ReportPrinter

Component Suite Version 1.1

Special Trial Run Demo

Introduction

Thank you for taking the time to look at ReportPrinter for Delphi. We're sure you'll see the time and effort that went into developing this suite of Delphi components that give the power of reporting back to the programmer where it belongs. ReportPrinter does this by offering a powerful suite of printing components that simplify the task of creating professional reports. ReportPrinter does not need any extra .DLL's, .VBX's or .EXE's. Reports are written in Delphi and are compiled into the application for easier distribution and faster execution. ReportPrinter also features a powerful print preview component that allows reports to be displayed quickly and accurately to the user's screen. Read through this manual and the examples on the accompanying disk and you'll soon be turning nightmare printing jobs into dream reports.

Some of ReportPrinter main features are:

* ReportPrinter does not have any extra files (like DLL's or VBX's). Reports are compiled right into your application using native Delphi code (Object Pascal).

* ReportPrinter comes with full source code and is written entirely in Delphi Object Pascal.

* ReportPrinter has much faster execution of reports over report writers like ReportSmith because it's not a seperate program to load and connect to the database engine (which can be quite slow over a network).

* ReportPrinter is compatible with any database engine that Delphi supports, however, ReportPrinter does not require a database engine. This is very useful for non-database applications that still need to produce printed output of some type.

* ReportPrinter can print memo fields (or any text stream) with word wrapping (even across multiple pages).

* ReportPrinter does not need a separate installation program.

* ReportPrinter has a footprint of less than 50K.

* ReportPrinter can left, center and right justify text within tabbed columns (with boxes

around text for a grid or horizontal/vertical lined style table listing)

* ReportPrinter has columns that will automatically wrap the cursor to the next column.

* ReportPrinter has full graphics capability (bitmaps, lines, ellipses,...).

* ReportPrinter uses real-world measurements like inches, mm and can even use a custom unit scale.

* ReportPrinter generates precise page positioning (useful for pre-printed forms).

* ReportPrinter has a fast and powerful Print Preview component with zooming, panning and after preview printing capabilities.

* ReportPrinter uses event handlers for printing parts of each page (header or footer) as well as notify events for handling events such as a new page or new column.

Installation

Installing the ReportPrinter component suite is an easy process. If you have installed an older or demo copy of ReportPrinter, first remove all instances of those files from the component library and your hard disk before continuing with the installation process. If you do not do this you may get type mismatch or other errors.

Step 1: Backup the component library file called COMPLIB.DCL in your \DELPHI\BIN directory. If you encounter any problems during the installation process, you can restore your component library by restoring this file.

Step 2: Copy the files found the in the INSTALL.ZIP archive into a directory on your hard disk (\DELPHI\LIB or \DELPHI\ADDON are good choices).

Step 3: Load up Delphi and select Options|Install Components from the Delphi menu bar. Once the Install Components dialog window is displayed, click on the Add button and select the file RPREG.PAS in the directory that the ReportPrinter files were copied into in Step 2. You should use the Browse button to do this.

Step 4: Click OK to return to the Install Components dialog box and then click OK to accept the changes. Delphi will now re-compile your component library. Once it is finished, you should have a new page on your component bar called Report that contains the new ReportPrinter components.

NOTE: If you want to change the component page that the ReportPrinter components are installed into, modify the RegisterComponents call in RPREG.PAS before beginning the installation.

Ordering Information

ReportPrinter is released as demoware and as such you must register it if you wish to use it past a 30 day evaluation period. This demoware trial-run version will only run while Delphi is running. The registration price is being for a special introductory price of **$59.95** plus **$5** shipping and handling ($10 outside North America). Registration includes printed documentation, examples on how to use ReportPrinter, complete source code to ReportPrinter and compiled versions of the ReportPrinter components that will run outside of Delphi. Your license for ReportPrinter will allow you to use the source code in an executable form without further royalties, however, you cannot release the source code, either modified or unmodified, in a text or non-executable for (such as a .DCU) without prior written consent from the author, Jim Gunkel. Also, the license is

for a single copy of ReportPrinter and additional copies must be registered if multiple programmers are going to be using ReportPrinter at the same time. Special pricing and discounts are available, please contact the author for more information. There are several easy ways you can register:

Credit Card: To order directly from Nevrona Designs you can send your credit card (**Visa, Master Card and American Express accepted**) and shipping information (see form below) in one of the following ways:

Voice Phone Number: 602-899-0794 (United States)

24-Hour Fax Phone Number: 602-530-4823 (United States)

Postal Mail: Nevrona Designs
Jim Gunkel
2581 E. Commonwealth Circle
Chandler, AZ 85225-6019

CompuServe ID: 70711,2020

Internet ID: jgunkel@primenet.com

NOTE: If you send your credit card information across the Internet, it may be visible to other parties before it reaches Nevrona Designs e-mail address. A PGP public key is provided in the file NEVRONA.KEY to allow you to encrypt your ordering information. If you decide not to encrypt your credit card information, it is done at your own risk.

Through CompuServe: Go to the **SWREG** forum (GO SWREG) and register program **ID #5660**, ReportPrinter 1.1. You will be asked for shipping information and after you complete the registration I will receive automatic notice through E-Mail.

Check/Money Order: Send check or money order to
Nevrona Designs
Jim Gunkel
2581 E. Commonwealth Circle
Chandler, AZ 85225-6019

Order Form

Please make sure to include the following information:

Name :_____

Company :_____

Address :_____

_____

City :_____ State:_____ Zip Code:_____

Country (if not U.S.):_____

Phone # :_____

E-mail Address:_____

Credit Card #:_____Expiration Date:_____

Name on Credit Card:_____

Copies of ReportPrinter:_____

Once I receive your registration I will send out the current released version of ReportPrinter with examples, complete source code to ReportPrinter and printed documentation via Postal mail. Alternative modes of shipping (example: Overnight) are available for an extra shipping charge.

Commonly Asked Questions

Why should I use ReportPrinter instead of a traditional report writer like ReportSmith?
With ReportPrinter, you code your reports using Delphi so you have all of the power of Object Pascal and whichever database engine you use (BDE, Apollo, ...). There is no need to learn yet another language (usually an interpreted macro language) to do more complicated reports and you have the power to create any report format report necessary instead of conforming your reports to pre-designed formats. Plus, complex data relationships can be sent to your report instead of creating temporary tables for a report writer to read and dump to the report.

Another benefit of ReportPrinter is that there are no .DLL's, .VBX;s or other files to distribute with your application; all code for the report is compiled into your executable file resulting in a more resource efficient and faster running report.

Still another advantage is that if your application requires printing that is not dependent upon an existing database (example: calculated data) you can print your report without having to create a temporary table or even have a database engine installed at all.

Are there any advantages to using a traditional report writer?
Yes, but most of these are only an issue in specific circumstances. In a large client/server environment, you may wish to give your users the ability to create or customize reports as they need to without requiring changes to the application code. This however usually only works for reports with simple formats and a report writer can still be used in conjunction with ReportPrinter to solve your clients needs. The only feature that a report writer has that ReportPrinter does not address is a visual report designer (although one is planned for a future release.) For most reports this is not a major issue and the visual report designer is the main reason that report writers are usually limited in the report formats that they support.

Where can I find the latest demo version of ReportPrinter?
The latest demo version of ReportPrinter can be found in at least two places:

1) On CompuServe in the DELPHI forum, under the name of RPRINTER.ZIP.

2) On the Internet at the ftp site ftp.primenet.com/users/j/jgunkel/delphi/rprinter.zip

This file will contain a compiled version of ReportPrinter which is fully functional but will only run while Delphi is running. Other files included will be documentation similar to this file and some sample applications that demonstrate the features of ReportPrinter.

Tutorial

The following items will present a short tutorial on some of the main features of ReportPrinter. After reading through these topics, you will have a much better understanding of how ReportPrinter works and what it is capable of.

Tutorial: Hello World!

In the true spirit of programming, let's code a simple report that prints the text 'Hello World!' to the center of the page. Follow the 4 simple steps listed below:

Step 1: Create a blank form. This will be your report status form.

Step 2: Place a ReportPrinter component (from the Reports page on your component bar) onto the report status form. It does not matter where since all of the ReportPrinter components are non-visual components. Make sure the properties in the Object Inspector are set to the desired values.

Step 3: Select the report status form and double click on the OnActivate event in the Object Inspector and type the following code in the new event handler that Delphi has just created for you:

```
ReportPrinter1.Execute;
```

Step 4: Select the ReportPrinter component then double click on the OnPrint event in the Object Inspector and type the following code:

```
With Sender as TBaseReport do begin
  SetFont('Arial',12);
  YPos := PageHeight/2;
  PrintCenter('Hello World!',PageWidth/2);
end;
```

That's it! When you activate the report status form a page will be printed to the default printer with the text Hello World! printed at the center of the page. The only difference between this report and a more useful report would be additional code in the OnPrint event handler. Read on to learn how to create reports that use more of the features of ReportPrinter.

Tutorial: Tables and Tabs

A simple report like the one above is good for learning the basics of ReportPrinter but I'm sure you're wondering how easy it is to create a report that actually does something. Below, we'll create a report that prints specific fields for each record in a database. We'll use a feature of ReportPrinter called tabs. Tabs allow you to print text at a specific horizontal position, center, left or right justified. Tabs also have an optional width that will define a tab box with optional lines around each tab box. These lines can be used to create automatic horizontal, vertical or grid separators.

Step 1: Follow step 1 through 3 for the Hello World! report to create a base report status form.

Step 2: Place a TTable component on your form, and initialize its properties, to access the table to report on. In this example we'll print the first three fields of the table which we'll assume are Name, Phone and Amount.

Step 3: Select the ReportPrinter component then double click on the OnBeforePrint event in the Object Inspector and type the following code:

```
Table1.Open; { Make sure the table is open }
Table1.First; { Goto the first record }
```

Step 4: Select the ReportPrinter component then double click on the OnPrintPage event in the Object Inspector and type the following code:

```
With Sender as TBaseReport do begin
  SetFont('Arial',10);
  Home;
{ Print the grid header }
  SetTab(1.0,pjCenter,2.0,2,BOXLINESALL,0);
  SetTab(NA,pjCenter,1.5,2,BOXLINESALL,0);
  SetTab(NA,pjCenter,1.5,2,BOXLINESALL,0);
  Println(#9'Name'#9'Phone #'#9'Amount');
{ Prepare the tabs for the data grid }
  SetTab(1.0,pjLeft,2.0,2,BOXLINESALL,0);
  SetTab(NA,pjCenter,1.5,2,BOXLINESALL,0);
  SetTab(NA,pjRight,1.5,2,BOXLINESALL,0);
{ Print as many lines as possible to page }
  With Table1 do begin
    While (LinesLeft > 0) and not EOF do begin
      Print(#9 + Fields[0].AsString);
      Print(#9 + Fields[1].AsString);
      Println(#9 + Fields[2].AsString);
      Next;
    end; { while }
  end; { with }
  Result := not Table1.EOF;
end; { with }
```

There are many different ways of doing this type of report. In this example we used the OnPrintPage event handler to print one page at a time and passed back a result of whether we needed to print more pages. Typically, reports that are the same from page to page will benifit from using OnPrintPage while reports that are more complicated should use OnPrint and call NewPage to start a new page.

Tutorial: Columns

In ReportPrinter, Columns are used to print a stream of text in evenly spaced columns across the page. Columns are defined by the methods SetColumns and SetColumnWidth and are undefined by the method ClearColumns. You can reference the properties ColumnStart, ColumnEnd, ColumnWidth, Columns and ColumnNum to find out more about the current column settings. To change the current column, set the desired value to the property ColumnNum (this will only change the horizontal cursor position and not the vertical).

During a call to Println or CRLF, if the text cursor passes below the current SectionBottom, it will be automatically placed at the top of the next column. There are two methods, LinesLeft and ColumnLinesLeft that will return the number of lines left in the current column and the number of lines left in all remaining columns respectively.

There is a special event handler to deal with new columns called OnNewColumn. This event handler will be called whenever SetColumns or SetColumnWidth is called as well as when the text cursor wraps over to a new column from a call to Println or CRLF. This event handler can be useful for printing column headers.

Tabs and Memos can be combined with columns, so that if the text cursor is on a column other than the first, the horizontal positions will be offset by the column width (plus the space between each column).

Example (Tabs combined with Columns):

LL	CLL	L#	C#
9	39	2	1
8	38	3	1
7	37	4	1
6	36	5	1
5	35	6	1
4	34	7	1
3	33	8	1
2	32	9	1
1	31	10	1

LL	CLL	L#	C#
9	29	2	2
8	28	3	2
7	27	4	2
6	26	5	2
5	25	6	2
4	24	7	2
3	23	8	2
2	22	9	2
1	21	10	2

LL	CLL	L#	C#
9	19	2	3
8	18	3	3
7	17	4	3
6	16	5	3
5	15	6	3
4	14	7	3
3	13	8	3
2	12	9	3
1	11	10	3

LL	CLL	L#	C#
9	9	2	4
8	8	3	4
7	7	4	4
6	6	5	4
5	5	6	4
4	4	7	4
3	3	8	4
2	2	9	4
1	1	10	4

Tutorial: Margins, Sections and Waste

Margins, Sections and Waste may appear to be similar, but there are significant differences. The waste values (LeftWaste,RightWaste,TopWaste and BottomWaste) are retrieved from the printer driver to find out the margins of unprintable area for the page. Most dot matrix printers will not have any waste margins, laserjet printers may have waste margins from 0.22 to 0.5 inches and some inkjet printers may have waste margins up to 0.75 inches. Almost every printer is different so it is important to pay attention to the waste areas if you are printing close to the edges of the page.

The Margin and Section values are a little closer in definition. Sections (SectionLeft, SectionRight, SectionTop and SectionBottom) control where the text cursor begins each new line (SectionLeft) and where line one begins (SectionTop). Sections are also used during SetColumns or SetColumnWidth to determine how wide or how many columns can fit between SectionLeft and SectionRight. So, as you can see, the section values determine the area on your page where you have told ReportPrinter that you are going to print. Margins on the other hand do not control anything other than to provide a base set of values that the sections are reset to whenever a new page is begun or ResetSections is called. Changing the value of a margin also changes the value of the same section property to that same measurement. Note that sections are always measured from the top or left sides of the page while margins are measured from the closest side of the page.

Tutorial: Memo Fields and Long Strings

Memos and long strings are easily printed in ReportPrinter using two special classes, TMemoBuf and TDBMemoBuf. TMemoBuf (found in the RPMEMO unit) can store text streams up to 64K. TDBMemoBuf (found in the RPDBUTIL unit) has the same functionality as TMemoBuf plus a new property called Field that can be used to assign a TMemoField's contents to the memo buffer. The reason that TDBMemoBuf is in a separate unit from TMemoBuf is so that applications can be generated that do not require the BDE.

ReportPrinter interfaces with a memo buffer object through the two methods MemoLines and PrintMemo. However, before passing a memo buffer to these two methods, the memo buffer must be initialized. The following code shows a simple example of how to initialize and print a memo buffer:

```
var
  MemoBuf: TMemoBuf;
  StringVar: string;
  StartPos: double;
  EndPos: double;

begin
  .
  . { Initialize StringVar, StartPos and EndPos }
  .
  MemoBuf := TMemoBuf.Create;
  try
    MemoBuf.Text := StringVar;
    MemoBuf.PrintStart := StartPos;
    MemoBuf.PrintEnd := EndPos;
    PrintMemo(MemoBuf,0,false);
  finally
    MemoBuf.Free;
  end; { try }
end;
```

With PrintMemo, you can specify to print only a specific number of lines. The position that PrintMemo ends printing will be remembered by the memo buffer so that the next time that memo buffer is printed, it will start at this new position. This can be very useful for printing memo fields across multiple pages. You can also load a memo buffer from a stream using the LoadFromStream method.

Tutorial: The Text Cursor and Line Height

Understanding the text cursor is central to creating good reports. The baseline of text is printed at the cursor location to provide an easy way to print fonts of different sizes on a single line. Most printing methods modify the text cursor position in a different way but most of the time the cursor is left at the end of the text that was just printed. The positioning methods will move the text cursor to a specific spot on the page so that the next print method that is called, provided it doesn't move the cursor (like PrintCenter), it will start at that position.

The current line height can be calculated in one of two ways. The property LineHeightMethod controls whether the line height is calculated by the LinePerInch property or is determined by the font height. When the line height is being determined by the font height, the line height doesn't change until the cursor is moved vertically to a new line. Therefore you can change the font size in the middle of a line and as long as you set the font back to the original setting the line height won't change.

Tutorial: Setting up the Printer

ReportPrinter gives you many methods to customize the way the printer prints your report. Select the printer with SelectPrinter or PrinterIndex. You can set other printing parameters with SetPageSize, Orientation, Copies and DevMode. You can also find out much information about the current printer with DeviceName, DriverName, Port, PaperWidth, PaperHeight, XDPI, YDPI, LeftWaste, RightWaste, TopWaste and BottomWaste. To send printer codes directly to the printer without interpretation by the Windows Print Manager, use the PrintData method.

NOTE: Not all printers have the same features and functionality. This is a printer or printer driver limitation. Be careful about developing non-standard printer requirements (such as paper size or paper bins) if your application will be distributed on a wide variety of hardware systems.

Tutorial: Print Preview

The TFilePreview component requires a special report file that was generated by a TReportFiler component. Once this file is generated, it is passed to the TFilePreview component either on the disk (for StreamMode = smFile or smMemory) or through a user defined stream assigned to the Stream property (for StreamMode = smUser).

The preview screen can be one of the most visible elements of ReportPrinter in your application. Because of this, ReportPrinter was made to hook into your user interface and not dictate the style of the preview screen. When you place a TFilePreview component on your preview form, the only requirement for a standard visual component is a TScrollBox (or one if its descendants) must be placed on the form and assigned to the ScrollBox property.

Use PrevPage, NextPage and PrintPage to go to other pages. Use ZoomIn, ZoomOut and ZoomFactor to change the scaling of the page within the preview window. ZoomPageFactor and ZoomPageWidthFactor will return the zoom factors necessary to zoom the entire page or zoom the entire page width respectively. The event handlers, OnPageChange and OnZoomChange will be called whenever the current page or zoom factor changes. This will allow you to keep your visual controls in sync with the current status of the preview screen.

Study the preview screen in the demo source code for more help and hints on how to create a preview screen for your application.

Component Overview

TReportPrinter

The TReportPrinter component is the primary reporting component and sends report output directly to the printer. To create a report with TReportPrinter, place the component on a form. The properties and events can then be customized or coded for the report. To run the report, simply call the Execute method. Execute will first call the OnBeforePrint event, then call the printing events, then call the OnAfterPrint event.

The OnBeforePrint and OnAfterPrint events are called before and after the print job is active and therefore, the printer canvas is not valid. Code inside of these events should call only non-printing code such as opening or closing of tables or the setting of published TReportPrinter properties.

Printing the body of the report is done with one of two event handlers: OnPrint and OnPrintPage. See their descriptions in the reference section for more information on how these events differ. OnPrintHeader, OnPrintFooter and OnNewPage are called for each page that is printed. OnNewPage is called before any printing code is done for that page to allow you to set the page up. OnPrintHeader is called after OnNewPage. OnPrintFooter is called after all printing for a page is completed but before the next page is activated.

TReportFiler

The TReportFiler component is used to generate a special report file that can later be used by the TFilePrinter and TFilePreview components. A report is generated the same way as the code in a TReportPrinter report except that the report is sent to a file (or other stream) instead of the printer.

TFilePrinter

The TFilePrinter component is used to send a report file previously generated from TReportFiler to the printer. To print a report with TFilePrinter, place the component on a form. Then initialize the FileName property for the report file to print. To print the report file, simply call the Execute method. You can also call ExecuteCustom to print a portion of the report file or to change the number of copies.

TFilePreview

The TFilePreview component is used to send a report file generated from TReportFiler to a preview form. The construction of this form must include a scrollbox and could include other features such as buttons or spin boxes to control the current page and zoom factor.

Alphabetical Method/Property Reference

Listed below is an alphabetical listing of all public and published properties and methods that make up the ReportPrinter component suite. Methods are defined by their calling syntax, category, components they are members of, a short description and any other properties or methods they may be related to. Properties are defined by their data type, access type (read, write or published), category, components they are members of, a short description and any other properties or methods they may be related to.

Abort method

Syntax: `procedure Abort;`
Category: Control
Components: TReportFiler, TReportPrinter
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 many allocate in your reporting code.
See Also: Execute, Aborted

Aborted:boolean (read only)

Category: Control
Components: TReportFiler, TReportPrinter
Description: This property will be set to true after a call to Abort has been made.
See Also: Abort

AbortPage method

Syntax: `procedure AbortPage;`
Category: Control
Components: TReportFiler, TReportPrinter
Description: This method will abort the printing of the current page and start printing a new page.
See Also: Abort

AccuracyMethod:TAccuracyMethod (read/write/pub)

Category: Control
Components: TReportFiler
Description: This property controls how text is written to the report file. If AccuracyMethod is equal to amPositioning then the text is written out one character at a time so that the preview screen will represent as accurately as possible the position and width that each text string will occupy. If equal to amAppearance then the text string is written out as a complete string. The problem with amPositioning is that text may not appear as pleasing to the eye since screen fonts size differently than printer fonts (especially spacing between letters and underlined text). 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. This property has no effect for TFilePrinter.

Arc method

Syntax: `procedure Arc(X1,Y1,X2,Y2,X3,Y3,X4,Y4: double);`
Category: Graphics
Components: TReportFiler, TReportPrinter
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

BKColor:TColor (read/write)

Category: Graphics
Components: TReportFiler, TReportPrinter
Description: This property returns or sets the current background property for certain graphics functions.

See Also: TextBKMode

Bold:boolean (read/write)
Underline:boolean (read/write)
Italic:boolean (read/write)
Strikeout:boolean (read/write)

Category: Text
Components: TReportFiler, TReportPrinter
Description: These properties return or set the attributes for the current font.

BrushCopy method

Syntax:
```
procedure BrushCopy(const Dest: TRect;
                          Bitmap: TBitmap;
                    const Source: TRect;
                          Color: TColor);
```
Category: Graphics
Components: TReportFiler, TReportPrinter
Description: Copies a portion of a bitmap, Bitmap, specified by the rectangle, Source, to the printer canvas with a specific color of the bitmap, Color, replaced by the current brush of the destination canvas. The rectangle, Dest, defines the region to copy the bitmap to.
See Also: CreateRect

Canvas:TCanvas (read only)

Category: Printer
Components: TReportFiler, TReportPrinter
Description: This method returns the TCanvas object that is being printed on. **NOTE:** Direct manipulation of the canvas is not supported by TReportFiler (and thus TFilePrinter and TFilePreview).

Chord method

Syntax:
```
procedure Chord(X1,Y1,X2,Y2,X3,Y3,X4,Y4: double);
```
Category: Graphics
Components: TReportFiler, TReportPrinter
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

ClearColumns method

Syntax:
```
procedure ClearColumns;
```
Category: Columns
Components: TReportFiler, TReportPrinter
Description: This method removes all current column settings.
See Also: SetColumns, SetColumnWidth

ClearTabs method

Syntax:
```
procedure ClearTabs;
```
Category: Tabs
Components: TReportFiler, TReportPrinter
Description: This method removes all current tab settings.
See Also: SetTab, ResetTabs

ColumnEnd:double (read only)

Category: Columns
Components: TReportFiler, TReportPrinter
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

ColumnLinesLeft method

Syntax: `function ColumnLinesLeft: integer;`
Category: Columns
Components: TReportFiler, TReportPrinter
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: LinesLeft, all column methods, SectionBottom

ColumnNum:integer (read/write)

Category: Columns
Components: TReportFiler, TReportPrinter
Description: This property will return or set the current column number that the text cursor is on.

Columns:integer (read only)

Category: Columns
Components: TReportFiler, TReportPrinter
Description: This property returns the number of columns that are available from the last call to SetColumns or SetColumnWidth.
See Also: SetColumns, SetColumnWidth, ColumnNum

ColumnStart:double (read only)

Category: Columns
Components: TReportFiler, TReportPrinter
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

ColumnWidth:double (read only)

Category: Columns
Components: TReportFiler, TReportPrinter
Description: This method returns the width of the current column.
See Also: SetColumns, SetColumnWidth

Copies:integer (read/write/pub)

Category: Printer
Components: TReportFiler, TReportPrinter
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. For these printers, just call the report multiple times or use TReportFiler and TFilePrinter to speed up report generation. Use a value of 0 to retain the setting defined by TPrinterSetupDialog.

CopyRect method

Syntax:
```
procedure CopyRect(const Dest: TRect;
                         Canvas: TCanvas;
                   const Source: TRect);
```
Category: Graphics
Components: TReportFiler, TReportPrinter
Description: This method copies part of an image defined by the rectangle, Source, from another canvas, Canvas, to the area on the printer canvas defined by the rectangle, Dest.
See Also: CreateRect

CR method

Syntax: `procedure CR;`
Category: Position
Components: TReportFiler, TReportPrinter
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 is defined by either the current SectionLeft setting or the setting of ColumnStart if columns are in use.
See Also: CRLF, LF, ColumnStart, SectionLeft

Create method

Syntax: `constructor Create(AOwner: TComponent);`
Category: Misc
Components: TReportFiler,TReportPrinter,TFilePrinter,TFilePreview
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

CreateBrush method

Syntax:
```
function CreateBrush(NewColor: TColor;
                     NewStyle: TBrushStyle;
                     NewBitmap: TBitmap): TBrush;
```
Category: Graphics
Components: TReportFiler, TReportPrinter
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. (Example: Canvas.Brush := MyBrush;) **NOTE:** The brush object returned must be released by calling the free method of TBrush.
See Also: SetBrush

CreateFont method

Syntax:
```
function CreateFont(NewName: string;
                    NewSize: integer): TFont;
```
Category: Text
Components: TReportFiler, TReportPrinter
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. (Example: Canvas.Font := MyPen;) **NOTE:** The font object returned must be released by calling the free method of TFont.
See Also: SetFont

CreatePen method

Syntax:
```
function CreatePen(NewColor: TColor;
                   NewStyle: TPenStyle;
                   NewWidth: integer;
                   NewMode: TPenMode): TPen;
```
Category: Graphics
Components: TReportFiler, TReportPrinter
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. (Example: Canvas.Pen := MyPen;) **NOTE:** The pen object returned must be released by calling the free method of TPen.
See Also: SetPen

CreatePoint method

Syntax: `function CreatePoint(X,Y: double): TPoint;`
Category: Graphics
Components: TReportFiler, TReportPrinter
Description: This method will return a TPoint record initialized to the point (X,Y).

CreateRect method

Syntax: `function CreateRect(X1,Y1,X2,Y2: double): TRect;`
Category: Graphics
Components: TReportFiler, TReportPrinter
Description: This method will return a TRect record initialized to the rectangle defined by the points (X1,Y1) and (X2,Y2).

CRLF method

Syntax: `procedure CRLF;`
Category: Position
Components: TReportFiler, TReportPrinter
Description: This method performs a carriage return (CR) followed by a line feed (LF), then resets the tabs.
See Also: CR, LF, ResetTabs

CurrentPage:integer (read only)
Category: Control
Components: TReportFiler, TReportPrinter
Description: This property returns the current page number.

CursorXPos:longint (read only)
CursorYPos:longint (read only)
Category: Position
Components: TReportFiler, TReportPrinter
Description: These properties return the horizontal and vertical text cursor position in printer units (dots).
See Also: XPos, YPos

Destroy method
Syntax:
```
destructor Destroy;
```
Category: Misc
Components: TReportFiler, TReportPrinter, TFilePrinter, TFilePreview
Description: The Destroy destructor should never be called directly. To destroy a component created with Create, call the Free method.
See Also: Create

DeviceName:string (read only)
Category: Printer
Components: TReportFiler, TReportPrinter
Description: This property will return the device name for the currently selected printer.
See Also: PrinterIndex

DevMode:PDevMode (read/write)
Category: Printer
Components: TReportFiler, TReportPrinter
Description: This property provides access to the TDevMode structure for the current printer. After any changes to DevMode are made, ResetPrinter should be called.

Draw method
Syntax:
```
procedure Draw(X,Y: double;
                    Graphic: TGraphic);
```
Category: Graphics
Components: TReportFiler, TReportPrinter
Description: This method draws the graphic object, Graphic, to the printer canvas at the location (X,Y). NOTE: Do not use Draw for bitmaps, instead use PrintBitmap or PrintBitmapRect.
See Also: PrintBitmap, PrintBitmapRect, StretchDraw

DrawFocusRect method
Syntax:
```
procedure DrawFocusRect(const Rect: TRect);
```
Category: Graphics
Components: TReportFiler, TReportPrinter
Description: This method will draw a rectangle, defined by Rect, in the style used to indicate that the rectangle has focus.
See Also: CreateRect

DriverName:string (read only)
Category: Printer
Components: TReportFiler, TReportPrinter
Description: This property will return the driver name for the currently selected printer.

Ellipse method
Syntax:
```
procedure Ellipse(X1,Y1,X2,Y2: double);
```
Category: Graphics
Components: TReportFiler, TReportPrinter

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

Execute method

Syntax: `procedure Execute;`
Category: Control
Components: TReportFiler,TReportPrinter,TFilePrinter
Description: This method will begin to print the report. For report generation components (TReportFiler, TReportPrinter) the event handlers OnBeforePrint, OnPrint, OnPrintPage, OnNewPage, OnNewColumn, OnPrintHeader, OnPrintFooter and OnAfterPrint will be called at their appropriate times. For TFilePrinter the contents of the report stream from a TReportFiler will be sent to either the printer or the preview screen. See Start for printing the report for a TFilePreview component.
See Also: Abort, Printing, all printing event handlers

ExecuteCustom method

Syntax:
```
procedure ExecuteCustom(NewFirstPage: integer;
                        NewLastPage: integer;
                        NewCopies: integer);
```
Category: Control
Components: TFilePrinter
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: Execute

FileName:string (read/write/pub)

Category: Control
Components: TReportFiler, TFilePrinter, TFilePreview
Description: If StreamMode is equal to smFile or smMemory, then FileName specifies the file to write the report to or read the report from.
See Also: StreamMode

FillRect method

Syntax: `procedure FillRect(const Rect: TRect);`
Category: Graphics
Components: TReportFiler, TReportPrinter
Description: This method fills the rectangle defined by Rect with the current brush.
See Also: CreateRect

Finish method

Syntax: `procedure Finish;`
Category: Preview
Components: TFilePreview
Description: This method finishes a preview session.
See Also: Start

FirstPage:integer (read/write/pub)

LastPage:integer (read/write/pub)
Category: Control
Components: TReportFiler, TReportPrinter
Description: These properties define the 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

FloodFill method

Syntax:
```
procedure FloodFill(X,Y: double;
                    Color: TColor;
                    FillStyle: TFillStyle);
```
Category: Graphics
Components: TReportFiler, TReportPrinter

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 boundry 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.)

FrameRect method

Syntax: `procedure FrameRect(const Rect: TRect);`
Category: Graphics
Components: TReportFiler, TReportPrinter
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

GetTab method

Syntax: `function GetTab(Index: integer): PTab;`
Category: Tabs
Components: TReportFiler, TReportPrinter
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.

GotoFooter method

Syntax: `procedure GotoFooter;`
Category: Position
Components: TReportFiler, TReportPrinter
Description: This method will position the text cursor just above the current SectionBottom.
See Also: PrintFooter, MarginBottom, SectionBottom

GotoHeader method

Syntax: `procedure GotoHeader;`
Category: Position
Components: TReportFiler, TReportPrinter
Description: This method will position the text cursor just below the current SectionTop.
See Also: PrintHeader, MarginTop, SectionTop

GotoXY method

Syntax:
```
procedure GotoXY(NewXPos: double;
                 NewYPos: double);
```
Category: Position
Components: TReportFiler, TReportPrinter
Description: This method will move the text cursor to the position NewXPos, NewYPos.
See Also: XPos, YPos

Home method

Syntax: `procedure Home;`
Category: Position
Components: TReportFiler, TReportPrinter
Description: This method will move the text cursor to the beginning of line 1.
See Also: LineNum

LeftWaste:double (read only)
RightWaste:double (read only)
TopWaste:double (read only)
BottomWaste:double (read only)

Category: Printer
Components: TReportFiler, TReportPrinter
Description: These properties return the waste area on the page that the printer cannot print into. It is a good idea to make sure that the reports margins are greater than or equal to these waste areas.

LF method

Syntax: `procedure LF;`
Category: Position
Components: TReportFiler, TReportPrinter
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, CRLF, LineHeight, LineNum, SectionTop, SectionBottom

LineHeight:double (read only)

Category: Position
Components: TReportFiler, TReportPrinter
Description: This property returns the current height of a line.
See Also: LineHeightMethod

LineHeightMethod:TLineHeightMethod (read/write/pub)

Category: Position
Components: TReportFiler, TReportPrinter
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.
See Also: LinesPerInch

LineNum:integer (read/write)

Category: Position
Components: TReportFiler, TReportPrinter
Description: This property returns or sets the current line number. This property is highly dependent upon the current LineHeightMethod as well as the size of the current font if LineHeightMethod is equal to lhmFont. LineNum may not represent the actual line number if the report is jumping around the page instead of calling Prints and Printlns.
See Also: LineHeightMethod, LineHeight

LinesLeft method

Syntax: `function LinesLeft: integer;`
Category: Position
Components: TReportFiler, TReportPrinter
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

LinesPerInch:integer (read/write/pub)

Category: Position
Components: TReportFiler, TReportPrinter
Description: This property will return or set the number of lines per inch if the LineHeightMethod property is equal to lhmLinesPerInch.
See Also: LineHeightMethod

LineTo method

Syntax: `procedure LineTo(X,Y: double);`
Category: Graphics
Components: TReportFiler, TReportPrinter
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: MoveTo

MarginLeft:double (read/write/pub)
MarginRight:double (read/write/pub)
MarginTop:double (read/write/pub)
MarginBottom:double (read/write/pub)

Category: Position
Components: TReportFiler, TReportPrinter
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: section properties, ResetSection

MarginMethod:TMarginMethod (read/write/pub)

Category: Preview
Components: TFilePreview
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

MarginPercent:double (read/write/pub)

Category: Preview
Components: TFilePreview
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

MemoBuf.Empty method

Syntax:
```
function Empty: boolean;
```
Category: Memo
Components: TMemoBuf, TDBMemoBuf
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: MemoBuf.Pos, MemoBuf.Size

MemoBuf.Field: TMemoField (write only)

Category: Memo
Components: TDBMemoBuf
Description: This property will assign the contents of a TMemoField component to the memo buffer.
See Also: MemoBuf.Pos, MemoBuf.Size

MemoBuf.LoadFromStream method

Syntax:
```
procedure LoadFromStream(Stream: TStream;
                         BufSize: longint);
```
Category: Memo
Components: TMemoBuf, TDBMemoBuf
Description: This method will load the memo buffer from the stream, Stream, for BufSize number of bytes.
See Also: MemoBuf.SaveToStream

MemoBuf.Pos: longint (read/write)

Category: Memo
Components: TMemoBuf, TDBMemoBuf
Description: This property will return or set the current position marker for the memo buffer.
See Also: MemoBuf.Reset

MemoBuf.PrintEnd: double (read/write)

Category: Memo
Components: TMemoBuf, TDBMemoBuf
Description: This property will return or set the rightmost position that the memo field will print in.
See Also: MemoBuf.PrintStart

MemoBuf.PrintStart: double (read/write)

Category: Memo
Components: TMemoBuf, TDBMemoBuf

Description: This property will return or set the leftmost position that the memo buffer will print in.
See Also: MemoBuf.PrintEnd

MemoBuf.Reset method

Syntax: `procedure Reset;`
Category: Memo
Components: TMemoBuf, TDBMemoBuf
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.

MemoBuf.SaveToStream method

Syntax:
```
procedure SaveToStream(Stream: Tstream);
```
Category: Memo
Components: TMemoBuf, TDBMemoBuf
Description: This method will save the memo buffer to the stream, Stream.
See Also: MemoBuf.LoadFromStream

MemoBuf.SetData method

Syntax:
```
procedure SetData(var Buffer;
                  BufSize: longint);
```
Category: Memo
Components: TMemoBuf, TDBMemoBuf
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 longer than 255 characters.
See Also: MemoBuf.Text

MemoBuf.Size: longint (read only)

Category: Memo
Components: TMemoBuf, TDBMemoBuf
Description: This property will return the current size of the memo buffer in bytes.

MemoBuf.Text: string (read/write)

Category: Memo
Components: TMemoBuf, TDBMemoBuf
Description: This property will set the memo buffer to a string assigned to it. If this property is referenced, the first 255 characters of the memo buffer (or the size of the memo buffer, whichever is less) will be returned.

MemoLines method

Syntax:
```
function MemoLines(MemoBuf: TMemoBuf): longint;
```
Category: Memo
Components: TReportFiler, TReportPrinter
Description: This method will return the number of lines necessary to print the memo buffer, MemoBuf, for the current font.
See Also: TMemoBuf, PrintMemo

Monochrome:boolean (read/write/pub)

Category: Preview
Components: TFilePreview
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.

MoveTo method

Syntax:
```
procedure MoveTo(X,Y: double);
```
Category: Graphics
Components: TReportFiler, TReportPrinter
Description: This method will move the current graphic cursor position to the point specified by (X,Y).
See Also: LineTo

NewPage method

Syntax: `procedure NewPage;`
Category: Control
Components: TReportFiler, TReportPrinter
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

NextPage method
Syntax: `procedure NextPage;`
Category: Preview
Components: TFilePreview
Description: This method will goto and print the next page to the preview window. The OnPageChange event handler will be called if the current page number changes.
See Also: PrevPage, CurrentPage, OnPageChange

OnAfterPrint:TNotifyEvent (read/write/pub)
Category: Control
Components: TReportFiler, TReportPrinter
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: OnBeforePrint, Execute

OnBeforePrint:TNotifyEvent (read/write/pub)
Category: Control
Components: TReportFiler, TReportPrinter
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: OnAfterPrint, Execute

OnNewColumn:TNotifyEvent (read/write/pub)
Category: Control
Components: TReportFiler, TReportPrinter
Description: This event will be called whenever a new column has begun (after a call to Println, CRLF, SetColumns or SetColumnWidth). This can be useful for printing column headers.
See Also: SetColumns, SetColumnWidth, CRLF, Println

OnNewPage:TNotifyEvent (read/write/pub)
Category: Control
Components: TReportFiler, TReportPrinter
Description: This event will be called whenever a new page is generated. This can be useful to initialize page related items.
See Also: NewPage

OnPageChange:TNotifyEvent (read/write/pub)
Category: Preview
Components: TFilePreview
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

OnPrint:TNotifyEvent (read/write/pub)
Category: Control
Components: TReportFiler, TReportPrinter
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: OnPrintPage, Execute, NewPage

OnPrintFooter:TNotifyEvent (read/write/pub)

Category: Control
Components: TReportFiler, TReportPrinter
Description: This event will be called after the body for each page has been printed. This can be useful for printing similar footers for each page.
See Also: OnPrintHeader, GotoFooter, PrintFooter

OnPrintHeader:TNotifyEvent (read/write/pub)

Category: Control
Components: TReportFiler, TReportPrinter
Description: This event will be called before the body for each page has been printed. This can be useful for printing similar headers for each page.
See Also: OnPrintFooter, GotoHeader, PrintHeader

OnPrintPage:TPrintPageEvent (read/write/pub)

Category: Control
Components: TReportFiler, TReportPrinter
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: OnPrint, Execute

OnZoomChange:TNotifyEvent (read/write/pub)

Category: Preview
Components: TFilePreview
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: ZoomIn, ZoomOut, RedrawPage

Orientation:TOrientation (read/write/pub)

Category: Printer
Components: TReportFiler, TReportPrinter
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.

OriginX:double (read/write)
OriginY:double (read/write)

Category: Position
Components: TReportFiler, TReportPrinter
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: labels).

OutputName:string (read/write)

Category: Printer
Components: TReportFiler, TReportPrinter
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'. 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 (example: 'copy printer.dmp lpt1 /b').
See Also: Port

PageHeight:double (read only)
PageWidth:double (read only)

Category: Printer
Components: TReportFiler, TReportPrinter
Description: These properties return the height and width of the currently selected paper size.

PageInvalid:boolean (read only)

Category: Control
Components: TReportFiler, TReportPrinter
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

Pages:integer (read only)

Category: Preview
Components: TFilePreview
Description: This property returns the total number of pages that exist inside the report file for a preview screen.

Pie method

Syntax: `procedure Pie(X1,Y1,X2,Y2,X3,Y3,X4,Y4: double);`
Category: Graphics
Components: TReportFiler, TReportPrinter
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

Polygon

Syntax: `procedure Polygon(const Points: array of TPoint);`
Category: Graphics
Components: TReportFiler, TReportPrinter
Description: This method will draw a polygon using the current pen defined by the points contained in the open array, Points, then closes the shape between the first and last points and fills it using the current brush.
See Also: CreatePoint

Polyline method

Syntax: `procedure Polyline(const Points: array of TPoint);`
Category: Graphics
Components: TReportFiler, TReportPrinter
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

PopPos method

Syntax: `function PopPos: boolean;`
Category: Position
Components: TReportFiler, TReportPrinter
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: PushPos

Port:string (read only)

Category: Printer
Components: TReportFiler, TReportPrinter
Description: This property will return the port name for the currently selected printer.
See Also: PrinterIndex, OutputName

PrevPage method

Syntax: `procedure PrevPage;`
Category: Preview
Components: TFilePreview
Description: This method will goto and print the previous page to the preview window. The OnPageChange event handler will be called if the current page number changes.
See Also: NextPage, CurrentPage, OnPageChange

Print method

Syntax: `procedure Print(Text: string);`
Category: Text
Components: TReportFiler, TReportPrinter
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

PrintBitmap method

Syntax:
```
procedure PrintBitmap(X,Y: double;
                      ScaleX,ScaleY: double;
                      Bitmap: TBitmap);
```
Category: Graphics
Components: TReportFiler, TReportPrinter
Description: This method will draw the bitmap, Bitmap, to the printer canvas at the point defined by (X,Y). The bitmap will be scaled by the factors ScaleX and ScaleY. (Example: A scaling factor would draw each pixel in the bitmap as 2 pixels on the printer canvas.)
See Also: PrintBitmapRect

PrintBitmapRect method

Syntax:
```
procedure PrintBitmapRect(X1,Y1,X2,Y2: double;
                          Bitmap: TBitmap);
```
Category: Graphics
Components: TReportFiler, TReportPrinter
Description: This method will draw the bitmap, Bitmap to the printer canvas stretched or shrunken to fit within the rectangle defined by the points (X1,Y2) and (X2,Y2).
See Also: PrintBitmap

PrintCenter method

Syntax:
```
procedure PrintCenter(Text: string;
                      Pos: double);
```
Category: Text
Components: TReportFiler, TReportPrinter
Description: This method will print the string, Text, on the current line centered horizontally at the position, Pos.
See Also: all other print functions

PrintData method

Syntax: `procedure PrintData(Value: string);`
Category: Printer
Components: TReportFiler, TReportPrinter
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: 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.
See Also: all other print functions

PrinterIndex:integer (read/write)

Category: Printer
Components: TReportFiler, TReportPrinter
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 or -2 to use the setting that already has been set with a TPrinterSetupDialog.
See Also: SelectPrinter

PrintFooter method

Syntax:
```
procedure PrintFooter(Text: string;
                      Justify: TPrintJustify);
```
Category: Text

Components: TReportFiler, TReportPrinter
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

PrintHeader method

Syntax:
```
procedure PrintHeader(Text: string;
                      Justify: TPrintJustify);
```
Category: Text
Components: TReportFiler, TReportPrinter
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

Printing:boolean (read only)

Category: Control
Components: TReportFiler, TReportPrinter
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

PrintLeft method

Syntax:
```
procedure PrintLeft(Text: string;
                    Pos: double);
```
Category: Text
Components: TReportFiler, TReportPrinter
Description: This method will print the string, Text, on the current line left justified at the position, Pos.
See Also: all other print functions

Println method

Syntax:
```
procedure Println(Text: string);
```
Category: Text
Components: TReportFiler, TReportPrinter
Description: This method will print the string, Text, just like the Print method does, then call CRLF to go to the next line.
See Also: all other print functions, CRLF

PrintMemo method

Syntax:
```
procedure PrintMemo(MemoBuf: TMemoBuf;
                    Lines: longint;
                    PrintTabs: boolean);
```
Category: Memo
Components: TReportFiler, TReportPrinter
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: TMemoBuf, MemoLines

PrintPage method

Syntax:
```
procedure PrintPage(PageNum: word);
```
Category: Preview
Components: TFilePreview
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: RedrawPage, OnPageChange

PrintRight method

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

Category: Text
Components: TReportFiler, TReportPrinter
Description: This method will print the string, Text, on the current line right justified at the position, Pos.
See Also: all other print functions

PrintXY method

Syntax:
```
procedure PrintXY(X,Y: double;
                  Text: string);
```
Category: Text
Components: TReportFiler, TReportPrinter
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

PushPos method

Syntax:
```
function PushPos: boolean;
```
Category: Position
Components: TReportFiler, TReportPrinter
Description: This method will push the current text cursor position onto an internal stack for later retrieval by PopPos.
See Also: PopPos

Rectangle method

Syntax:
```
procedure Rectangle(X1,Y1,X2,Y2: double);
```
Category: Graphics
Components: TReportFiler, TReportPrinter
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

RedrawPage method

Syntax:
```
procedure RedrawPage;
```
Category: Preview
Components: TFilePreview
Description: This method will redraw the current page for the preview screen.
See Also: PrintPage

Reset method

Syntax:
```
procedure Reset;
```
Category: Control
Components: TReportFiler, TReportPrinter
Description: This method will reset certain settings (Pen, Brush, Origins, Columns, Tabs, Sections and Text Cursor position) to their default values.

ResetLineHeight method

Syntax:
```
procedure ResetLineHeight;
```
Category: Position
Components: TReportFiler, TReportPrinter
Description: This method will reset the property LineHeight to the current font if the LineHeightMethod property is equal to lhmFont. Otherwise, ResetLineHeight sets LineHeight to the value of 1.0/LinesPerInch.
See Also: LineHeight, LineHeightMethod

ResetPrinter method

Syntax:
```
procedure ResetPrinter;
```
Category: Printer
Components: TReportFiler, TReportPrinter
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: DevMode

ResetSection method

Syntax: `procedure ResetSection;`
Category: Page
Components: TReportFiler, TReportPrinter
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

ResetTabs method

Syntax: `procedure ResetTabs;`
Category: Tabs
Components: TReportFiler, TReportPrinter
Description: This method resets the current tab to the beginning. CRLF calls this function to reset the current tab.
See Also: ClearTabs, SetTab

RestorePos method

Syntax: `function RestorePos(Index: byte): boolean;`
Category: Position
Components: TReportFiler, TReportPrinter
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

RoundRect method

Syntax: `procedure RoundRect(X1,Y1,X2,Y2,X3,Y3: double);`
Category: Graphics
Components: TReportFiler, TReportPrinter
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: Rectangle

SavePos method

Syntax: `function SavePos(Index: byte): boolean;`
Category: Position
Components: TReportFiler, TReportPrinter
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: RestorePos

ScaleX:double (read/write/pub)
ScaleY:double (read/write/pub)

Category: Control
Components: TReportFiler, TReportPrinter
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:

ScrollBox:TScrollBox (read/write/pub)

Category: Preview
Components: TFilePreview
Description: This property defines the scroll box on the preview form that the report will be drawn in.

SectionLeft:double (read/write)
SectionRight:double (read/write)
SectionTop:double (read/write)

SectionBottom:double (read/write)

Category: Position
Components: TReportFiler, TReportPrinter
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: 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 the same 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: SectionRight := 8.0 would be the same as MarginRight := 0.5 for 8.5 inch wide paper.)
See Also: margin properties, ResetSection

SelectPrinter method

Syntax:
```
function SelectPrinter(SubStr: string): boolean;
```
Category: Printer
Components: TReportFiler, TReportPrinter
Description: This method will set the current printer to the first printer in Printer.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

SetBrush method

Syntax:
```
procedure SetBrush(NewColor: TColor;
                   NewStyle: TBrushStyle;
                   NewBitmap: TBitmap);
```
Category: Graphics
Components: TReportFiler, TReportPrinter
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

SetColumns method

Syntax:
```
procedure SetColumns(NewColumns: integer;
                     Between: double);
```
Category: Columns
Components: TReportFiler, TReportPrinter
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: SetColumnWidth, ColumnWidth, SectionLeft, SectionRight

SetColumnWidth method

Syntax:
```
procedure SetColumnWidth(Width: double;
                         Between: double);
```
Category: Columns
Components: TReportFiler, TReportPrinter
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: SetColumns, Columns, SectionLeft, SectionRight.

SetFont method

Syntax:
```
procedure SetFont(NewName: string;
                  NewSize: integer);
```
Category: Text
Components: TReportFiler, TReportPrinter
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).
See Also: CreateFont

SetPaperSize method

Syntax:
```
procedure SetPaperSize(Size: integer;
                       Width: double;
```

```
                                    Height: double);
```
Category: Printer
Components: TReportFiler, TReportPrinter
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.

SetPen method

Syntax:
```
procedure SetPen(NewColor: TColor;
                 NewStyle: TPenStyle;
                 NewWidth: integer;
                 NewMode: TPenMode);
```
Category: Graphics
Components: TReportFiler, TReportPrinter
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

SetTab method

Syntax:
```
procedure SetTab(NewPos: double;
                 NewJustify: TPrintJustify;
                 NewWidth: double;
                 NewMargin: double;
                 NewLines: byte;
                 NewShade: byte);
```
Category: Tabs
Components: TReportFiler, TReportPrinter
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) justfied. 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 BOXLINExxxxx 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

SetTopOfPage method

Syntax:
```
procedure SetTopOfPage;
```
Category: Position
Components: TReportFiler, TReportPrinter
Description: This method will set SectionTop to the bottom of the current line.
See Also: SectionTop, MarginTop

Start method

Syntax:
```
procedure Start;
```
Category: Preview
Components: TFilePreview
Description: 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.
See Also: Finish

StatusFormat:string (read/write/pub)

Category: User
Components: TReportFiler, TReportPrinter
Description: This property defines the format for the text printed to StatusLabel during a UpdateStatus call. There are several special formatting character pairs that can be used within the string:

%p - Current Page
%f - First Page
%l - Last Page
%d - Printer Device Name

%r - Printer Driver Name
%t - Printer Port
%0 through %9 - Status Text Line (see StatusText)
%% - % character

See Also: StatusLabel, StatusText, UpdateStatus

StatusLabel:TLabel (read/write/pub)

Category: User
Components: TReportFiler, TReportPrinter
Description: This property defines the TLabel component that UpdateStatus will put the status text, StatusFormat, into.
See Also: StatusFormat, StatusText, UpdateStatus

StatusText:TStrings (read/write/pub)

Category: User
Components: TReportFiler, TReportPrinter
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

Stream:TStream (read/write/pub)

Category: Control
Components: TReportFiler, TFilePrinter, TFilePreview
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: StreamMode, FileName

StreamMode:TStreamMode (read/write/pub)

Category: Control
Components: TReportFiler, TFilePrinter, TFilePreview
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. 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: Stream, FileName

StretchDraw method

Syntax:
```
procedure StretchDraw(const Rect: TRect;
                      Graphic: TGraphic);
```
Category: Graphics
Components: TReportFiler, TReportPrinter
Description: This method draws the graphic object, Graphic, to the printer canvas strecthed 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

Tab method

Syntax:
```
procedure Tab(LeftWidth: integer;
              RightWidth: integer;
              TopWidth: integer;
              BottomWidth: integer;
              ShadeOverride: integer);
```
Category: Tabs
Components: TReportFiler, TReportPrinter
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. 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

TabEnd method

Syntax: `function TabEnd(Index: integer): double;`
Category: Tabs
Components: TReportFiler, TReportPrinter
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

TabShade:integer (read/write/pub)

Category: Tabs
Components: TReportFiler, TReportPrinter
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: SetTab, Tab

TabStart method

Syntax: `function TabStart(Index: integer): double;`
Category: Tabs
Components: TReportFiler, TReportPrinter
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

TabWidth method

Syntax: `function TabWidth(Index: integer): double;`
Category: Tabs
Components: TReportFiler, TReportPrinter
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: TabStart, TabEnd

TextBKMode:TBKMode (read/write/pub)

Category: Graphics
Components: TReportFiler, TReportPrinter
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

TextRect method

Syntax:
```
procedure TextRect(      Rect: TRect;
                         X,Y: double;
                   const Text: string);
```
Category: Graphics
Components: TReportFiler, TReportPrinter
Description: This method will draw the text, Text, clipped within the rectangle defined by Rect. The point (X,Y) defines the starting point of the text.
See Also: CreateRect, all print methods

TextWidth method

Syntax: `function TextWidth(Text: string): double;`
Category: Text
Components: TReportFiler, TReportPrinter
Description: This method will return the length of the string, Text.

Title:string (read/write/pub)

Category: User
Components: TReportFiler, TReportPrinter
Description: This property defines the title for the current print job that will be displayed in the Windows Print Manager.

Units:TPrintUnits (read/write/pub)

Category: Units
Components: TReportFiler, TReportPrinter
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

UnitsFactor:double (read/write/pub)

Category: Units
Components: TReportFiler, TReportPrinter
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. (Example: unCM = 2.54 since 2.54 centimeters equal an inch.)
See Also: Units

UpdateStatus method

Syntax:
```
procedure UpdateStatus;
```
Category: User
Components: TReportFiler, TReportPrinter
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

XD2I and YD2I methods

Syntax:
```
function XD2I(Pos: longint): double;
function YD2I(Pos: longint): double;
```
Category: Units
Components: TFilePrinter,TFilePreview
Description: These methods will convert printer canvas measurements (dots) to inch measurements.
See Also: all other units conversion functions

XD2U and YD2U methods

Syntax:
```
function XD2U(Pos: longint): double;
function YD2U(Pos: longint): double;
```
Category: Units
Components: TReportFiler, TReportPrinter
Description: These methods will convert printer canvas measurements (dots) to unit measurements (defined by Units and UnitsFactor).
See Also: Units, UnitsFactor, all other units conversion functions

XDPI:integer (read only)
YDPI:integer (read only)

Category: Printer
Components: TReportFiler, TReportPrinter
Description: These properties return the horizontal and vertical dots per inch for the current printer.

XI2D and YI2D methods

Syntax:
```
function XI2D(Pos: double): longint;
function YI2D(Pos: double): longint;
```
Category: Units
Components: TReportFiler, TReportPrinter
Description: These methods will convert inch measurements to printer canvas measurements (dots).
See Also: all other units conversion functions

XI2U and YI2U methods

Syntax: `function XI2U(Pos: double): double;`
`function YI2U(Pos: double): double;`
Category: Units
Components: TReportFiler, TReportPrinter
Description: These methods will convert inch measurements to unit measurements (defined by Units and UnitsFactor).
See Also: Units, UnitsFactor, all other units conversion functions

XPos:double (read/write)
YPos:double (read/write)

Category: Position
Components: TReportFiler, TReportPrinter
Description: These properties set or return the horizontal and vertical text cursor position.
See Also: CursorXPos, CursorYPos

XU2D and YU2D methods

Syntax: `function XU2D(Pos: double): longint;`
`function YU2D(Pos: double): longint;`
Category: Units
Components: TReportFiler, TReportPrinter
Description: These methods will convert unit measurements (defined by Units and UnitsFactor) to printer canvas measurements (dots).
See Also: Units, UnitsFactor, all other units conversion functions

XU2I and YU2I methods

Syntax: `function XU2I(Pos: double): double;`
`function YU2I(Pos: double): double;`
Category: Units
Components: TReportFiler, TReportPrinter
Description: These methods will convert unit measurements (defined by Units and UnitsFactor) to inch measurements.
See Also: Units, UnitsFactor, all other units conversion functions

ZoomFactor:double (read/write/pub)

Category: Preview
Components: TFilePreview
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

ZoomIn method

Syntax: `procedure ZoomIn;`
Category: Preview
Components: TFilePreview
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

ZoomInc:integer (read/write/pub)

Category: Preview
Components: TFilePreview
Description: This property defines the amount that ZoomIn and ZoomOut modifies ZoomFactor.
See Also: ZoomFactor, ZoomIn, ZoomOut

ZoomOut method

Syntax: `procedure ZoomOut;`
Category: Preview
Components: TFilePreview

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: ZoomIn, ZoomInc, ZoomFactor, OnZoomChange

ZoomPageFactor:double (read only)

Category: Preview
Components: TFilePreview
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.
See Also: ZoomFactor, ZoomPageWidthFactor

ZoomPageWidthFactor:double (read only)

Category: Preview
Components: TFilePreview
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.
See Also: ZoomFactor, ZoomPageFactor