

Get the world's fastest report designer
working for
you!



PrintDAT!
Tutorials



GREBAR SYSTEMS INC,

<http://www.grebarsys.com/>

© 1998-2012 Grebar Systems Inc.
All Rights Reserved

Phone 204-942-3301
support@grebarsys.com

Table of Contents

Introduction.....	1
What is PrintDAT!?.....	1
.....	1
Conventional Report Writer Inadequacies.....	2
The Smart Solution.....	2
PrintDAT! Gives You Instant Reports.....	2
How PrintDAT! Works.....	3
Some of PrintDAT! Features:.....	3
Improved Productivity.....	4
How good is PrintDAT!?.....	5
We think our customers say it best.....	5
Send us your quotes and testimonials.....	9
PrintDAT! makes all	
these components printable!.....	10
Print standard Delphi Components:.....	10
Client Server.....	10
Datsnap™/Midas™.....	10
Print any BDE table or TDataset Descended Component.....	10
Print 3rd Party Databases.....	11
Conventions Used in this Manual.....	14
Lesson #1 - Getting Started.....	16
Your first PrintDAT! Report.....	16
Steps.....	16
Steps Explained.....	18
Accessing the Printer and the Report Options.....	18
Congratulations.....	19
Create a report in 1 second. Believe it or not!.....	19
Lesson #2 – Report Options.....	21
PrintDAT! has 5 defining factors that makes it unique:.....	21
Text File Output.....	21
Opening the Report Options Window.....	24

Export ASCII Delimited File.....	25
Importing the file into MS Excel™	25
Report Alignment.....	27
Wide Reports.....	27
Really Wide Reports.....	27
Really Really Wide Reports.....	28
Really Really Really Wide Reports.....	28
Narrow Reports.....	29
3 Panels Across Reports.....	29
Reports For Overhead Projectors.....	30

Lesson #3 - Creating TTable and TQuery Reports.....31

Any Table, Anytime.....	32
-------------------------	----

Lesson #4 – Printing TStringGrid.....33

Memory Grids & TrimGridDim.....	33
Override Field Alignment.....	33
Smart Field Alignment for Memory Grids.....	34
Fixed Columns.....	35

Lesson #5 – Column Totals.....36

Adding A Column Total.....	36
Text Non-Total.....	39
Getting Numeric Field Value From The Grid.....	39
Numeric Conversion Exception Handling.....	40
Do your own field conversion.....	40
Numerical Accuracy.....	41
Formatting.....	41
Two Line Totals.....	42
Preventing Column Totals From Printing.....	42
Column Totals That Wrap.....	42
Invalid Totals.....	43
Blank Fields.....	43
Standard Deviation & Variance: Population vs Sample.....	43

Lesson # 6

The ever elusive TDecisionGrid Report.....45

Additional Features.....	46
Sample Report #1.....	47
Sample Report #2.....	48
Sample Report #3.....	48

It's dimensions are amazing.....	52
Lesson #7 - Automatic Printing.....	53
Lesson #8 - Setting Report Options from your application.....	55
AfterOptionsLoaded Event - Example #1.....	55
Example#2 - Using a String for Page Title.....	56
Explanation.....	57
Saving Report Options to File.....	58
Example#2 - Using a Memo for the Report Header.....	58
Explanation.....	59
AfterOptionsLoaded Event: Example#3 - Making Report Options Read Only & Hiding Report Options Pages.....	59
Lesson #9 – OnFieldEdit Event.....	61
Changing cell data before it gets into the report.....	61
Changing a cell's data with a lookup.....	61
Flagging certain rows of a report.....	64
Printing Column Titles onto 2 or more lines.....	64
Printing Cell Data onto 2 or more lines.....	65
Lesson #10 – Printing Selected Rows & Report Filters....	67
Printing Multi-Selected Grid Rows.....	67
Ask the user if they want to print just the selected rows.....	69
Report Filter.....	69
Report Filter Example.....	70
Lesson #11 – Printing Group Breaks.....	73
How to create a group break report:.....	75
Group Break Rules.....	77
Breaking all the rules.....	78
Breaking Rule #1.....	78
Breaking Rule #2.....	81
Lesson #12 – Unicode Output.....	84
Disadvantages of ANSI:.....	84
Disadvantages of Unicode:.....	84

Languages Supported.....85
 How does an English Unicode report compare to an ANSI report using our
 Letter Gothic Line PD font?.....89
 Activating Unicode.....93

Lesson #13 – Storing Report Options.....94

.PXT Text File.....94
 Unicode Reports.....94
 TDataset Table.....94
 Table Schema.....95

Lesson #14 - Distributing PrintDAT! with your Application.....97

1) Distribute the Report Settings File.....97
 .PXT File.....97
 Database Table.....97
 2) Distribute the Linedraw Font (optional).....97

 3) User Help File.....98
 4) Distribute PrintDAT! Runtime Package (Optional).....98

Lesson #13--Helpful Hints.....100

PrintDAT! has several features you may have missed that will make your job easier. 100

Introduction

*Contrary to popular belief,
creating reports does not have to be like root canal.*

What is PrintDAT!?

PrintDAT! was created out of necessity in order to fill a major need in the programming language. Grids are the cornerstones of all database applications. Grids are essential in getting the data from the database to the computer screen. Unfortunately once on the screen there is no easy way to print the grid or export it to another program. This *logical link* of getting the data from the grid to the report is missing.

Over the years, programmers have been filling this gap with conventional report writers. Unfortunately report writers are either too complicated and have a steep learning curve, or are too laborious and designing reports becomes much like manual labor. Ask any novice programmer any they'll probably agree that writing reports is one of their greatest obstacles. Most can't understand why creating reports has to be so difficult. At the other end of the spectrum, power programmers who have mastered report writers will find writing reports too time consuming and their productivity slows to a crawl. In order to come up with a better solution we must first recognize the problems that are inherent in conventional report writers.

Conventional Report Writer Inadequacies

Creating a columnar report with a conventional report writer is a lot of hard work. You must:

- Create a new program unit to contain the report
- Link the new unit to your program
- Instantiate the report form at runtime
- Add the proper dataset to the report
- Drag & drop fields onto the report
- Coax the fields into position
- Adjust the column widths to suit the underlying data
- Try and allow enough space for memo fields
- Squeeze the columns together so all of the columns fit across the page, being careful not to truncate any of the data.
- If the columns don't fit, choose a smaller font for the fields and reposition all of them
- If more columns are added later, redo the last 6 steps!
- If there are too many columns for the page width, give up!

It's no wonder programmers don't like writing reports. It's too much hard work. They can spend half a day just creating a few little reports. Not only that, but a lot of this work may seem like *deja vu* because they're already done most of the work when they designed their database grids. Why on earth would anyone want to do it all over again?

The Smart Solution

Since you already have a grid with all of the information in it, why can't we use the grid to create the report? What we really need is a report component that will *quickly produce reports* from a grid with all of the fields *properly formatted* and *error free* the very first time the report is printed. It should eliminate all of the steps outlined above so the report can be completed in less than 30 seconds. Does this sound like you're dreaming? Get ready to pinch yourself because you're about to find out how easy it is to create a report.

PrintDAT! Gives You Instant Reports

PrintDAT! is the missing link that connects the grid to the report. Our motto is:

"If you have a grid, you have a report"

Memorize this statement and let it become your mantra because it will save you hours and hours of work over the next few weeks. If you have PrintDAT! on your component palette, every time you see a grid you should also see the opportunity to produce an **Instant Report**. Forget about the antiquated and laborious methods of using report writers to print a grid. Let your competitors use manual labor if they like, but you have better things to do with your time. PrintDAT! can design and print a grid report in only a few seconds and with only 1 line of code. *It doesn't get any easier than this!*TM

How PrintDAT! Works

PrintDAT! is much easier to use than a report writer because it is a dialog component. Dialog components are much easier to use because you do not need to create a new unit to store the report. We made PrintDAT! intelligent so it will do most, if not all of the work for you. All you need to do is drop the PrintDAT! component onto a form that has a grid, then double click it to see the report. What could be easier than that?

Some of PrintDAT! Features:

- It takes just 1 line of code to print a report.
- Does all of the field positioning
- Calculates the column widths by scanning the underlying data whenever the report is run.
- Automatically adjusts the report width to the page width. This eliminates wasted white space by automatically adding columns to fill the entire width of the page. This means when you print in landscape mode, the report will automatically print more columns across the page. If you switch to legal size paper, the report automatically prints more lines per page.
- Long strings and memos are no longer a problem because they are automatically word-wrapped onto several lines thereby eliminating wide columns that hog the entire page,
- *Shrink to page* automatically rescales the font sizes to produce wider reports
- For the really wide reports, PrintDAT! can use horizontal page breaks to print grids up to 1,000 columns across. This means the report can be 100 pages across! Your reports can be wide enough to wallpaper your office.

- PrintDAT! has dozens of report options that can be changed at runtime without having to modify and recompile the program for every little change to the report. All of these options can also be changed under program control.
- PrintDAT! can produce column totals at runtime, and there are no messy formulas to enter.
- You can even create and run the report without leaving the Delphi or C++Builder IDE.
- Prints all Delphi grids and dataset components like TTable and TQuery as well as most 3rd party databases and 3rd party grids like InfoPower™.
- All of the code is native Delphi and gets compiled into your .exe. Supports 32-bit and now 64-bit applications with Delphi XE2.
- Unicode support for printing reports in over 40 languages.

Improved Productivity

Because PrintDAT! has a very low learning curve with its drop and click methodology, novice programmers are very productive and will produce bug free reports in just seconds. This is great for building their self-esteem. Power users will also improve their productivity because they no longer have to waste their time formatting columnar reports. They can see the results instantly.

How good is PrintDAT!?

We think our customers say it best.

"My current project requires multiple Decision Cube analyses from many tables. Before taking on this project I should have considered the consequences of having to produce printed reports for such varied analyses. When I began my search on the Internet for a way to print Decision Grids I felt I was in trouble. With difficulty I finally found PrintDAT!, downloaded it and installed it easily. I added it to my demo using a Decision Grid and added the one liner in an OnClick event of a button: 'PrintDAT.Print()'. After seeing the result and the available options I paid for a license.

I tell others that I bought PrintDAT! after evaluating it for only 10 minutes because I feared that it might disappear on me if I didn't act quickly. :)

It enables me to be unbelievably productive. I can define the settings I think are appropriate and yet users can change it to their own liking! It even comes with its own end-user Help file!! Such professionalism I have not seen before in a Delphi component.

It's not just the huge amount of time one saves, but also the lack of stress that is usually involved in producing reports with some report writers.

When I wanted to do something special with a field not only did I receive a prompt reply from the author that included useful information, tips and a reference to an event in the developer help file, but also some extremely useful code. Want something more? He also sent me a demo!"

Raymond Kennington
B.Sc.(Ma.Sc.)(Hons) Grad.Dip.Ed., B.Comp.Inf.Sc.(Hons)
Programming Solutions

"I just thought I would let you know that I've installed PrintDat successfully and have had a chance to try it out. PrintDat has far exceeded my expectations! It is an extremely powerful, well written and well designed Package.

Very seldom do you find software that lives up to it's hype, however this is one very pleasant exception. You people at Grebar Systems can be very proud of this product."

Thanks again for your help,
Kenneth Kaye
Clestra Hauserman

"I'm a registered user of PrintDat! and think it's a super component. I originally purchased it a year ago to print DecisionCube results for Delphi 3. I am just embarking on a new project and am looking forward to incorporating PrintDat!."

Curtis Brettin
MHSA

"We love PrintDat!. You certainly gave us a merry Christmas and happy new year. All the best. Looking forward to working with you in the future."

Kenny B.
Horizon Global Trading

"We've owned it a relatively short time and I found it to be THE BEST grid reporter around."

Kevin Morris,
SPAR Aviation Services

"I'm with Horizon Global Trading, in New York City's Financial District. We are a Delphi shop that services the financial industry with applications that manage securities trading and inventory, and we have purchased PrintDat 1.5 with source code.

I am happy to report that we are finding PrintDat to be quite flexible and reliable, and we hope to use it in an upcoming release of our flagship product for many important reporting tasks. Thank you for a work saving component like this!"

Mike Nachison
@Horizon

"PrintDat is marvelous. It's THE component I'm waiting to complement the grid!"

Marcos Cunha Lima/Brazil

*"Wow! – PrintDAT! now lekker
Yep!, lekker is the best word invented ever. It is a Afrikaans word
meaning anything that is good, tasty, feels good etc. etc. etc."*

Theo Pistorius

"This (PrintDAT!) is an excellent product."

Humberto

"The component looks great. You've done a good job!"

Mike

*"I think PrintDAT is fabulous!! Not only is it easy to use but it is fast and
generates exceptional output. This is the only reporting tool I use."*

Jeff Carter
Sr. Software Engineer
Rockwell Collins

*"What a cool tool. I've only been using it a couple of days and it's
changed my way of thinking about reporting so that I almost forgot what
it's like not to use it. It is simple and good enough for my developer busy
life..."*

Ricardo Rego

I want to thank for your (very fast) help. I think your support is excellent. This makes using Printdat! even better!

Harry

"I have found PrintDAT to be one of the most useful components I've purchased. The ease with which I've developed a reporting facility on my database program was surprising. I'm using Microsoft Access as the data base, Diamond Access as my MS to Delphi interface, and PrintDAT to do most of the work. Having attempted to develop reports in MS Access, this combination has made life much easier. Thanks for a great component."

Neil Pollard
Quality Assurance Manager
AMC Search Limited

"We've been working with PrintDAT for about a day now. I just wanted to tell you that after experimenting with a number of report writers : Crystal, Piparti, %\$^&# Quick Report, Report Maker - and about 40 hours of wasted efforts, we had thought it was impossible to find a report writer/printing component to just allow us to print what we saw on the screen (in a grid or list). We always had to create a special report which took way too much time (and in most cases did not work).*

I wanted to let you know that we added all of the printing capabilities that we had hoped for to our application within 4 hours of work. Our users are now very happy with the new feature to print anything that they see and after posting an update 2 hours ago, we've already gotten compliments from some of our larger corporate customers.

I wanted to thank you personally for making us look good, and second for making a print component so that we no longer have to dread adding a print button to a dialog or form.

I hope you stick with your development of this product. I know you have a lot of "competitors" - but in my mind they are no competition for you.

Let us know when the source code is available - we want it!"

Eric Siko
President
Linked Systems

Send us your quotes and testimonials.

We'd like to hear from you. If you like PrintDAT! please send us a quote or a testimonial that we can share with other developers. There are a lot of programmers out there who haven't heard of PrintDAT! and this is one way of getting the word out.. You can send your comments to support@grebarsys.com.

And if you think PrintDAT! is just for printing DbGrid's, take a look at the next section. You'll be amazed at all of the components PrintDAT! can make printable.

PrintDAT! makes all these components printable!

Print standard Delphi Components:

TdbGrid
TCustomDbGrid
TStringGrid
TListBox
TListView
TTable
TQuery
TDataSet
TADOTable
TADOQuery

Client Server

TDecisionGrid : Decision Cube
TDecisionQuery: Decision Cube

Datasnap™/Midas™

TClientDataSet

Print any BDE table or TDataset Descended Component

Paradox™
DBase™
Interbase™
Informix™
Oracle™
AS/400™
and more...

Print 3rd Party Databases

(The following databases and grids have all been verified to work with PrintDAT!)

Advantage™ www.advantagedatabase.com/

Advantage Database Server is a scalable, high performance client/server DBMS for networked, standalone, Internet and mobile database applications. The Advantage Database Server supports the NetWare, Windows NT/2000, and Windows 95/98 operating systems.

TadsTable, TadsQuery, Views (Note: ADS 6 has not been tested.)

NexusDb™ <http://www.nexusdb.com/>

NexusDB is an ultra-fast, client/server database engine originally designed for the Delphi and C++Builder developer. For the current version 2 it has been extended to provide most of it's functionality to .NET programmers no matter which programming language they use. NexusDB is nimble enough to be fully embedded into your desktop applications and powerful enough to be your primary database server.

TnxTable, TnxQuery

CodeBase™ www.sequiter.com/

Fast database with free distribution. CodeBase replaces the BDE with smaller memory and disk requirements without sacrificing speed. Supports client/server, stand-alone, querying, transactions, security, and has xBase file compatibility. OLE DB/ADO, ODBC, and SQL support is also available.

TCBTableSet, TCBQuerySet (from Codebase Components II)

DBISAM™ www.elevatesoft.com/

TDBISAMTable, TDBISAMQuery, TDBISAMGrid

ElevateDb™ www.elevatesoft.com/

TEDBTable, TEDBQuery

DiamondAccess™www.islamov.com/diamond/

Diamond Access is a set of Delphi components, that provide high-speed performance when working with Microsoft's Access databases. Diamond Access uses Data Access Objects 3.5 (DAO) to work directly with a Jet engine, providing fastest possible interface to the Access databases.

TDAOTable, TDAOQuery, TDAOMasterDetailTable

DiamondADO™www.islamov.com/diamondado

Diamond ADO is a Delphi component library, that provide high-speed performance when working with any OLE DB provider or ODBC data source. Diamond ADO uses ActiveX Data Objects 2.1 (ADO) to access data without requiring the BDE. Using Diamond ADO you can connect to any OLE DB provider. Currently OLE DB providers exists for SQL Server, Oracle, Access, ODBC, Active Directory Services and the Index Server.

TdADOTable, TdADOQuery

FlashFiler™ 1.5 & 2www.turbopower.com

FlashFiler is a professional, compact, multi-user database engine for programmers developing applications with Borland Delphi and C++Builder. FlashFiler may be used as a client/server database system or as a high-performance database engine embedded within your own application.

TffTable, TffQuery, TffStringGrid

**InfoPower™ 2000 &
InfoPower 4.x for Delphi 3**www.woll2woll.com

InfoPower is an award-winning suite of visual components specifically designed to give professional database developers unmatched power in their Delphi and C++Builder applications. InfoPower includes a greatly enhanced data-aware grid, advanced edit controls, visual filtering dialogs, picture mask validation support, record viewers, incremental searching, QBE support, and more!

TwwDbGrid, TwwTable, TwwQuery

Topaz™www.softsci.com

"TOPAZ for Delphi" is a component library for Borland Delphi and C++ builder that is designed to provide support for accessing BASE tables and indexes. Database applications written with TOPAZ are totally self-contained, do not require the BDE or any other DLLs, and fit on one floppy disk.

TzDbf



Plus print virtually any TDataset descended table/query component.

For a list of non-BDE databases, visit DelphiBAG.com

Conventions Used in this Manual

This manual is made up of tutorials that will show you how to get the most out of PrintDAT!. The first lesson will demonstrate how to create a simple report from a dbGrid in just a few seconds. The remaining tutorials will reveal the incredible dynamic flexibility of PrintDAT!. So hang on to your socks, the show is about to begin

ICON KEY



Notes containing valuable information



Hint



Warning. This information must be read before proceeding.



Technical information that can be skipped by novice users.

The icons on the left will appear throughout the manual to indicate points of interest.



PrintDAT! has two help files.

PD_Dev.Hlp or PD_Dev.CHM

The developer's help file lists all of the properties and methods for the various classes. This is the help that appears when you press F1 on a TpdTPrintDAT component or the Object Inspector.

PD_User.Hlp or PD_User.CHM

The end user's help file can be distributed with your application. It has the screen snap shots of the Report Options window and hot spots for all of the settings. This is the help file that appears when F1 is pressed at runtime from a PrintDAT! window.

These help files are found in the \PrintDAT\Doc directory as well as in the PrintDAT! program group. This manual is contained in the file Tutorial.PDF and is located in the same directory.



The PrintDAT! demo program (\PrintDAT\Demo\PDDemo.DPR) is full of useful reports you won't want to miss.

Lesson #1 - Getting Started

Your first PrintDAT! Report

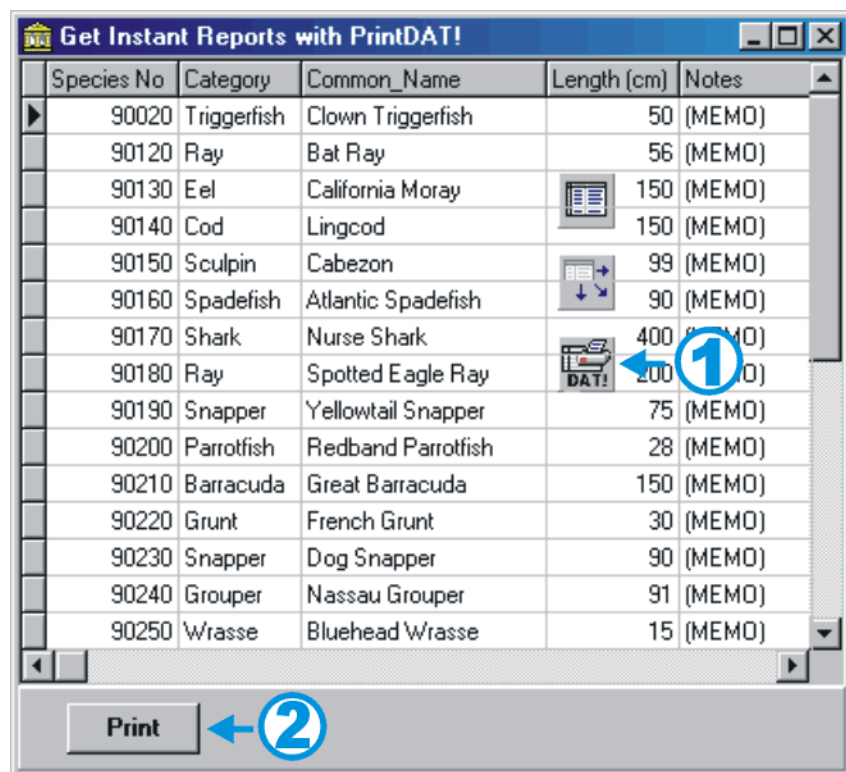
Creating a PrintDAT! report is as easy as dropping a component onto a form and adding 1 line of code. I know you're anxious to get started, so let's get down to business.

Steps

After PrintDAT has been installed in Delphi, you're ready to produce your first PrintDAT! report. All you need is something to print. Let's print a TdbGrid.



If you have a stop watch you may want to start timing to see how long it takes to create a report using PrintDAT!. Ready? Start timing now.



Creating a PrintDAT! report is as easy as 1, 2, 3.

- 1) Drop a TpdtdPrintDAT component onto a form that has a TdbGrid. This component can be found under the **Grebar** tab of the component palette.
- 2) Add a TButton and change its caption to "Print". Double click on the button and add the following code to its

OnClick event:
pdtPrintDAT1.Print;

- 3) Stop timing and don't forget to smile. ☺
That's right. You're finished. You now have a report that can print the grid and it probably took you only 30 seconds Not bad for your first time out.

What's that? You don't believe it's that simple?
Compile and run the program. When you press the Print button you'll see the report in the preview window.

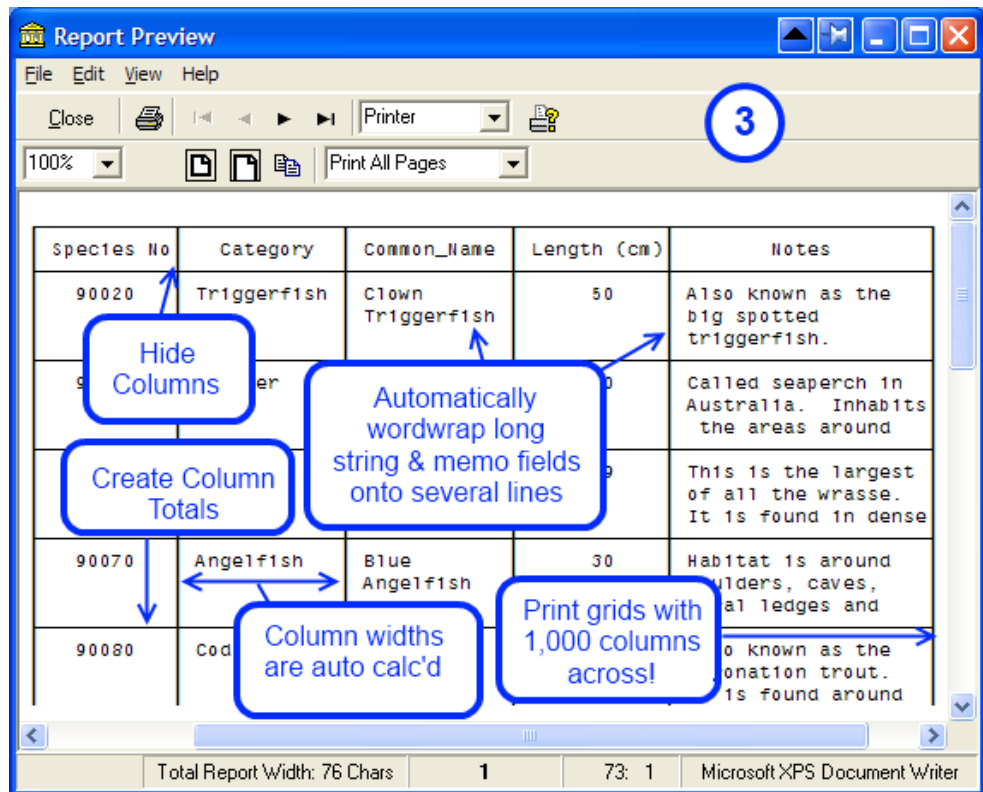


Illustration 1: Build Intelligent Reports with PrintDAT!



For more information on the options in the Report Preview window, press F1 for help.



If the compiler said the PrintDAT! unit wasn't found, check your Delphi Library path. See \PrintDAT\ReadMe.pdf for more information.

Steps Explained

Here's what happened. When you dropped a TPdtPrintDAT component onto the form it automatically scanned the components on the form looking for an object that it can print. PrintDAT! is smart enough to find the most likely object to print and will attach itself to it using its ObjectToPrint property. If you want to print a different control than the one that was selected, simply select it using the ObjectToPrint property in the Object Inspector.

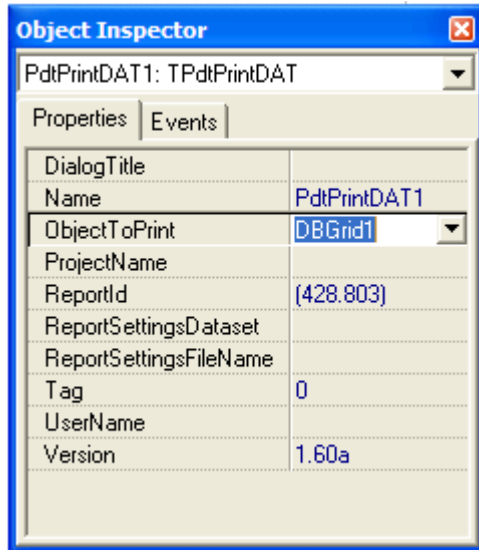





Illustration 2: TpdtdPrintDAT in the Object Inspector

The **ObjectToPrint** is automatically assigned when the TpdtdPrintDAT component is dropped onto the form.

A unique **ReportId** was also created for the TPdtPrintDAT1 object which uniquely identifies this object in the report settings file. Do not change this value without first reading the help file.

Accessing the Printer and the Report Options

- 4) With the report displayed in the report preview window, press the Print button  at the top of the preview window (or press **Ctrl-P**) to send the report to the printer. If you don't have a printer attached to your computer, select the Output Destination dropdown  and change it to **Text File**. When you press the Print button the report will be sent to the output file.
- 5) If you want to access the Report Options window, select **File > Report Options** or press **Ctrl-O**. You can return to the Report Preview window by pressing the preview  button or by pressing **Ctrl-W**. If you want the Report Options window to appear first or have the report go directly to the printer, see Lesson #7 - Automatic Printing on page 53.

Congratulations

You've just created and printed your first PrintDAT! report and it looks great. Your boss will think you spent an hour laying out all the fields when it only took you about 30 seconds.

Create a report in 1 second. Believe it or not!

In the last example you broke the world speed record for designing a report. It only took you about 30 seconds. If you try to create a similar report using a conventional reporting tool you might spend 15 to 30 minutes laying out fields and the line drawing which is tedious boring work, and you can forget about trying to print wide grids with horizontal page breaks. Conventional reporting tools can't handle it because they stop printing at the right margin.

As easy as the last example was, it's going to get even easier, so get ready to break a new report design record. Get your stop watch ready.

Find a form with something you want to print, a TdbGrid, TTable, etc and make sure the table is open. Drop a TPdtPrintDAT component onto the form and as soon as the component hits the form, start timing. Ok, stop timing. The report has been completed. We skipped steps 2 & 3, which saved us a few seconds. But you're probably scratching your head and wondering

"Without adding the code for the print button, without compiling the application, and without running the application, how on earth are we going to print the report?"

That's easy, just **double click the TpdtdPrintDAT1** object on the form. The report preview window will appear where you can press the Print button to print the report. Now that's easy! This type of temporary report will come in real handy when you're debugging your application and you need to print a TTable, TQuery, or a TdbGrid. Not only can you print the data, but you can also send it to a text file or export it into a spreadsheet or graphics program, all without writing one line of code or leaving the IDE. Think of the time you're going to save over the next year with PrintDAT!.

When you need to print a simple report using data from a TdbGrid, TwwdbGrid, TStringGrid, TTable, TQuery or the TDecisionGrid component, you won't find anything easier to use or faster than PrintDAT!.

Ok, enough of the commercial. Let's see what options we can play with by moving on to Lesson #2.



If you just started a new Delphi project by selecting **File > New Application**, you must **save the project** before you drop a TpdPrintDAT object onto your form. PrintDAT! uses the project's directory as the location to store its report settings. If the Project has never been saved, then it will erroneously put the report settings file in the wrong directory. For more information on the *Report Settings File*, see the PD_Dev.Chm file.

Lesson #2 – Report Options


Lesson #1 demonstrated how easy it is to create a PrintDAT! report. Now we're going to find out how powerful it is.

PrintDAT! has 5 defining factors that makes it unique:

- 1) Increases developer's productivity by creating bug free reports in 30 seconds or less.
- 2) Dynamic reports are powerful and flexible because it implements options that can be changed and saved at runtime without recompiling your program.
- 3) Reports are compiled into your application so they run as fast as Delphi and there are no DLL's or Active-X components to distribute.
- 4) Temporary reports can be designed in as little as 1 second without leaving the IDE.
- 5) Data manipulation. Cell data can be intercepted as it comes off of the grid using the OnFieldEdit event, so the data can be changed before it is printed. An OnFilterRecord event can filter records or terminate the report when a certain record is found.

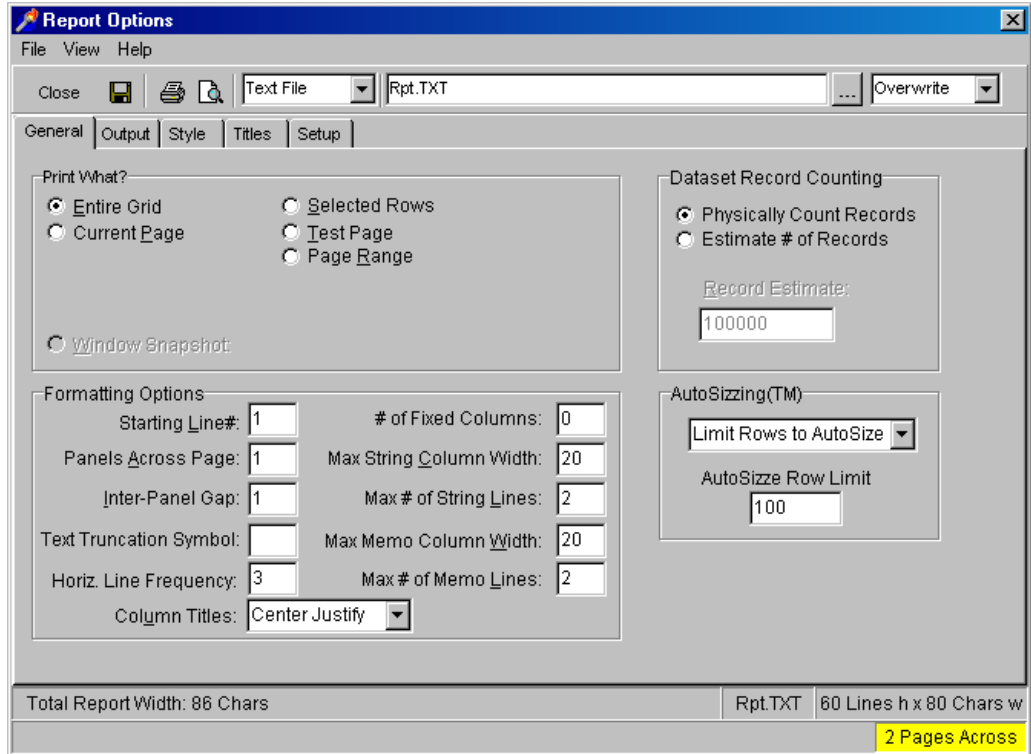
Now we're going to put option #2 to the test to see what we can do with the report options.

Text File Output

- 1) Rerun the PrintDAT! report that you created in lesson #1. If you haven't yet created the application, you can compile and run the PrintDAT! demo program in \PrintDAT\Demo\PDDemo.DPR and select the Print button from the form called "Simple".
- 2) When the Report Preview window appears press **Ctrl-O** to display the Report Options window. Then click on the Output Destination  pulldown on the toolbar. Select the item **Text File** from the list. The name of the output file appears on the status bar at the bottom of the page.

To change the name of the output file, enter the new file name in

the edit field to the right. A suffix of .TXT will indicate it is a text file. You can use a different file suffix if you prefer.



The Report Options window can be accessed by pressing Ctrl-O from the Report Preview window

(Press F1 to learn more about the report options)

- 3) Select the **Style page** and select either "**IBM LineDraw**" or "ASCII LineDraw" from the Output Quality dropdown box. If your text editor can display the IBM Linedrawing set then "IBM LineDraw" is your best choice. When IBM Linedraw is chosen, your text editor or printer will need to be able to use a character set that is capable of displaying the IBM linedraw characters. PrintDAT! by default uses the font "Letter Gothic Line PD" (LetGOTHL_PD.TTF) and was installed by the PrintDAT! installation program. Here is a sample of Letter Gothic Line font that uses the IBM Linedraw character set.

#	NAME	SIZE	WEIGHT	AREA
1	Angel Fish	2	2	Computer Aquariums
2	Boa	10	8	South America
3	Critters	30	20	Screen Savers
4	House Cat	10	5	New Orleans
5	Ocelot	40	35	Africa and Asia
6	Parrot	5	5	South America
7	Tetras	2	2	Fish Bowls

Illustration 3: Report using IBM Linedraw Characters



If you are viewing this report from a text editor, make sure you change the font to "Letter Gothic Line" or something similar like "MS Linedraw".

If your text editor or printer can't display the IBM linedraw characters, choose "ASCII Linedraw" which will use dashes and vertical bars to draw the lines as shown below:

```

-----
| #|   NAME   | SIZE| WEIGHT|   AREA   |
|---+-----+-----+-----+-----|
| 1| Angel Fish|  2 |  2 | Computer Aquariums|
| 2|  Boa     | 10 |  8 | South America     |
| 3| Critters | 30 | 20 | Screen Savers     |
| 4| House Cat| 10 |  5 | New Orleans       |
| 5| Ocelot   | 40 | 35 | Africa and Asia   |
| 6| Parrot   |  5 |  5 | South America     |
| 7| Tetras   |  2 |  2 | Fish Bowls        |
-----

```

Illustration 4: Report Using ASCII Linedraw Characters

- 4) Press the Print button and the data gets written to the text file that was specified in step 2.

This file can be read by any text editor, wordprocessor and can be cut and pasted into the clipboard.

- 5) When the output is going to a text file, you can control the width and height of the report page by changing the *Lines per Page* and *Page Width* on the Output Page.

When the output is going to the printer, PrintDAT! will automatically determine the page dimension depending on the

size of paper in your printer and by the page orientation.

- 6) If you want to remove the “#” column, simply open the Report Options window and go to the Style page. In the columns grid, uncheck the “Line#” column to hide the column.

Opening the Report Options Window

To open the Report Options window, either double click the TpdPrintDAT component from the IDE, or run the program and select “File > Report Options” from the report preview window.



To open the Report Options window, you can also press and hold the Shift-Ctrl keys while clicking on the button or menu item in your application that prints the report. Then release the Shift-Ctrl keys when the report options window appears. This allows you to get into the report options before the report is sent to the report preview window or printer.

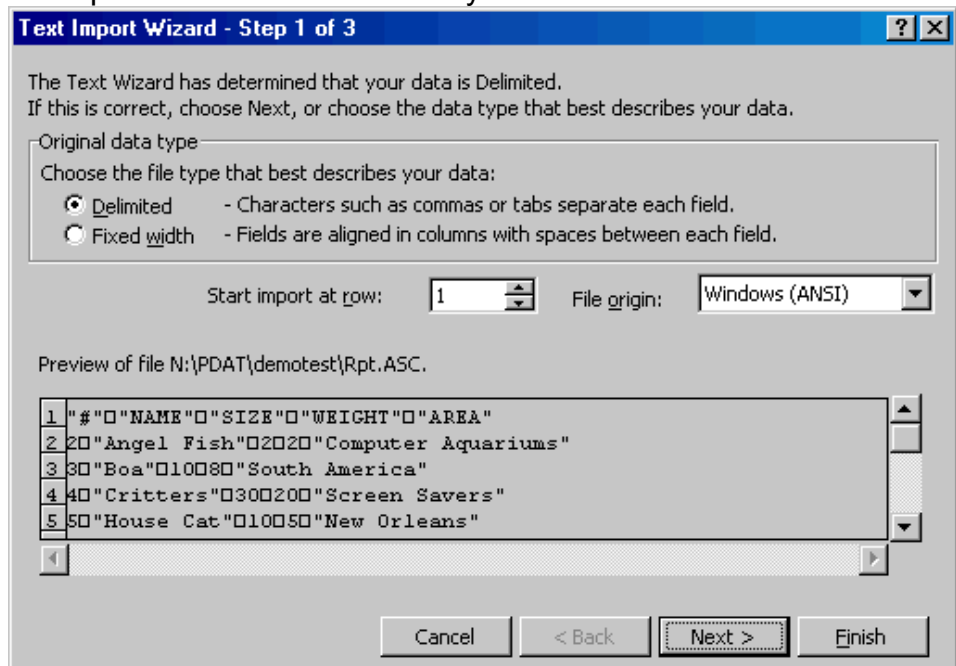
Export ASCII Delimited File

- 1) From the PrintDAT! report options window, select **Export File** from the Output Destination drop down on the status bar. The output file name changes to a file with the .ASC suffix to indicate it's an ASCII delimited file (see status bar). You can change this suffix when you enter the output file name.
- 2) The **Field Delimiter** should be set to a Tab and **Text Quotes** should set to a double quote ("). These fields are on the Output page.
- 3) Press the Print button and the data from the dbGrid gets written to the ASCII file.

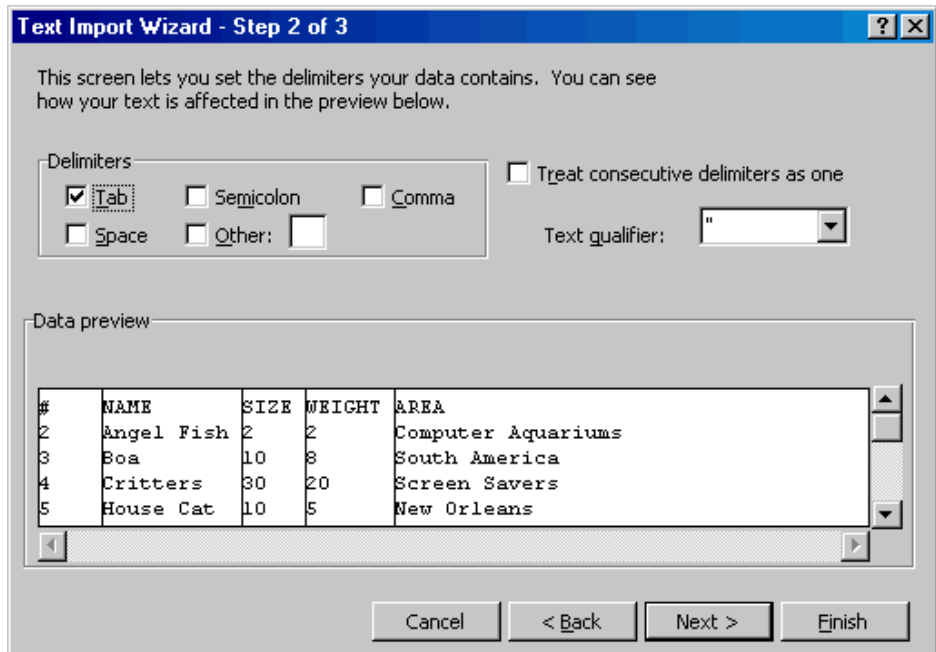
This ASCII delimited file can be read by most spreadsheets, database programs, and graphics programs. This is one terrific way of getting data out of your database and into another program. See "Exporting Data to other programs" in the PD_User.Hlp help file for more information.

Importing the file into MS Excel™

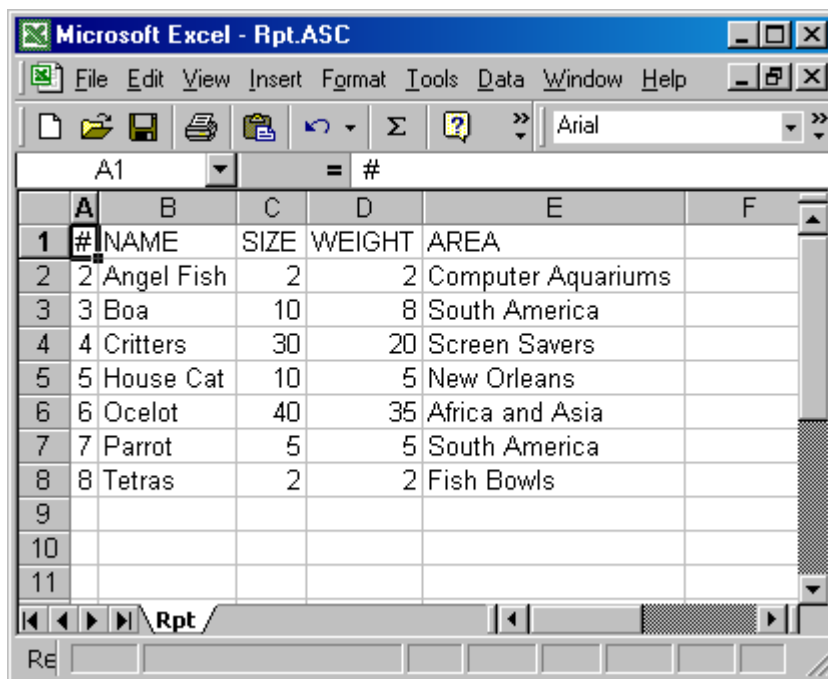
- 1) Run MS Excel and select File > Open and give it the name of the export file that was created by PrintDAT!.



- 2) Select "Delimited" default and press **Next**.



- 3) PrintDAT! defaults to Tab Delimited output, so make sure "Tab" is checked. The data should appear readable in the Data Preview window.
- 4) If you don't need to format any of the cells, press Finish to see the spreadsheet with the data from your dbGrid. Double click on the column boundaries to expand the column widths so it matches the data width.



Your data is now in MS Excel™



If you want to export certain fields or add more fields to be exported, simply create a dbGrid with the fields in it. Use lookup fields if you need to export fields from related tables. you can also export directly from a TTable or TQuery component rather than going through a dbGrid. The TTable index or TQuery "order by" clause can sort the data before it is exported. The Style page will allow you to hide fields and prevent them from being exported.

If you want to export or print just the data and not the column titles, click on the Styles Page and turn off "**Column Titles**" in the Include list. Click on the check box or use the space bar to toggle the option on or off..



If you need to do a lot of customizing or filtering before exporting the data, you can create a TStringGrid and load it with the data from your table, then have PrintDAT! export the TStringGrid. The TStringGrid does not have to be visible on your form and you can even instantiate it on the fly in your export routine.

Report Alignment

Reports are normally centered on the report. To make the report left justified, select the General page and change the Alignment drop down to Left Justify or Right Justify.


Wide Reports

Grids and tables can have a lot of columns in them. The easiest way to print a wide report is to print it in landscape mode (if your printer supports it).

- 1) Go to the Output Page and set Orientation to **Landscape**.
- 2) Print the report and the output now appears on the page so the width is the widest dimension.

PrintDAT! will automatically expand the report by filling the page with more columns.

Really Wide Reports

Occasionally you'll come across a report that won't fit on the page even when using landscape mode. So what do you do? Print on legal size paper with landscape mode. To change to a different paper size, click on the Printer Setup button  (Output Destination must be set to

"Printer" for this button to appear) and change the printer to Legal Size paper. PrintDAT! will automatically adjust to the new size paper and will fill it up with more columns. There is no need to recompile your program when the paper size changes. PrintDAT! is smart enough to use as much of the paper as possible.

Really Really Wide Reports

If your report is close to fitting on the page, try shrinking it down to size. **Shrink to Page** is turned on by default and I bet you probably didn't even know it was working behind the scenes. When **Shrink To Page** is turned on (found on the Output Page) the font size of the text will be adjusted downward until the report fits across the page. Unlike other report programs, PrintDAT! won't shrink the text so small that it is unreadable. The **Smallest Font Size** field will prevent the text from being reduced to small. If the Smallest Font Size is set to 6 points the smallest font on the report will not be less than 6 points. If you have a microscope handy you can try setting Smallest Font Size to 0 which means there is no lower limit to the font size. This could come in handy for printing annual statements for shareholders, especially if you want to hide the details of the report!

Really Really Really Wide Reports

Ok, eventually you're going to find a report so wide, that the text either shrinks down to micro-dots or it requires paper so large that it won't fit into your printer. So how are you going to print it? What do you do?

- Is the report going to truncate columns and lose important data in the process like your conventional report writer?
- Or is it going to require extensive programming changes that requires a consultant to come in and occupy your desk to all hours of the night?
- Do you start splitting the file or grid up into multiple components and print one piece at a time?
- Do you put in a request for a large format printer and hope the boss doesn't catch on in time?

How are you going to solve the problem with printing really really really wide reports? The answer is "Nothing, absolutely nothing at all.". That's right, you do absolutely nothing. (That was easy wasn't it?). When there are too many columns for the report, PrintDAT! will simply move those columns to another page. The first 10 columns might fit on the first page, the next 10 columns may fit on the next page, and so on

until all of the columns are printed. After the report has printed, you simply tape or staple the pages horizontally across to form the report. This means there is virtually no physical limit to the number of columns that can be printed. PrintDAT! can print reports with over 1,000 data columns across!

To make piecing the pages of this report easier, the page numbers are numbered "y-x" where the "y" represents the usual vertical page number and "x" represents the horizontal page number across. So if the report was 3 pages wide, the pages would be numbered "1-1", "1-2", "1-3". The leading digit "1-" tells you the page row whereas the "-1", "-2", and "-3" tells you the page column. Even if all the pages were dropped on the floor and were mixed up, it would be quite easy to sort them back into the correct order. With PrintDAT! it's now possible to produce a report wide enough to wallpaper your office!

Narrow Reports

Often you will have a report that is quite narrow and won't take up much space on the page but it may be over a hundred pages long. Printing a report like this will waste a great deal of paper and flipping through a hundred pages can make it difficult to find the information you're looking for. Fortunately there is an easy solution to the problem. PrintDAT! can print several grids across the page. In other words it can print 2-across, 3-across, up to 10 grids across the page! Each of these grids is called a Panel. You can think of a panel as a rectangle that can hold one grid. The number of panels that can fit across the page depends on the width of the grid, font size, and the width of the page.

3 Panels Across Reports

Drop a TPdtPrintDAT component on a form with a narrow grid that has one or two columns in it. Run the program and go to the Report Options window, select the General Page and under Formatting Options you'll find the field Panels Across Page. Enter a 3 for this field and the report will print 2 panels across. As soon as a new value is entered into the Panels Across Page field, the report width on the status bar will change. If you enter too many panels, PrintDAT! will reduce the # of panels so they will fit across the page. Switching to legal size paper, landscape mode, and using Shrink to Page will allow more panels to fit across the page.

CITIES OF THE WORLD

<p>Aberdeen, Scotland Adelaide, Australia Algiers, Algeria Amsterdam, Netherlands Ankara, Turkey Asunciun, Paraguay Athens, Greece Budapest, Hungary Buenos Aires, Argentina Cairo, Egypt Calcutta, India Canton, China Caracas, Venezuela Cayenne, French Guiana Chihuahua, Mexico Chongqing, China Copenhagen, Denmark Cordoba, Argentina Dakar, Senegal Darwin, Australia Djibouti, Djibouti Dublin, Ireland</p>	<p>Durban, South Africa Edinburgh, Scotland Frankfurt, Germany Georgetown, Guyana Glasgow, Scotland Guatemala City, Guatemala Guayaquil, Ecuador Madrid, Spain Manchester, England Manila, Philippines Marseilles, France Mazatlán, Mexico Mecca, Saudi Arabia Milan, Italy Montevideo, Uruguay Moscow, Russia Munich, Germany Nagasaki, Japan Nagoya, Japan Nairobi, Kenya Nanjing (Nanking), China Naples, Italy</p>	<p>Newcastle-on-Tyne, England Odessa, Ukraine Osaka, Japan Oslo, Norway Panama City, Panama Paramaribo, Suriname Paris, Sofia, Bulgaria Stockholm, Sweden Sydney, Australia Tananarive, Madagascar Teheran, Iran Venice, Italy Veracruz, Mexico Vienna, Austria Vladivostok, Russia Warsaw, Poland Wellington, New Zealand Zurich, Switzerland</p>
---	---	---

Illustration 5: 3 Panels Across Report

Reports For Overhead Projectors

We've all attended presentations where overhead transparencies were pretty much unreadable because the text in the table was all printed using a 10 point font. That's why we created the Expand To Page option. Click on the Output page and under Output Options select **Expand To Page**. When this is turned on, a narrow report will be expanded to the full size of the page so it's easier to read. To prevent the fonts sizes from getting too large, set **Largest Font** to around 20 points. None of the fonts on the page will be larger than this font size. You can also increase the page margins to prevent the report from printing near the edge of the page.

There is a table waiting for us at the next lesson.

Lesson #3 - Creating TTable and TQuery Reports

Printing a TTable or TQuery component is no different than a TdbGrid. You will need to check the ObjectToPrint property in the Object Inspector to make sure it is indeed pointing to the TTable or TQuery component, because if the form also has a grid on it, it will automatically set the ObjectToPrint to the grid instead of the dataset component. That's because by default PrintDAT! chooses grids over datasets when it chooses what to print..

Printing a TTable or TQuery with PrintDAT! can give you a powerful reporting tool. The TTable.DatabaseName and TTable.TableName properties can be specified at runtime or the TQuery component can be used to specify the columns and sort them on the fly. Until now there was no easy way to print these dynamic datasets. Lucky for us we have PrintDAT! to make quick work out of any table or query that we care to throw at it.

The demo program in \PrintDAT\Demo demonstrates the power behind printing any table from any database. Look at Major.Pas and DynGrid.Pas to view the actual code.

#	Population	Name	Capital	Continent	Area
1	249200000	United States of America	Washington	North America	9363130
2	150400000	Brazil	Brasilia	South America	8511196
3	88600000	Mexico	Mexico City	North America	1967180
4	33000000	Colombia	Bagota	South America	1138907
5	32300003	Argentina	Buenos Aires	South America	2777815
6	26500000	Canada	Ottawa	North America	9976147
7	21600000	Peru	Lima	South America	1285215
8	19700000	Venezuela	Caracas	South America	912047
9	13200000	Chile	Santiago	South America	756943
10	10600000	Ecuador	Quito	South America	455502
11	10600000	Cuba	Havana	North America	114524
12	7300000	Bolivia	La Paz	South America	1098575
13	5300000	El Salvador	San Salvador	North America	20865
14	4660000	Paraguay	Asuncion	South America	406576
15	3900000	Nicaragua	Managua	North America	139000
16	3002000	Uruguay	Montevideo	South America	176140
17	2500000	Jamaica	Kingston	North America	11424
18	800000	Guyana	Georgetown	South America	214969

Illustration 6: TQuery report example used SQL to sort the countries by population in descending order. The PrintDAT! line# column ranks them.

You'll notice the Name "United States of America" prints on two lines. This is because the report options had specified the maximum string column width to be 20 characters. If the text exceeds this length it will automatically be wordwrapped onto a second line. This prevents large text or memo values from occupying too much of the report width. We can also specify the maximum number of lines to use when wordwrapping text.

Any Table, Anytime

Delphi and C++Builder are terrific because they will access any information from any table regardless of whether the data resides in Paradox, dBase, Oracle, Interbase, MS SQL Server etc. without any code changes! PrintDAT! now allows you to print or export this information so it is no longer locked onto your computer screen. You will be able to share the information with others quite easily without writing dozens of reports. In fact, a single procedure can do it all!

The next lesson will string us along

Lesson #4 – Printing TStringGrid

The TStringGrid component is often overlooked by developers because it doesn't access a TDataSet. But with a little code it can display data in a format that is not easily reproducible using a TdbGrid. A TStringGrid can display something as simple as a calendar, or it can show summarized values read from several datasets and formatted with labels and group totals.

Memory Grids & TrimGridDim

The TStringGrid, TListBox, and TListView are called memory grids because all of the data it contains resides in memory. A TdbGrid is not a memory grid because it uses a TDataSet to access its data.

The TReportSettings.IncludeOpts. has an option that applies only to memory grids. To prevent the blank columns on the right side of the grid and the blank rows at the bottom of the grid from printing, turn on the **Trim Grid Dim** option (see Include Options on the Styles Page). This will help to eliminate empty cells from the report. Note: This has no effect on empty columns on the left side of the grid or empty rows at the top of the grid.

Override Field Alignment

Since the cells in a TStringGrid are treated as strings, the grid left justifies the data. But columns may contain numbers and left justified numbers makes for an ugly report because the decimal places do not line up. Lucky for us PrintDAT! can align-on-the-fly the cells using its OnFieldAlign event. Here's an example of how it works. Select a TpdtPrintDAT component and on the events page of the Object Inspector, double click on the OnFieldAlign event and enter the following code.

```
procedure TfmMajorFeatures.pdtStringGridFieldAlign(Sender: TObject;
  var FldStr: String; var AlignField: Boolean; var Alignment: TAlignment;
  var DataRect: TRect; MaxRect: TRect; FieldInfo: TpdtFieldInfo);
begin
    {(1) Is this a data row? And not a column title?}
    if FieldInfo.RptLineType = rltCellData then
    {(2) Is this a data column?}
    if FieldInfo.GridColNum >= StringGrid1.FixedCols then
    {(And not a row title?)}
    Alignment := taRightjustify;    {(3) Then it is a number so right justify it}
end;
```



This event works on any type of grid, including TdbGrid. But since PrintDAT! already justifies dbGrid numbers correctly using the TField.Alignment property, you may only need to use this

OnFieldAlign event to override the default alignments. For more detailed information see the TpdtdPrintDAT.OnFieldAlign event in the PD_Dev.Hlp file.

Smart Field Alignment for Memory Grids

When printing columns from a TdbGrid, PrintDAT! will use the underlying TField.Alignment property to determine how to align the columns.

The OnFieldAlign event is only used to override the default field alignment for an individual field. We need to do this occasionally because the columns of a memory grid like a TStringGrid do not have a TField.Alignment property so PrintDAT! can't properly determine how to align the column data. Or can it?

Actually when printing a memory grid, PrintDAT! will guess at the alignment by examining the data in the column to see if it is numeric. If all of the cells in a column are numeric then it will right justify the column. If any of the cells in the column contains a non-numeric string like "abc", then the column data is left justified. Here is a report example. Keep in mind that PrintDAT! did the alignments automatically without us having to flip any switches or write any code.

TStringGrid Report

#	ColTitle0	ColTitle1	ColTitle2	ColTitle3	ColTitle4
1	RowTitle1	2	1.02	1.03	\$1.04
2	RowTitle2	3	2.02	2.03	\$2.04
3	RowTitle3	4	3.02	3.03	\$3.04
4	RowTitle4	5	4.02	4.03	\$4.04
5	RowTitle5	6	5.02	5.03	\$5.04
6	RowTitle6	7	6.02	6.03	\$6.04
7	RowTitle7	8	7.02	7.03	\$7.04
8	RowTitle8	9	8.02	8.03	\$8.04
9	RowTitle9	10	9.02	9.03	\$9.04
10	RowTitle10	11	10.02	10.03	\$10.04
11	RowTitle11	12	11.02	11.03	\$11.04

Illustration 7: TStringGrid Report

PrintDAT! is smart enough to right justify numeric columns and left justify non-numeric columns, even in a TStringGrid

So our earlier example with the OnFieldEdit event to right align numeric columns in a TStringGrid isn't really necessary after all. PrintDAT! will do it for us automatically.

Fixed Columns

A TStringGrid component has a FixedCols property that prevents the first 'FixedCols' columns from scrolling on the screen. PrintDAT! will set ReportSettings.NumFixedColumns to this same value so these columns will appear on every horizontal report page. This value can be changed on the General Page. PrintDAT! can also assign NumFixedColumns when printing other grids like a TDbGrid. For example. This allows you to fix the customer name to the left side of all horizontal report pages.



In the rare case that you are using a non-BDE table that is not descended from a TDataset then PrintDAT! can't print it (unless you change the PrintDAT! source code). But to get around this problem, you can build a TStringGrid and load it with records from the data file and then print the TStringGrid instead.

We can really count on the next lesson.

Lesson #5 – Column Totals

Quite often you may want to add a total to a report column. This can be done easily in PrintDAT! without writing one line of code and without entering any formulas. The column totals are defined on the Styles page of the Report Options window.

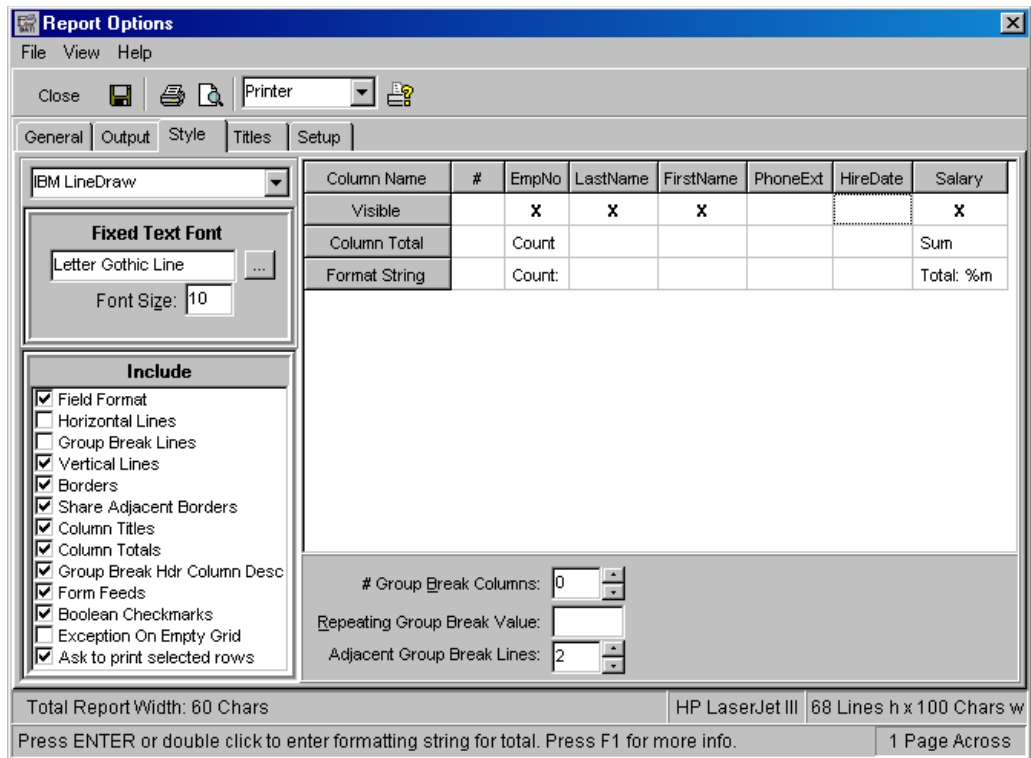
PrintDAT! can perform several different types of column totals:

- Count Records
- Sum
- Minimum Value
- Maximum Value
- Average
- Standard Deviation (Population & Sample)
- Variance (Population & Sample)

If you have the PrintDAT! source code, you can add new types of column totals.

Adding A Column Total

To add a column total, select the Style Page and find the column you wish to total and select the type of column total. That's all you need to do. You may optionally add a Format String to format the number with the proper number of decimal places or you may add other text in the Format String to describe the total. (See below)



Column Totals can have a formatting string

Column totals can be added to virtually any grid column, including string columns that have numbers embedded in them. This means even TListBox and TStringGrid can have column totals.



When a formatting string is added to the column total, the "Total Report Width" and "x Pages Across" on the status bar are automatically updated. This will alert you whenever the total widths have widened the columns so they no longer fit onto one page.



If the column total doesn't appear in the report, make sure the "Column Totals" check box is selected in the Include list on the Style page.

EmpNo	LastName	FirstName	Salary
2	Nelson	Roberto	40000
4	Young	Bruce	55500
5	Lambert	Kim	25000
8	Johnson	Leslie	25050
9	Forest	Phil	25050
11	Weston	K. J.	33292.9375
12	Lee	Terri	45332
14	Hall	Stewart	34482.625
15	Young	Katherine	24400
20	Papadopoulos	Chris	25050
24	Fisher	Pete	23040
28	Bennet	Ann	34482.8
29	De Souza	Roger	25500
34	Baldwin	Janet	23300
36	Reeves	Roger	33620
37	Stansbury	Willie	39224
44	Phong	Leslie	40350
45	Ramanathan	Ashok	33292.94
46	Steadman	Walter	19599
52	Nordstrom	Carol	4500
61	Leung	Luke	34500
65	O'Brien	Sue Anne	31275
71	Burbank	Jennifer M.	45332
72	Sutherland	Claudia	35699
83	Bishop	Dana	45000
85	MacDonald	Mary S.	35699
94	Williams	Randy	28900
105	Bender	Oliver H.	36799
107	Cook	Kevin	35500
109	Brown	Kelly	27000
110	Ichida	Yuki	25689
113	Page	Mary	48000
114	Parker	Bill	35000
118	Yamamoto	Takashi	32500
121	Ferrari	Roberto	40500
127	Yanowski	Michael	44000
134	Glon	Jacques	24855
136	Johnson	Scott	30588.99
138	Green	T.J.	36000
141	Osborne	Pierre	35600
144	Montgomery	John	35699
145	Guckenheimer	Mark	32000
Count: 42			Total: \$1,386,202.29

Illustration 8: Report with column totals

Text Non-Total

The dropdown menu has an entry for "Text". The Text Total is not a total at all but a means of adding text below a column without producing a total. If the text is too wide for the column width (determined by Max String Column Width), then the text will wrap onto 2 or more lines. To keep the column narrow, you can also use a vertical bar to indicate a line break.

Example: "Inventory Levels | as of 9/1/99" will produce a column total with the text:

Inventory Levels
as of 9/1/99

Getting Numeric Field Value From The Grid

Column totals require numeric field values and there are two ways to get them from the grid.

1. If the grid has an underlying TDataSet, as is the case with a TdbGrid, then PrintDAT! will use the TField AsFloat to retrieve the field value. This is faster than trying to convert a string value into a numeric and is more reliable since there shouldn't be any conversion errors to worry about.
2. If the grid does not have an underlying TDataSet (as with a TStringGrid) then the cell values from the grid are string and PrintDAT! will convert the string cell to a numeric value. In doing so it will automatically remove non-numeric characters such as letters and symbols from the cell string before the string is converted to a number. Example: A cell that has the value "Bob Smith \$12,321.22" will be converted to 12321.22 automatically without creating an exception during the conversion process. The DecimalSeparator variable is used to find the decimal place for internationally formatted numbers. See the Delphi help for more information on the DecimalSeparator constant.

Numeric Conversion Exception Handling

Occasionally a grid cell may have an invalid number such as "123.456.78" which has 2 decimal separators. If PrintDAT! tries to convert this number it will produce an EConvertError exception. PrintDAT! will then call your TpdtPrintDAT.OnCnvtFldError event if you have this event defined. If you do not have this event defined, or if the code in this event does not successfully convert this number, then this number will be ignored in the total. It will be as if this cell never existed. The number of rows that went into making the total will not count this cell. So totals that use the row count, such as averages and standard deviations, will still be correct because neither the cell value or its count will be included in the total. In other words, instead of using 'n' rows, the total will be based on 'n-1' rows.

Do your own field conversion

Occasionally you may have a grid column that has a number embedded in a string or memo field and this number needs to be totaled. How can PrintDAT! extract the number and use it in the column total? Example: The field "07/28/1999 \$4,123.40" needs to have the "\$4,123.40" totaled but the date needs to be ignored. To solve this problem you can use the OnFieldEdit event to substring the dollar amount from the string. You then have the option of letting PrintDAT! convert the new string "\$4,123.40" or you can convert it yourself.

You can also selectively asterisk out the salaries of the user's superiors while still printing the salaries of his assistants. (To prevent the user from see the salaries in the grid, either hide the grid or print the report directly from a TTable or TQuery component. You can of course also add a calculated column to the dataset and display the salary there, and have the calculated column event asterisk out the field if the user does not have the privilege to see it.) See the TpdtPrintDAT.OnFieldEdit event in the help file for more information.

Converting the field yourself leads to another advantage. Let's say the column you're totaling contains non-numeric string values like "United States", "Greenland", "Canada". You can intercept the country string, do a lookup into another table to get a number like the country's population, or perform some internal calculation on the country string to produce a number, and then use that number in the column total. Or you can reference other fields in the record in the calculation of this column total. All this can be done in the TpdtPrintDAT.OnFieldEdit event.

Numerical Accuracy

The calculations for the totals are all done using Extended data type which provides for 19-20 significant digits with a numeric range of 3.6×10^{-4951} .. 1.1×10^{4932} . This is more than sufficient for any scientific or monetary calculations. Not even Bill Gates has $\$10^{4932}$ in his bank account! Well at least not yet. ☺

Formatting

Column totals can be unformatted which means it can appear as "452132.41223" or you can use a Delphi Format string on the total to produce a more readable "\$452,132.41". All totals are real numbers so floating point format types are allowed (see table below). Example: "%F", "%6.2F", "Average: %6.2F" are all valid format strings. You cannot use "%6d" because "d" is for formatting integer values and the total is a real number. If you wish to remove the decimal places from the total then use a format like "%6.0F" or "%.0F". See the Delphi help file under "Format Strings" for more information on how the number can be formatted with %F.

Formatting String	Description	Example	Output	Hint
	No formatting symbol will format the number with all decimal places.		123456.2432	Leave the formatting string empty if the appearance of the number is unimportant.
%f	Fixed. If no decimal specifier is used, then all decimal digits is used.	%5.2F %.0	123456.24 123456	%F Does not use commas. Use %.0 to display the number as an integer (eliminates the decimals).
%e	Scientific. Uses an exponent so the number is formatted as "-d.ddd...E+ddd".	%5.3e	1.23E+005	Good for scientific calculations where there are very large or very small numbers.
%g	Will format the number with either %f or %e so it occupies the smallest width.	%g	123456.2432	Use this format if you want the number to occupy the smallest amount of space
%n	Similar to %f except it uses thousands separates like "-d,ddd,ddd.ddd..."	%5n	123,456.24	The most readable format for large numbers but the commas will widen the column slightly.
%m	Currency. Formats the number with currency format.	%m	\$123,456.24	Should be used on all currency columns.

Table 1: Column totals can use any formatting string that can be used with a real number

Two Line Totals

If the column is narrow and you want to add a description to the total as in "Average: 123.456" without expanding the column width, you can have it appear on 2 lines by inserting a vertical bar "|" to indicate a line break. Example: "Average|%7.3F" produces the following column total:

```
Average
1233.456
```



You can add spaces before the "|" to move the text "Average" to the left.

Preventing Column Totals From Printing

To prevent a single column total from printing, set the Column Total Type to "None". To prevent all the column totals from printing, disable the "Column Totals" option found in the Include Options listbox that is on the Styles Page. You can then re-activate the column totals at a later date.

Column Totals That Wrap

Column totals, like any other String field, will wrap if it is wider than the maximum column string width which defaults to around 20 characters. See `TpdtReportSettings.MaxStringColWid`. So if you're using a large format string like "The Invoice Sum is: %12.2M" it may print on 2 lines. This is still quite readable and wrapping the column is the only way to prevent column bloat.

To prevent the total from wrapping you can:

- shorten the text in front of the total,
- reduce the format size from 12 to 10 for example, or
- increase the `MaxStringColWid`.

If the total does wrap, make sure you have `MaxStringLines >=2` otherwise the total will not get printed if it has to appear on more than 1 line. You'll just be left with a 1 line total that says "The Invoice Sum Is:" with nothing underneath it because you have `MaxStringLines` set to 1 which prevents wrapping.

Invalid Totals

Some totals such as Variance (Sample) and Standard Deviation (Sample) will divide by n-1 where n is the number of fields from the grid that went into making up the total. If n is one or less, then the total cannot be calculated and the column total will have "n/a" which stands for "not applicable".

Blank Fields

If a blank string field is retrieved from the grid, it will be recognized as a zero. If you wish to ignore blank fields, then use the OnFieldEdit event with code similar to this:

```
if FieldInfo.IsTotalColumn and (FldStr='') then
    FieldInfo.IgnoreValueForTotal := true;
```

Standard Deviation & Variance: Population vs Sample

Standard deviation is the measurement of how widely the data is scattered about the mean (average). The variance is simply the square of the standard deviation.

Some of you may be wondering what the difference is between the two types of variances and standard deviations. One is called "Population" and the other is called "Sample".

Standard Deviation (Sample)

$$\sqrt{\frac{n \sum x^2 - (\sum x)^2}{n(n-1)}}$$

Use this calculation when the data is from a sample of the population. That is to say, you are not dealing with the entire set of data but only a subset of it. This uses an "n-1" or an "unbiased" method to determine the standard deviation.

If on the other hand the data represents the entire population then use the Population calculation shown below.

Standard Deviation (Population)

$$\sqrt{\frac{n \sum x^2 - (\sum x)^2}{n^2}}$$

Use this calculation if the data represents the entire population.

For more information on Standard Deviation and Variance, consult an introductory book on statistics or take your local statistician to lunch.

The next lesson will do the impossible

Lesson # 6

The ever elusive TDecisionGrid Report

The TDecisionGrid component that appears in the Client Server versions of Delphi is a remarkable tool for data mining. It can summarize raw data into crosstab reports and manipulate the dimensions to produce various subtotal breaks. If you haven't seen a TDecisionGrid, here's what one looks like on the screen:

		+	ShipVIA						
PaymentMethc	Terms		DHL	Emery	FedEx	UPS	US Mail	dEx	Sum
AmEx	FDB					\$68,059.45			\$68,059.45
	Net 30		\$7,513.80		\$31,982.45	\$17,544.70	\$9,653.00		\$66,693.95
	Sum		\$7,513.80		\$31,982.45	\$85,604.15	\$9,653.00		\$134,753.40
COD	Net 30					\$12,396.65	\$21,379.90		\$33,776.55
	Sum					\$12,396.65	\$21,379.90		\$33,776.55
Cash	FDB				\$19,022.00		\$2,150.00		\$21,172.00
	Net 30				\$124,000.70		\$18,830.95		\$142,831.65
	Sum				\$143,022.70		\$20,980.95		\$164,003.65
Check	FDB		\$150,738.70		\$15,818.80	\$25,207.95	\$20,139.70		\$211,905.15
	Net 30					\$58,587.00			\$58,587.00
	Sum		\$150,738.70		\$15,818.80	\$83,794.95	\$20,139.70		\$270,492.15
Credit	FDB		\$126,287.15		\$31,372.80	\$255,949.35	\$373,286.85		\$786,896.15
	Net 30		\$67,746.65	\$3,249.85	\$148,097.30	\$303,910.75	\$22,529.55		\$545,534.10
	Sum		\$194,033.80	\$3,249.85	\$179,470.10	\$559,860.10	\$395,816.40		\$1,332,430.25
MC	FDB			\$33,664.80	\$63,228.25	\$54,024.15			\$150,917.20
	Net 30		\$13,406.35			\$38,391.70	\$47,448.00		\$99,246.05
	Sum		\$13,406.35	\$33,664.80	\$63,228.25	\$92,415.85	\$47,448.00		\$250,163.25
Visa	FDB		\$10,152.00	\$12,169.60	\$16,852.00	\$85,588.20	\$96,086.25		\$220,848.05
	Net 30		\$55,609.00		\$85,165.45	\$77,695.80	\$4,872.20		\$223,342.25
	Sum		\$65,761.00	\$12,169.60	\$102,017.45	\$163,284.00	\$100,958.25		\$444,190.30
sh	t 30Ca							\$3,650.00	\$3,650.00
	Sum							\$3,650.00	\$3,650.00
Sum			\$431,453.65	\$49,084.25	\$535,539.75	\$997,355.70	\$616,376.20	\$3,650.00	\$2,633,459.55

Sample screen shot of a TDecisionGrid component

By clicking on the plus sign that is on the **Terms** dimension, the grid will expand showing yet another dimension, called "Ship Date". Clicking on the "-" symbol will compress that dimension and the "ShipDate" detail will be summarized into the parent dimension. So as the DecisionGrid expands to show more dimensions, it displays more detail. When a dimension is compressed, you lose that detail and it produces a more global view of your data.

The dimensions are also pivot tables. By dragging one dimension onto another they will swap positions. So dragging ShipVia onto the PaymentMethod will swap these dimensions so ShipVia now appears as the left column and PaymentMethod appears across the top. The dimensions can also be moved so the ShipVia can appear between

PaymentMethod and Terms. Whenever you move dimensions the cells and totals will be recalculated. The rows and columns of the grid will expand or compress accordingly.

As you can see, DecisionGrids can be a lot of fun. They allow you to view your data from a variety of angles. A single DecisionGrid like this one can be arranged into a dozen or more different views just by swapping and moving dimensions.

Unfortunately there is no way to print a TDecisionGrid, until now!

Printing a TDecisionGrid is the same as printing any other component with PrintDAT!. Just drop the TpdPrintDAT component onto the form and add the one line of code "pdtPrintDAT1.Print;" to a button and you're done. Or double click the TpdPrintDAT component to see the report while still in the Delphi IDE.

Additional Features

You'll be happy to discover PrintDAT! can print the DecisionGrid with the same dimensions that are displayed on the screen. So as the dimensions are swapped, compressed or expanded to produce a different DecisionGrid, PrintDAT! will produce a report showing those exact changes.

PrintDAT! has an option under the Include list on the Styles page called *Group Break Lines*. This is turned on by default whenever a TDecisionGrid is printed. It will print a horizontal line between group breaks on the report to make the report more readable.

Sample Report #1

This report was produced by our demo program.

**** DecisionGrid Report ****

Courier Shipping Expenses

PaymentMethod	Terms	ShipVIA DHL	Emery	FedEx	UPS
AmEx	FOB				\$68,059.45
	Net 30	\$7,513.80		\$31,982.45	\$17,544.70
	Sum	\$7,513.80		\$31,982.45	\$85,604.15
COD	Net 30				\$12,396.65
	Sum				\$12,396.65
Cash	FOB			\$19,022.00	
	Net 30			\$127,650.70	
	Sum			\$146,672.70	
Check	FOB	\$150,738.70		\$15,818.80	\$25,207.95
	Net 30				\$58,587.00
	Sum	\$150,738.70		\$15,818.80	\$83,794.95
Credit	FOB	\$126,287.15		\$31,372.80	\$255,949.35
	Net 30	\$67,746.65	\$3,249.85	\$148,097.30	\$303,910.75
	Sum	\$194,033.80	\$3,249.85	\$179,470.10	\$559,860.10
MC	FOB		\$33,664.80	\$63,228.25	\$54,024.15
	Net 30	\$13,406.35			\$38,391.70
	Sum	\$13,406.35	\$33,664.80	\$63,228.25	\$92,415.85
Visa	FOB	\$10,152.00	\$12,169.60	\$16,852.00	\$85,588.20
	Net 30	\$55,609.00		\$85,165.45	\$77,695.80
	Sum	\$65,761.00	\$12,169.60	\$102,017.45	\$163,284.00
Sum		\$431,453.65	\$49,084.25	\$539,189.75	\$997,355.70

Sample Report #2

Courier Shipping Expenses

PaymentMethod	ShipVIA DHL	Emery	FedEx	UPS	US Mail	Sum
AmEx	\$7,513.80		\$31,982.45	\$85,604.15	\$9,653.00	\$134,753.40
COD				\$12,396.65	\$21,379.90	\$33,776.55
Cash			\$146,672.70		\$20,980.95	\$167,653.65
Check	\$150,738.70		\$15,818.80	\$83,794.95	\$20,139.70	\$270,492.15
Credit	\$194,033.80	\$3,249.85	\$179,470.10	\$559,860.10	\$395,816.40	\$1,332,430.25
MC	\$13,406.35	\$33,664.80	\$63,228.25	\$92,415.85	\$47,448.00	\$250,163.25
Visa	\$65,761.00	\$12,169.60	\$102,017.45	\$163,284.00	\$100,958.25	\$444,190.30
Sum	\$431,453.65	\$49,084.25	\$539,189.75	\$997,355.70	\$616,376.20	\$2,633,459.55

This is the same report but with the Terms dimension compressed. Just by clicking on the "+" next to "Terms" we have a different report without changing so much as one line of code.

Sample Report #3

In this report we swapped a couple of columns and expanded the ShipDate dimension. Now we have a different view of the same data. Because all the reports shown on this page is summarizing the same data (after all it is the same TDecisionGrid), the dimension subtotals will be the same. If you look at the Sum for DHL it is \$431,453.65 which is the same on all 3 reports.

The report starts on the next page and because of its length, is split across several pages.

Courier Shipping Expenses

ShipVIA	PaymentMethod	ShipDate	Terms FOB	Net 30	Sum
DHL	AnEx	1988		\$930.00	\$930.00
		1993		\$6,583.80	\$6,583.80
		Sum		\$7,513.80	\$7,513.80
	Check	1988	\$7,885.00		\$7,885.00
		1992	\$51,673.15		\$51,673.15
		1993	\$21,614.00		\$21,614.00
		1994	\$61,994.55		\$61,994.55
1995		\$7,572.00		\$7,572.00	
Sum		\$150,738.70		\$150,738.70	
Credit	1988	\$7,227.00	\$9,955.00	\$17,182.00	
	1989	\$92,224.00	\$36,794.00	\$129,018.00	
	1993	\$9,756.40	\$15,342.85	\$25,099.25	
	1994	\$17,079.75	\$5,654.80	\$22,734.55	
	Sum	\$126,287.15	\$67,746.65	\$194,033.80	
MC	1994		\$5,983.00	\$5,983.00	
	1995		\$7,423.35	\$7,423.35	
	Sum		\$13,406.35	\$13,406.35	
Visa	1988	\$10,152.00		\$10,152.00	
	1989		\$19,812.00	\$19,812.00	
	1994		\$35,797.00	\$35,797.00	
	Sum	\$10,152.00	\$55,609.00	\$65,761.00	
Sum		\$287,177.85	\$144,275.80	\$431,453.65	
Emery	Credit	1988		\$3,249.85	\$3,249.85
		Sum		\$3,249.85	\$3,249.85
	MC	1988	\$20,321.75		\$20,321.75
1994		\$8,305.95		\$8,305.95	
1995		\$5,037.10		\$5,037.10	
Sum		\$33,664.80		\$33,664.80	
Visa	1988	\$343.80		\$343.80	
	1992	\$3,531.80		\$3,531.80	
	1993	\$8,294.00		\$8,294.00	
	Sum	\$12,169.60		\$12,169.60	

	Sum		\$45,834.40	\$3,249.85	\$49,084.25
FedEx	AmEx	1989 Sum		\$31,982.45 \$31,982.45	\$31,982.45 \$31,982.45
	Cash	1988	\$16,603.00		\$16,603.00
		1989		\$114,205.80	\$114,205.80
		1993	\$2,419.00		\$2,419.00
		1994		\$13,444.90	\$13,444.90
		Sum	\$19,022.00	\$127,650.70	\$146,672.70
	Check	1988	\$766.80		\$766.80
		1994	\$15,052.00		\$15,052.00
		Sum	\$15,818.80		\$15,818.80
Credit	1988	\$14,247.80		\$14,247.80	
	1989		\$41,763.00	\$41,763.00	
	1993	\$12,895.20		\$12,895.20	
	1994	\$4,229.80	\$106,334.30	\$110,564.10	
	Sum	\$31,372.80	\$148,097.30	\$179,470.10	
MC	1988	\$20,667.60		\$20,667.60	
	1989	\$31,847.00		\$31,847.00	
	1993	\$7,304.80		\$7,304.80	
	1994	\$104.00		\$104.00	
	1995	\$3,304.85		\$3,304.85	
	Sum	\$63,228.25		\$63,228.25	
Visa	1988	\$7,346.00		\$7,346.00	
	1989	\$4,495.00	\$77,178.55	\$81,673.55	
	1994	\$5,011.00		\$5,011.00	
	1995		\$7,986.90	\$7,986.90	
	Sum	\$16,852.00	\$85,165.45	\$102,017.45	
Sum		\$146,293.85	\$392,895.90	\$539,189.75	
UPS	AmEx	1988	\$6,287.85	\$3,891.70	\$10,179.55
		1993	\$0.00	\$13,653.00	\$13,653.00
		1994	\$52,832.00		\$52,832.00
		1995	\$8,939.60		\$8,939.60
		Sum	\$68,059.45	\$17,544.70	\$85,604.15
	COD	1988		\$7,675.85	\$7,675.85
		1992		\$4,720.80	\$4,720.80
		Sum		\$12,396.65	\$12,396.65

	Check	1988	\$18,532.00	\$11,571.00	\$30,103.00
		1993	\$6,675.95	\$47,016.00	\$53,691.95
		Sum	\$25,207.95	\$58,587.00	\$83,794.95
	Credit	1988	\$64,610.00	\$31,439.65	\$96,049.65
		1989	\$36,535.45	\$35,015.00	\$71,550.45
		1992	\$42,641.70		\$42,641.70
		1993		\$45,674.00	\$45,674.00
1994		\$34,947.95	\$167,977.10	\$202,925.05	
1995		\$77,214.25	\$23,805.00	\$101,019.25	
Sum		\$255,949.35	\$303,910.75	\$559,860.10	
MC	1988	\$1,809.85		\$1,809.85	
	1989	\$8,560.00	\$19,293.70	\$27,853.70	
	1994	\$43,654.30	\$19,098.00	\$62,752.30	
	Sum	\$54,024.15	\$38,391.70	\$92,415.85	
Visa	1988	\$4,807.00	\$25,210.00	\$30,017.00	
	1989	\$35,595.00	\$14,188.00	\$49,783.00	
	1992	\$31,219.95		\$31,219.95	
	1993		\$25,071.00	\$25,071.00	
	1994	\$10,901.25	\$13,226.80	\$24,128.05	
	1995	\$3,065.00		\$3,065.00	
	Sum	\$85,588.20	\$77,695.80	\$163,284.00	
Sum		\$488,829.10	\$508,526.60	\$997,355.70	
US Mail	AmEx	1988		\$7,807.00	\$7,807.00
		1993		\$1,846.00	\$1,846.00
		Sum		\$9,653.00	\$9,653.00
	COD	1992		\$203.00	\$203.00
		1994		\$20,711.90	\$20,711.90
1995			\$465.00	\$465.00	
Sum			\$21,379.90	\$21,379.90	
Cash	1989	\$2,150.00	\$18,830.95	\$20,980.95	
	Sum	\$2,150.00	\$18,830.95	\$20,980.95	
Check	1989	\$15,355.00		\$15,355.00	
	1994	\$4,784.70		\$4,784.70	
	Sum	\$20,139.70		\$20,139.70	
Credit	1989	\$106,416.60	\$12,736.00	\$119,152.60	
	1994	\$266,870.25	\$9,793.55	\$276,663.80	
	Sum	\$373,286.85	\$22,529.55	\$395,816.40	

MC	1989		\$21,609.00	\$21,609.00
	1994		\$7,922.00	\$7,922.00
	1995		\$17,917.00	\$17,917.00
	Sum		\$47,448.00	\$47,448.00
Visa	1988	\$7,036.80	\$3,632.00	\$10,668.80
	1992	\$28,389.00		\$28,389.00
	1993	\$47,710.75	\$1,240.00	\$48,950.75
	1994	\$12,949.70		\$12,949.70
	Sum	\$96,086.25	\$4,872.00	\$100,958.25
Sum		\$491,662.80	\$124,713.40	\$616,376.20
Sum		\$1,459,798.00	\$1,173,661.55	\$2,633,459.55

Illustration 9: Fully Expanded DecisionGrid Report

It's dimensions are amazing

What's really amazing about these reports is that they were all produced from the same TDecisionGrid object. This one form can produce a dozen or more different crosstab reports by re-arranging the dimensions and by changing the calculations, and the same PrintDAT! component can print them all, without so much as changing one line of code! It is completely automatic. PrintDAT! automatically senses the dimensions in the TDecisionGrid and prints the grid accordingly.

PrintDAT! doesn't stop at the right margin like so many other report writers. The grid can expand horizontally up to 1,000 data columns (100 pages) across by using horizontal page breaks. So it doesn't matter how large the grid is you're trying to print. PrintDAT! will handle it and it does it at runtime without changing one line of code or any of the report options. It's automatic. (This is true of any grid or dataset that PrintDAT! can print, not just the TDecisionGrid component.)

The next lesson has something to hide.

Lesson #7 - Automatic Printing

Occasionally you may want the report to print right away without displaying the Report Options window. Then when the report has finished printing it should return control to the calling procedure. To accomplish this, do the following:

- 1) Run the report and display the Setup page in Report Options window
- 2) Set the following properties:

Property	Value
Display Report Options Window	False
Display Progress Window	True (it is ok to turn if off for small reports)
Pause After Printing	False
Preview First	False

Automatic Printing

- 3) Set the Output Destination to Printer or Text file.
If outputting to a Text file, give it the name of the output file.
- 4) Save the report options and exit the Report Options window.
- 5) Run the report again and now it will print directly to the output device without pausing. When it has finished printing control will return to the calling procedure.



Ok, by now some of you may have a sinking feeling like the last time you locked the keys in your car. *Click...oops! "Ahhh... didn't we just lock ourselves out of the Report Options window?"* Well, yes we did. But fortunately there is another way in.

To get back into the Report Options window when the window is being bypassed, simply unravel a coat hanger and insert it between the window and the title bar and . oops, I mean press and hold the **Shift** and **Ctrl** keys when you click on the button on your form to start the report. Keep holding the Shift and Ctrl keys

until the Report Option window reappears. this may take 5 seconds or so.

The next lesson will put the report options under program control.

Lesson #8 - Setting Report Options from your application



This topic is definitely for the techies out there. If you're satisfied with changing report options using our Report Options window then you can skip this lesson. Just keep in mind that virtually all of the options that appears on the Report Options window can be changed under program control using a TpdPrintDAT event.

99.9% of the time the report options will be changed and saved using the Report Options window. Of course times when you may need to change the report options under program control, which is why we've created the TpdPrintDAT.AfterOptionsLoaded event. This event is executed after PrintDAT! reads the report options from its report settings file. Inside the AfterOptionsLoaded event you will have access to all of the Report Option settings for this report, and you may change any of these settings (except for a few ReadOnly properties like the ReportId).

AfterOptionsLoaded Event - Example #1

When you double click on the AfterOptionsLoaded event you'll see the event handler:

```
procedure TfmSimpleReport.PdtPrintDAT1AfterOptionsLoaded
(   Sender      : TObject;
    var ReportSettings: TpdReportSettings);
begin
end;
```

This event is executed after the report options have been read from the report settings file. The second parameter is TpdReportSettings is a class that contains the properties that were read from the report settings file and these properties can now be modified.

The help topic TpdReportSettings class in PD_Dev.Chm lists all of the properties that can be changed. This manual will provide you with a quick demonstration and will mention some of the more common properties that can be changed. For more information please consult the

PD_Dev.chm or PD_Dev.hlp files found in the d:\PrintDAT\Doc directory.

Let's start off with something easy. let's say you want to turn the report on its side so it prints in landscape mode so we can fit more columns across the page.

```
procedure TfmSimpleReport.PdtPrintDAT1AfterOptionsLoaded
(   Sender: TObject;
    var ReportSettings: TpdtrReportSettings);
begin
    ReportSettings.OutputDest := odPrinter;
    ReportSettings.Orientation := poLandscapeOrient;
end;
```

This code will ensure the output is going to the printer and the report is printed in landscape mode. If the property DisplayReportOptionsWin is set to True, the user can still override the Orientation property using the Report Options window, otherwise the report goes directly to the printer (or to the report viewer if PreviewFirst is True). See "Lesson #7 - Automatic Printing" on page 53 for more information.

Example#2 - Using a String for Page Title

Let's say for example your program runs a query and you want the page title to describe the query. The page title might look like this:

"Parts Inventory Report
for the Midwest Branch"

You could display the Report Options window, select the Titles Page and change the title manually, but it can also be done under program control as the examples will illustrate.

```
procedure TForm1.PdtPrintDAT1AfterOptionsLoadedEvent
(Sender: TObject; var ReportSettings: TpdtrReportSettings);
{-----}
{ This example will take a String variable and assigns it to the Page Title. }
{ The Page Title will be centered on the page. }
{-----}
var
    NewTitle: String;
    Lng      : Integer;

begin
    ReportSettings.RptMemo[rmPgTi].MemoBuf := 'Parts Inventory Report'+
        #13#10+'for the Midwest Branch'+#13#10#13#10;
end;
```



All titles, headers, and footers for the report are stored as memos and are accessed using the `TpdtReportSettings.RptMemo` array. The index into this array determines which title, header or footer to access. Memo Locations

To add a new line to the report memo, use `#13#10`. The linefeed character `#10` is needed to display the memo properly on the Titles Page.

Explanation

The parameter "ReportSettings" is a class whose properties contain all of the report options that appear on the Report Options window, including the report memos. When `TpdtPrintDAT.Print` is executed, it allocates memory for all of its internal classes and buffers. The `Print` method will then use the `TpdtPrintDAT.ReportId` property to read the report options from the report settings file and loads them into `ReportSettings`. Then the `AfterOptionsLoaded` event gets called and receives `ReportSettings` as a variable parameter where you can override any of its stored properties.

The report memos are stored in an array called `TReportSettings.RptMemo` and each report memo is referenced by an index, in this case we're using the page title index **rmPgTi** which stands for "report memo: page title".

Report Memo Indexes that can be used are:

TpdtRptMemoNdx	Description
rmRptTi	Report Title
rmRptHdr	Report Header
rmPgTi	Page Title
rmPgHdr	Page Header
rmPgFtr	Page Footer
rmPgNum	Page Number
rmRptFtr	Report Footer
rmCopyright	Copyright (bottom of page)

Table 2: These indexes allow access to the various report memos under program control.

See the PD_Dev.chm file for more information on TpdtReportSettings.RptMemo.

If report options are modified in the AfterOptionsLoaded event, they will be used by the report when the report prints. Otherwise the report will use the previously stored values.

After this event is executed, the report options window will be displayed, or the report will be printed depending on the options in the Setup page. See *“Hide the Report Options Window”* on page 53. The report settings do not need to be saved in order for the report to use them.

Saving Report Options to File

If you need to make these changes permanent, execute the TpdtPrintDAT.SaveReportSettings method from within the AfterOptionsLoaded event. This will save the ReportSettings back to its report settings file. The SaveReportSettings method cannot be executed outside this AfterOptionsLoaded event because ReportSettings class has not been allocated or the property values loaded prior to running the Print method.

Example#2 - Using a Memo for the Report Header

You may find a String variable too awkward when entering a lot of text for a report title or a report header. Since PrintDAT! titles are

wordwrapped "memos", it makes sense to ask the user to fill in a TMemo component on your form that describes the report, then use that memo as one of PrintDAT!'s report memos. Here's how it's done.

```

procedure TForm1.PdtPrintDAT1AfterOptionsLoadedEvent(Sender: TObject; var ReportSettings:
TpdtReportSettings);
{-----}
{ This example will take a TMemo and assigns it to the Page Title. The Page Title will be }
{ centered on the page. }
{-----}
var
  NewTitle: String;
  Lng      : Integer;
begin
  with ReportSettings.RptMemo[rmPgTi] do {Use the Page Title Memo}
  begin
    MemoBuf      := Memo1.Text;
    MemoXPosPer  := 15;           {Start the report memo 15% in from the left}
    MemoWidPer   := 70;          {The report memo will occupy 70% of the report width}
    MemoAlignment := taCenter;   {Center justify the text in the report memo}
  end;
end;

```

Explanation

This example is similar to the last example except we're using Memo1.Text to move the contents of Form1.Memo1 to MemoBuf. We're using rmRptHdr index which will add the text to the report header memo and this text will be left justified. We can put the memo (or string) into any one of the report memos and justify it however we like. See Types of Report Memos on the previous page.



When the report options window is displayed, the Report Header memo will be displaying our new memo on the Title Page. If we save the report options using the Save button on the toolbar, the new memo will be saved to the report options file.

AfterOptionsLoaded Event: Example#3 - Making Report Options Read Only & Hiding Report Options Pages

You may want to prevent certain users from saving the Report Options. Ordinarily to do this we would only have to set ReadOnly to true on the Setup page. But this would set it to ReadOnly for everyone. But in this example we want to set it to ReadOnly for everyone except for users "Master" and "Barry" who should also have write access. Here is the code to do it:

```

procedure TForm1.PdtPrintDAT1AfterOptionsLoadedEvent(Sender: TObject; var ReportSettings:
TpdtReportSettings);
{-----}
{ This example will examine the current user name string that was retrieved by Win32's }
{ GetUserName and will make the report options ReadOnly for everyone except the users }

```

```

{ "Master" and "Barry". }
{ If the user does not have write access, hide the General, Setup, and Styles pages. }
{-----}
var
  UBuf      : Array[0..25] of char;
  UBufSize: Integer;
  UName     : String;
begin
  UBufSize := SizeOf(UBuf);
  GetUserName(UBuf, UBufSize);
  UName     := UpperCase(StrPas(UBuf));
  ReportSettings.SettingsReadOnly := not( (UName = 'MASTER') or (UName = 'BARRY') );
  if ReportSettings.SettingsReadOnly then
    ReportSettings.HidePages := 'General,Setup,Styles'      {Hide these tabs if readonly}
  Else {ReadOnly access}
    ReportSettings.HidePages := '';                          {Otherwise don't hide any pages}
end;

```

There are more property examples described in the TpdReportSettings properties in PD_Dev.Hlp. The more frequently used properties are listed below.

In the PD_Dev.chm help file, see also

PreviewFirst, DisplayReportOptionsWin, OutputDest, OutputFileName, OutputQuality, DefaultPrinter, PrinterName, PageMargin, RptMemo

The next lesson will give us something to look up to.

Lesson #9 – OnFieldEdit Event

Changing cell data before it gets into the report

We can use the power of TpdPrintDAT's OnFieldEdit event to modify the data of the grid cell that was retrieved or even replace the data with something else entirely. In this lesson we will take the two letter State and Province code that appears in the dbGrid (or TDataSet) and use it to lookup the State/Province name. See the PrintDAT! demo program \PrintDAT\Demo\ColLkUp.Pas for the complete source code.

Changing a cell's data with a lookup

We need to create a TpdPrintDAT.OnFieldEdit event with the following code.

Sample code to replace a field in the grid with a lookup value

```
procedure TfmColumnTotals.PdtPrintDAT1FieldEdit( Sender : TObject;
var FldStr : String;
FieldInfo : TpdFieldInfo);
begin
with FieldInfo do
begin
Field := nil;
if FieldsIndex >= 0 then {Is this a grid field? and not the line# field}
Field := dbGrid1.DataSource.DataSet.Fields[FieldsIndex]; {Make it easier to reference}
{ the Field class}
if not Assigned(Field) then {This field is not part of }
exit; { the dataset then exit}
{What part of the grid does }
case RptLineType of { this field belong to?}
rptColTitle: ; {Column titles?}
rptCellData: {Grid data cell?}
begin { (1)When the StateProv field}
if UpperCase(Field.FieldName) = 'STATEPROV' then { is read, take the letter }
begin { abbrev. that was read }
{ into FldStr and use }
{ it to look up and retrieve the }
{ full State name. Then assign the }
{ full State name back to FldStr }
{ which puts it into the report.}
FldStr := ProvStates.Values[FldStr];
end;
end;
end;
end;
end;
```

Steps Explained

- 1) When the State column is read, FldStr contains the 2 digit State abbreviation and that is used to look up the State name which then gets assigned back to FldStr. The variable ProvStates is a TStringList that contains all of the State abbreviations and their State names. But we could just as easily performed a file lookup into a

States table to retrieve the State name, or call another procedure, or perform some sort of calculation to get our new FldStr value. By having the calculations done in an event, you have the full power of the Delphi language at your disposal.

PrintDAT! will automatically expand or narrow the column to accommodate the new value so the report always looks perfect. Plus if the column is a String or Memo and we supply it with too much text, it gets automatically wordwrapped in the cell according to the Max String Column Width. We don't need to lift a finger. PrintDAT! does all of the column formatting for us.

We can also change column titles the same way, only we would use the rltColTitle part of the case statement.

Here is a code snippet for the OnFieldEdit event that will flag customers who owe more than \$1000

```
case RptLineType of
    rltColTitle: ;           {What part of the grid does this field belong to?}
    rltCellData:           {Column titles?}
    rltCellData:           {Grid data cell?}
begin
    {Flag customer names with }
    if Assigned(Field) and { asterisks if they owe >= $1000}
        UpperCase(Field.FieldName) = 'CUSTOMERNAME' then
    if FieldInfo.DataSet.FieldByname('AmtOwing').AsCurrency >= 1000 then
        FldStr := '***'+FldStr;           {Asterisks get added to the front}
end;                               { of their name}
```

Here is the sample report with the new State/Prov name. Notice the State/Prov column width was automatically expanded and when the name exceeded our Max String Column Width, it gets wordwrapped automatically.

#	OrderNo	CustNo	State/Prov	SaleDate	PaymentMethod	ItemsTotal	Freight	AmountPaid
1	#1103	CN 1351	Arizona	4/12/88	Credit	\$1,250.00	\$0.00	\$0.00
2	#1104	CN 2156	California	4/17/88	Check	\$7,885.00	\$0.00	\$7,885.00
3	#1105	CN 1356	Colorado	4/20/88	Visa	\$4,807.00	\$0.00	\$4,807.00
4	#1106	CN 1380	Connecticut	11/6/94	Visa	\$31,987.00	\$0.00	\$0.00
5	#1107	CN 1384	District of Columbia	5/1/88	Visa	\$6,500.00	\$0.00	\$6,500.00
6	#1108	CN 1510	Delaware	5/3/88	Visa	\$1,449.50	\$0.00	\$0.00
7	#1109	CN 1513	Florida	5/11/88	COD	\$5,587.00	\$0.00	\$0.00
8	#1110	CN 1551	Georgia	5/11/88	COD	\$4,996.00	\$0.00	\$4,996.00
9	#1111	CN 1560	Hawaii	5/18/88	COD	\$2,679.85	\$0.00	\$2,679.85
10	#1112	CN 1563	Iowa	5/19/88	Credit	\$5,201.00	\$0.00	\$5,201.00
11	#1113	CN 1624	Idaho	5/25/88	Credit	\$3,115.00	\$0.00	\$3,115.00
12	#1114	CN 1645	Illinois	5/25/88	Credit	\$134.85	\$0.00	\$134.85
13	#1115	CN 1651	Indiana	5/25/88	MC	\$20,321.75	\$0.00	\$20,321.75
14	#1116	CN 1680	Kansas	6/2/88	AmEx	\$2,605.00	\$0.00	\$0.00
15	#1117	CN 1984	Kentucky	6/12/88	Check	\$10,195.00	\$0.00	\$0.00
16	#1118	CN 2118	Louisiana	6/18/88	Check	\$5,256.00	\$0.00	\$0.00
17	#1119	CN 2195	Massachusetts	6/24/88	Credit	\$20,602.00	\$0.00	\$0.00
18	#1120	CN 2156	Maryland	6/24/88	Credit	\$9,955.00	\$0.00	\$9,955.00
19	#1121	CN 2163	Maine	6/24/88	Credit	\$3,719.00	\$0.00	\$3,719.00
20	#1122	CN 2163	Michigan	6/30/88	Credit	\$10,064.65	\$0.00	\$10,064.65
21	#1123	CN 1221	Minnesota	7/1/88	Check	\$4,674.00	\$0.00	\$4,674.00
22	#1124	CN 3151	Missouri	7/2/88	Check	\$6,897.00	\$0.00	\$6,897.00
23	#1125	CN 1510	Mississippi	7/3/88	AmEx	\$930.00	\$0.00	\$930.00
24	#1126	CN 1624	Montana	7/7/88	AmEx	\$2,920.00	\$0.00	\$2,920.00
25	#1127	CN 1384	North Carolina	7/7/88	Visa	\$25,210.00	\$0.00	\$25,210.00
26	#1128	CN 1651	North Dakota	7/7/88	Visa	\$343.80	\$0.00	\$343.80
27	#1129	CN 1645	Nebraska	7/18/88	MC	\$20,108.00	\$0.00	\$20,108.00
28	#1130	CN 3615	New Hampshire	7/25/88	MC	\$559.60	\$0.00	\$559.60
29	#1131	CN 2118	New Jersey	7/28/88	Credit	\$12,685.00	\$0.00	\$12,685.00
30	#1132	CN 2163	New Mexico	7/28/88	Credit	\$775.00	\$0.00	\$775.00
31	#1133	CN 1384	Nevada	8/1/88	Cash	\$1,238.00	\$0.00	\$1,238.00
32	#1134	CN 1680	New York	8/13/88	Check	\$18,532.00	\$0.00	\$18,532.00
33	#1135	CN 1560	Ohio	8/16/88	Credit	\$560.00	\$0.00	\$560.00
34	#1136	CN 5384	Oklahoma	8/25/88	Credit	\$4,110.00	\$0.00	\$4,110.00
35	#1137	CN 1984	Oregon	8/26/88	Credit	\$3,117.00	\$0.00	\$3,117.00
36	#1138	CN 1645	Pennsylvania	8/26/88	Visa	\$10,152.00	\$0.00	\$10,152.00
37	#1139	CN 3158	Rhode Island	8/29/88	Visa	\$536.80	\$0.00	\$536.80
38	#1140	CN 6812	South Carolina	9/4/88	Visa	\$3,632.00	\$0.00	\$3,632.00
39	#1141	CN 4652	South Dakota	9/16/88	AmEx	\$7,807.00	\$0.00	\$7,807.00
40	#1142	CN 3984	Tennessee	9/24/88	AmEx	\$971.70	\$0.00	\$971.70
41	#1143	CN 5132	Texas	9/30/88	Credit	\$12,455.00	\$0.00	\$12,455.00
42	#1144	CN 5515	Utah	10/8/88	Credit	\$64,050.00	\$0.00	\$64,050.00
43	#1145	CN 9841	Virginia	10/16/88	Credit	\$787.80	\$0.00	\$787.80
44	#1146	CN 2315	Vermont	11/12/88	Check	\$766.80	\$0.00	\$766.80
45	#1147	CN 5515	Washington	11/27/88	Cash	\$15,365.00	\$0.00	\$15,365.00
46	#1148	CN 5432	Wisconsin	12/2/88	Visa	\$7,346.00	\$0.00	\$7,346.00
47	#1149	CN 9841	West Virginia	12/13/88	MC	\$1,809.85	\$0.00	\$1,809.85
48	#1150	CN 3052	Wyoming	12/24/88	AmEx	\$6,287.85	\$0.00	\$6,287.85
49	#1151	CN 2165	Alberta	1/4/89	Credit	\$325.00	\$0.00	\$325.00
50	#1152	CN 1351	British Columbia	1/6/89	Credit	\$16,788.00	\$0.00	\$16,788.00
51	#1153	CN 6516	Manitoba	1/15/89	Credit	\$24,650.00	\$0.00	\$24,650.00
52	#1154	CN 5432	New Brunswick	1/28/89	Visa	\$14,188.00	\$0.00	\$14,188.00
53	#1155	CN 1351	Newfoundland	2/4/89	Credit	\$23,406.00	\$0.00	\$23,406.00
54	#1156	CN 1513	Nova Scotia	2/8/89	MC	\$19,293.70	\$0.00	\$19,293.70
55	#1157	CN 1563	Ontario	2/18/89	Credit	\$1,975.00	\$0.00	\$1,975.00
56	#1158	CN 2118	Prince Edward Island	2/21/89	Credit	\$12,736.00	\$0.00	\$12,736.00

Instead of seeing two letter State codes that were in the dbGrid, PrintDAT! replaced them with the State name using a Lookup.



If you're using an InfoPower™ grid that has special control fields to do lookups you will want to replicate that logic using our OnFieldEdit event. We chose not to access the fields directly off of the TwwDbGrid because in order to read the TwwDbGrid's calculated fields we would

have to scroll the grid and that would be extremely slow for large reports. Their implementation is fine for displaying a couple dozen records on the screen, but not for printing thousands or even hundreds of thousands of rows. Instead we have given you a mechanism that is hundreds of times faster and more flexible than if we had tried to read the information directly off of the grid.



PrintDAT! can print calculated fields from a TTable or TDataSet component extremely fast.

Flagging certain rows of a report

It is important to note that we can assign anything we like to FldStr and it will replace the original FldStr value. It's like intercepting a value and replacing it with something else. If we don't intercept and modify the value, the original value gets put into the report. So if we wanted to flag certain values in order to get them noticed, we could prefix it with asterisks by adding "*****" to certain FldStr values. Don't forget, for dbGrids we have access to FieldInfo.DataSet so we can access any field in the current record even if the field is not being printed, and include it as part of our calculation. So if the AmtOwing field is > \$1000 we can asterisk the customer name.

Printing Column Titles onto 2 or more lines

Quite often your column title may be too wide for the report column. You can of course go back to the grid and change it there, but there is an easier way.

If the grid has a "~" or "]" character in the column title, PrintDAT! will interpret it as a carriage return and will split the text onto two (or more) lines.

If you don't want these characters to appear in the grid, we can still persuade PrintDAT! to split the column title onto 2 lines without modifying the grid by using the TpdTPrintDAT.OnFieldEdit event.

```
procedure TpdTPrintDAT1.OnFieldEdit(   Sender   : TObject;
                                       var FldStr   : String;
                                       FieldInfo: TpdTFieldInfo);
begin
  if FieldInfo.RptLineType = rltColTitle then {Is this a column title?}
  if FldStr = 'Common_Name' then
    FldStr := 'Common~Name'                {We can add a "~" to signify a line break}
  else if FldStr = 'Species No' then
    FldStr := 'Species'+#13+'Number';      {Or we can insert a Return ourselves}
end;
```

Rather than seeing “Common_Name” and “Species No” on one line of the column titles, we’ll now see:

Putting Column Titles
Onto 2 Lines

#	Species Number	Category	Common Name	Notes
1	90020	Triggerfish	Clown Triggerfish	Also known as the big spotted triggerfish.
2	90160	Spadefish	Atlantic Spadefish	Found in mid-water areas around reefs, wrecks and bridges.
3	90200	Parrotfish	Redband Parrotfish	Inhabits reef areas. The parrotfish's teeth are fused

You’ll notice the “*Species Number*” column width has decreased in size because the column title is now on two lines. The “*Common Name*” column didn’t decrease because the data underneath is still quite wide. But we can solve this too in the next section.



For the title to be wrapped onto two or more lines, the *Max # of String Lines* on the General report options page needs to be set to the maximum number of lines to wrap. In this case it needs to be set to at least two. Keep in mind this will also wrap the cell data and cell footer onto the same number of lines if their cell data exceeds the column width for the cell. Thankfully PrintDAT! will only wrap text up to this number of lines only if it needs to. If we had set *Max # of String Lines* to 5 lines and the text only requires 2 lines, then only 2 lines are printed. It won’t print 3 blank lines.

Printing Cell Data onto 2 or more lines

The ability to split text onto 2 or more lines is not limited to the column titles or column footers. The OnFieldEdit event can also be used to insert a carriage return (#13) into the cell data itself so the data appears on 2 (or more) lines.

In this example we will replace the blanks with a carriage return (#13) for the “*Common Name*” and “*Species Name*” fields to force each word onto a new line. The OnFieldEdit event looks like this:

```

procedure TpdtdPrintDAT1.OnFieldEdit(   Sender   : TObject;
                                       var FldStr  : String;
                                       FieldInfo: TpdtdFieldInfo);
begin
  if FieldInfo.RptLineType = rltColTitle then {Is this a column title?}
  if FldStr = 'Common_Name' then
    FldStr := 'Common~Name'                {We can add a "~" to signify a line break}
  else if FldStr = 'Species No' then
    FldStr := 'Species'+#13+'Number';      {Or we can insert a Return ourselves}

  if FieldInfo.RptLineType = rltCellData then {Is this cell data? And not a title/footer?}
  if Assigned(FieldInfo.Dataset) and (FieldInfo.FieldsIndex >= 0) then {Not line# col?}
  if (FieldInfo.Dataset.Fields[FieldInfo.FieldsIndex].FieldName = 'Common_Name') or
    (FieldInfo.Dataset.Fields[FieldInfo.FieldsIndex].FieldName = 'Species Name') then
    repeat
      p := pos(' ', FldStr);
      if p > 0 then
        FldStr := copy(FldStr,1,p-1) + #13 + Copy(FldStr,p+1,high(Integer));
      until p = 0;
end;

```

Putting Cell Data
Onto 2 Lines

#	Species Number	Category	Common Name	Species Name	Notes
1	90020	Triggerfish	Clown Triggerfish	Ballistoides conspicillum	Also known as the big spotted triggerfish.
2	90160	Spadefish	Atlantic Spadefish	Chaetodiperus faber	Found in mid-water areas around reefs, wrecks and bridges.
3	90200	Parrotfish	Redband Parrotfish	Sparisoma Aurofrenatum	Inhabits reef areas. The parrotfish's teeth are fused

Every word in the “*Common Name*” and “*Species Name*” column now appears on a separate line. Because the report is also using column AutoSizing, the column widths have decreased making for a more compact report.

If some of these fish had 3 or more words in their name, then it would require 3 or more wrapped lines which means *Max # of String Lines* would have to be set to at least 3. Otherwise only the first 2 words will be displayed.

The next lesson is quite selective.

Lesson #10 – Printing Selected Rows & Report Filters

Quite often you may have a grid where only a few rows need to be printed. If you have a TdbGrid the conventional way of printing the various rows would be to add a filter to the table so only those rows are displayed. PrintDAT! will then print the grid with those rows.

But creating a filter can be a lot of work if the user only wants to print a few rows and they're already visible in the grid. An easier way would be to let the user select the rows from the grid with the mouse, and then print the selected rows. Most grids have a multi-selected row option that allows the user to select one or more rows at one time.

Printing Multi-Selected Grid Rows

Select a form that has a TdbGrid and use the Object Inspector to activate the grid option **dgMultiSelect**. The grid can now have more than one row selected at a time.

Create a PrintDAT! report in the normal fashion. Drop a TpdtPrintDAT component on the form and add a TButton with a caption that says "&Print". Double click the button and add the usual line to execute the print event:

```
PdtPrintDAT1.Print;
```

Now run the program and display the form that has the TdbGrid. When the grid is displayed, hold down the Ctrl key while clicking on several of the grid rows. If the dgMultiSelect grid option is set properly, you should see several rows selected as shown below.

Species No	Common_Name	Species Name	Length (cm)	Notes
90020	Clown Triggerfish	Ballistoides conspicillum	50	(MEMO)
90120	Bat Ray	Myliobatis californica	56	(MEMO)
90130	California Moray	Gymnothorax mordax	150	(MEMO)
90140	Lingcod	Ophiodon elongatus	150	(MEMO)
90150	Cabezon	Scorpaenichthys marmoratus	99	(MEMO)
90160	Atlantic Spadefish	Chaetodiperus faber	90	(MEMO)
90170	Nurse Shark	Ginglymostoma cirratum	400	(MEMO)
90180	Spotted Eagle Ray	Aetobatus narinari	200	(MEMO)
90190	Yellowtail Snapper	Ocyurus chrysurus	75	(MEMO)
90200	Redband Parrotfish	Sparisoma Aurofrenatum	28	(MEMO)
90210	Great Barracuda	Sphyræna barracuda	150	(MEMO)
90220	French Grunt	Haemulon flavolineatum	30	(MEMO)
90230	Dog Snapper	Lutjanus jocu	90	(MEMO)
90240	Nassau Grouper	Epinephelus striatus	91	(MEMO)
90250	Bluehead Wrasse	Thalassoma bifasciatum	15	(MEMO)
90260	Yellow Jack	Gnathanodon speciosus	90	(MEMO)
90270	Redtail Surfperch	Amphistichus rhodoterus	40	(MEMO)
90280	White Sea Bass	Atractoscion nobilis	150	(MEMO)
90290	Rock Greenling	Hexagrammos lagocephalus	60	(MEMO)
90300	Senorita	Oxyjulis californica	25	(MEMO)
90310	Surf Smelt	Hypomesus pretiosus	25	(MEMO)

Illustration 10: A typical TdbGrid with several rows selected

Now press the Print button.

The report preview window will display all the rows from the grid because by default PrintDAT! is set to print all the rows. We need to change this in the following steps.

- 1) From the report preview window, select **File > Report Options** (or press Ctrl-O) to display the report options window.
- 2) Select the **General Tab** and set **Print What?** to “Selected Rows”.
- 3) If you Preview the report now, only the selected rows will appear in the report.



If no rows are selected in the grid, then all rows will be printed. PrintDAT! assumes you don't want an empty report.

Ask the user if they want to print just the selected rows

You may want to give the user a choice whether to print the selected rows or all the rows from the grid.

Select the Style Page and turn on “*Ask to print selected rows*” in the Include Options list. Now when the report is run the user will see a dialog window:

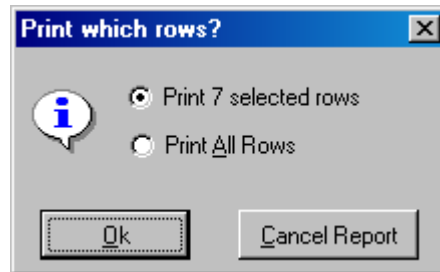


Illustration 11: User can be asked whether to print the selected rows or all rows.

This dialog only appears if **Print What?** is set to “*Selected Rows*” and the “*Ask to print selected rows*” is turned on. The number of selected rows will appear in the first radio button. If this value is zero, it will give the user a chance to go back to the grid in case they forgot to select any rows.



PrintDAT! can print selected rows from any grid that supports this feature, including 3rd party grids. A TStringGrid is even more unusual because it can allow a rectangular block of cells to be selected from several rows, where only some of the cells from the row are selected. It’s much like taking a rectangular cookie-cutter and selecting a block of cells at a time. Not only can PrintDAT! print just that block of cells but if the selected region is wide enough so it needs to fit onto two or more pages, PrintDAT! will also repeat the fixed columns of the TStringGrid on all horizontal pages so you always have a means of identifying the rows on all the horizontal pages.

Report Filter

If you have a large subset of records to print then selecting the records individually may be too much work for the user.

You can of course add a filter to the table and display only those records in the grid. This is the normal approach and it works reasonably well. But what happens if you want to print a subset of the displayed records without re-filtering the grid? Re-filtering the grid data just prior to printing a report may confuse the user because the grid data changes unexpectedly. We need to find a better way.

The rows that go into a PrintDAT! report can be filtered by adding code to the TpdtdPrintDAT.OnReportFilter event to accept or reject records for the report. If this event sets the Accept variable to false, the record will not appear in the report. Because the filtering is done using code in an event, it provides us with a great deal of flexibility.

Report Filter Example

To create a report filter, double click on the OnFilterRecord event in the Object Inspector for the TpdtdPrintDAT object.

Here we have an example where a TpdtdPrintDAT component prints a TDbGrid that displays the BioLife table. The Biolife table has a Common_Name field and the form has a corresponding TEdit field where the user can enter a value for the filter. This TEdit field (edFilter) allows the user to limit the report to only a specific Common_Names value.

We will program the filter event so if the TEdit field is blank then all the rows are printed. If this TEdit field has a search string, then the only the records where the Common_Name field contains our search string will be printed. This makes for a clever but highly effective filter that the user can turn on or off just by entering or removing the TEdit value.

Species No	Common_Name	Species Name	Length (cm)	Notes
90020	Clown Triggerfish	Ballistoides conspicillum	50	(MEMO)
90120	Bat Ray	Myliobatis californica	56	(MEMO)
90130	California Moray	Gymnothorax mordax	150	(MEMO)
90140	Lingcod	Ophiodon elongatus	150	(MEMO)
90150	Cabezon	Scorpaenichthys marmoratus	99	(MEMO)
90160	Atlantic Spadefish	Chaetodiperus faber	90	(MEMO)
90170	Nurse Shark	Ginglymostoma cirratum	400	(MEMO)
90180	Spotted Eagle Ray	Aetobatus narinari	200	(MEMO)
90190	Yellowtail Snapper	Ocyurus chrysurus	75	(MEMO)
90200	Redband Parrotfish	Sparisoma Aurofrenatum	28	(MEMO)
90210	Great Barracuda	Sphyaena barracuda	150	(MEMO)
90220	French Grunt	Haemulon flavolineatum	30	(MEMO)
90230	Dog Snapper	Lutjanus jocu	90	(MEMO)
90240	Nassau Grouper	Epinephelus striatus	91	(MEMO)
90250	Bluehead Wrasse	Thalassoma bifasciatum	15	(MEMO)
90260	Yellow Jack	Gnathanodon speciosus	90	(MEMO)
90270	Redtail Surfperch	Amphistichus rhodoterus	40	(MEMO)
90280	White Sea Bass	Atractoscion nobilis	150	(MEMO)
90290	Rock Greenling	Hexagrammos lagocephalus	60	(MEMO)
90300	Senorita	Oxyjulis californica	25	(MEMO)
90310	Surf Smelt	Hypomesus pretiosus	25	(MEMO)

Illustration 12: In this example a report filter will print only those records where the Common_Name contains the word “fish”.

```

procedure TForm1.GenericPrintDATFilterRecord(   Sender      : TObject;
                                                Dataset    : TDataset;
                                                GridRow     : Integer;
                                                var Accept  : Boolean;
                                                var TerminateReport: Boolean);
begin
  Accept := (Trim(edFilter.Text) = '') or
            (0 <> Pos(UpperCase(edFilter.Text),
                    UpperCase(Dataset.FieldByName('Common_Name').AsString)));
end;

```

The **Dataset** parameter is the dataset that belongs to the grid that is being printed and is pointing at the row that is about to be added to the report. The dataset fields can be examined to determine if the record should be excluded from the report.

The **GridRow** parameter is only used if the object being printed is a memory grid like a TStringGrid, because it doesn't use a dataset and the Dataset parameter will be nil.

If the event sets the **Accept** variable to False, the record will not be included in the report. This variable is initialized to True before the event is called, so you do not need to set it to True for the record to be included in the report.

The **TerminateReport** parameter can be set to True if the report should be terminated gracefully when a certain record has been encountered.

Because our code uses the Pos() function in the event, the filter will accept any record that has “fish” in the Common_Name field. The event provides us with a powerful report filter because we can use any code we like to accept or reject the records. For example, the filter can examine memo fields or lookup fields in other tables to determine if the record should be printed. For example, we may want to prevent the user from printing any personnel records when the record belongs to his superior. He can see the record but he can't print it. But the filter event will allow him to print personnel records for his assistants.

#	Species No	Common_Name	Species Name	Length (cm)	Notes
1	90020	Clown Triggerfish	Ballistoides conspicillum	50	Also known as the big spotted
2	90160	Atlantic Spadefish	Chaetodiperus faber	90	Found in mid-water areas around reefs,
3	90200	Redband Parrotfish	Sparisoma Aurofrenatum	28	Inhabits reef areas. The parrotfish's

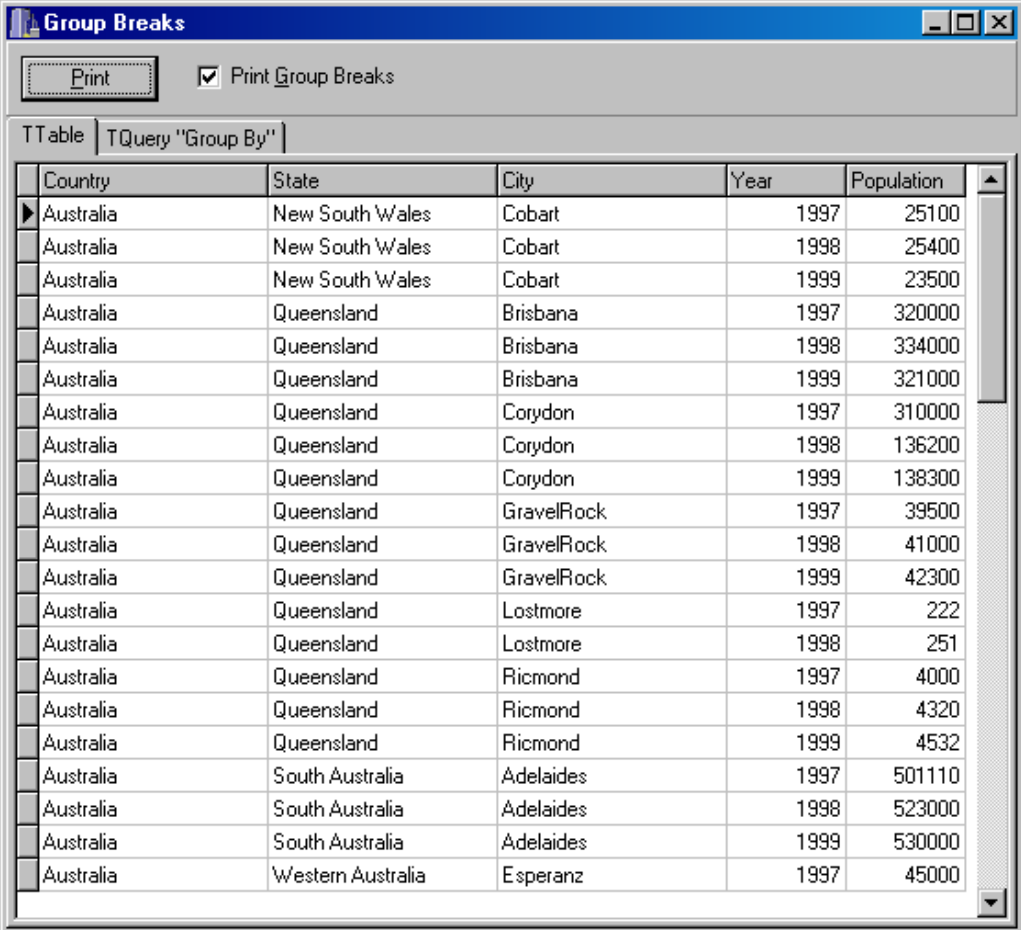
Illustration 13: The report filter produces a report where only the records where Common_Name contains “fish” are printed

The next lesson will give us a break.

Lesson #11 – Printing Group Breaks

Sometimes you will need to print a report where the information is grouped so it gives the appearance of a hierarchical report.

For PrintDAT! to group the information properly the data has to be sorted before the Print method is executed. If you are using a TTable then use its index to sort the columns. If you are using a TQuery then use its “Order By” clause to sort the columns.



Country	State	City	Year	Population
Australia	New South Wales	Cobart	1997	25100
Australia	New South Wales	Cobart	1998	25400
Australia	New South Wales	Cobart	1999	23500
Australia	Queensland	Brisbana	1997	320000
Australia	Queensland	Brisbana	1998	334000
Australia	Queensland	Brisbana	1999	321000
Australia	Queensland	Corydon	1997	310000
Australia	Queensland	Corydon	1998	136200
Australia	Queensland	Corydon	1999	138300
Australia	Queensland	GravelRock	1997	39500
Australia	Queensland	GravelRock	1998	41000
Australia	Queensland	GravelRock	1999	42300
Australia	Queensland	Lostmore	1997	222
Australia	Queensland	Lostmore	1998	251
Australia	Queensland	Ricmond	1997	4000
Australia	Queensland	Ricmond	1998	4320
Australia	Queensland	Ricmond	1999	4532
Australia	South Australia	Adelaides	1997	501110
Australia	South Australia	Adelaides	1998	523000
Australia	South Australia	Adelaides	1999	530000
Australia	Western Australia	Esperanz	1997	45000

When the data is sorted, PrintDAT! can produce group breaks to make the report more readable.

Country	State	City	Year	Population
Australia	New South Wales	Cobart	1997	25100
Australia	New South Wales	Cobart	1998	25400
Australia	New South Wales	Cobart	1999	23500
Australia	Queensland	Brisbana	1997	320000
Australia	Queensland	Brisbana	1998	334000
Australia	Queensland	Brisbana	1999	321000
Australia	Queensland	Corydon	1997	310000
Australia	Queensland	Corydon	1998	136200
Australia	Queensland	Corydon	1999	138300
Australia	Queensland	GravelRock	1997	39500
Australia	Queensland	GravelRock	1998	41000
Australia	Queensland	GravelRock	1999	42300
Australia	Queensland	Lostmore	1997	222
Australia	Queensland	Lostmore	1998	251
Australia	Queensland	Ricmond	1997	4000
Australia	Queensland	Ricmond	1998	4320
Australia	Queensland	Ricmond	1999	4532
Australia	South Australia	Adelaides	1997	501110
Australia	South Australia	Adelaides	1998	523000
Australia	South Australia	Adelaides	1999	530000
Australia	Western Australia	Esperanz	1997	45000
Australia	Western Australia	Esperanz	1998	46400
Australia	Western Australia	Joggalong	1997	3320
Australia	Western Australia	Joggalong	1998	3199
Australia	Western Australia	Joggalong	1999	3410
Australia	Western Australia	Kanunnarra	1997	42000
Australia	Western Australia	Kanunnarra	1998	42400
Australia	Western Australia	Kanunnarra	1999	45211
Australia	Western Australia	Pert	1997	53000
Australia	Western Australia	Pert	1998	54300
Australia	Western Australia	Pert	1999	53221
Australia	Western Australia	Roeburnt	1997	50000
Australia	Western Australia	Roeburnt	1998	51200
Australia	Western Australia	Roeburnt	1999	52100
Canada	British Columbia	Nanamu	1997	4900
Canada	British Columbia	Vancouvert	1997	46500
Canada	Manitoba	Brandom	1997	36000
Canada	Manitoba	Flin Floni	1997	6400
Canada	Manitoba	Winterpeg	1997	615000
Canada	Manitoba	Winterpeg	1998	620000
Canada	Saskatchewan	Saskatoonian	1997	131000
USA	Colorado	Denvert	1997	290000
Venezuela		Caraca	1997	1943000
Venezuela		Caraca	1998	1954900
Venezuela		Caraca	1999	1930000

Illustration 14: A normal PrintDAT! report created from the grid

This conventional PrintDAT! report is full of data with similar column values and it is difficult for our eyes to perceive the overall pattern. We

know the pattern is there but we just can't focus on it. It sort of reminds me of the silly pattern puzzles where if you stare at it long enough an image is suppose to appear. I think in a few years time someone's going to admit it was all a big hoax and there never was any picture there in the first place. Lucky for us PrintDAT! can show us the big picture by re-organizing the report.

How to create a group break report:

- 1) Sort the data in the grid (or dataset) before executing the TpdtdPrintDAT1.Print method.
- 2) Create a PrintDAT! report in the usual way.
- 3) On the Style page of the Report Options window, specify the “# of Group Break Columns”. This usually corresponds to the number of sorted columns. These columns must start from the left side of the report.
- 4) Make sure the Include options list box has “Group Break Lines” checked.

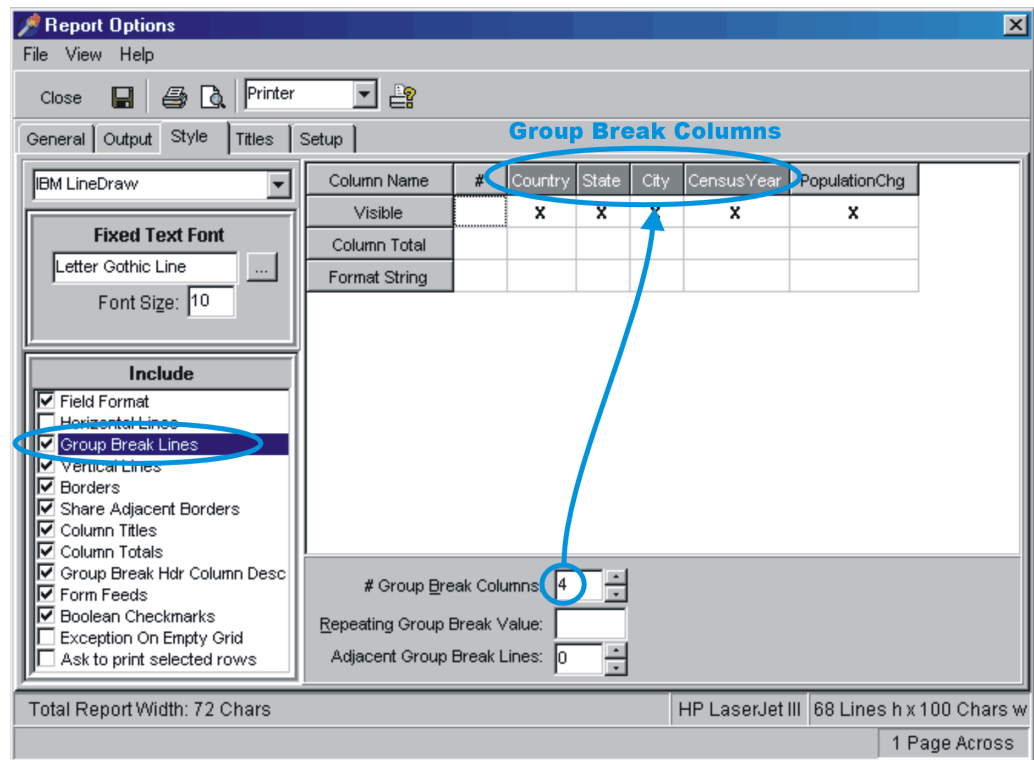


Illustration 15: The Group Break columns are identified by a darker title background.

That's all you have to do. Press the Print button to see the report:

Which report would you rather read? The original report that printed lines of redundant data? Or the new group break report that neatly organizes the data into “boxes”? Three out of four dentists surveyed

prefer the group break report. Sorry, just seeing if you were paying attention. ☺

Country	State	City	Year	Population	
Australia	New South Wales	Cobart	1997	25100	
			1998	25400	
			1999	23500	
	Queensland	Brisbana	1997	320000	
			1998	334000	
			1999	321000	
			Corydon	1997	310000
				1998	136200
				1999	138300
	GravelRock	1997	39500		
		1998	41000		
	1999	42300			
	Lostmore	1997	222		
		1998	251		
	Richmond	1997	4000		
		1998	4320		
		1999	4532		
	South Australia	Adelaides	1997	501110	
1998	523000				
1999	530000				
Western Australia	Esperanz	1997	45000		
		1998	46400		
	Joggalong	1997	3320		
		1998	3199		
		1999	3410		
	Kanunnarra	1997	42000		
1998		42400			
1999		45211			
Pert	1997	53000			
	1998	54300			
	1999	53221			
Roeburnt	1997	50000			
	1998	51200			
	1999	52100			
Canada	British Columbia	Nanamu Vancouver	1997	4900	
			1997	46500	
	Manitoba	Brandon Flin Floni Winterpeg	1997	36000	
			1997	6400	
1997			615000		
1998	620000				
Saskatchewan	Saskatoonian	1997	131000		
USA Venezuela	Colorado	Denvert Caraca	1997	290000	
			1997	1943000	
			1998	1954900	
			1999	1930000	

Illustration 16: Report with group breaks

Group Break Rules

The appearance of the report is remarkably different from that of the normal report. PrintDAT! has removed a lot of redundant data and has redrawn the lines.

- 1) Horizontal lines are only drawn between rows when the next Group Break column is at a higher level (outdented).
- 2) Cell values for the Group Break columns are blanked if they have the same value as those on the previous row.

With these two rules in mind, we'll examine how the report was created. The first Country printed is "Australia" and all the values below it have been blanked until the Country name changes to "Canada". This removed a lot of redundant data and added more white space. A horizontal line is drawn above "Canada" because the previous line had been a subordinate row (indented). This gives us the impression that Australia "owns" all of the rows in this group, which of course it does.

You will notice that there is no horizontal line drawn between "USA" and "Venezuela" because both rows have their group break columns at the same break level, namely at level one. PrintDAT! doesn't waste space by putting in horizontal lines between rows that are at the same group break level. But if this concerns you, we can change it with the *"Adjacent Group Break Lines"* field that is discussed a little later on in this chapter.

The same group break rules apply to "Queensland" and all of its subordinate columns. By glancing at this report we know right away that within Australia we have Queensland and within Queensland are the cities of Brisbane, Corydon, GravelRock, Lostmore, and Richmond. (In case you're wondering, all of the cities are fictitious so put your map away.)

The "Year" column is also a group break column but since there is only one year per city, setting it to a group break column has no effect. Perhaps later on if the user adds more years per city or if the user adds a "Month" column, then the "Years" column will automatically be grouped without having to change the report settings or recompile the report.



Any number of columns can be designated as group break columns and these columns can be sorted in either ascending or descending order or a mixture of both. The group break columns should be sorted if the group breaks are to appear correctly. However no error will be generated if some or all of the group break columns remain unsorted.

The group break columns do not have to be a string value. They can be numeric, date, or any other printable field type.

Breaking all the rules

Rules are made to be broken and we certainly didn't waste any time.

Breaking Rule #1

The first rule said "*Horizontal lines are only drawn between rows when the next Group Break column is at a higher level (outdented).*"

Sometimes you may want to print horizontal break lines between the rows that are at the same break level (adjacent). Take "USA" and "Venezuela" in the previous example. No break line was drawn between these two rows because they were at the same break level.

If you want a horizontal line between rows at the same break level, in this case at level 1, then set "Adjacent Group Break Lines" to 1 on the Style page. The report will be almost the same except for a few additional break lines.

Below you will see the bottom of the report. You will notice there is now a line separating "USA" and "Venezuela".

Canada	British Columbia	Nanamu Vancouver	1997 1997	4900 46500
	Manitoba	Brandon Flin Floni Winterpeg	1997 1997 1997 1998	36000 6400 615000 620000
	Saskatchewan	Saskatoonian	1997	131000
USA	Colorado	Denvert	1997	290000
Venezuela		Caraca	1997 1998	1943000 1954900

Illustration 17: Adjacent Group Break Lines are set to level 1, Country column

This only affects lines for adjacent group breaks that are at level 1 because that is what we have set "Adjacent Group Break Lines" to. The cities "Nanamu" and "Vancouver" do not have a break line between them because they are adjacent at break level 3. Same with "Brandom", "Flin Floni", and "Winterpeg". If we increase "Adjacent Group Break Lines" to level 3, we get line breaks for rows that are adjacent at these break levels. It looks like this:

Canada	British Columbia	Nanamu	1997	4900
		Vancouver	1997	46500
	Manitoba	Brandom	1997	36000
		Flin Floni	1997	6400
		Winterpeg	1997 1998	615000 620000
	Saskatchewan	Saskatoonian	1997	131000
USA	Colorado	Denvert	1997	290000

Illustration 18: Adjacent Group Break Lines are set to level 3, the Cities column

And if we increase the Adjacent Group Break Lines to level 4 (Year) we get:

Country	State	City	Year	Population
Australia	New South Wales	Cobart	1997	25100
			1998	25400
			1999	23500
	Queensland	Brisbana	1997	320000
			1998	334000
			1999	321000
		Corydon	1997	310000
			1998	136200
			1999	138300
		GravelRock	1997	39500
			1998	41000
			1999	42300
		Lostmore	1997	222
			1998	251
		Ricmond	1997	4000
			1998	4320
			1999	4532
		South Australia	Adelaides	1997
	1998			523000
	1999			530000
	Western Australia	Esperanz	1997	45000
1998			46400	
Joggalong		1997	3320	
		1998	3199	

Illustration 19: Adjacent Group Break Lines are at level 4.

We see the Year column now has break lines between every row because the years are all adjacent at break level 4. It is also interesting to note that when break lines occur, they extend through all subordinate columns (columns to the right of the break level).

Subsequent pages of the report will repeat the current Group Break values at the top of each page. If PrintDAT! didn't do this, the group break fields (Country, State, City, Year) fields would be blank if the page break occurred in the middle of their group. You would then have to go back one or more pages to determine the proper group break value.

Country	State	City	Year	Population
*Australia	*Western Australia	*Joggalong	1999	3410

Illustration 20: The second page of the report repeats the current Group Break Values

At the start of a new page, an "*" is automatically prefixed to the repeated group break value(s) to indicate they started on a previous page and did not start at the top of the current page. In this example the Country "*Australia", State "*Western Australia", and City "*Joggalong" are all prefixed with an "*" to indicate these are repeated break values and they first appeared on a prior page and not on this page. This example also has the year 1999 without an asterisk because it does appear for the first time at the top of the second page for this Country, State, and City.



It is a good idea to leave the cell left margin set to 1 (default) so it has enough room for the additional "*" character. Otherwise the value may be wrapped onto two lines. (The cell margins are defined on the Output page.)

Breaking Rule #2

The second rule states: "*Cell values for the Group Break columns are blanked if they have the same value as those on the previous row*".

We don't have to blank the group break values that are repeated. We can substitute a string constant instead. If we go to the Style page and

enter "(Same)" as the "Repeating Group Break Value", then this constant will be used instead of blanks. The report will look something like this:

Country	State	City	Year	Population
Australia	New South Wales	Cobart	1997	25100
(Same)	(Same)	(Same)	1998	25400
(Same)	(Same)	(Same)	1999	23500
(Same)	Queensland	Brisbana	1997	320000
(Same)	(Same)	(Same)	1998	334000
(Same)	(Same)	(Same)	1999	321000

Illustration 21: Repeating Group Break Values set to "(Same)"

A simple double quote may be more readable because it uses less space.

Country	State	City	Year	Population
Australia	New South Wales	Cobart	1997	25100
"	"	"	1998	25400
"	"	"	1999	23500
"	Queensland	Brisbana	1997	320000
"	"	"	1998	334000
"	"	"	1999	321000

Illustration 22: Repeating Group Break Values set to "

The Repeating Group Break Value can be any length you like. If it is too large for the column, then it will be wrapped like any other string. Keep in mind that a short string is less likely to expand narrow columns which is important if you are trying to print a lot of columns on the report.



The next time you have a grid, table, or query that is sorted with similar data values, assign it Group Break Columns and your report will come alive. Group breaks also come in handy when printing views.



If you are adding group breaks to a wide report that prints on more than one horizontal page, then you may want to set “# of Fixed Columns” (on the General Page) to repeat the group break columns at the left side of every page.

The next lesson will send you around the world.

Lesson #12 – Unicode Output

There are two ways the PrintDAT! report output can be encoded:

1. [ANSI](#)
2. [Unicode \(UTF-16\)](#)

By "encoded" we mean how the text information is stored in the report. For decades people have been content with using ANSI output which stores each character as 1 byte. This is very fast and efficient, but has two main drawbacks.

Disadvantages of ANSI:

1. ANSI fonts are typically used for the English language and a few Western languages
2. If you want to have IBM linedraw characters in the report, it means installing a special font like "Letter Gothic Line PD". If the report is printed to a text file and is then sent to someone else, that person will need the same font installed on their computer in order to read the report. (We supply you with a font install program in that situation.)

If none of these drawbacks concern you, then feel free to keep using ANSI encoded reports by unchecking the "Unicode Support" in the Include Options on the Style page of the Report Options window.

But if your reports consist of other languages besides English, or you don't want to use the "Letter Gothic Line PD" font, then Unicode may be what you are looking for.

Disadvantages of Unicode:

1. Unicode is only available on computers running Windows XP, NT4, Windows 2000, Windows 7 or later. Computers running Windows '95, ME, or 98 do not support Unicode (very well).
2. Reports printed to a file will produce a Unicode text file. You will need a modern wordprocessor or text editor to view the report with the IBM linedraw characters. An older text editor or wordprocessor will still display the report properly but it will convert the IBM linedraw characters into [ASCII linedraw characters](#).

3. Unicode text files will use more disk space because each character uses 2 to 4 bytes.
4. Reports exported to a CSV file will produce a Unicode text file. The program importing the data from this text file will need to be able to read Unicode files.

Languages Supported

PrintDAT! can print reports in over 40 different languages using Unicode output and Delphi 2010 or later. A standard fixed spaced font like “Courier New” is used for the report.

But not all languages are supported. Certain languages do not have fixed spaced characters, even when a fixed-spaced font is used, which makes it impossible for the report columns to line up correctly. Here is a list of languages we have tested.

Supported Languages	Unsupported Languages
Afrikaans	Arabic
Albanian	Armenian
Azerbaijani	Belarusian
Basque	Bengali
Catalan	Bulgarian
Croatian	Chinese
Czech	Georgian
Danish	Gujjarati
Dutch	Hebrew
English	Hindi
Estonian	Japanese
Filipino	Kannada
Finnish	Korean
French	Persian
Galician	Tamil
German	Telugu
Greek	Thai

Supported Languages	Unsupported Languages
Haitian Creole	Ukrainian
Hungarian	Urdu
Icelandic	Yiddish
Indonesian	
Irish	
Italian	
Latin	
Latvian	
Lithuanian	
Macedonian	
Malay	
Maltese	
Norwegian	
Polish	
Portuguese	
Romanian	
Russian	
Serbian	
Slovak	
Slovenian	
Spanish	
Swahili	
Swedish	
Turkish	
Vietnamese	
Welsh	

Our Demo program has a Unicode report that can print all of these languages, a sample of which is shown on the next page.

PrintDAT! now supports Unicode,
and can print data from many different languages
in the same report.

Language	Quote
Afrikaans	Uit, uit, kort kers! Life se maar 'n loop skaduwee, 'n swak speler wat die stutte en Frets sy uur op die verhoog en dan word nie meer gehoor word nie: dit is 'n verhaal vertel van 'n idioot, vol
Albanian	Jashtë, jashtë, qiri i shkurtër! Të jetë , por një hije në këmbë, një lojtar i dobët që struts dhe frets orë e tij mbi skenë dhe më pas është dëgjuar jo më shumë: është një tregim i thënë nga një
Azerbaijani	Out həyata, qısa şam! Yaşam lakin bir gəzinti kölgə, bir kasıb oyunçu struts sonra mərhələ sonra onun saat frets və daha çox eşidilir ki, bir nağıl deyil heç bir şey signifying, səs və qəzəb tam
Basque	Out,, kandela laburra! Bizitzaren oinez, baina itzal bat, jokalaria bat pobrea struts eta bere orduko frets eszenatokiaren gainean eta, ondoren entzun ez den gehiago: ipuin bat da,,
Catalan	Fora, fora, una petita espelma! La vida és una ombra que camina, un pobre actor que es vanta i es desgasta durant la seva hora sobre l'escenari i després es va sentir res més: és un conte explicat
Croatian	Out, out, kratka svijeća! Život je, ali hodanje sjena, loš igrač koji opruge i meandri njegov sat na pozornici, a zatim se čuje više: to je priča rekao idiot, pun zvuk i bijesa, što znači ništa.
Czech	Ven, ven, stručný svička! Život je ale chůze stín, chudý hráč, který vzpěry a pražců svou hodinu na jevišti a pak je slyšet víc: je to příběh řekl, idiot, plný zvuk a zuřivost, znamenat nic.
Danish	Ud, ud, korte stearinlys! Livets men en omvandrende skygge, en dårlig spiller, stivere og bånd hans time på scenen, og derefter hørte ikke mere: Det er en fortælling, fortalt af en idiot, fuld af
Dutch	Out, out, korte kaars! Life's, maar een wandelende schaduw, een slechte speler die veerpoten en frets zijn uurtje op het podium en dan wordt niet meer gehoord: het is een verhaal verteld door

Illustration 23: Sample Unicode Report from our Demo program

As you can see from the previous report, the output can contain accented letters from a variety of different languages. It doesn't matter how many different languages the report has, PrintDAT! can print it.

Did I say "print it"? Yes, but you can also:

1. save the report to a Unicode text file
2. preview it on the screen
3. export the contents to a Unicode CSV file so it can be imported to another program like MS Excel™.

How does an English Unicode report compare to an ANSI report using our *Letter Gothic Line PD* font?

The reports will look very similar, and I challenge you to spot the difference.

ANSI Report

#	VendorNo	VendorName	Address1	City	Preferred
1	2014	Cacor Corporation	161 Southfield Rd	Southfield	x
2	2641	Underwater	50 N 3rd Street	Indianapolis	x
3	2674	J.W. Luscher Mfg.	65 Addams Street	Berkely	
4	3511	Scuba Professionals	3105 East Brace	Rancho Dominguez	x
5	3819	Divers' Supply Shop	5208 University Dr	Macon	
6	3820	Techniques	52 Dolphin Drive	Redwood City	
7	4521	Perry Scuba	3443 James Ave	Hapeville	x
8	4642	Beauchat, Inc.	45900 SW 2nd Ave	Ft Lauderdale	x
9	4651	Amor Aqua	42 West 29th Street	New York	x
10	4652	Aqua Research Corp.	P.O. Box 998	Cornish	x
11	4655	B&K Undersea Photo	116 W 7th Street	New York	
12	4681	Diving International Unlimited	1148 David Drive	San Diego	x
13	4682	Nautical Compressors	65 NW 167 Street	Miami	x
14	5385	Glen Specialties, Inc.	17663 Campbell Lane	Huntington Beach	x
15	5641	Dive Time	20 Miramar Ave	Long Beach	x
16	6415	Undersea Systems, Inc.	18112 Gotham Street	Huntington Beach	x
17	6451	Felix Diving	310 S Michigan Ave	Chicago	
18	6452	Central Valley Skin Divers	160 Jameston Ave	Jamaica	
19	6481	Parkway Dive Shop	241 Kelly Street	South Amboy	x
20	6482	Marine Camera & Dive	117 South Valley Rd	San Diego	x
21	6588	Dive Canada	275 W Ninth Ave	Vancouver	x
22	7382	Dive & Surf	P.O. Box 20210	Indianapolis	
23	7685	Fish Research Labs	29 Wilkins Rd Dept. SD	Los Banos	x

ANSI Font: "Letter Gothic Line PD"
IBM Linedraw

Unicode Report

#	VendorNo	VendorName	Address1	City	Preferred
1	2014	Cacor Corporation	161 Southfield Rd	Southfield	x
2	2641	Underwater	50 N 3rd Street	Indianapolis	x
3	2674	J.W. Luscher Mfg.	65 Addams Street	Berkely	
4	3511	Scuba Professionals	3105 East Brace	Rancho Dominguez	x
5	3819	Divers' Supply Shop	5208 University Dr	Macon	
6	3820	Techniques	52 Dolphin Drive	Redwood City	
7	4521	Perry Scuba	3443 James Ave	Hapeville	x
8	4642	Beauchat, Inc.	45900 SW 2nd Ave	Ft Lauderdale	x
9	4651	Amor Aqua	42 West 29th Street	New York	x
10	4652	Aqua Research Corp.	P.O. Box 998	Cornish	x
11	4655	B&K Undersea Photo	116 W 7th Street	New York	
12	4681	Diving International Unlimited	1148 David Drive	San Diego	x
13	4682	Nautical Compressors	65 NW 167 Street	Miami	x
14	5385	Glen Specialties, Inc.	17663 Campbell Lane	Huntington Beach	x
15	5641	Dive Time	20 Miramar Ave	Long Beach	x
16	6415	Undersea Systems, Inc.	18112 Gotham Street	Huntington Beach	x
17	6451	Felix Diving	310 S Michigan Ave	Chicago	
18	6452	Central Valley Skin Divers	160 Jameston Ave	Jamaica	
19	6481	Parkway Dive Shop	241 Kelly Street	South Amboy	x
20	6482	Marine Camera & Dive	117 South Valley Rd	San Diego	x
21	6588	Dive Canada	275 W Ninth Ave	Vancouver	x
22	7382	Dive & Surf	P.O. Box 20210	Indianapolis	
23	7685	Fish Research Labs	29 Wilkins Rd Dept. SD	Los Banos	x

Unicode Font: "Courier New"
IBM Linedraw

Were you able to spot the difference in the two reports?

Spoiler Alert.

The ANSI report is slightly narrower than the Unicode report because the characters in the *Letter Gothic Line PD* font are narrower than the *Courier New* font. Which report looks better on paper is up to each person's own personal taste.

If on the other hand the report is saved to a text file, then the differences become more apparent. The Unicode report requires slightly more disk space, but can be viewed in any Unicode-compatible text editor or wordprocessor with the "*Courier New*" font (using Windows XP or later).

If the ANSI report is saved to a text file, then the computer needs to have the "*Letter Gothic Line PD*" font installed in order to view the report correctly in a text editor or wordprocessor. This isn't usually a big issue because we supply you with a free font install program that you can distribute to anyone viewing your reports with a text editor or wordprocessor. With this font installed, the report stored in the ANSI text file can be viewed on any Windows machine from Win'95 onwards and the computer doesn't have to be Unicode compatible. Of course if the

person is running your compiled application to produce the report, the TpdPrintDAT component automatically installs the "Letter Gothic Line PD" font whenever it is necessary, and it does so the user isn't even aware it is happening.

The following two reports are printed using standard ASCII line-draw characters found on most keyboards. The first report is printed in ANSI and the other in Unicode. I doubt you'll be able to spot the difference.

ANSI Report

```

-----
# | VendorNo | VendorName | Address1 | City | Preferred |
-----
1 | 2014 | Cacor Corporation | 161 Southfield Rd | Southfield | x |
2 | 2641 | Underwater | 50 N 3rd Street | Indianapolis | x |
3 | 2674 | J.W. Luscher Mfg. | 65 Addams Street | Berkely | |
4 | 3511 | Scuba Professionals | 3105 East Brace | Rancho Dominguez | x |
5 | 3819 | Divers' Supply Shop | 5208 University Dr | Macon | |
6 | 3820 | Techniques | 52 Dolphin Drive | Redwood City | |
7 | 4521 | Perry Scuba | 3443 James Ave | Hapeville | x |
8 | 4642 | Beauchat, Inc. | 45900 SW 2nd Ave | Ft Lauderdale | x |
9 | 4651 | Amor Aqua | 42 West 29th Street | New York | x |
10 | 4652 | Aqua Research Corp. | P.O. Box 998 | Cornish | x |
11 | 4655 | B&K Undersea Photo | 116 W 7th Street | New York | |
12 | 4681 | Diving International | 1148 David Drive | San Diego | x |
| | Unlimited | | | | |
13 | 4682 | Nautical Compressors | 65 NW 167 Street | Miami | x |
14 | 5385 | Glen Specialties, | 17663 Campbell Lane | Huntington Beach | x |
| | Inc. | | | | |
15 | 5641 | Dive Time | 20 Miramar Ave | Long Beach | x |
16 | 6415 | Undersea Systems, | 18112 Gotham Street | Huntington Beach | x |
| | Inc. | | | | |
17 | 6451 | Felix Diving | 310 S Michigan Ave | Chicago | |
18 | 6452 | Central Valley Skin | 160 Jameston Ave | Jamaica | |
| | Divers | | | | |
19 | 6481 | Parkway Dive Shop | 241 Kelly Street | South Amboy | x |
20 | 6482 | Marine Camera & Dive | 117 South Valley Rd | San Diego | x |
21 | 6588 | Dive Canada | 275 W Ninth Ave | Vancouver | x |
22 | 7382 | Dive & Surf | P.O. Box 20210 | Indianapolis | |
23 | 7685 | Fish Research Labs | 29 Wilkins Rd Dept. | Los Banos | x |
| | SD | | | | |
-----

```

ANSI Font: "Courier New"
 ASCII Linedraw

Unicode Report

#	VendorNo	VendorName	Address1	City	Preferred
1	2014	Cacor Corporation	161 Southfield Rd	Southfield	x
2	2641	Underwater	50 N 3rd Street	Indianapolis	x
3	2674	J.W. Luscher Mfg.	65 Addams Street	Berkely	
4	3511	Scuba Professionals	3105 East Brace	Rancho Dominguez	x
5	3819	Divers' Supply Shop	5208 University Dr	Macon	
6	3820	Techniques	52 Dolphin Drive	Redwood City	
7	4521	Perry Scuba	3443 James Ave	Hapeville	x
8	4642	Beauchat, Inc.	45900 SW 2nd Ave	Ft Lauderdale	x
9	4651	Amor Aqua	42 West 29th Street	New York	x
10	4652	Aqua Research Corp.	P.O. Box 998	Cornish	x
11	4655	B&K Undersea Photo	116 W 7th Street	New York	
12	4681	Diving International Unlimited	1148 David Drive	San Diego	x
13	4682	Nautical Compressors	65 NW 167 Street	Miami	x
14	5385	Glen Specialties, Inc.	17663 Campbell Lane	Huntington Beach	x
15	5641	Dive Time	20 Miramar Ave	Long Beach	x
16	6415	Undersea Systems, Inc.	18112 Gotham Street	Huntington Beach	x
17	6451	Felix Diving	310 S Michigan Ave	Chicago	
18	6452	Central Valley Skin Divers	160 Jameston Ave	Jamaica	
19	6481	Parkway Dive Shop	241 Kelly Street	South Amboy	x
20	6482	Marine Camera & Dive	117 South Valley Rd	San Diego	x
21	6588	Dive Canada	275 W Ninth Ave	Vancouver	x
22	7382	Dive & Surf	P.O. Box 20210	Indianapolis	
23	7685	Fish Research Labs	29 Wilkins Rd Dept. SD	Los Banos	x

Unicode Font: "Courier New"
ASCII Linedraw

Were you able to spot the difference in the last two reports?

Spoiler Alert

Well, there isn't any difference. Both reports are using the same font, "Courier New" and look identical.

There are minor differences if the files are stored to a text file, but will look the same in a text editor or wordprocessor when using the "Courier New" font. Since neither report are using special linedraw characters, both files will have almost the same physical size, except the Unicode report will have a Unicode BOM marker (byte order marker) at the start of the file.

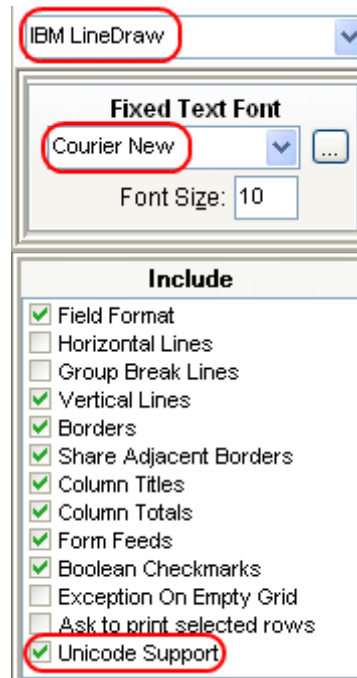
Differences in Exporting data to CSV

If English text is exported to a CSV file, then it is overkill to use Unicode because the same text will be exported in either case. The linedraw characters of course are not exported. The only difference is the Unicode CSV file will have a BOM marker and the program importing the file must be able to handle a Unicode file.

If you are exporting non-English text, then you should of course use Unicode when creating a CSV file.

Activating Unicode

To activate Unicode for a report, select “Unicode Support” from the Include Options on the Style page of the Report Options window.



When “Unicode Support” is checked, the font changes to “Courier New” and the output should be “IBM LineDraw”. The report will now produce the report using Unicode.

If at a later date you wish to switch back to ANSI output, just uncheck “Unicode Support”. It's that simple.

If you don't see “Unicode Support” in the Include options, then you either don't have Delphi 2010 or later, or the UseUnicode option is disabled in \PrintDAT\Source\PDAT.Inc when you compiled the packages. If this is the case, you will need to modify PDAT.Inc by uncommenting the line:

```
//{$DEFINE UseUniCode}
```

and recompile the PrintDAT! packages. (see ReadMe.Pdf).

Lesson #13 – Storing Report Options

PrintDAT! stores the report options in either:

1. A text file called *ApplicationName.PXT* (default)
2. A TDataset database table.

.PXT Text File

By default TpdtdPrintDAT will use a .PXT file to store the report options. This works well for most stand-alone applications. This .pxt file is usually located in the same directory as the application's .exe file. The user will need write access to this .pxt file if the user is to save the report settings. If the user does not have write access to this directory the Object Inspector can specify a .pxt file in a different directory using the ReportSettingsFileName property.

If you do not want the user to save the report settings, then change the .PXT file attribute to Read-Only and PrintDAT! will not attempt to write to the .PXT file.

This .pxt file can be edited with a text editor. If any settings are deleted from this file, PrintDAT! will revert back to the default value.

Unicode Reports

If some of your reports are using Unicode, then this .pxt file will be Unicode encoded so you will need to use a text editor that can read and write a Unicode file, like Notepad++ from <http://notepad-plus-plus.org/>.

TDataset Table

If your application is going to be running on a network or the user has restricted write access to the computer, you may prefer to store the report settings in a database table. Any number of your applications can use the same report settings table, by specifying a different PDAT_Proj for each application.

To use a TDataSet to store the report settings, use the TpdPrintDAT.ReportSettingsDataset property to point to a table in your database.

The report settings table can be created by our PrintDAT! Report Settings Utility Program (PRSU) which is located in the d:\PrintDAT\Utils directory. This program can also import report settings from existing .pjt files and store them in the dataset. The following information is taken from the PRSU_ReadMe.pdf file.

Table Schema

The layout for this table is quite simple. There is one row for each PrintDAT! report.

<i>Field Name</i>	<i>Data Type</i>	<i>Size</i>
<i>PDAT_Proj</i>	String	50
<i>PDAT_Key</i>	String	50
<i>PDAT_Data</i>	Memo	n/a

PDAT_Proj

Each program using this report settings table should have its own project name and this defaults to the name of your executable program. Example: "Project123" without the quotes. If only one program is using this report settings table then the PDAT_Proj can be left empty.

This value is retrieved from the TpdPrintDAT.ProjectName property.

PDAT_Key

The field stores the PrintDAT! Report Id property, namely TpdPrintDAT.ReportId. This uniquely identifies the report within the project. Whenever you drop a TpdPrintDAT component onto a form, this report id gets automatically generated.

The size of the PDAT_Proj and PDAT_Key fields are arbitrary. We chose 50 characters because it is large enough for most project names and report id's. You can change these field sizes if you feel it is necessary. When you change these field sizes, you do not need to change the TpdPrintDAT component or re-compile your programs that use PrintDAT!

The PDAT_Proj ; PDAT_Key combination must be unique.

PDAT_Data

This memo field contains the PrintDAT! report settings for the report. The contents of this field is in plain text and can be edited manually by the developer if he wishes to modify the report settings outside of the normal PrintDAT! report options window. Care of course must be taken not to unintentionally modify other settings for the report. It is important to realize that like the .PXT file, this memo contains only those report settings that are overriding the default values. Any report option that is removed from this memo field will cause that option to revert back to its default value. Any row deleted from this table will delete the report settings for that report. The next time the report is run, a new record for this report will be created in this table. So deleting a record will not cause your program to stop working. It will just revert the report back to its default report settings.

Please note, if the report settings table is deleted (dropped), then your PrintDAT! reports will stop working because the TpdtdPrintDAT component does not have the rights to create this table. The report settings table will have to be re-created using the PRSU program.

PRSU supports several database engines like: BDE, DBISAM, ElevateDb, NexusDb, Advantage, etc.. If the database engine you use is not supported you can easily modify PRSU source code and recompile it. You do not have to make any changes to the TpdtdPrintDAT component to get it to recognize the new table engine. As long as the table is TDataset compatible, it should work just fine.

If any of your reports are using Unicode, then your table must be able to store Unicode data for this field.

The PD_Dev.Chm help file and the PRSU_Readme.Pdf has more information on the Report Settings File.

Lesson #14 - Distributing PrintDAT! with your Application

1) Distribute the Report Settings File

.PXT File

You will need to include this file when you distribute your program. By default this file is located in a directory with the same name as your .exe file except with a .PXT file suffix. So if your program is called d:\Pgms\SalesCom.Exe, then the Report Settings file would be called d:\Pgms\Sales.PXT.

If you don't include the Report Settings file (.PXT) when you distribute your application, PrintDAT! will automatically create it when the report is run. The reports will then revert back to their default settings which is probably not what you want to do.

The Report Settings file does not have to be in the same directory as your program. The TpdPrintDAT.ReportSettingsFile can specify the location of the .PXT file.

Database Table

If you are using a TDataset to store the report settings, then you can ignore the .PXT file and distribute your report settings table instead. See the previous section for more information.

2) Distribute the Linedraw Font (optional)

LetGothL_PD.TTF is the Letter Gothic Line PrintDAT! font used to put the linedrawing characters on the report. The TpdPrintDAT component automatically embeds this font into your .exe file. If the font is not currently installed on the user's machine when a PrintDAT! report is run, the font will be installed automatically by your .exe file and is immediately usable without rebooting. No other program or file, including LetGothL_PD.TTF, needs to be distributed. This is all built into your .exe file. Pretty neat, eh?

So when should the LetGothL_PD.TTF file be distributed?

If the user is printing or viewing reports from your program that has

PrintDAT! Installed, then the font gets installed automatically and you DO NOT need to distribute it.

You only need to distribute and install this font on a machine if they are going to be viewing the report output without ever having run your program. For example, if the user outputs the report to a text file and e-mails it to someone else, then the person that receives the file will need the LetGothL_PD.TTF font. We have a font install program called "*PDATAnsiFontInstall.exe*" in the d:\PrintDAT\Utils directory that you may wish to distribute. The sole purpose of this program is to install the LetGothL_PD.TTF file on a Windows™ 32/64 bit computer. If you don't want to use our PDATAnsiFontInstall.exe program then use your own install program to install this font.

You can of course print the report to a PDF file using many of the free or moderately priced PDF printer drivers. Just make sure the printer driver handles fixed-spaced fonts properly. Some of the cheap PDF printer drivers will not print characters at a uniform width so the report columns will not line up properly. One of the better PDF printer drivers is "PDF Factory Pro" from www.fineprint.com.

3) User Help File

You may wish to distribute our PrintDAT! User's help file PD_USER.chm or PD_User.Hlp that lists the report options screens and has hot links to all of its options. This file is found in the \PrintDAT\DOC subdirectory. We deliberately made this help file as generic as possible by eliminating references to "PrintDAT!" as much as possible so it can pass for one of your own help files.

DO NOT distribute the PrintDAT! Developer's help file called PD_DEV.HLP.

4) Distribute PrintDAT! Runtime Package (Optional)

The PrintDAT! runtime package (VclPD*.?PL) only need to be distributed if your program was compiled with Runtime Packages (Project Options > Packages, Build with runtime packages is turned on). This means the compiler did not use the PrintDAT! DCU files to link the PrintDAT! component into your .exe file. So you will need to distribute the PrintDAT! runtime package with your application. It can be found in your Delphi Projects directory.

DO NOT distribute any of the PrintDAT! .DCU files. These files get compiled into your .exe program if you are not using packages. Your

.exe file is then totally self contained and will not need any additional PrintDAT! runtime packages.


The next lesson is a big help.

Lesson #13—Helpful Hints

PrintDAT! has several features you may have missed that will make your job easier.

Running from IDE

To run PrintDAT! without compiling your program, double click on the

TpdtPrintDAT  object. Make sure the Object Inspector has the ObjectToPrint and ReportId properties filled in.

Display name of Object being printed, ReportId, and location of report settings file (.PXT) at runtime.

When the PrintDAT! report options window is displayed, click on the "# of pages across" box on the status bar. The window caption will give you the name of the object being printed and the ReportId, and the status bar displays the location of PrintDAT!'s report settings file. Be careful not to move the mouse because this will trigger the redisplay of the hints in the status bar.

PrintDAT! isn't limited to just printers

Did you know you could output the report directly to your fax modem? You can even change the output to landscape to print a grid with a lot of columns. Just set Output Destination as "Printer" and use Printer Setup button to select your fax modem. If you always want this report to send the output to the fax modem, turn off Default Printer on the Output page and select the fax modem from the dropdown. Now whenever this report prints, your fax modem gets the output. The size of the report page is automatically adjusted to the size of the fax page so nothing gets truncated. Neat, eh?

Cut and paste a report page into your e-mail

Since the report viewer displays the report in text, you can easily copy (Ctrl-A and Ctrl-C) and paste (Ctrl-V) the text into an e-mail message.

Great for debugging a database program

How many times do you wish you had a hard copy of the data that's in your table? Well now you can. Just drop the PrintDAT! component on

the same form as your TTable component, set its ObjectToPrint property to the name of the TTable, double click the TPdtPrintDAT1 object and print the table to the printer or to a text file. If you want to print a group of tables, simply write a small procedure to create the TPdtPrintDAT object at runtime, create a loop that assigns the ObjectToPrint the table name, and execute a pdtPrintDAT1.Print.. All of your tables will be printed. Remember to use the Setup page to eliminate seeing the report settings window each time. (See below) If you want to send the output to a file, use the Append to File option.

Debugging: Seeing which fields are changed in a table by your application

Use PrintDAT! to print the contents of the table to a text file before your application is run. Use a large page width like 999 characters to prevent horizontal page breaks and left justify the report. Run your application which updates the table in some way. Use PrintDAT! to print the same table this time to a second text file. Now use a file compare program that displays the differences in the 2 text files. (There are plenty of shareware file compare programs to choose from on the web and CIS. WinDiff is one that comes with Windows PowerTools) The file compare program will flag the lines that are different in the 2 text files. You'll know instantly which records were changed. Since PrintDAT! uses text output, printing 10,000+ records to a text file is quite fast and doesn't use much disk space.

HTML Output?

Even though PrintDAT! doesn't support HTML output directly, you can export your grid as an ASCII delimited file to MS Excel™ where it can output it out to an HTML file.

That's it for the tutorials. Class dismissed. But before you go, be sure to check out the FAQ's and help screens in the PrintDAT! User's help file (PD_User.Hlp). These screens are automatically invoked when you press F1 from the Report Options window. The developer's help file (PD_Dev.Hlp) has all of the properties and methods for PrintDAT! and is necessary reading if you want to control the report options under program control. And don't forget about the demo program which has a treasure trove of useful reports. Consider these topics as your homework assignment for tonight.☺

In no time at all, you'll be writing better reports with no effort at all.

The help files: PD_User.hlp/chm and PD_Dev.hlp/chm have FAQ's and more information about PrintDAT!.

Index

A

Adjacent group break lines.....	78
Any Table, Anytime.....	32
Automatic.....	
Printing.....	53
Automatic Printing.....	53

C

Car.....	
Locked Out Of.....	53
Columns.....	
2 line.....	64
Copyright.....	58
Coventional Report Writers.....	2

D

Debugging.....	100
----------------	-----

E

Events.....	
AfterOptionsLoaded.....	55
OnFieldEdit.....	61
Expand to page.....	30
Export ASCII Delimited File.....	25

F

Field.....	
Conversion.....	40
Files.....	
.DCU.....	98
.PXT.....	97
Fixed.....	
Columns.....	35

Footer.....

Page.....	58
Report.....	58

G

Getting Numeric Field Value.....	
Grid.....	39
Group Break Lines.....	46
Group Breaks.....	
At page breaks.....	81
Breaking all the rules.....	78
Breaking rule #2.....	81
How to create.....	75
Lines.....	75
Rules.....	77

H

Header.....	
Page.....	58
Report.....	58
Header Memo.....	
Report.....	59
Help Files.....	
PD_USER.chm.....	98

I

Icon Key.....	14
Importing the file into MS Excel™.....	25
InfoPower™.....	63
Instant Reports.....	2, 3
Invalid.....	
Totals.....	43

L			
Landscape Printing.....	27	PD_USER.chm.....	98
Lesson #4 – Printing TStringGrid....	33	Prevent Printing.....	
Lessons.....		Totals.....	42
Printing Group Breaks.....	73	PreviewFirst.....	56
# 1 - Getting Started.....	16	Printable Components.....	10
# 2 - Report Options.....	21	PrintDAT!.....	
# 3 - Printing Datasets.....	31	Advantages.....	1
# 4 - Printing TStringGrid.....	33	How it works.....	3
# 5 - Column Totals.....	36	What is it?.....	1
# 6 - Printing TDecisionGrid.....	45	Printer Orientation.....	27
# 7 - Automatic Printing.....	53		
# 8 - Setting Report Options.....	55	R	
# 9 - OnFieldEdit Event.....	61	Read Only.....	59
#10 - Printing Selected Rows & Report Filters.....	67	Report Alignment.....	27
#11 - Printing Group Breaks.....	73	Report Options.....	
#12 - Unicode Output.....	84	General Page.....	21
#13 - Storing Report Options.....	94	Hiding.....	59
#13 -- Helpful Hints.....	100	Locked Out Of.....	53
#14 - Distributing PrintDAT!.....	97	PDAT_Data.....	96
Linedraw.....		PDAT_Key.....	95
ASCII Linedraw.....	22	PDAT_Proj.....	95
		Storing.....	94
		TDataset Table.....	94
		.PXT File.....	97
		.PXT Text File.....	94
		Report Settings File.....	97
M		ReportId.....	100
MemoBuf.....	59	Reports.....	
Memory Grids.....	33	Memos.....	
Missing link.....	2	RmCopyright.....	58
		RmPgFtr.....	58
		RmPgHdr.....	58
		RmPgNum.....	58
		RmPgTi.....	58
		RmRptFtr.....	58
		RmRptHdr.....	58
		RmRptTi.....	58
		TStringGrid.....	32
		Root canal.....	1
		Running PrintDAT! from.....	
		IDE.....	100
		S	
		Shrink to page.....	28
		Smart Field.....	

Alignment.....	34	Wide Reports.....	27
Standard Deviation.....			
Totals.....	43	#	
		# of Group Break Columns.....	75
T			
TDecisionGrid.....	45		
Text File Output.....	21		
Text Non-Total.....			
Total.....	39		
Title.....			
Page.....	58		
Report.....	58		
Total.....			
Adding.....			
Column.....	36		
Totals.....			
Adding.....	36		
Average.....	36		
Blank Fields.....	43		
Count Records.....	36		
Formatting.....	41		
Maximum.....	36		
Minimum.....	36		
Numeric Conversion.....	40		
Numerical Accuracy.....	41		
Standard Deviation.....	36		
Sum.....	36		
Two Line Totals.....	42		
Variance.....	36		
Wrapping.....	42		
TpdtPrintDAT.....	100		
TStringGrid.....	33		
TStringGrid, printing.....	33		
U			
Unicode.....			
Activating.....	93		
Exporting data.....	92		
V			
Variance.....			
Totals.....	43		
VcIPD*.?PL.....	98		
W			



PrintDAT!
empowers
your
components
by
making
them
printable