

### Abstract

This paper describes how to achieve presentation–quality tabular output using PROC REPORT, ODS Styles, and the ODS Printer destination for PS and PDF output. Covered are tricks for control of cell width and cell height, borders and shading, splitting and stacking cell values, and all those gnarly tricks you have to know to have (mostly) complete control over the look and feel of your tables. The techniques described can also be applied to PROC TABULATE and other SAS® output as well. But hey, let's face it, REPORT is cool so why would you want to use anything else? I will also cover indenting and other secret formatting abilities of the ODS PRINTER code, and show a preview of coming 8.2 features.

### Introduction

Now that many of you have had a chance to start using ODS, we are hearing more and more questions from people who are trying to customize the output in any number of ways, both in ways that we had anticipated and in ways that we had not anticipated when we designed the system. In this paper I will examine some of the table formats that people have asked us about and show how to achieve these using the ODS PRINTER destination.

Based on the feedback we have received on these issues, we are designing new features to cover some of the "holes" in the original v8 interface for ODS, and at the end of the paper I will give you a preview of some of the features coming in v8.2 and beyond. However, the primary focus of this paper will be on what you can do right now in the latest production release of the SAS system (v8.1). Because this paper focuses squarely on the ODS PRINTER output destination, however, I will feel free to reveal tricks that would not work on other destinations. Whenever something is not portable across destinations, however, I will point that out, and tell you whether we've addressed it in a more robust way for v8.2 or not.

### Tables with fewer rules

One of the more common things that people want is a table with far fewer rules than the very "boxy" tables we produce by default. This one should be easy to do, since we have a pre–packaged style intended to do this very sort of thing. Unfortunately, it doesn't work quite right. Oops.

Nonetheless, it makes a pretty decent start, and we're going to take a look at what it does and how it does it.

First, this is the default appearance of a simple table in ODS PRINTER:

Obs	name	age	height
1	Jessie	12	57.3
2	James	12	59.8
3	Meowth	5	10.2

What we'd like instead is something more like this:

Obs	name	age	height
1	Jessie	12	57.3
2	James	12	59.8
3	Meowth	5	10.2

If things worked as intended, all it would take is this:

```
ods printer style=sasdocPrinter;
```

However, what this produces instead (in v8.1) is this:

Obs	name	age	height
1	Jessie	12	57.3
2	James	12	59.8
3	Meowth	5	10.2

### How was that done?

Ok, how do we fix it? Well, first let's see how "sasdocPrinter" is (incorrectly) defined:

```
proc template;
  source styles.sasdocPrinter;
run;
```

Here's what it shows (edited a bit for readability):

```
define style Styles.sasdocPrinter;
  parent = styles.Printer;
  replace fonts /
    'TitleFont2' =
      ("Arial, Helvetica",12pt,Bold)
    'TitleFont' =
      ("Arial, Helvetica",13pt,Bold)
    'StrongFont' =
      ("Bookman, Times",10pt,Bold)
    'EmphasisFont' =
      ("Bookman, Times",10pt,Italic)
    'FixedEmphasisFont' =
      ("Courier New, Courier",9pt,Italic)
    'FixedStrongFont' =
      ("Courier New, Courier",9pt,Bold)
    'FixedHeadingFont' =
      ("Courier New, Courier",9pt,Bold)
    'BatchFixedFont' =
      ("SAS Monospace, Courier",7pt)
    'FixedFont' =
      ("Courier New, Courier",9pt)
```

```

'headingEmphasisFont' =
  ("Bookman, Times",11pt,Bold Italic)
'headingFont' =
  ("Bookman, Times",11pt,Bold)
'docFont' =
  ("Bookman, Times",10pt);
style Table from Output /
  background = _undef_
  frame = HSIDES
  cellpadding = 4pt
  cellspacing = 0.75pt
  borderwidth = 0.75pt;
end;

```

So it resets the fonts to look more like the SAS documentation—a change we don't even especially desire here—and it resets the table to get rid of the extra rules. Note that this is a bit more involved than you might think at first blush: Not only must you do the obvious thing and get rid of the default borders ("frame = HSIDES"); you must also get rid of the internal rules.

Now, you might expect to see RULES=GROUPS in order to get rules only between the groups and not around all of the cells. The reason you don't is that, as odd as it might seem, this is already the default (inherited from the Output style element). So why do there appear to be rules between all of the cells in the default style? That's because by default the background color of the *table* is black, and the table's background color "shows through" in the space between the cells. In order to avoid this, the code above sets "background = \_undef\_." (Please note that, in order to make things easier to work with in v8.2, we have changed this default.)

### Getting the background colors right

That is all well and good, but it fails to do anything to the background colors of the cells. There are two possible approaches for resetting the background color of the cells: we could change the definition used for the header and footer cells so that it no longer requests a background color, or we could change the color used for those cells. Here is the latter approach:

```

proc template;
  define style styles.fixeddocPrinter1;
  parent = styles.sasdocPrinter;
  replace color_list /
    "bg" = white
    "fg" = black
    "bgH" = white
    "link" = blue ;
  end;
run;

```

And here are the results it produces:

Obs	name	age	height
1	Jessie	12	57.3
2	James	12	59.8
3	Meowth	5	10.2

### Knowing what to change

Now the obvious question here is: *how did I know to do that?*

Well, in order to figure that out, we have to see how the default styles are defined. As you might already know, the default style for the printer destination is "printer" (this is documented in the description of the ODS PRINTER STYLE= option). So we examine the printer style definition:

```

proc template;
  source styles.printer;
run;

```

This style is longer and more complex than the sasdocPrinter style reproduced previously, and thus too long to include in its entirety here, but the relevant part is pretty simple:

```

style Table from Output /
  rules = ALL
  cellpadding = 4pt
  cellspacing = 0.25pt
  borderwidth = 0.75pt;
replace color_list
  "Colors used in the default style" /
  'link' = blue
  'bgH' = grayBB
  'fg' = black
  'bg' = white;

```

In the example above, of course, I simply modified this color list. When we want to modify the definition of the header cells, however, it's a bit more difficult. We will search the printer style in vain for any sign of a definition of the header cells themselves. We will also find ourselves quite unable to find how the borders for the table are controlled, because these attributes are not specified in the printer style at all. Instead, they are *inherited* from the parent style, which is shown along with the other information about the printer style from the "source" command above. The relevant line looks like this:

```
parent = styles.default
```

Now we can get the styles.default source in the same way that we got the styles.printer and styles.sasdocPrinter source. It is the base for almost all of the styles in ODS, and thus is enormous. However, you can search in the log window for "Header" (or, as I did, run in batch and use your favorite text editor to search the log file). You will soon discover that the base style for all the table headers and footers is called HeadersAndFooters, and is defined as follows:

```

style HeadersAndFooters from Cell
  "Abstract. Controls table headers" /
  BackGround = colors("headerbg")
  ForeGround = colors("headerfg")
  font = fonts("HeadingFont");

```

We can also find the definition for Output (from which table inherits) and thereby solve the mystery of the default borders and rules, but I'll leave that one as an exercise for the reader.

**Putting it all together**

So we can change the definition of the headers and footers themselves. In this example, I have gone a bit further—I have also based the style directly off of "printer" rather than "sasdocPrinter" and redefined the table myself. This allows me to avoid the font change that sasdocPrinter introduces, and also to get rid of the rules at the top and bottom of the table, by specifying FRAME=VOID.

```
proc template;
  define style styles.fixeddocPrinter2;
    parent = styles.printer;
    replace HeadersAndFooters from Cell /
      font = fonts("HeadingFont");
  style Table from Output /
    background = _undef_
    frame = VOID
    cellpadding = 4pt
    cellspacing = 0.75pt;
  end;
run;
```

This finally produces the desired results:

Obs	name	age	height
1	Jessie	12	57.3
2	James	12	59.8
3	Meowth	5	10.2

**Controlling cell sizes**

Sometimes you need for the cells to be bigger or smaller than the default size that ODS will pick for you. Perhaps you are designing a form to print out that people will fill in, and you want to leave extra space, or perhaps you want to force the printer to try even harder than it usually would to avoid paneling (though it tries pretty hard without help!), or perhaps you simply want to make the columns equal widths to improve the aesthetics of your particular report.

Whatever the reason, it's easy to accomplish. Just set the CELLWIDTH or CELLHEIGHT on the cells of interest. The subtlety here is that in ODS PRINTER the cell will be made *precisely* the size that you request, even if it's smaller than the other cells in the column, and the visual effect of this is pretty odd. Thus, you want to either make all cells in the column be the same width (in particular, be sure to make the header cell the same width as the data cells), or be sure that you are *increasing* the cell width from the default (the column is made as wide as the widest cell).

Changing the cell sizes (or other attributes of specific cells) differs from the proceeding section because rather than changing the overall document style, we want to affect different portions of the document in different ways. Since this paper is specific to PROC REPORT, it will cover how these are specified in PROC REPORT. There is similar syntax in PROC TABULATE; the output of most procedures is controlled by table templates handled through PROC TEMPLATE; and a few procedures do not allow this sort of control at all. (PROC PRINT will have similar syntax to PROC REPORT starting in v8.2.)

**Making it wider and taller**

Let us say that you wanted to create a small form for somebody to fill out. You want to leave enough room for them to fill in the blanks, but you'd prefer to use something more aesthetically pleasing than resorting to the listing tactic of using a string of underscores. You could do this:

```
data;
  length left $20 right $20;
  input left right;
  cards;
  Name Age
  Address Sex
  City/State Phone
run;

proc report nowindows noheader
  style(column)={cellheight=1cm
    font_weight=bold
    just=r vjust=b
    posttext=":"}
  style(report)={background=_undef_
    rules=rows};
  columns left lv right rv;
  define lv / computed
    style={cellwidth=3in posttext=":"};
  define rv / computed
    style={cellwidth=1in posttext=":"};
  compute lv / char length=1;
  lv = " "; endcomp;
  compute rv / char length=1;
  rv = " "; endcomp;
run;
```

Note that what the STYLE(COLUMN) on the PROC REPORT statement itself really does it to provide a default for all of the table cells. We take advantage of this and even go so far as to specify a POSTTEXT= value there. We are forced to over-ride this default for the two blank columns, but it avoids the necessity to explicitly define either of the actual data columns at all. This produces results like those shown below (except that the output here is compressed horizontally so as to fit into the format of this paper).

<b>Name:</b>	<b>Age:</b>
<b>Address:</b>	<b>Sex:</b>
<b>City/State:</b>	<b>Phone:</b>

**Making it narrower**

There are also times when you'd like to make a column narrower. For example, consider this table:

a	b	c
abc	3	4
def	5	6
supercalifragilisticexpialadociouslyantidisestablishmentarianism	1	2
xyz	7	8

In an effort to accommodate all of the data, ODS PRINTER has widened the table to the very edges of the margins. But the result is not very visually pleasing, since almost all of the data items are quite narrow, and the wide column winds up wrapping anyway. Now the most obvious fix (and probably the most sensible, in this case) would be to simply add a space about the middle of that preposterous word and be done with it. But this is an example in a paper, so we'll do it the hard way instead. Actually, it's not all that hard:

```
proc report nowd;
  define a / style={cellwidth=2in};
run;
```

a	b	c
abc	3	4
def	5	6
supercalifragilisticexpialadociouslyantidisestablishmentarianism	1	2
xyz	7	8

That worked well, but I also want to illustrate the problems you can run into with cellwidth if you don't specify it for all of the cells in a column. I should think of some reason why you might run into this in real life, but I can't. PROC REPORT is a little too user-friendly; the only realistic examples I can think of would use a template. Nonetheless I want to show the effect of mis-matched cellwidth specifications in the hopes that it will help you recognize them if you encounter them while developing a report.

```
proc report nowd;
  define a /
    style(header)={cellwidth=1in};
run;
```

a	b	c
abc	3	4
def	5	6
supercalifragilisticexpialadociouslyantidisestablishmentarianism	1	2
xyz	7	8

**Double rules**

A formatting convention sometimes encountered in tables is the double-rule; that is, at "breaks" in the data, or just after the header, the rule is doubled, like this (well, somewhat like this anyway; normally the gap between the lines would be rather smaller):

team	player	Batting average
Green Sox	Jeff O'Malley	0.289
	Jose Ng	0.333
Purple Sox	Fred Jeffries	0.300
	Jeff Fredrick	0.189
	Cedric Halfonts	0.267

Now, in an ideal world, this would be quite simple; in truth, the BREAK BEFORE / SKIP option should probably do exactly this; however, it does not, so we have a little more work to do. The first thing to try is to use the summarize option but somehow make the output disappear. This easily enough done by making the foreground white; however, it is still full height. So we also shrink the cell, as shown below:

```
proc report nowd;
  define team / order;
  break before team /
    summarize style={foreground=white
                      cellheight=1pt};
run;
```

Now, this seems logical enough, but it has this effect:

team	player	Batting average
Green Sox	Jeff O'Malley	0.289
	Jose Ng	0.333
Purple Sox	Fred Jeffries	0.300
	Jeff Fredrick	0.189
	Cedric Halfonts	0.267

The problem is that the height of the cell is forced down by the cellheight attribute, but the height of the text is not, so the white text partially over-writes the cell's rule. Definitely not the effect we were hoping for. So rather than using cellheight, in this case, we can use the font\_size attribute to shrink the cell, as follows:

```
summarize style={foreground=white
                  font_size=1pt};
```

And thus we get this table again:

team	player	Batting average
Green Sox	Jeff O'Malley	0.289
	Jose Ng	0.333
Purple Sox	Fred Jeffries	0.300
	Jeff Fredrick	0.189
	Cedric Halfonts	0.267

This is the best we can do, I fear. The gap is still a lot larger than the 1pt we requested, due to the cell padding. However, if we get rid of the cell padding, the results are perfectly hideous:

team	player	Batting average
Green Sox	Jeff O'Malley	0.289
	Jose Ng	0.333
Purple Sox	Fred Jeffries	0.300
	Jeff Fredrick	0.189
	Cedric Halfonts	0.267

There are tricks we could do to add space at the start and finish of the cells, but nothing (that I can think of anyway!) to get vertical space on the cells where we desire them but not on the others. And for all that work, we can't make it as close as we'd like it anyway; font sizes smaller than 1pt aren't supported. Nonetheless, since this paper is supposed to cover all sorts of tricks, I will show how we can add horizontal space to the data cells. Note that just using *pretext* and/or *posttext* isn't sufficient; without the *asis* attribute, leading and trailing spaces are stripped.

```
proc report style={cellpadding=0} nowd
  style(column) = {pretext=" "
                  posttext=" "
                  asis=yes
                  };
  define team / order;
```

Here is the result. Note that it's not really complete, even for what it intends to do; to complete it we'd need to mimic the column attributes for the header so that it didn't bump up against the left and right edges, either.

team	player	Batting average
Green Sox	Jeff O'Malley	0.289
	Jose Ng	0.333
Purple Sox	Fred Jeffries	0.300
	Jeff Fredrick	0.189
	Cedric Halfonts	0.267

### Splitting or stacking

As you may already know, PROC REPORT supports stacked columns already, in a certain sense. But in the usual PROC REPORT terms, this is essentially a shorthand for defining a computed statistic on a variable without requiring a define statement, or at least I can't figure out anything else useful that can be done with it myself. But that's not what we are talking about here. Furthermore, PROC REPORT supports `split=` to specify a split character to be used in PROC REPORT headers to specify where newlines go. That's a lot closer, but it doesn't work for the data cells, just the header cells.

We can take advantage of secret internal features of ODS PRINTER, however, to get "split" or "stacked" values into the data cells of the REPORT output. As it turns out, the "magical" sequence

```
'03'x || "n"
```

will force a newline any time the ODS PRINTER output destination encounters it, even in the middle of a cell. Consider the following report:

```
proc report data=sashelp.class split='*' nowd;
  columns age weight height ratio;
  define age / group;
  define ratio / computed
    "wt / ht*= ratio"
    style={just=r};
  define weight / analysis mean noprint;
  define height / analysis mean noprint;
  compute ratio / character length=25;
    ratio =
      compress(put(weight.mean, f12.))
      || " / " ||
      compress(put(height.mean, f12.))
      || '03'x || "n" || "= " ||
      compress(put(weight.mean/
        height.mean, f12.3));
  endcomp;
run;
```

It produces this output:

Age	wt / ht = ratio
11	68 / 54 = 1.245
12	94 / 59 = 1.588
13	89 / 61 = 1.443
14	102 / 65 = 1.570
15	117 / 66 = 1.789
16	150 / 72 = 2.083

### Indenting and other secrets

As it turns out, the 03n sequence is not the only special sequence. There are a few other such special sequences. These are actually derived from sequences used internally in SAS code, designed primarily for use in notes. Please note that, even though you are reading it here, these are officially *undocumented*. They have not been through our usual QA process for 8.1, and there are some known "quirks" when using them. If you have trouble with them, please use the contact information at the end of this paper to ask about them; *do not* contact the official SAS technical support folks about them. They are swell folks who would probably try to help you anyway, but they are officially supposed to disavow all knowledge of these (for 8.1, anyway). Here is set of such marks that appear to work as expected in 8.1:

- '03'x||'\_' Non-breakable space. The column width will be expanded in preference to breaking here.
- '03'x||'m' Set mark. This position will be remembered, and if the line is too long and must be wrapped, it will wrap to here.
- '03'x||'n' New-line. Forces a line-break here, as shown above.
- '01'x Copyright symbol (©). Actually present throughout SAS but, I believe, little-known.
- '02'x Registered trademark (®). Actually present throughout SAS but, I believe, little-known.
- '04'x Trademark symbol (™). Actually present throughout SAS but, I believe, little-known.

NOTE: The '03'x||'n' sequence will behave differently in v8.2 than in v8.1 in the presence of a mark. To get consistent behavior, use the sequence '03'x||'-2n' instead.

### Examples

Here is a little example of what some of these do.

```
data;
  a = "Mark here: *" || '03'x || 'm' ||
      "This text wraps around to where " ||
      "the mark is rather than to the " ||
      "very start of the line." ||
      '03'x || '-2n' ||
      "This silly example is " ||
      '01'x || " 2000 SAS" || '02'x ||
      " Institute Inc.";
  output;
run;
proc report noheader nowd;
run;
```

Mark here: \*This text wraps around to where the mark is rather than to the very start of the line.  
This silly example is © 2000 SAS® Institute Inc.

### Indenting

One application of the marks, together with asis mode, is to have indented values in a cell. To be perfectly honest I don't really understand why somebody would want this effect, but it is apparently required by some FDA reports, so here we go. Note that leading spaces are normally stripped by ODS code, and using ASIS will prevent line wrapping entirely, so we have to put something innocuous at the start of the line. I chose a mark, which is invisible and harmless since it is superseded later by the intended real mark.

```
data;
  length a $ 255;
  a = "First obs, not indented, and " ||
      "of course wrapping to the start";
  b = "The other column, taking up space";
  output;
  a = '03'x || "m      " || '03'x || "m" ||
      "Second obs, indented, and " ||
      "forced to wrap to the indent point";
  output;
run;
proc report noheader nowd;
run;
```

First obs, not indented, and of course wrapping to the start	The other column, taking up space
Second obs, indented, and forced to wrap to the indent point	The other column, taking up space

### Controlling page-breaks

Sometimes you don't want each procedure to start a new page. You can use this option:

```
options debug="pso_skip_explicit";
```

When this option is in effect, the implicit page breaks before each procedure are suppressed, but we will still go to a new page when we run out of room on the page.

### Other effects

There are numerous other effects that you can get using ODS PRINTER and PROC REPORT that are not covered in this paper. Here, we have covered only those features specific to ODS PRINTER and PROC REPORT, and I have stuck almost exclusively to features not covered in previous ODS papers. There are many style attributes not covered here. There is also traffic highlighting, for example.

Traffic highlighting in PROC REPORT under ODS usually takes advantage of the embedded data—step computation code that is already used for special effects in v6 PROC REPORT. It also supports the SAS format lookup—table approach to traffic lighting, but I won't be covering either one in this paper. If you want to see how it works, though, there are numerous previous papers that go over this sort of thing in detail, available on the SAS web pages. (See the last section of this paper for web and contact information.)

### What the future holds [8.2 preview]

There are a number of new features in the *next* release of SAS, version 8.2, that will simplify the sort of formatting we have covered in this report. I will also touch on a few other highlights of v8.2 enhancements for ODS PRINTER.

#### ESCAPECHAR=

As you have seen, there is some embedded formatting you can already get, but specifying it is very tedious. To alleviate this, we will provide the ESCAPECHAR= option on the ODS PRINTER command. This allows you to specify a character that will be substituted for the '03' x above. The biggest advantage of this is that you need not be in an expression context in order to use these formatting features. The remaining examples in this section will assume that we have specified

```
ods escapechar='\' ;
```

#### New formatting support

In addition to the codes listed above (which of course can be specified with the escape character if desired), these are supported in v8.2:

<code>\w</code>	Preferred line—break. If the line breaks, it breaks there, but if there's enough room, it won't break.
<code>\z</code>	Error code. Formats the output like a SAS error message. Also available: <code>\1z</code> = error <code>\2z</code> = warning <code>\3z</code> = note <code>\4z</code> = fatal
<code>\{arg\}n</code>	Arguments allowed for new—line: <code>-2</code> = same as <code>\n</code> in v8.1: wrap to mark; otherwise, an explicit <code>\n</code> forces a wrap to the left margin Other numbers are taken as how much far to advance vertically; e.g., <code>\1.5n</code> inserts an extra half—line of vertical space.
<code>\S={style-attributes}</code>	Change style. This allows you to specify style attribute values that change in the middle of the cell or paragraph. An empty style reverts to the default.
<code>\{super val\}</code>	Superscript.

<code>\{sub val\}</code>	Subscript.
<code>\R/[tag]"raw-text"</code>	Insert raw text into the output. Allows a tag; if present, the raw text appears only in the output for the named destination.

Here are some of these in action:

```
data;
  a = "\3zIn 8.2, you can put " ||
      "\S={font_weight=bold}bold\S={} " ||
      "and " ||
      "\S={font_style=italic}italic\S={} " ||
      "|| "text into your cells. It's " ||
      "\S={font_style=italic}n\S={}" ||
      "\{super 2} the fun.";
run;
proc report noheader nowd;
run;
```

NOTE: In 8.2, you can put **bold** and *italic* text into your cells. It makes for  $n^2$  the fun.

#### Formatting macros

Even that's more difficult than one would hope. That's why we have available for downloading from our web page some formatting macros to make this easier. The above can now be reduced to this:

```
data;
  a = "\3zIn 8.2, you can put " ||
      "%bo(bold) and %it(italic) " ||
      "text into your cells. It's " ||
      "%it(n)%super(2) the fun.";
run;
proc report noheader nowd;
run;
```

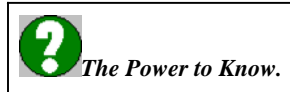
NOTE: In 8.2, you can put **bold** and *italic* text into your cells. It's  $n^2$  the fun.

Actually, the macros work in 8.1, too, but many of them are "dummy" macros that don't do anything since they are interfaces to features that aren't in SAS v8.1.

#### Images

In v8.2, ODS PRINTER supports images. They work just like in ODS HTML, except of course that the image must be available to SAS run—time rather than to the web server. For example, here is the new SAS logo:

```
data;
  a = "The Power to Know."; run;
proc report noheader nowd;
  define a / display
    style={preimage="saslogo.gif"
           pretext=" "
           font_style=italic
           font_weight=bold};
run;
```

**STARTPAGE=**

There will be an official, documented replacement for `pso_skip_explicit`; namely, `STARTPAGE=`. Not only can it be turned ON or OFF, but it can be set to NOW, to force an immediate page break, or to NEVER to prevent page breaks even around graphs. (Normally, `STARTPAGE=NO` will still do page breaks around SAS/GRAPH output because SAS/GRAPH uses the entire page, but by using new [experimental] support to limit the graphs to only a portion of the page, and `STARTPAGE=NEVER`, you can put multiple graphs per page, or mixed graphs and text. You can even overlay text with graphs, though this is very, very rough in 8.2.)

**And more secret stuff**

Even in 8.2, there are experimental and undocumented features that may become "official" in later releases. For example, you use `COLUMNS=` and `TEXT=` on the ODS PRINTER statement to produce multi-column output and to directly insert paragraphs, respectively.

There's also a secret ability to move up an down or back and forth on a line; it's very rough and thus undocumented but could come in handy for special effects. The biggest drawback is that it works only in printer pixels, not in measured units, so the document might have to be altered if you go to another printer. If this proves a particularly popular pastime, it could be promoted to production and proper unit support provided. A short example follows:

```
data;
  a = "In 8.2, you can format like " ||
      "T\7y\~5xE\~7y\~5xX if you want" ||
      "\~73x----- need to.";
run;
proc report noheader nowd;
run;
```

In 8.2, you can format like  $\text{T}\_E\text{X}$  if you want— need to.

**Destinations**

We will also increase the flexibility of the ODS output destinations. First of all, we have, in response to the feedback we've gotten from numerous users, re-thought the way that we present the format options for the printer. While of course the current syntax will continue to work, the "new" way of looking at is to think of each SAS-supported output format as a separate destination. Because they are separate destinations, it is very natural to produce multiple types of "printer" (page layout) output at once:

```
ods printer; /* To physical printer */
ods ps file="foo.ps";
ods pdf file="foo.pdf";
ods pcl file="foo.pcl";
ods display; /* Pop up a window */
```

Note especially that PDF will become a production output destination in v8.2.

Unfortunately, this new paradigm did not make it into the official 8.2 documentation [oddly, the documentations deadlines are much earlier than the code deadlines], but it works just fine, and is the way I recommend using the code in v8.2+.

Also, in 8.2 you will be able to tag your destinations in order to get multiple copies of the same destination with different options. When you are done, you can close all the output destinations with a single statement.

```
ods ps file="foo.ps";
ods ps(2) file="foobw.ps" nocolor;
:
:
ods _all_ close;
```

**Seeing is believing!**

This is one of my favorite things about this paper. In SAS version 8.2, at least, the presentation-quality output abilities are strong enough that it's actually possible to format an entire paper in SAS itself. In fact, this paper was written in SAS, using the macros available in the ODS PRINTER section of the SAS web pages.

Now, I'll be real about this. Would you really want to toss out your Microsoft Word® or  $\text{T}\_E\text{X}$  program in favor of formatting directly in SAS? No, you'd have to be nutty to do that. Primarily due to limitations in the SAS language itself, which was *not* designed for natural-language input, the input is rather awkward, and ODS PRINTER wasn't designed for this task, either, so it lacks (for example) full justification and hyphenation support.

I, on the other hand, *am* a nut, so it doesn't bother me, and it sure makes it easy to include sample SAS input. Of course, the real reason that I did this paper in SAS was to prove that we had provided the tools that made it possible. Actually, it proved no such thing at the outset, but since I wrote and support the ODS PRINTER code, I have upgraded the ODS PRINTER code as necessary to get this paper written, and all of these enhancements will ship with SAS v8.2, though many of them will not be officially documented.

The real reason, of course, that we have this support is to allow you to enhance and annotate your SAS output, not to replace your word processor. I figure that if it's possible to write an entire paper in the system, that's a pretty good indication of our capability to meet your needs.

The formatting macros themselves actually create data sets and print them with PROC REPORT in order to render the paragraphs of the report. The implementation code is much too large to include in this paper, but it is available for download with other ODS PRINTER-related documentation. The source for this paper is also available. Here is a brief example of what the paper source ("qual.sas") looks like:

```
%section( "Section header" )
%para( "A paragraph, with %it(italic)
text. "Quotes" are a pain.
And some example code:" )

```

### ***Still missing***

The biggest feature still missing, in my opinion, having looked at the reports that you folks are trying to do and can't, is the ability in PROC REPORT to combine arbitrary cells, both horizontally and vertically. The second biggest feature is support for data step-based reports. This one we are working at but not in time for 8.2. (Actually, there's a *really* secret SCL interface shipping with 8.2 that could wind up being unofficially documented if there's sufficient demand.) We'd like to hear *your* opinions about what's missing, though. Please contact us at ods@sas.com with questions and comments.

### ***Conclusion***

I'd conclude that ODS PRINTER and PROC REPORT are the greatest thing since sliced bread, but I suppose that I need a more boring conclusion than *that*, so let's try this:

There are a number of tricks you can do to get formatting output using PROC REPORT and ODS PRINTER in version 8.1 of the SAS system. (Most of them, incidentally, work in v8.0 as well.) Even more features are available in 8.2, and we welcome your feedback on what else is needed.

### ***Contact and Web information***

To get to the web pages, you can simply start at [www.sas.com](http://www.sas.com) and navigate your way down to the Base SAS pages (as I write this, you get there by picking *Service & Support* and then *Base SAS* [listed under "Communities"]). From there it's pretty straightforward; many branches lead to information on ODS. Most particularly, the *Tips, Techniques, and SUGI Papers* link leads to previous papers we've presented. To navigate directly to the Base SAS pages, just go directly to "<http://www.sas.com/rnd/base>". To navigate directly to the printer pages, go to "<http://www.sas.com/rnd/base/topics/odsprinter/>".

There's a section of ODS PRINTER-oriented papers in there, in the *Tips, Techniques, and SUGI Papers* section, labeled *ODS PRINTER Information*. In particular, if you use ODS PRINTER much, you should really read the faq. It's pretty hard to miss; there are a number of ways to navigate to it.

### **Stuff nobody reads**

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are registered trademarks or trademarks of their respective companies.