## 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.

## Example

```
Delphi
```

```
RvProject1.SetParam('UserName',UserName);
C++Builder
RvProject1->SetParam("UserName",UserName);
```

# SetPen method

## Declaration

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

## Category

Graphics

## Component/Class

TBaseReport

## 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

CreatePen, TColor, TPenMode, TPenStyle

```
Delphi
RvNDRWriter1.SetPen(clBlack,psSolid,-2,pmCopy);
C++Builder
RvNDRWriter1->SetPen(clBlack,psSolid,-2,pmCopy);
```

## SetPIVar method

## Declaration

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

## Category

Printing

## Component/Class

TBaseReport

## 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

<u>PIVar</u>

## Example

see PIVar

# SetTab method

## Declaration

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

## Category

Tabs

## Component/Class

TBaseReport

#### 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.

#### See also

ClearTabs, ResetTabs

## 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;
```

#### C++Builder

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

# SetTopOfPage method

#### Declaration

procedure SetTopOfPage;

#### Category

Position

#### Component/Class TBaseReport

#### Description

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

#### See also

MarginTop, SectionTop

#### Example

```
Delphi
RvNDRWriter1.SetTopOfPage;
C++Builder
RvNDRWriter1->SetTopOfPage();
```

## ShadeToColor method

#### Declaration

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

Category Graphics

## Component/Class

TBaseReport

#### Description

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

## See also

SetBrush, TColor

## ShadowDepth property (read/write/published)

## Declaration

property ShadowDepth: integer;

#### Default

0

Category Preview

Component/Class TBaseReport

#### 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

**Monochrome** 

#### Example

Delphi ShadowDepth := 5; C++Builder ShadowDepth = 5;

## ShowPrintDialog method

#### Declaration

function ShowPrintDialog: boolean;

## Category

Printer

#### Component/Class TBaseReport

#### Description

Brings up the standard Windows PrintDialog. Use this function instead of Delphi's TPrintDialog component.

See also

**ShowPrinterSetupDialog** 

## Example

```
Delphi
if RvNDRWriter1.ShowPrintDialog then begin
    RvNDRWriter1.Execute;
end; { if }
```

#### C++Builder

```
if (RvNDRWriter1->ShowPrintDialog()) {
    RvNDRWriter1->Execute();
}/ if
```

# ShowPrinterSetupDialog method

## Declaration

function ShowPrinterSetupDialog: boolean;

## Category

Printer

## Component/Class

TBaseReport

## Description

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

## See also

ShowPrintDialog

## Example

```
Delphi
if RvNDRWriter1.ShowPrinterSetupDialog then begin
    RvNDRWriter1.Execute;
end; { if }
```

## C++Builder

```
if (RvNDRWriter1->ShowPrinterSetupDialog()) {
    RvNDRWriter1->Execute();
}/ if
```

## Size property (read/write)

## Declaration

property Size: longint;

#### Category Memo

Component/Class TMemoBuf

## Description

This property will return the current size of the text in the memo buffer in bytes.

## See also

<u>MaxSize, Pos</u>

## Example

Delphi
MemoBytes := MemoBufl.Size;
C++Builder
MemoBytes = MemoBufl->Size;

## Start method

## Declaration

procedure Start;

## Category

Control

## Component/Class

TBaseReport

## 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

Execute, Finish

## Example

Delphi
RvRenderPreview1.Start;
C++Builder
RvRenderPreview1->Start();

### StatusFormat property (read/write/published)

#### Declaration

property StatusFormat: string;

#### Default

'Printing page '

#### Category

Misc

#### Component/Class TBaseReport

#### 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
%р -	Current Page
%f -	First Page
%I -	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	Status Text Line (see StatusText)
%9 -	
%-	% character

#### See also

CurrentPass, StatusLabel, StatusText, TotalPasses, UpdateStatus

#### Example

```
Delphi
RvNDRWriter1.StatusFormat := 'Generating page ';
RvNDRWriter1.StatusFormat := 'Printing page (Pass of )';
```

```
RvNDRWriter1->StatusFormat = "Generating page ";
RvNDRWriter1->StatusFormat = "Printing page (Pass of )";
```

### StatusLabel property (read/write/published)

#### Declaration

property StatusLabel: TLabel;

#### Default

nil

Category Misc

Component/Class TBaseReport

#### Description

This property defines the TLabel component that UpdateStatus will put the status text, StatusFormat, into.

#### See also

StatusFormat, StatusText, UpdateStatus

#### Example

```
Delphi
RvNDRWriter1.StatusLabel := StatusForm.Label1;
C++Builder
RvNDRWriter1->StatusLabel = StatusForm->Label1;
```

### StatusText property (read/write/published)

#### Declaration

property StatusText: TStrings;

#### Default

(empty)

Category Misc

Component/Class

TBaseReport

#### Description

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

StatusFormat, TStrings

#### Example

```
Delphi
```

```
StatusText[1] := 'Inform user of report status';
UpdateStatus;
```

```
rp->StatusText->Strings[1] = "Inform user of report status";
rp->UpdateStatus();
```

### StoreRAV property ZZZ read only/special/pub ZZZ

#### Declaration

property StoreRAV: boolean;

#### Default

false

### Category

Rave

### Component/Class

TRvProject

#### 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

ClearRaveBlob, LoadRaveBlob, ProjectFile, RaveBlobDateTime, SaveRaveBlob

## Stream property (read/write/published)

#### Declaration

property Stream: TStream;

#### Default

nil

Category

Control

Component/Class TBaseReport

#### 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

FileName, StreamMode

#### Example

```
Delphi
var ReportStream: TMemoryStream;
begin
ReportStream := TMemoryStream.Create;
try
with RvNDRWriter1 do begin
StreamMode := smUser;
Stream := ReportStream;
Execute;
end; { with }
finally
ReportStream.Free;
end; { tryf }
end;
```

```
TMemoryStream* ReportStream = new TMemoryStream();
try {
  rp->StreamMode = smUser;
  rp->Execute();
}
____finally {
  delete ReportStream;
}/ tryf
```

### StreamMode property (read/write/published)

#### Declaration

property StreamMode: TStreamMode;

#### Default

smMemory

#### Category

Control

#### Component/Class TBaseReport

#### Description

This property defines how the stream for the report file is maintained.

The setting *smFile* uses a TFileStream to store the report file and is very good for large reports, but may run a little slower.

*smTempFile* will send the ouput 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.

The setting *smMemory* 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* 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

FileName, Stream

#### Example

#### Delphi RvNDRWriter1.StreamMode := smMemory; RvNDRWriter2.FileName := 'TEMP.RpT'; RvNDRWriter2.StreamMode := smFile;

```
RvNDRWriter1->StreamMode = smMemory;
RvNDRWriter2->FileName = "TEMP.RPT";
RvNDRWriter2->StreamMode = smFile;
```

## StretchDraw method

#### Declaration

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

Category

Graphics

Component/Class

TBaseReport

#### 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

CreateRect, Draw, PrintBitmap, PrintBitmapRect, TGraphic, TRect

## Strikeout property (read/write)

#### Declaration

property Strikeout: boolean;

Default false

Category Font

Component/Class TBaseReport

#### Description

This property returns or sets the strikeout attribute for the current font.

#### See also

Bold, Italic, Underline

#### Example

```
Delphi
with RvNDRWriter1 do begin
  Strikeout := true;
  Print( 'Deleted Text' );
  Strikeout := false;
end; { with }
```

```
rp->Strikeout = true;
rp->Print( "Deleted Text" );
rp->Strikeout = false;
```

### Subscript property (read/write)

#### Declaration

property Subscript: boolean;

#### Default

false

### Category

Font

Component/Class TBaseReport

#### Description

Returns or sets the subscript setting for the current text font.

#### See also

Superscript

#### Example

Print a formula

#### Delphi

```
Print('Y = Pi * X');
Subscript := true;
Print('a');
Subscript := false;
```

```
rp->Print("Y = Pi * X");
rp->Subscript = true;
rp->Print("a");
rp->Subscript = false;
```

### Superscript property (read/write)

#### Declaration

property Superscript: boolean;

#### Default

false

#### Category Font

Component/Class

Description

Returns or sets the superscript setting for the current text font.

#### See also

Subscript

#### Example

Print a formula

#### Delphi

```
Print('E = MC');
Superscript := true;
Print('2');
Superscript := false;
```

#### C++Builder

```
rp->Print("E = MC");
rp->Superscript = true;
rp->Print("2");
rp->Superscript = false;
```

## SupportBin method

#### Declaration

function SupportBin(BinNum: integer): boolean;

#### Category

Printer

#### Component/Class TBaseReport

твазекероп

#### 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

SelectBin, other Support methods, TDevMode in Windows API help

## SupportCollate method

#### Declaration

function SupportCollate: boolean;

#### Category

Printer

Component/Class TBaseReport

#### Description

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

#### See also

Other Support methods

## SupportDuplex method

#### Declaration

function SupportDuplex: boolean;

### Category

Printer

### Component/Class

TBaseReport

#### Description

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

#### See also

Duplex, Other Support methods

## SupportOrientation method

#### Declaration

function SupportOrientation: boolean;

#### Category Printer

Component/Class TBaseReport

#### Description

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

#### See also

Other Support methods

### SupportPaper method

#### Declaration

function SupportPaper(PaperNum: integer): boolean;

#### Category

Printer

#### Component/Class

TBaseReport

#### 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

Other Support methods, TDevMode in Windows API help

### **SystemFiler** property (read/write/published)

#### Declaration

property SystemFiler: TSystemFiler;

#### Category

ReportSystem

#### Component/Class

TRvSystem

#### 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

Other System options

#### Example

Delphi
RvSystem1.SystemFiler.AccuracyMethod := amAppearance;
C++Builder
RvSystem1->SystemFiler->AccuracyMethod = amAppearance;

## SystemOptions property (read/write/published)

#### Declaration

property SystemOptions: TSystemOptions;

#### Category

ReportSystem

#### Component/Class

TRvSystem

#### 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	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.
soAllowPrintFromPreview	will determine whether the user can print 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

Other SystemXxxx options

#### Example

Disable the status screen

#### Delphi

```
RvSystem1.SystemOptions := RvSystem1.SystemOptions - (soShowStatus];
C++Builder
RvSystem1->SystemOptions = RvSystem1->SystemOptions >> soShowStatus;
```

### SystemPreview property (read/write/published)

#### Declaration

property SystemPreview: TSystemPreview;

#### Category

ReportSystem

#### Component/Class

TRvSystem

#### 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

Other SystemXxxx options

#### Example

```
Delphi
RvSystem1.SystemPreview.FormState := wsMaximized;
C++Builder
RvSystem1->SystemPreview->FormState = wsMaximized;
```

### SystemPrinter property (read/write/published)

#### Declaration

property SystemPrinter: TSystemPrinter;

#### Category

ReportSystem

#### Component/Class TRvSystem

TRVSyste

#### Description

SystemPrinter displays all the printer type options displayed in TRvRenderPrinter.

#### See also

Other SystemXxxx options

#### Example

```
Delphi
RvSystem1.SystemPrinter.MarginLeft := 0.5;
C++Builder
RvSystem1->SystemPrinter->MarginLeft = 0.5;
```

### SystemSetups property (read/write/published)

#### Declaration

property SystemSetups: TSystemSetups;

#### Default

[ssAllowSetup, ssAllowCopies, ssAllowCollate, ssAllowDuplex, ssAllowDestPreview, ssAllowDestPrinter, ssAllowDestFile, ssAllowPrinterSetup]

#### Category

ReportSystem

#### Component/Class

TRvSystem

#### 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

TSystemSetup

#### Example

Disable the setup screen

#### Delphi

```
RvSystem1.SystemSetups := RvSystem1.SystemSetups - [ssAllowSetup];
C++Builder
RvSystem1->SystemSetups = RvSystem1->SystemSetups >> ssAllowSetup;
```

## Tab method

#### Declaration

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

#### Category

Tabs

#### Component/Class

TBaseReport

#### 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*, *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

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 }
```

```
rp->Tab(-2,NA,-2,-2,NA);
rp->Print("First tab");
rp->Tab(NA,NA,-2,-2,NA);
rp->Print("Second tab");
```

### TabColor property (read/write/published)

#### Declaration

property TabColor: TColor;

#### Default

clBlack

Category Tabs

.....

Component/Class TBaseReport

#### 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

SetTab, TabShade, TColor

## TabEnd method

#### Declaration

function TabEnd(Index: integer): double;

Category

Tabs

Component/Class TBaseReport

#### 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

GetTab, TabStart, TabWidth

#### Example

End of current tab region

#### Delphi

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

## TabJustify property (read/write)

#### Declaration

property TabJustify: TTabJustify;

#### Default

tjNone

Category Tabs

Component/Class TBaseReport

#### 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.

#### Example

```
Delphi
TabJustify := tjCenter;
Println(#9'Name'#9'Number');
TabJustify := tjNone;
```

#### C++Builder

```
rp->TabJustify = tjCenter;
rp->Println("\tName\tNumber");
rp->TabJustify = tjNone;
```

### Table property (read/write/published)

Declaration
 property Table(MyPrinter: TRave);
Default
 nil
Category
 Rave
Component/Class
 TRvTableConnection
Description
 Specifies the TTable component that is connected to the TRvTableConnection component.
Example
 Delphi
 CustomerCXN.Table := CustomerQuery;
 C++Builder
 CustomerCXN->Table = CustomerQuery;

## TabShade property (read/write/published)

#### Declaration

property TabShade: integer;

#### Default

0

Category

Tabs

Component/Class TBaseReport

#### 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

<u>SetTab</u>, <u>Tab</u>

#### Example

alternate tab shading by even / odd line status

#### Delphi

```
if Odd(LineNum) then begin
  TabShade := 0;
end else begin
  TabShade := 15;
end; { else }
```

```
if ((rp->LineNum 2) == 1) {
   TabShade = 0;
}
else {
   TabShade = 15;
}/ else
```

## TabStart method

#### Declaration

function TabStart(Index: integer): double;

#### Category

Tabs

#### Component/Class

TBaseReport

#### 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

GetTab, TabEnd, TabWidth

#### Example

Start of current tab region

#### Delphi

```
CurrStart := RvNDRWriter1.TabStart( 0);
C++Builder
CurrStart = RvNDRWriter1->TabStart( 0);
```

## TabWidth method

#### Declaration

function TabWidth(Index: integer): double;

#### Category

Tabs

#### Component/Class

TBaseReport

#### 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

TabEnd, TabStart

#### Example

Width of current tab region

#### Delphi

```
CurrWidth := RvNDRWriter1.TabWidth( 0);
C++Builder
CurrWidth = RvNDRWriter1->TabWidth( 0);
```

## TAccuracyMethod type (type)

#### Declaration

TAccuracyMethod = (amPositioning, amAppearance);

#### Category

Control

#### Component/Class TBaseReport

#### Description

amPositioning:	This setting will cause the string to be written one character at a
	time.
amAppearance:	This setting will cause the whole string to be written at one time.

#### See also

AccuracyMethod

#### Example

see AccuracyMethod

## TBKMode type (type)

#### Declaration

TBKMode = (bkTransparent, bkOpaque);

#### Category

Graphics

#### Component/Class TBaseReport

#### Description

*bkTransparent: bkOpaque:* 

This setting will write the text without erasing the background. This setting will write the text after the background has been cleared.

## See also

<u>TextBKMode</u>

#### Example

See TextBKMode

### Text property (read/write)

#### Declaration

property Text: string;

#### Default

" (empty)

#### Category

Memo

#### Component/Class TMemoBuf

#### 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

<u>SetData</u>

#### Example

```
Delphi
MemoBuf1.Text := 'New text assigned into MemoBuf1';
C++Builder
MemoBuf1->Text = "New text assigned into MemoBuf1";
```

### Text property (read/write)

#### Declaration

property Text: string;

#### Category BarCode

#### Component/Class

TRpBarsBase

#### 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

Print, PrintXY, TextJustify, UseChecksum

#### Example

example of -- since "-" is not valid it will be stripped out

```
Delphi
PostNetBC1.Text := '85283-3558';
C++Builder
PostNetBC1->Text = "85283-3558";
```

### TextBKMode property (read/write/published)

#### Declaration

property TextBKMode: TBKMode;

#### Default

bkTransparent

Category

Graphics

Component/Class TBaseReport

#### 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

**BKColor** 

#### Example

```
Delphi
RvNDRWriter1.TextBKMode := bkOpaque;
C++Builder
RvNDRWriter1->TextBKMode = bkOpaque;
```

### TextJustify property (read/write)

#### Declaration

property TextJustify: TPrintJustify

Default

pjCenter

Category BarCode

Component/Class

TRpBarsBase

#### 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

PrintReadable, PrintTop, Text

### TextRect method

#### Declaration

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

#### Category

Graphics

#### Component/Class

TBaseReport

#### 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

CreateRect, All print methods, TRect

#### Example

```
Delphi
var TxtRect: TRect;
    TxtXPos: double;
    TxtYPos: double;
    Txt: string;
begin
    TxtRect := CreateRect(1.00,1.00,3.00,3.00);
    TxtXPos := 0.95;
    TxtYPos := 0.95;
    Txt := 'Text is clipped off!';
    TextRect(TxtRect, TxtXPos, TxtYPos, Txt);
end;
```

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

## TextWidth method

#### Declaration

function TextWidth(Text: string): double;

#### Category

Position

#### Component/Class TBaseReport

IbaseReput

### Description

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

#### Example

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

#### C++Builder

double TxtLen = rp->TextWidth("How long am I?");

## TFontAlign type (type)

#### Declaration

TFontAlign = (faBaseline, faTop, faBottom);

#### Category

Font

#### Component/Class TBaseReport

#### Description

faBaseline:	This setting will align the font at the baseline of the font.
faTop:	This setting will align the font at the top of the line.
faBottom:	This setting will align the font at the bottom of the line.

#### See also

<u>FontAlign</u>

#### Example

see FontAlign

### Title property (read/write/published)

#### Declaration

property Title: string;

Default

'Rave Report'

Category Misc

Component/Class TBaseReport

#### 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).

#### Example

This code causes the text "Sales Report" to show as the print job name in the print manager.

#### Delphi

RvNDRWriter1.Title := 'Sales Report'; C++Builder RvNDRWriter1->Title = "Sales Report";

### TitlePreview property (read/write/published)

#### Declaration

property TitlePreview: TFormatString;

#### Default

'Report Preview'

#### Category

ReportSystem

Component/Class TRvSystem

#### Description

This property defines the caption that will be used for the RvSystem report preview form.

See also

TitleSetup, TitleStatus

## TitleSetup property (read/write/published)

#### Declaration

property TitleSetup: TFormatString;

#### Default

'Report Setup'

#### Category

ReportSystem

#### Component/Class TRvSystem

Description

This property defines the caption that will be used for the RvSystem report setup form.

#### See also

TitlePreview, TitleStatus

## TitleStatus property (read/write/published)

#### Declaration

property TitleStatus: TFormatString;

#### Default

'Report Status'

### Category

ReportSystem

### Component/Class

TRvSystem

#### Description

This property defines the caption that will be used for the RvSystem report status form.

#### See also

TitlePreview, TitleSetup

## TLineHeightMethod type (type)

#### Declaration

TLineHeightMethod = (IhmLinesPerInch, IhmFont);

#### Category

Position

# Component/Class TBaseReport

#### Description

IhmLinesPerInch:	This setting will cause the number of lines to be fit per inch.
IhmFont:	This setting will cause the line to adjust to the font size.
IhmUser:	This setting will allow the user to define <i>LineHeight</i> directly.

#### See also

LineHeightMethod, LineHeight

#### Example

See LineHeightMethod

## TMacroID type (type)

#### Declaration

TMacroID = (midCurrDateShort, midCurrDateLong, midCurrDateUS, midCurrDateInter, midCurrTimeShort, midCurrTimeLong, midCurrTimeAMPM, midCurrTime24, midFirstPage, midLastPage, midTotalPages, midCurrentPage, midPrinterName, midDriverName, midPortName, midUser01..midUser20);

#### Category

Printing

#### **Component/Class**

TBaseReport

#### Description

midCurrDateShort:	Returns the short date format
midCurrDateLong:	Returns the long date format
midCurrDateUS:	Returns the date as MM/DD/YY
midCurrDateInter:	Returns the date as DD/MM/YY
midCurrTimeShort:	Returns the short time format
midCurrTimeLong:	Returns the long time format
midCurrTimeAMPM:	Returns the time in am/pm format
midCurrTime24:	Returns the time in 24 hour format
midFirstPage:	Returns the first page number
midLastPage:	Returns the last page number
midTotalPages:	Returns the total number of pages
midCurrentPage:	Returns the current page number
midPrinterName:	Returns the printer name
midDriverName:	Returns the driver name
midPortName:	Returns the port name
midUser01	
through	
midUser20:	Returns the n'th entry from MacroD

Returns the n'th entry from MacroData

#### See also

Macro, MacroData

#### Example

See Macro

## TMarginMethod type (type)

#### Declaration

TMarginMethod = (mmScaled, mmFixed);

#### Category

Preview

#### Component/Class

**TRvRenderPreview** 

#### Description

mmScaled:

This setting will cause the margin on the preview screen to be scaled according to MarginPercent.

m mF ixe d:

See also

MarginMethod, MarginPercent

Example

see MarginMethod

### Top property (read/write)

#### Declaration

property Top: double;

Category BarCode

Component/Cla

#### Component/Class TRpBarsBase

#### 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

BarTop, PrintReadable, PrintTop

#### Example

Print the bar code so the top is 3.5 inches down

```
Delphi
BarCode1.Top := 3.5;
C++Builder
BarCode1->Top = 3.5;
```

### TopWaste property (read only)

#### Declaration

property TopWaste: double;

#### Category

Printer

#### Component/Class

TBaseReport

#### 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

BottomWaste, LeftWaste, MarginTop, RightWaste

#### Example

See LeftWaste

## TOrientation type (type)

#### Declaration

TOrientation = (poPortrait, poLandscape, poDefault);

#### Category

Control

#### Component/Class TBaseReport

#### Description

poPortrait:	Portrait mode.
poLandscape:	Landscape mode.
poDefault:	Default mode on the current printer.

#### See also

**Orientation** 

#### Example

see Orientation example

## TotalPasses property (read/write)

#### Declaration

property TotalPasses: Integer;

#### Category

Misc

#### Component/Class TBaseReport

#### Description

This is the value that will be returned when a %s is encountered in a StatusFormat string.

#### See also

CurrentPass, StatusFormat, StatusLabel, StatusText, UpdateStatus

#### Example

```
Delphi
RvNDRWriter1.StatusFormat := 'Printing page (Pass of )';
C++Builder
RvNDRWriter1->StatusFormat = "Printing page (Pass of )";
```

## TPrintJustify type (type)

#### Declaration

TPrintJustify = (pjCenter, pjLeft, pjRight, pjBlock);

#### Category

Printing

#### Component/Class TBaseReport

### Description

pjCenter:	Center justify
pjLeft:	Justify to the left
pjRight:	Justify to the right
pjBlock:	Block (full) justify

#### See also

Justify, PrintFooter, PrintHeader, SetTab

#### Example

See SetTab

### TPrintUnits type (type)

#### Declaration

TPrintUnits = (unInch, unMM, unCM, unPoint, unUser);

#### Category

Units

#### Component/Class TBaseReport

#### 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

<u>Units</u>

#### Example

see Units

## TransparentBitmaps property (read/write)

#### Declaration

property TransparentBitmaps: boolean;

### Default

false

#### Category

Graphics

#### Component/Class

TBaseReport

#### 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

PrintBitmap, PrintBitmapRect

#### Example

```
Delphi
TransparentBitmaps := true;
C++Builder
TransparentBitmaps = true;
```

### TReportDest type (type)

#### Declaration

TReportDest = (rdPreview, rdPrinter, rdFile);

#### Category

ReportSystem

#### Component/Class TRvSystem

Try Oysten

#### 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

**DefaultDest** 

#### Example

see DefaultDest

### TruncateText property

#### Declaration

function TruncateText(Value: String; Width: Double): String;

#### Category

Printing

#### Component/Class

TBaseReport

#### 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

PrintTab, SetFont

#### Example

#### Delphi

RvNDRWriter1.SetFont( 'Arial', 14 ); TruncateText('This text is too long to fit within 2 inches', 2.0);

```
RvNDRWriter1->SetFont( "Arial", 14 );
TruncateText("This text is too long to fit within 2 inches", 2.0);
```

## TStreamMode type (type)

#### Declaration

TStreamMode = (smMemory, smTempFile, smFile, smUser);

#### Category

Control

#### Component/Class

TBaseReport

#### Description

smMemory:	This setting will use a memory stream for input and output.		
smFile:	This setting will use a file for input and output.		
smTempFile	will send the ouput 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.		
smUser:	This setting will use stream defined by user for input and output.		

#### See also

Stream, StreamMode

#### Example

See StreamMode

## TSystemOption type (type)

#### Declaration

TSystemOption = (soUseFiler, soWaitForOK, soShowStatus, soAllowPrintFromPreview, soPreviewModal);

Category

ReportSystem

#### Component/Class TRvSystem

Description

see SystemOptions

#### See also

SystemOptions

#### Example

see SystemOptions

## TSystemOptions type (type)

Declaration

TSystemOptions = Set of TSystemOption;

Category ReportSystem

Component/Class TRvSystem

Description see SystemOptions

See also SystemOptions

Example see SystemOptions

## TSystemSetup type (type)

#### Declaration

TSystemSetup = (ssAllowSetup, ssAllowCopies, ssAllowCollate, ssAllowDuplex, ssAllowDestPreview, ssAllowDestPrinter, ssAllowDestFile, ssAllowPrinterSetup);

#### Category

ReportSystem

### Component/Class

TRvSystem

#### Description

ssAllowSetup: ssAllowCopies:	If false, the setup screen will not be displayed. 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

SystemSetups

#### Example

see SystemSetups

## TSystemSetups type (type)

#### Declaration

TSystemSetups = Set of TSystemSetup;

Category

ReportSystem

#### Component/Class TRvSystem

Description see TSystemSetup

See also SystemSetups, TSystemSetup

#### Example

see SystemSetups

## TTabJustify type (type)

#### Declaration

TTabJustify = (tjCenter, tjLeft, tjRight, tjBlock, tjNone);

#### Category

Tabs

#### Component/Class TBaseReport

#### 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

TabJustify

#### Example

see TabJustify

# Underline property (read/write)

### Declaration

property Underline: boolean;

#### Default

false

# Category

Font

#### Component/Class TBaseReport

# Description

This property returns or sets the underline attribute for the current font.

# See also

Bold, Italic, Strikeout

# Example

#### Delphi

```
with RvNDRWriter1 do begin
  Underline := true;
  Print( 'Underlined text' );
  Underline := false;
end; { with }
```

#### C++Builder

```
rp->Underline = true;
rp->Print( "Underlined text" );
rp->Underline = false;
```

# Units property (read/write/published)

### Declaration

property Units: TPrintUnits;

#### Default

unInch

# Category

Units

#### Component/Class TBaseReport

#### 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*.

#### See also

UnitsFactor

#### Example

Delphi
RvNDRWriter1.Units := unInch;
C++Builder
RvNDRWriter1->Units = unInch;

# UnitsFactor property (read/write/published)

#### Declaration

property UnitsFactor: double;

#### Default

1.0

Category Units

Component/Class TBaseReport

#### 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

300 DPI conversion

#### Delphi

```
RvNDRWriter1.Units := unUser;
RvNDRWriter1.UnitsFactor := 300;
RvNDRWriter1.PrintXY( 300, 600, 'Text at 1", 2"' );
```

#### C++Builder

```
RvNDRWriterrl->Units = unUser;
RvNDRWriterl->UnitsFactor = 300;
RvNDRWriterl->PrintXY( 300, 600, "Text at 1\", 2\"" );
```

# UnregisterGraphic method

#### Declaration

procedure UnregisterGraphic( index: integer );

#### Category

Graphics

#### **Component/Class**

TBaseReport

#### 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

RegisterGraphic, ReuseGraphic

#### Example

See RegisterGraphic

# UpdateStatus method

#### Declaration

procedure UpdateStatus;

#### Category

Misc

#### Component/Class

TBaseReport

#### 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

StatusLabel, StatusText

#### Example

After report execution, depending on whether the user aborted the report's creation or not, the status bar is updated with the appropriate message.

#### Delphi

```
if Aborted then begin
   StatusFormat := #13'Report Canceled!';
   UpdateStatus;
end else begin
   StatusFormat := #13'Report Completed!';
   UpdateStatus;
end; { else }
```

#### C++Builder

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

# UseChecksum property (read/write)

### Declaration

property UseChecksum: boolean

### Default

false (Code128 := true)

# Category

BarCode

Component/Class TRpBarsBase

# Description

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

# See also

BarHeight, BarWidth, PrintReadable, Text, Width

# UseCompression property (read/write)

### Declaration

property UseCompression: boolean read FCompression write FCompression

### Default

false

# Category

Render

# Component/Class

TRpRender

# 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

**OnCompress** 

# UseSetRange property (read/write/published)

### Declaration

property UseSetRange: boolean;

#### Default

false

#### Category Rave

Tave

# Component/Class

TRvTableConnection

# Description

This property will determine whether filters are handled by the TTable.Filter property or the TTable.SetRange method.

# Version property (read/write/published)

### Declaration

property Version: String;

#### Category Misc

Component/Class TRpComponent

### Description

Returns the current release version of Rave.

# WideFactor property (read/write)

### Declaration

property WideFactor: double

### Default

3.0

Category BarCode

Darcout

Component/Class TRpBarsBase

Description

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

See also

BarHeight, BarWidth, Width

# Example

set wide to narrow bar ratio to be 2.5

# Delphi

WideFactor := 2.5; C++Builder WideFactor = 2.5;

# Width property (read only)

#### Declaration

property Width: double;

#### Category

BarCode

# Component/Class

TRpBarsBase

### Description

This property will return the calculated width of the entire bar code for the current value of Text.

# See also

BarWidth, Text, WideFactor

### Example

get width of bar code for ABC123

#### Delphi

```
var BarCodeWidth: double;
BarCode1.Text := 'ABC123';
BarCodeWidth := BarCode1.Width;
```

#### C++Builder

double BarCodeWidth; BarCode1->Text = "ABC123"; BarCodeWidth = BarCode1->Width;

# WriteBCDData method

#### Declaration

function WriteBCDData(FormatData: String; NativeData: Currency): String;

#### Category

Rave

#### Component/Class

**TRvCustomConnection** 

#### 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.

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
Tutorial:	see "customizing data connections" tutorial for more information.

#### See also

OnGetCols, OnGetRow, other WriteXxxxData methods

#### Example

```
Delphi
Connection.WriteBCDData( ' ',InvoiceAmount );
C++Builder
Connection->WriteBCDData( " ",InvoiceAmount );
```

# WriteBlobData method

#### Declaration

function WriteBlobData(var: Buffer; Len: Longint): String;

#### Category

Rave

#### Component/Class

TRvCustomConnection

#### 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

OnGetCols, OnGetRow, other WriteXxxxData methods

```
Delphi
Connection.WriteBlobData( '',CustomerPict );
C++Builder
Connection->WriteBlobData( "",CustomerPict );
```

# WriteBoolData method

### Declaration

function WriteBoolData(FormatData: String; NativeData: Boolean): String;

#### Category

Rave

### Component/Class

TRvCustomConnection

#### Description

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 preformatted output is needed.

NativeData parameter should contain the unmodified contents of the field

### See also

OnGetCols, OnGetRow, other WriteXxxxData methods

#### Example

```
Delphi
Connection.WriteBoolData( '',CustomerActive );
C++Builder
Connection->WriteBoolData( "",CustomerActive );
```

# WriteCurrData method

#### Declaration

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

#### Category

Rave

#### Component/Class

TRvCustomConnection

#### 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

OnGetCols, OnGetRow, other WriteXxxxData methods

```
Delphi
Connection.WriteCurrData( '',InvoiceAmount );
C++Builder
Connection->WriteCurrData( "",InvoiceAmount );
```

# WriteDateTime method

# Declaration

function WriteDateTime(FormatData: String; NativeData: TDateTime);

#### Category

Rave

### Component/Class

TRvCustomConnection

#### 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 preformatted output is needed.

NativeData parameter should contain the unmodified contents of the field

#### See also

OnGetCols, OnGetRow, other WriteXxxxData methods

#### Example

```
Delphi
```

```
Connection.WriteDateTime( '',Now );
C++Builder
Connection->WriteDateTime( "",Now );
```

# WriteFloatData method

#### Declaration

function WriteFloatData(FormatData: String; NativeData: Extended): String;

#### Category

Rave

#### Component/Class

TRvCustomConnection

#### 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 preformatted output is needed.

NativeData parameter should contain the unmodified contents of the field

#### See also

OnGetCols, OnGetRow, other WriteXxxxData methods

```
Delphi
Connection.WriteFloatData( '',CustomerBudget );
C++Builder
Connection->WriteFloatData( "",CustomerBudget );
```

# WriteIntData method

### Declaration

function WriteIntData(FormatData: String; NativeData: Integer): String;

#### Category

Rave

#### Component/Class

**TRvCustomConnection** 

#### 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-

FormatData parameter defines the formatted value of the field, but can be blank if no preformatted output is needed.

NativeData parameter should contain the unmodified contents of the field

### See also

OnGetCols, OnGetRow, other WriteXxxxData methods

Example

```
Delphi
Connection.WriteIntData( '',CustomerCount );
C++Builder
Connection->WriteIntData( "",CustomerCount );
```

# WriteNullData method

#### Declaration

function WriteNullData( no parameters );

#### Category Rave

Rave

# Component/Class

TRvCustomConnection

#### 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

OnGetCols, OnGetRow, other WriteXxxxData methods

```
Delphi
Connection.WriteNullData();
C++Builder
Connection->WriteNullData();
```

# WriteStrData method

### Declaration

function WriteStrData(FormatData: String; NativeData: String): String;

#### Category

Rave

#### Component/Class

**TRvCustomConnection** 

#### 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 preformatted output is needed.

NativeData parameter should contain the unmodified contents of the field

### See also

OnGetCols, OnGetRow, other WriteXxxxData methods

#### Example

```
Delphi
Connection.WriteStrData( '',CustomerName );
C++Builder
Connection->WriteStrData( "",CustomerName );
```

# XD2I method

#### Declaration

function XD2I(Pos: longint): double;

# Category

Units

# Component/Class

TRvRenderPreview

#### Description

This method will convert horizontal printer canvas measurements (dots) to inch measurements.

# See also

All other units conversion functions

#### Example

With Units currently set to unlnch

#### Delphi

```
XPos := RvNDRWriter1.XD2I( LastXDots );
C++Builder
XPos = RvNDRWriter1->XD2I( LastXDots );
```

# XD2U method

# Declaration

function XD2U(Pos: longint): double;

# Category

Units

# Component/Class

TBaseReport

# Description

This method will convert horizontal printer canvas measurements (dots) to unit measurements (defined by *Units* and *UnitsFactor*).

# See also

Units, UnitsFactor, All other units conversion functions

# Example

```
Delphi
XPos := RvNDRWriter1.XD2U( LastXDots );
C++Builder
XPos = RvNDRWriter1->XD2U( LastXDots );
```

# XDPI property (read only)

```
Declaration

property XDPI: integer;

Category

Printer

Component/Class

TBaseReport

Description

This property returns the horizontal dots per inch for the current printer.

Example

Delphi
```

```
CurrXDPI := RvNDRWriter1.XDPI;
C++Builder
CurrXDPI = RvNDRWriter1->XDPI;
```

# XI2D method

# Declaration

function XI2D(Pos: double): longint;

### Category

Units

# Component/Class

TBaseReport

### Description

This method will convert horizontal inch measurements to printer canvas measurements (dots).

### See also

All other units conversion functions

# Example

With Units currently set to unInch

#### Delphi

```
CurrXDots := RvNDRWriter1.XI2D( RvNDRWriter1.XPos );
C++Builder
CurrXDots = RvNDRWriter1->XI2D( RvNDRWriter1->XPos );
```

# XI2U method

### Declaration

function XI2U(Pos: double): double;

#### Category

Units

### Component/Class

TBaseReport

#### Description

This method will convert horizontal inch measurements to unit measurements (defined by *Units* and *UnitsFactor*).

#### See also

Units, UnitsFactor, All other units conversion functions

```
Delphi
XPos := RvNDRWriter1.XI2U( LastXInch );
C++Builder
XPos = RvNDRWriter1->XI2U( LastXInch );
```

# XPos property (read/write)

### Declaration

property XPos: double;

#### Default

0.0

# Category

Position

Component/Class TBaseReport

### Description

This property sets or returns the horizontal text cursor position.

### See also

CursorXPos, CursorYPos, YPos

# Example

Delphi
XPos := 0.45;
YPos := 0.95;
Print('Text at ( 0.45, 0.95 )');

### C++Builder

rp->XPos = 0.45; rp->YPos = 0.95; rp->Print("Text at ( 0.45, 0.95 )");

# XU2D method

#### Declaration

function XU2D(Pos: double): longint;

#### Category

Units

#### Component/Class

TBaseReport

#### Description

This method will convert horizontal unit measurements (defined by *Units* and *UnitsFactor*) to printer canvas measurements (dots).

# See also

Units, UnitsFactor, All other units conversion functions

```
Delphi
CurrXDots := RvNDRWriter1.XU2D( RvNDRWriter1.XPos );
C++Builder
CurrXDots = RvNDRWriter1->XU2D(RvNDRWriter1->XPos );
```

# XU2I method

# Declaration

function XU2I(Pos: double): double;

### Category

Units

# Component/Class

TBaseReport

### Description

This method will convert horizontal unit measurements (defined by *Units* and *UnitsFactor*) to inch measurements.

### See also

Units, UnitsFactor, All other units conversion functions

# Example

With units set to unCM

#### Delphi

```
CurrXInch := RvNDRWriter1.XU2I( RvNDRWriter1.XPos );
C++Builder
CurrXInch = RvNDRWriter1->XU2I( RvNDRWriter1->XPos );
```

# YD2I method

### Declaration

function YD2I(Pos: longint): double;

#### Category

Units

### Component/Class

TBaseReport

#### Description

This method will convert vertical printer canvas measurements (dots) to inch measurements

#### See also

All other units conversion functions

# Example

With Units currently set to unInch

#### Delphi

```
YPos := RvNDRWriter1.YD2I( LastYDots );
C++Builder
YPos = RvNDRWriter1->YD2I( LastYDots );
```

# YD2U method

# Declaration

function YD2U(Pos: longint): double;

### Category

Units

# Component/Class

TBaseReport

### Description

This method will convert vertical printer canvas measurements (dots) to unit measurements (defined by *Units* and *UnitsFactor*).

# See also

Units, UnitsFactor, All other units conversion functions

# Example

```
Delphi
RvNDRWriter1.YPos = RvNDRWriter1.YD2U( LastYDots );
C++Builder
RvNDRWriter1->YPos = RvNDRWriter1->YD2U( LastYDots );
```

# YDPI property (read only)

Declaration

property YDPI: integer;

Category

Printer

#### Component/Class TBaseReport

Description

This property returns the vertical dots per inch for the current printer.

# See also

All other units conversion functions

# Example

Delphi
CurrYDPI := RvNDRWriter1.YDPI;
C++Builder
CurrYDPI = RvNDRWriter1->YDPI;

# YI2D method

# Declaration

function YI2D(Pos: double): longint;

# Category

Units

#### Component/Class TBaseReport

# Description

This method will convert vertical inch measurements to printer canvas measurements (dots).

### See also

All other units conversion functions

# Example

With Units currently set to unInch

# Delphi

CurrYDots := RvNDRWriter1.YI2D( YPos ); C++Builder CurrYDots = RvNDRWriter1->YI2D( RvNDRWriter1->YPos );

# YI2U method

# Declaration

function YI2U(Pos: double): double;

# Category

Units

#### Component/Class TBaseReport

# Description

This method will convert vertical inch measurements to unit measurements (defined by *Units* and *UnitsFactor*).

# See also

Units, UnitsFactor, All other units conversion functions

```
Delphi
RvNDRWriter1.YPos := RvNDRWriter1.YI2U( LastYInch );
C++Builder
RvNDRWriter1->YPos = RvNDRWriter1->YI2U( LastYInch );
```

# YPos property (read/write)

### Declaration

property YPos: double;

#### Default

0.0

# Category

Position

Component/Class TBaseReport

### Description

This property sets or returns the vertical text cursor position.

### See also

CursorXPos, CursorYPos, XPos

# Example

Delphi
XPos := 0.45;
YPos := 0.95;
Print('Text at ( 0.45, 0.95 )');

# C++Builder

rp->XPos = 0.45; rp->YPos = 0.95; rp->Print("Text at ( 0.45, 0.95 )");

# YU2D method

#### Declaration

function YU2D(Pos: double): longint;

#### Category

Units

#### Component/Class

TBaseReport

#### Description

This method will convert vertical unit measurements (defined by *Units* and *UnitsFactor*) to printer canvas measurements (dots).

# See also

Units, UnitsFactor, All other units conversion functions

```
Delphi
CurrYDots := RvNDRWriter1.YU2D( RvNDRWriter1.YPos );
C++Builder
CurrYDots = RvNDRWriter1->YU2D( RvNDRWriter1->YPos );
```

# YU2I method

# Declaration

function YU2I(Pos: double): double;

### Category

Units

# Component/Class

TBaseReport

### Description

This method will convert vertical unit measurements (defined by *Units* and *UnitsFactor*) to inch measurements.

# See also

Units, UnitsFactor, All other units conversion functions

### Example

With units set to unCM

#### Delphi

CurrYInch := RvNDRWriter1.YU2I( RvNDRWriter1.YPos ); C++Builder CurrYInch = RvNDRWriter1->YU2I( RvNDRWriter1->YPos );

# ZoomFactor property (read/write/published)

#### Declaration

property ZoomFactor: double;

#### Default

100.0

Category

Preview

### Component/Class

TRvRenderPreview

#### 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

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.

#### Delphi

```
var S1: string[10];
begin
   Str(RvRenderPreview1.ZoomFactor:1:1,S1);
   ZoomEdit.Text := S1;
   RvRenderPreview1.RedrawPage;
end;
```

#### C++Builder

```
AnsiString S1;
S1 = FloatToStrF(RvRenderPreview1->ZoomFactor, ffGeneral,1,1);
ZoomEdit->Text = S1;
RvRenderPreview1->RedrawPage();
```

# ZoomIn method

#### Declaration

procedure ZoomIn;

#### Category

Preview

#### Component/Class

**TRvRenderPreview** 

#### 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

ZoomOut, ZoomInc, ZoomFactor, OnZoomChange

#### Example

This code causes the ZoomFactor to be incremented by ZoomInc percent.

# Delphi

RvRenderPreview1.ZoomIn; C++Builder RvRenderPreview1->ZoomIn();

# Zoominc property (read/write/published)

#### Declaration

property ZoomInc: integer;

#### Default

10

# Category

Preview

# Component/Class

TRvRenderPreview

#### Description

This property defines the amount that ZoomIn and ZoomOut modifies ZoomFactor.

#### See also

ZoomFactor, ZoomIn, ZoomOut

# Example

This code causes the *ZoomFactor* property to be incremented by 10when *ZoomIn* and *ZoomOut* are called.

#### Delphi

```
RvRenderPreview1.ZoomInc := 10;
C++Builder
RvRenderPreview1->ZoomInc = 10;
```

# ZoomOut method

#### Declaration

procedure ZoomOut;

#### Category

Preview

### Component/Class

TRvRenderPreview

#### Description

This method will subtract *ZoomInc* from the current *ZoomFactor* and will make the image smaller 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

Zoomln, Zoomlnc, ZoomFactor, OnZoomChange

# Example

Delphi
RvRenderPreview1.ZoomOut;
C++Builder
RvRenderPreview1->ZoomOut();

# ZoomPageFactor property (read only)

#### Declaration

property ZoomPageFactor: double;

#### Category

Preview

#### Component/Class

**TRvRenderPreview** 

#### 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

ShadowDepth, ZoomFactor, ZoomPageWidthFactor

### Example

use an OnPreviewShow event with the following

#### Delphi

```
with Sender As TRvRenderPreview do begin
ZoomFactor := ZoomPageFactor - (ShadowDepth + 5) / 10;
end; { with }
```

#### C++Builder

```
TRvRenderPreview* fp = dynamic_cast<TRvRenderPreview*>(Sender);
  fp->ZoomFactor = fp->ZoomPageFactor - (fp->ShadowDepth + 5) / 10;
```

# ZoomPageWidthFactor property (read only)

#### Declaration

property ZoomPageWidthFactor: double;

#### Category

Preview

#### Component/Class

**TRvRenderPreview** 

#### 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

ShadowDepth, ZoomFactor, ZoomPageFactor

### Example

use an OnPreviewShow event with the following

#### Delphi

```
with Sender As TRvRenderPreview do begin
ZoomFactor := ZoomPageWidthFactor - (ShadowDepth +3) / 10;
end; { with }
```

#### C++Builder

```
TRvRenderPreview* fp = dynamic_cast<TRvRenderPreview*>(Sender);
    fp->ZoomFactor = fp->ZoomPageWidthFactor - (fp->ShadowDepth + 3) /
10;
```

# INDEX

Α	
AlphaNumeric Items	23
C	
Controlling the Visibility of the connection	11
Custom Data Connections	11
Customizing Data Connections	13
D	
Data Bridge	11
Date / Time items	24
Description9,	17
E	
Engine Property	21
event	
OnAfterClose1	
OnAfterOpen1	
OnAfterPrint1	
OnBeforeClose	
OnBeforeOpen1	
OnBeforePrint1	
OnCreate	
OnDecodeImage1	
OnDesignerSave	
OnDesignerSaveAs1	
OnDesignerShow1	
OnDestroy1 OnEOF1	
OnEOF	
OnGetCols1	
OnGetRow1	
OnGetSorts1	
OnNewColumn1	
OnNewPage1	
OnNext1	
OnOpen1	
OnPageChange1	
OnPreviewSetup1	
OnPreviewShow1	
OnPrint1	
OnPrintFooter1	61
OnPrintHeader1	62
OnPrintPage1	63
OnRestore1	64
OnSetFilter1	64
OnSetSort1	
OnValidateRow1	65

OnZoomChange16 OverridePreview17	
OverrideSetup17	
OverrideStatus17	
Events	
F	
Formatting2	23
Getting Started	3
I	
Introduction to RaveK	
Keyboard / Mouse Shortcuts2	
Limited Warranty	.3
method	
Abort	29
AbortPage	31
AdjustLine	
AllowAll	
AllowPreviewOnly	
AllowPrinterOnly	
Append	
AppendMemoBuf	
Arc	
AssignFont	
BrushCopy	
CalcGraphicHeight	
CalcGraphicWidth	
Chord	
Clear	
ClearAllTabs	
ClearColumns	55
ClearRaveBlob	
ClearTabs	56
Close	56
ConstraintHeightLeft6	
CopyRect	
CR6	
Create	37
CreateBrush6	
CreateFont6	
CreatePen	
CreatePoint	

CreateRect	71
Delete	74
Design	75
DesignReport	76
Destroy	76
Draw	80
DrawFocusRect	79
Ellipse	83
Empty	
Execute	85
ExecuteCustom	85
ExecuteReport	86
FillRect	89
Finish	89
FinishTabBox	90
FloodFill	91
FrameRect1	02
FreeSaved1	02
GetMemoLine1	
GetNextLine1	03
GetReportCategoryList1	
GetReportList1	
GetTab1	
GotoFooter1	05
GotoHeader1	06
GotoXY1	
GraphicFieldToBitmap1	
Home1	
Insert1	
InsertMemoBuf1	
IsValidChar1	
LF119	
LinesLeft1	23
LineTo1	24
LoadFromFile1	
LoadFromStream1	
LoadRaveBlob1	
Macro1	29
MemoHeightLeft1	
MemoLines1	
MemoLinesLeft1	
MoveTo1	
NewColumn1	
NewLine1	
NewPage1	
NextPage1	

NoPrinters       145         Open       167         Pie       175         Polygon       177         Polyline       178         PopFont       179         PopFos       180         PopTabs       180         PrevPage       182         Print       183         PrintBitmap       184         PrintBitmapRect       185         PrintCenter       186         PrintCharJustify       186         PrintDataStream       188         PrintFimA       190         PrintFimB       191         PrintFooter       192         PrintHeader       192         PrintHeight       193         PrintLeft       196         PrintLeft       196         PrintLeft       197         PrintBageRect       194         PrintMemo       198         PrintPage       199         PrintRight       200         PrintTab       201         PrintAge       201         PrintAge       202         PushFont       203         PushFos       204		
Pie       175         Polygon       177         Polyline       178         PopFont       179         PopPos       180         PopTabs       180         ProvPage       182         Print       183         PrintBitmap       184         PrintBlock       185         PrintCenter       186         PrintData       187         PrintData       187         PrintData       187         PrintFimA       190         PrintFimB       191         PrintFimC       191         PrintFimB       191         PrintFimG       192         PrintHeader       192         PrintHeader       193         PrintILines       196         PrintLeft       196         PrintLame       197         PrintMemo       198         PrintRight       200         PrintRight       200         PrintTab       201         PrintXY       202         PushFont       203         PushPos       204         RecoverPrinter       206         Rectangl	NoPrinters	.145
Pie       175         Polygon       177         Polyline       178         PopFont       179         PopPos       180         PopTabs       180         ProvPage       182         Print       183         PrintBitmap       184         PrintBlock       185         PrintCenter       186         PrintData       187         PrintData       187         PrintData       187         PrintFimA       190         PrintFimB       191         PrintFimC       191         PrintFimB       191         PrintFimG       192         PrintHeader       192         PrintHeader       193         PrintILines       196         PrintLeft       196         PrintLame       197         PrintMemo       198         PrintRight       200         PrintRight       200         PrintTab       201         PrintXY       202         PushFont       203         PushPos       204         RecoverPrinter       206         Rectangl	Open	.167
Polygon       177         Polyline       178         PopFont       179         PopPos       180         PopTabs       180         PrevPage       182         Print       183         PrintBitmap       184         PrintBitmapRect       185         PrintCenter       186         PrintCharJustify       186         PrintCharJustify       186         PrintThata       187         PrintBeac       188         PrintFimA       190         PrintFimB       191         PrintFooter       192         PrintHeader       192         PrintHeight       193         PrintLeft       196         PrintLeft       196         PrintLeft       197         PrintMemo       198         PrintRight       200         PrintRight       201         PrintRight       202         PushFont       203         PushFos       204         PushTabs       204         PushTabs       204         PrintRight       206         PrintRight       207		
Polyline       178         PopFont       179         PopPos       180         ProvPage       182         Print       183         PrintBitmap       184         PrintBitmapRect       185         PrintCenter       186         PrintCharJustify       186         PrintData       187         PrintData       187         PrintFimA       190         PrintFimB       191         PrintFooter       192         PrintHeight       193         PrintHeight       193         PrintLeft       196         PrintLines       197         PrintRight       198         PrintLapge       199         PrintRight       200         PrintRight       201         PrintRight       202         PushFont       203         PushFont       203         PushFos       204         PushTabs       207         RecoverPrinter       206         Rectangle       207         ReleasePrinter       208         ReleasePrinter       209         ReportDescToMemo       21	Polygon	.177
PopFont       179         PopPos       180         PopTabs       180         PrevPage       182         Print       183         PrintBitmap       184         PrintBitmapRect       185         PrintChers       186         PrintCharJustify       186         PrintCharJustify       186         PrintData       187         PrintDataStream       188         PrintFimA       190         PrintFimB       191         PrintFooter       192         PrintHeader       192         PrintHeader       193         PrintLeft       196         PrintLeft       196         PrintLeft       196         PrintLans       197         PrintMemo       198         PrintPage       199         PrintRight       200         PrintTab       201         PrintXY       202         PushFont       203         PushFos       204         PushTabs       204         PecoverPrinter       206         Rectangle       207         RegisterGraphic       208		
PopPos       180         PopTabs       180         PrevPage       182         Print       183         PrintBitmap       184         PrintBitmapRect       185         PrintBlock       185         PrintCenter       186         PrintCharJustify       186         PrintCharJustify       188         PrintData       187         PrintDataStream       188         PrintFimA       190         PrintFimB       191         PrintFooter       192         PrintHeader       192         PrintHeader       193         PrintLotstify       196         PrintLeft       196         PrintLeft       196         PrintLeft       197         PrintMemo       198         PrintPage       199         PrintRight       200         PrintXY       202         PushFont       203         PushFos       204         PushTabs       204         PushTabs       207         RedrawPage       207         RegisterGraphic       208         ReleasePrinter		
PopTabs       180         PrevPage       182         Print       183         PrintBitmap       184         PrintBitmapRect       185         PrintBlock       185         PrintCenter       186         PrintCharJustify       186         PrintData       187         PrintDataStream       188         PrintFimA       190         PrintFimB       191         PrintFooter       192         PrintHeader       192         PrintHeader       193         PrintLottstify       196         PrintLeft       196         PrintLotes       197         PrintLeft       196         PrintLeft       196         PrintLames       197         PrintBege       199         PrintRight       200         PrintTab       201         PrintXY       202         PushFont       203         PushPos       204         PushTabs       204         RecoverPrinter       206         Rectangle       207         RedrawPage       207         ReplaceAll       210 <td></td> <td></td>		
PrevPage       182         Print       183         PrintBitmap       184         PrintBitmapRect       185         PrintBlock       185         PrintCenter       186         PrintCharJustify       186         PrintData       187         PrintDataStream       188         PrintFimA       190         PrintFimB       191         PrintFooter       192         PrintHeader       192         PrintHeight       193         PrintLeft       196         PrintLeft       196         PrintLeft       197         PrintMemo       198         PrintPage       199         PrintRight       200         PrintTab       201         PrintXY       202         PushFont       203         PushPos       204         PushTabs       204         RecoverPrinter       206         Rectangle       207         RedrawPage       207         RedrawPage       207         ReplaceAll       210         ReportDescToMemo       211         ResetLineHeight       <		
Print       183         PrintBitmap       184         PrintBitmapRect       185         PrintBlock       185         PrintCenter       186         PrintCharJustify       186         PrintData       187         PrintDataStream       188         PrintFimA       190         PrintFimB       191         PrintFooter       192         PrintHeader       192         PrintHeader       192         PrintLageRect       194         PrintLustify       196         PrintLageRect       194         PrintLageRect       194         PrintLageRect       194         PrintLageRect       196         PrintLage       197         PrintLage       197         PrintLage       197         PrintRight       200         PrintRight       200         PrintRight       201         PrintXY       202         PushFont       203         PushPos       204         PushTabs       204         Rectangle       207         RedrawPage       207         RegisterGraphic		
PrintBitmap       184         PrintBitmapRect       185         PrintBlock       185         PrintCenter       186         PrintCharJustify       186         PrintData       187         PrintData       187         PrintDataStream       188         PrintFimA       190         PrintFimB       191         PrintFooter       192         PrintHeader       192         PrintHeight       193         PrintLeght       193         PrintLeft       196         PrintLines       197         PrintLeft       196         PrintLines       197         PrintMemo       198         PrintPage       199         PrintRight       200         PrintTab       201         PrintRight       202         PushFont       203         PushFos       204         PushTabs       204         RecoverPrinter       206         Rectangle       207         RegisterGraphic       208         ReleasePrinter       209         ReportDescToMemo       211         ResetLine		
PrintBitmapRect.       185         PrintBlock.       185         PrintCenter       186         PrintCharJustify       186         PrintData       187         PrintDataStream       188         PrintFimA       190         PrintFimB       191         PrintFooter       192         PrintHeader       192         PrintHeader       193         PrintLeght       193         PrintLeft       196         PrintLines       197         PrintLeft       196         PrintLines       197         PrintBege       198         PrintPage       199         PrintRight       200         PrintTab       201         PrintTab       202         PushFont       203         PushFont       203         PushPos       204         PushTabs       207         Rectangle       207         RedrawPage       207         ReplaceAll       210         ReportDescToMemo       211         Reset       213, 214		
PrintBlock.       185         PrintCenter       186         PrintCharJustify       186         PrintData       187         PrintDataStream       188         PrintFimA       190         PrintFimB       191         PrintFooter       192         PrintHeader       192         PrintHeight       193         PrintLeft       196         PrintLeft       196         PrintLeft       197         PrintMemo       198         PrintPage       199         PrintRight       200         PrintTab       201         PrintXY       202         PushFont       203         PushFos       204         PushTabs       207         Rectangle       207         RedrawPage       207         RegisterGraphic       208         ReleasePrinter       209         ReplaceAll       210         Reset       213, 214		
PrintCenter       186         PrintCharJustify       187         PrintData       187         PrintDataStream       188         PrintFimA       190         PrintFimB       191         PrintFooter       192         PrintHeader       192         PrintHeight       193         PrintLines       194         PrintLines       195         PrintLeft       196         PrintLanes       197         PrintMemo       198         PrintPage       199         PrintRight       200         PrintTab       201         PrintXY       202         PushFont       203         PushFos       204         RecoverPrinter       206         Rectangle       207         RedrawPage       207         RedrawPage       207         ReportDescToMemo       211         Reset       213, 214		
PrintCharJustify       186         PrintData       187         PrintDataStream       188         PrintFimA       190         PrintFimB       191         PrintFooter       192         PrintHeader       192         PrintHeight       193         PrintJustify       196         PrintLeft       196         PrintLeft       197         PrintLeft       196         PrintLeft       197         PrintMemo       198         PrintPage       199         PrintRight       200         PrintTab       201         PrintXY       202         PushFont       203         PushTabs       204         RecoverPrinter       206         Rectangle       207         RedrawPage       207         RegisterGraphic       208         ReleasePrinter       209         ReplaceAll       210         ReportDescToMemo       211         Reset       213, 214		
PrintData       187         PrintDataStream       188         PrintFimA       190         PrintFimB       191         PrintFimC       191         PrintFooter       192         PrintHeader       192         PrintHeight       193         PrintJustify       196         PrintLeft       196         PrintLeft       197         PrintLeft       197         PrintMemo       198         PrintPage       199         PrintRight       200         PrintTab       201         PrintXY       202         PushFont       203         PushFos       204         PushTabs       204         RecoverPrinter       206         Rectangle       207         RedrawPage       207         RegisterGraphic       208         ReleasePrinter       209         ReplaceAll       210         Reset       213, 214         Reset       213, 214		
PrintDataStream       188         PrintFimA       190         PrintFimB       191         PrintFimC       191         PrintFooter       192         PrintHeader       192         PrintHeight       193         PrintJustify       196         PrintLines       197         PrintLines       197         PrintRight       193         PrintPage       199         PrintRight       200         PrintTab       201         PrintXY       202         PushFont       203         PushFos       204         PushTabs       204         RecoverPrinter       206         Rectangle       207         RedrawPage       207         RegisterGraphic       208         ReleasePrinter       209         ReplaceAll       210         ReportDescToMemo       211         Reset       213, 214	•	
PrintFimA       190         PrintFimB       191         PrintFimC       191         PrintFooter       192         PrintHeader       192         PrintHeight       193         PrintJustify       196         PrintLeft       196         PrintLeft       197         PrintMemo       198         PrintPage       199         PrintRight       200         PrintTab       201         PrintXY       202         PushFont       203         PushPos       204         PushTabs       204         RecoverPrinter       206         Rectangle       207         RedrawPage       208         ReleasePrinter       209         ReplaceAll       210         ReportDescToMemo       211         Reset       213, 214		
PrintFimB       191         PrintFimC       191         PrintFooter       192         PrintHeader       192         PrintHeight       193         PrintJustify       196         PrintLeft       196         PrintLeft       196         PrintLeft       197         PrintMemo       198         PrintPage       199         PrintRight       200         PrintTab       201         PrintXY       202         PushFont       203         PushFos       204         PecoverPrinter       206         Rectangle       207         RedrawPage       209         ReleasePrinter       209         ReplaceAll       210         ReportDescToMemo       211         Reset       213, 214         ResetLineHeight       214		
PrintFimC       191         PrintFooter       192         PrintHeader       192         PrintHeight       193         PrintImageRect       194         PrintJustify       196         PrintLeft       196         PrintLeft       197         PrintLeft       197         PrintMemo       198         PrintPage       199         PrintRight       200         PrintTab       201         PrintXY       202         PushFont       203         PushPos       204         PushTabs       204         RecoverPrinter       206         Rectangle       207         RedrawPage       207         ReportDescToMemo       211         Reset       213, 214         ResetLineHeight       214		
PrintFooter       192         PrintHeader       192         PrintHeight       193         PrintImageRect       194         PrintJustify       196         PrintLeft       196         PrintLeft       196         PrintLines       197         PrintLn       197         PrintPage       199         PrintRight       200         PrintTab       201         PrintXY       202         PushFont       203         PushPos       204         PushTabs       204         RecoverPrinter       206         Rectangle       207         RedrawPage       207         ReplaceAll       210         ReportDescToMemo       211         Reset       213, 214         ResetLineHeight       214		
PrintHeader       192         PrintHeight       193         PrintImageRect       194         PrintJustify       196         PrintLeft       196         PrintLines       197         PrintLeft       197         PrintMemo       198         PrintPage       199         PrintRight       200         PrintTab       201         PrintXY       202         PushFont       203         PushFos       204         RecoverPrinter       206         Rectangle       207         RegisterGraphic       208         ReleasePrinter       209         ReplaceAll       210         ReportDescToMemo       211         Reset       213, 214		
PrintHeight       193         PrintImageRect.       194         PrintJustify       196         PrintLeft       197         PrintLines       197         PrintMemo       198         PrintPage       199         PrintRight       200         PrintTab       201         PrintXY       202         PushFont       203         PushPos       204         RecoverPrinter       206         Rectangle       207         RedrawPage       209         ReleasePrinter       209         ReplaceAll       210         ReportDescToMemo       211         Reset       213, 214         ResetLineHeight       214		
PrintImageRect.       194         PrintJustify       196         PrintLeft       196         PrintLines       197         PrintLn       197         PrintMemo       198         PrintPage       199         PrintRight       200         PrintTab       201         PrintXY       202         PushFont       203         PushPos       204         RecoverPrinter       206         Rectangle       207         RegisterGraphic       208         ReleasePrinter       209         ReplaceAll       210         Reset       213, 214         ResetLineHeight       214		
PrintJustify       196         PrintLeft       196         PrintLines       197         PrintLn       197         PrintMemo       198         PrintPage       199         PrintRight       200         PrintTab       201         PrintXY       202         PushFont       203         PushFos       204         RecoverPrinter       206         Rectangle       207         RedrawPage       207         ReleasePrinter       208         ReleasePrinter       209         ReplaceAll       210         Reset       213, 214         ResetLineHeight       214		
PrintLeft       196         PrintLines       197         PrintLn       197         PrintMemo       198         PrintPage       199         PrintRight       200         PrintTab       201         PrintXY       202         PushFont       203         PushPos       204         PushTabs       204         RecoverPrinter       206         Rectangle       207         RedrawPage       207         ReplaceAll       210         ReportDescToMemo       211         Reset       213, 214         ResetLineHeight       214	-	
PrintLines       197         PrintLn       197         PrintMemo       198         PrintPage       199         PrintRight       200         PrintTab       201         PrintXY       202         PushFont       203         PushPos       204         PushTabs       204         RecoverPrinter       206         Rectangle       207         RedrawPage       207         ReleasePrinter       209         ReplaceAll       210         Reset       213, 214         ResetLineHeight       214	-	
PrintLn       197         PrintMemo       198         PrintPage       199         PrintRight       200         PrintTab       201         PrintXY       202         PushFont       203         PushFos       204         PushTabs       204         RecoverPrinter       206         Rectangle       207         RedrawPage       207         ReleasePrinter       209         ReplaceAll       210         Reset       213, 214         ResetLineHeight       214		
PrintMemo       198         PrintPage       199         PrintRight       200         PrintTab       201         PrintXY       202         PushFont       203         PushPos       204         PushTabs       204         RecoverPrinter       206         Rectangle       207         RegisterGraphic       208         ReleasePrinter       209         ReplaceAll       210         Reset       213, 214         ResetLineHeight       214		
PrintPage       199         PrintRight       200         PrintTab       201         PrintXY       202         PushFont       203         PushPos       204         PushTabs       204         RecoverPrinter       206         Rectangle       207         RegisterGraphic       208         ReleasePrinter       209         ReplaceAll       210         Reset       213, 214         ResetLineHeight       214		
PrintRight       200         PrintTab       201         PrintXY       202         PushFont       203         PushFos       204         PushTabs       204         RecoverPrinter       206         Rectangle       207         RedrawPage       207         RegisterGraphic       208         ReleasePrinter       209         ReplaceAll       210         Reset       213, 214         ResetLineHeight       214		
PrintTab       201         PrintXY       202         PushFont       203         PushPos       204         PushTabs       204         RecoverPrinter       206         Rectangle       207         RegisterGraphic       208         ReleasePrinter       209         ReplaceAll       210         Reset       213, 214         ResetLineHeight       214		
PrintXY       202         PushFont       203         PushPos       204         PushTabs       204         RecoverPrinter       206         Rectangle       207         RedrawPage       207         RegisterGraphic       208         ReleasePrinter       209         ReplaceAll       210         Reset       213, 214         ResetLineHeight       214	PrintRight	.200
PushFont203PushPos204PushTabs204RecoverPrinter206Rectangle207RedrawPage207RegisterGraphic208ReleasePrinter209ReplaceAll210Reset213, 214ResetLineHeight214	PrintTab	.201
PushPos204PushTabs204RecoverPrinter206Rectangle207RedrawPage207RegisterGraphic208ReleasePrinter209ReplaceAll210Reset213, 214ResetLineHeight214	PrintXY	.202
PushTabs204RecoverPrinter206Rectangle207RedrawPage207RegisterGraphic208ReleasePrinter209ReplaceAll210ReportDescToMemo211Reset213, 214ResetLineHeight214	PushFont	.203
RecoverPrinter206Rectangle207RedrawPage207RegisterGraphic208ReleasePrinter209ReplaceAll210ReportDescToMemo211Reset213, 214ResetLineHeight214	PushPos	.204
Rectangle207RedrawPage207RegisterGraphic208ReleasePrinter209ReplaceAll210ReportDescToMemo211Reset213, 214ResetLineHeight214	PushTabs	.204
RedrawPage207RegisterGraphic208ReleasePrinter209ReplaceAll210ReportDescToMemo211Reset213, 214ResetLineHeight214	RecoverPrinter	.206
RedrawPage207RegisterGraphic208ReleasePrinter209ReplaceAll210ReportDescToMemo211Reset213, 214ResetLineHeight214	Rectangle	.207
RegisterGraphic208ReleasePrinter209ReplaceAll210ReportDescToMemo211Reset213, 214ResetLineHeight214		
ReleasePrinter209ReplaceAll210ReportDescToMemo211Reset213, 214ResetLineHeight214		
ReplaceAll210ReportDescToMemo211Reset213, 214ResetLineHeight214		
ReportDescToMemo211 Reset213, 214 ResetLineHeight214		
Reset213, 214 ResetLineHeight214		
ResetLineHeight214	Reset	214

ResetSection	.215
ResetTabs	.216
RestoreBuffer	.216
RestoreFont	.217
RestorePos	.217
RestoreState	
RestoreTabs	
ReuseGraphic	.219
RoundRect	
RTFLoadFromFile	
RTFLoadFromStream	
Save	
SaveBuffer	
SaveFont	
SavePos	
SaveRaveBlob	
SaveState	
SaveTabs	
SaveToFile	
SaveToStream	
Saverostream	
SearchNext	
SelectBin	
SelectPaper	
SelectPrinter	
SelectReport	
SelectReport	
SetColumns	
SetColumnWidth	
SetData	
SetFont	
SetPaperSize	
SetParam	
SetPen	
SetPIVar	
SetTab	
SetTopOfPage	
ShadeToColor	
ShowPrintDialog	
ShowPrinterSetupDialog	
Start	
StretchDraw	
SupportBin	
SupportCollate	
SupportDuplex	
SupportOrientation	250

SupportPaper	260
Tab	
TabEnd	265
TabStart	268
TabWidth	268
TextRect	272
TextWidth	273
UnregisterGraphic	289
UpdateStatus	290
WriteBCDData	295
WriteBlobData	295
WriteBoolData	296
WriteCurrData	296
WriteDateTime	297
WriteFloatData	297
WriteIntData	298
WriteNullData	298
WriteStrData	299
XD2I	299
XD2U	300
XI2D	301
XI2U	301
XU2D	302
XU2I	303
YD2I	303
YD2U	304
YI2D	305
YI2U	305
YU2D	306
YU2I	307
ZoomIn	309
ZoomOut	310
Ν	
Naming the Data Connection	11
0	
OnEOF event	15
OnFirst Event	14
OnGetCols Event	14
OnGetRow event	15
OnNext event	15
OnOpen event	14
OnRestore event	
OnValidateRow event	
Output Components	
Overview	21

# Ρ

Page Designer Only	.27
Page Designer or Project Tree	.27
Project Tree Only	.27
Properties	9
Properties and Events	.17
property	
Aborted	
AccuracyMethod	
Active	
AscentHeight	
BarBottom	.38
BarCodeJustify	.39
BarCodeRotation	.40
BarHeight	.41
BarTop	.41
BarWidth	.42
BaseReport42, 43,	44
Bins	.45
BKColor	
Bold	
Bottom	.46
BottomWaste	
BoxLineColor	.47
Buffer	
BufferInc	
CacheDir	
Canvas	
Center	
CheckSum	
CodePage	
Collate	
ColumnEnd	
ColumnLinesLeft	
ColumnNum	
Columns	
ColumnStart	
ColumnWidth	
Copies	
CurrentPage	
CurrentPass	
CursorXPos	
CursorYPos	
DataSet	
DataSet	
DescentHeight	
DeviceName	. 1 1

DevMode	
DisplayName	.78
DLLFile	.78
DriverName	.81
Duplex	.82
Engine	.84
Extended	.86
ExtendedText	.87
Field	.87
FieldAliasList	.88
FileName	.88
FirstPage	.91
FontAlign	.92
FontBaseline	.93
FontBottom	.93
FontCharset	.94
FontColor	.94
FontHandle	.95
FontHeight	.95
FontName	.96
FontPitch	.96
FontRotation	.97
Fonts	.98
FontSize	.99
FontTop1	00
FontWidth1	00
FrameMode1	01
GridHoriz1	08
GridPen1	08
GridVert1	09
Height1	10
IgnoreFileSettings1	12
ImageQuality1	12
Italic1	14
Justify1	15
LastPage1	
Left	
LeftWaste1	18
LineBottom1	19
LineHeight1	20
LineHeightMethod1	
LineMiddle1	
LineNum1	22
LinesPerInch1	
LineTop1	25
LoadDesigner1	

LocalFilter	128
MacroData	130
MarginBottom	131
MarginLeft	132
MarginMethod	133
MarginPercent	133
MarginRight	134
MarginTop	134
MaxCopies	
MaxSize	136
Memo	136
MetafileDPI	138
Monochrome	139
NoBufferLine	142
NoCRLF	142
NoNewLine	143
NoNTColorFix	143
NoPrinterPageHeight	144
NoPrinterPageWidth	
OnCompress	
Orientation	
OriginX	
OriginY	168
OutputFileName	169
OutputInvalid	
OutputName	
PageHeight	
PageInc	172
PageInvalid	
Pages	173
PageWidth	
Papers	
PIVar	
Port	181
Pos	182
Position	
PrintChecksum	
PrintEnd	189
PrinterIndex	
Printers	
Printing	
PrintReadable	
PrintStart	
PrintTop	
ProjectFile	
Query	

RaveBlobDateTime	.205
ReadableHeight	.206
ReportDateTime	
ReportDesc	
ReportDest	.212
ReportFullName	
ReportName	.213
RichEdit	.219
Right	.220
RightWaste	
RTFField	
RTFText	
RulerType	
RuntimeVisibility	
ScaleX	
ScaleY	
ScrollBox	
SectionBottom	
SectionLeft	
SectionRight	
SectionTop	
Selection	
ServerMode	
ShadowDepth	
Size	
StatusFormat	
StatusLabel	
StatusText	
StoreRAV	
Stream	
StreamMode	
Strikeout	
Subscript	
Superscript	
SystemFiler	
SystemOptions	
SystemPreview	
SystemPrinter	
SystemSetups	
TabColor	
TabJustify	
Table	
TabShade	
Text	
TextBKMode	
TextJustify	

Title274	ŀ
TitlePreview274	
TitleSetup275	)
TitleStatus275	
Тор	
TopWaste279	
TotalPasses	
TransparentBitmaps281	
TruncateText	
Underline287	,
Units	
UnitsFactor289	
UseChecksum	
UseCompression291	
UseSetRange292	
Version	
WideFactor293	3
Width294	ŀ
XDPI	)
XPos	)
YDPI	ŀ
YPos	5
ZoomFactor308	3
ZoomInc	)
ZoomPageFactor311	
ZoomPageWidthFactor312	
Property Descriptions	)
Q	
Quick Start with Rave7	,
R	
Rave Classes6	
Rave Data Connection11	
Rave Toolbar5	,
Rendering components19 S	)
Single User License Agreement	3
Structure of Rave7	

-	

1	
Technical Support	4
TRvNDRWriter component	17
TRvProject Component	21
TRvRenderHTML Description	20
TRvRenderPDF Description	19
TRvRenderPreview Description	19
TRvRenderPreview Events	19
TRvRenderPreview Properties	19
TRvRenderPrinter Description	19
TRvRenderPrinter Properties and Eve	nts 19
TRvRenderRTF Description	20
TRvRenderText Description	20
TRvSystem Component	9
type	
TAccuracyMethod	
TBKMode	
TFontAlign	
TLineHeightMethod	
TMacroID	
TMarginMethod	
TOrientation	
TPrintJustify	
TPrintUnits	
TReportDest	
TStreamMode	
TSystemOption	
TSystemOptions	
TSystemSetup	
TSystemSetups	
TTabJustify	286
U	
Using Events to Customize your Data	40
Connection	
Using TRvProject <b>W</b>	21
	-
Welcome to RAVE!	1