Paper 2-26

Changes & Enhancements for ODS by Example (through Version 8.2)

Sandy McNeill, SAS, Cary, NC

ABSTRACT

The purpose of this paper is to show through example some of the new ODS features and options that have been added for SAS Version 8.2. Some of the highlights are: production PDF destination; template options which allow the invocation of a macro or evaluation of an expression using values from other columns on the same row; in-line formatting for the PRINTER and RTF destinations which allows a finer control over page breaks, subscripting, superscripting, formatting of particular words in a cell instead of the entire cell; and style control for titles and footnotes for the PRINTER and RTF destinations. This paper will talk about each of these topics along with some examples.

INTRODUCTION

With each new version of SAS, the ODS developers stay hard at work adding new features or improving existing features to hopefully make your jobs easier at producing output the way you need it from SAS®. This paper will describe and show examples of some of these new features that we have added for the latest version of SAS Software®.

PDF DESTINATION

I debated calling this a new production destination since some users have been using this in its experimental form, but PDF is now a production destination. This is native PDF without the need for the Adobe Acrobat Distiller. Prior to Version 8.2, you could obtain PDF output by outputting to a postscript file and then invoking the Adobe Acrobat Distiller to distill the postscript into PDF. However, now there is no need for the distiller since we write the native PDF code. The syntax for this is very easy:

```
ods Printer PDF file='foo.pdf'
```

Where foo is, of course, the filename.

Even though you no longer need a distiller, you will still need some type of viewer to view the PDF output. A common viewer for the PCs is Adobe® Acrobat Reader and a common viewer for Unix is ghostview.

STYLE CONTROL FOR TITLE/FOOTNOTE STATEMENTS

In SAS Version 8.1, some of the graph options allowed on the title and footnote statements were passed on to HTML. What this allowed was the stylistic customization of your title and footnote statements such as changing the font, color, height, and justification. Keep in mind that I am talking about titles and footnotes in HTML located OUTSIDE the graph image. If indeed you are running a graph proc, use the options NOGTITLE and NOGFOOTER to place the titles and footnotes outside of the graph image.

In addition to the options on the title/footnote statement, some of the GOPTIONS were also respected. Unfortunately, we ran out of time for Version 8.1 and did not get to add this functionality into all the destinations. The good news is that this same capability is now available in Version 8.2 for the Printer and RTF destinations. The graph options that are respected on the title/footnote statements are: COLOR, FONT, HEIGHT, and UNDERLIN. The GOPTIONS that are supported are: FTITLE, FTEXT, GUNIT, HTITLE, and HTEXT.

Here is an example using the PDF destination with several of these options being used:

```
Title font=courier height=8pt color=blue
    j=left 'Catch the ' font=times
    height=12pt 'Wave';
ods printer pdf file='catch.pdf';
proc print data=sashelp.class;run;
ods printer close;
```

The GOPTIONS work as documented in the SAS/Graph documentation. The values set in the GOPTIONS will be used unless the corresponding style attribute is specified on the title/footnote statement. In that case, the value actually specified on the title/footnote statement will win. For example, if I set the GOPTION HTITLE=5pt, this is setting the HEIGHT style attribute. If you do not specify a HEIGHT option on your TITLE statement, then the height of your text will be 5 points. However, if you specify HEIGHT=10pt on the TITLE statement, then the HEIGHT option on the TITLE statement supercedes the GOPTION. Now if you are familiar with ODS and style templates, you are probably asking yourself "But what about the style attributes specified in the style template? The order of precedence is:

Style template
Goptions
Option specified on the title/footnote statement

Here's what this means: Say you are using a style with a particular font size for titles and footnotes (the default font size for titles is 5 for HTML (styles.default) and 13pt for the PRINTER destinations (styles.printer)). So let's say we are using the PRINTER destination with a default font size of 13pt. If you do not specify HTITLE in the goptions and if you do not specify HEIGHT on the title statement, then the title will be in 13pt. However, if you specify HEIGHT=8pt on the title statement, then the font size for the title will be 8 pt. If you use GOPTIONS HTITLE=8pt and do not use a HEIGHT option on the title statement, then the size will be retrieved from the goptions statement. The goptions HTITLE overrides the font size in the ODS style that is being used.

One caveat to the interaction with the ODS styles: by default, the font for the titles and the footers in the default ODS style template is BOLD and ITALIC. We received some feedback from SAS/GRAPH users that since there is no option on the title or footnote statement to remove the BOLD or ITALIC, but there are options to turn on those attributes, they would like the BOLD and ITALIC turned off automatically when a FONT is specified. This only is in effect when you use the options NOGTITLE or NOGFOOTER on the ODS statement. You can, of course, turn off the BOLD and ITALIC for all your titles and footnotes by modifying the style template.

Here's an example of using the default style and specifying a font which will turn off the BOLD and ITALIC that is received from the default style. :

```
Ods html file='foo.html' nogtitle ;
Title font=COURIER 'This is a title';
proc gchart data=sashelp.class;
vbar sex;
run;
ods html close;
```

If you want to change the font for a particular title, but you still want BOLD and ITALIC, then just use those options on the title statement.

```
Ods html file='foo2.html' nogtitle;
```

```
Title font=courier bold italic 'Bolded
Title';
Proc gchart data=sashelp.class;
Vbar sex;
Run;
Ods html close;
```

Now on to other examples.

Here is an example in which I am creating a new style called SmallTFfont. I am going to be using the RTF destination, so this style will inherit from Styles.RTF and it will specify a font size of 6 pt for both the title and the footnote statements.

```
Proc template ;
Define style SmallTFfont;
Parent = Styles.RTF;

Style SystemTitle from SystemTitle /
    Font = ("Arial, Helvetica", 6pt, bold );
Style SystemFooter from systemFooter /
    Font = ("Arial, Helvetica", 6pt, bold );
End;
Run;
```

TITLE CUSTOMIZATION EXAMPLE 1

The font size of the title statement will be 6pt because we are using the new smallTFfont that we defined above. There are two text strings with one of them left justified and the other one right justified. We did not specify a font, so the font weight will be BOLD because it is picking up the BOLD from the SmallTFfont. The foreground color of the text will be BLACK, which the style SmallTFfont inherits from Styles.RTF which inherits from Styles.Printer.

```
Title j=right 'WAVE'  j=left 'Catch the';
Ods rtf file='titles1.rtf' style=smallTFfont;
Proc Print data=sashelp.class; run;
Ods rtf close;
```

TITLE CUSTOMIZATION EXAMPE 2

The font size of the title statement will be 9pt because now we are specifying a HEIGHT option on the title statement. This HEIGHT option will override the font size that was specified in the style SmallTFfont. All the rest of the attributes will remain the same as in Title Example 1. The height attribute remains 9pt until the end of the statement or until another height option is specified.

```
title HEIGHT=9pt  j=right 'WAVE' j=left
     'Catch the';
Ods rtf file='titles2.rtf' style=smallTFfont;
Proc Print data=sashelp.class; run;
Ods rtf close;
```

TITLE CUSTOMIZATION EXAMPLE 3

Now we are using a font option. Because of this, the BOLD font weight will be cleared. The only attributes that change in this example are the font name and the font weight which is no longer BOLD.

```
title HEIGHT=9pt FONT=courier j=right
     'WAVE' j=left 'Catch the';
Ods rtf file='titles3.rtf' style=smallTFfont;
Proc Print data=sashelp.class; run;
Ods rtf close;
```

TITLE CUSTOMIZATION EXAMPLE 4

We don't like a title statement that is not bolded, so we are going to turn the font weight BOLD back on. The rest of the attributes will remain the same. There is no option to turn the bold off, so once you turn it on, it is on for the remainder of the statement.

```
title BOLD HEIGHT=9pt FONT=courier j=right
     'WAVE' j=left 'Catch the';
Ods rtf file='titles4.rtf' style=smallTFfont;
Proc Print data=sashelp.class; run;
Ods rtf close;
```

TITLE CUSTOMIZATION EXAMPLE 5

Better example of interaction between the style template and the title statement. Create a new style and modify the SYSTEMTITLE style element such that the text of the titles will be the color blue. Then use some options on the title statement and see how the color of the titles is blue unless the color is overridden on the statement. "Catch" is blue from the blue foreground set in the SystemTitle. 'the' is green from the title statement color option. 'WAVE' is blue again from the style template and the varying heights from the title2 statement

```
proc template;
define style ex5; parent = styles.rtf;
style SystemTitle from SystemTitle /
     foreground = blue;
end;
run;

title j=left font='Comic Sans MS'
     'Catch ' color=green 'the';
title2 j=center font='Comic Sans MS'
     height=24pt 'W'
     height=20pt 'A'
     height=16pt 'V'
     height=12pt 'E';
ods rtf file='titles5.rtf' style=ex5;
Proc Print data=sashelp.class;run;
Ods rtf close;
```

IN-LINE FORMATTING AND PAGE BREAK CONTROL

The PRINTER and RTF destinations now support the ability to insert simple formatting text within a given cell or paragraph. The types of formatting available are: specifying a style option to customize the font, color, etc; specifying a superscript and subscript; specifying a dagger or sigma character; specifying raw text to be inserted into the document for the open destination or you can specify that the raw text be targeted to a particular destination.

To use any of these special formatting tricks, you need to use an escape character. By default, '03'x is expected as the escape character. However, using this everywhere you want special formatting is a pain, so we added a special ODS statement just for specifying the escape character that you would like to use.

```
Ods escapechar = '\';
```

This statement above would specify the backslash character as the escape character to be used when specifying the use of in-line formatting.

Assuming we are using the above statement and have specified the backslash as our escape character, here is a table of the formatting codes and their definitions.

\w	Preferred line break. If the line breaks, it breaks there; but if there's enough room, it won't break
\z	Error code. Formats the output like a SAS error message. Also available: \1z = error \2z = warning \3z = note \4z = fatal
\m	Set a mark. The position is remembered

	and if the line wraps, it will wrap to this position.
\-2n	Wrap to mark
\n	Wrap to left margin
\[arg]n	Wrap where the value specified for the arg indicates how far to advance vertically.
\S = {style attributes}	Lets you specify style attributes. See discussion about the style option in this paper under New Template Features, or see the Style statement documentation for Proc Template in the on-line documentation.
\{super text}	Superscript what is denoted by text
\{sub text}	Subscript what is denoted by text
\R</tag>"raw-text"	Insert raw text into document for the current output destination. The tag is optional and represents a particular destination. If the tag value is present, then the raw text only appears in the output for the specified destination. The values for tag would be: RTF, PDF, PS

I am going to go over an example, but for more information see the paper written by Brian Schellenberger (the in-line formatting guru) at http://www.sas.com/rnd/base/topics/odsprinter/qual.pdf.

IN-LINE FORMATTING EXAMPLE

Example of In-Line formatting using both the Postscript destination and the RTF destination. This example demonstrates the ability to subscript, superscript, and change the formatting of words within a particular cell.

```
data b;
   x= "\3z\mIn 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." ||
      "\-2nWhere do we end up?";
   y=2;
   z=3;
   run;

proc template;
define style foo;
parent = styles.rtf;
style systemtitle from systemtitle /
   protectspecialchars = off;
end;
run;

ods escapechar = '\';

ods printer ps file='InLineFormatEx.ps';
ods rtf file='InLineFormatEx.rtf'
     style = foo;

proc report nowd data=b;
title2 '001 PROC REPORT: Using Functions';
title3 'In a title too:
     \{super super} \{sub sub}';
title4 '\{dagger} \{sigma}';
run;

ods rtf close;
ods printer close;
```

A couple things are worth pointing out about this example. This example uses the MARK (escapechar m) formatting; however, this will currently only work for the PRINTER destinations such as postscript and PDF. If you run this example and look at the RTF output, you'll see that the second line "Where do we end up" is not indented, but it IS indented in the PRINTER destinations. Another difference with the destinations is that for RTF, you must tweak the SystemTitle style element and turn off the PROTECTSPECIALCHARS attribute if you want to use the in-line formatting in the title statement (or tweak SystemFooter for footnote statements). You don't, however, have to do this for the PRINTER destinations. We are looking into making this more consistent for Version 9.

CONTROLLING PAGEBREAKS FOR RTF AND PRINTER

Another option that has been added to the PRINTER and the RTF destinations is the ability to control page breaks. Currently, each procedure call starts a new page. Sometimes you don't want that, and with the number of questions that we have received asking if it is possible to control the page breaks, I'd say the number of you wanting this functionality is pretty high.

The option to control page breaks for the PRINTER and the RTF destinations is STARTPAGE= and is an option on the ODS statement. The values for this option are: NOW, ON, OFF. So if you use

Ods printer ps file='foo.ps' startpage=no;

The implicit page breaks before each procedure will be suppressed. We will, however, still go to a new page when we run out of room on the page.

PAGE BREAK CONTROL EXAMPLE

This example is to demonstrate the usage of the STARTPAGE option to control page breaking. On our initial ODS statement, we specify STARTPAGE=NO which tells ODS to suppress the page breaks which you would normally get for each procedure invocation. Before the third PROC PRINT, we issue another ODS statement, but this time with STARTPAGE=NOW. This will start a new page at the next procedure call, but after that the page breaks are still suppressed until a STARTPAGE=YES or a STARTPAGE=NOW is processed.

```
Data x;a=1;b=2;c=3;run;
Ods printer ps file='PrinterPageBreakEx.ps'
     Startpage = no;
/* Page 1 */
Proc print data=x; run;
Proc Print data=x; run;
Ods printer startpage = now;
/* Page 2 */
Proc print data=sashelp.class; run;
Proc print data=x; run;
Ods printer startpage = yes;
/* Page 3 */
Proc print data=x; run;
/* Page 4 */
Proc print data=x; run;
Ods printer close;
```

TEMPLATE OPTIONS

Several new options have been added to Proc Template for Version 8.2. One of these options is the ability to use a format name as the attribute value in the STYLE option. This functionality has been available in Proc Tabulate in Version 8.1, but for consistency this has now been made available in Proc Report, Proc Template, and in the new Proc Print styles. The nice this about this is that you can write your user-defined formats for traffic lighting, and then use these formats across several different procedures.

Here is a Proc Template example to illustrate the use of creating a user-defined format and then using that format as the value for a style attribute. One of the reasons for using this technique is to perform traffic lighting. This example uses the dataset exprev

and the formats found under Proc Print style example 3.

```
proc template;
   define table RegionReport;
      define column Region; end;
      define column State;end;
      define column month;end;
      define column revenues;
         style = {background=revfmt.
                flyover=revfly.};
      end;
   end;
run;

ods html file='TemplateEx1.html';
ods listing close;

title 'Regional Activity';
data _null_;
   set work.exprev;
   file print ods=(template='RegionReport'
       columns=(region
                state
                month
                revenues));
   put _ODS_;
run;

ods html close;
ods listing;
```

A second new functionality available in Proc Template is the EXPRESSION function which can be used to evaluate an expression which is given as the value for a style attribute. The expression to be evaluated within the parentheses follows the same rules as expressions used in the TRANSLATE and the CELLSTYLE…AS functions. The powerful thing about this is that you can get to the values in other columns on the same row instead of just being able to use the global symbol _VAL_ to access the value of the current cell.

Here's a second Proc Template example which illustrates the use of the EXPRESSION function. In this example, we use both an expression and the global symbol _VAL_ as parameters in the EXPRESSION function. The global symbol _VAL_ is the current value of the cell. Notice in the expression how we are accessing the values from other columns.

```
ods listing close;

ods html file='TemplateEx2.html';

data colors;
  length lightness saturation hue $ 20;
  input lightness $ & saturation $ & hue $ &;
  datalines;
dark  grayish  blue
moderate  reddish  purple
dark  moderate  red
medium  strong  orange
light  greenish  yellow
strong  bluish  green
;

proc template;
   define table colortab;
   define header tabhd;
      text 'Expression function';
      style={foreground=#FF00FF};
   end;
   column lightness saturation hue
       lightnesshue lightnessSaturHue;
   define lightness;
      header='Lightness';
   end;

   define saturation;
      header='Saturation';
   end;

   define hue;
      header='Hue';
      style={background=expression("_val_")};
   end;

   define lightnesshue;
      compute as lightness||' '||hue;
      header='Foreground is lightness and
                  hue';
      style = {background =
         expression("lightness||' '||hue")};
   end;

   define lightnessSaturHue;
      compute as lightness || ' ' ||
                 saturation || ' ' || hue;
      header='Background is lightness
              saturation hue';
      style = {background =
              expression("lightness ||
                  ' ' || saturation ||
                  ' ' || hue")};
   end;
end;

data _null_;
   set colors;
   file print ods=
     ( template='colortab'
       columns(
         lightness
         saturation
         hue
        )
      );

   put _ods_;
run;

ods _all_ close; /* closes ALL the open */
         /* destinations - even listing */
ods listing;     /* reopen the listing */
```

A third new functionality that's very cool is the ability to invoke a macro by using the RESOLVE function. &_Val_ is a global macro variable which is set with the value of the current cell before the call to the macro. This lets you use the value of the current cell within the macro.

This third Proc Template example illustrates the invocation of a macro called DOURL which is evaluated and create the attribute value for the URL style attribute. After running this example, you should see that each of the columns headers is a unique link. Column A will have a link to http://www.yourcompany.com/a, column B will have a link to http://www.yourcompany.com/b, and column C will have a link to http://www.yourcompany.com/c.

```
data x;
   a=1;
   b=2;
   c=3;
   run;

%macro dourl;
```

```
   http://www.yourcompany.com/&_val_
%mend;

proc template;
   define table testit;
   column a b c;

   define a;
      define header hdra;
         style={url = resolve('%dourl')};
      end;
      header = hdra;
   end;

   define b;
      define header hdrb;
         style={url = resolve('%dourl')};
      end;
      header = hdrb;
   end;

   define c;
      define header hdrc;
         style={url = resolve('%dourl')};
      end;
      header = hdrc;
   end;
end;
run;

ods listing close;
ods html file='TemplateEx3.html';
data _null_;
   set x;

   file print ods=
     ( template='testit'
       columns=(
         a
         b
         c
        )
      );

   put _ods_;
 run;

ods html close;
ods listing;
```

PROC PRINT STYLE OPTION

For Version 8.2, Proc Print now has a style option which allows style customizations just like Proc Report, Proc Tabulate, and Proc Template. The general form of the style option is exactly the same as for the other Procedures:

STYLE<(location-name(s))> =
<style-element-name>[style-attribute-specification(s)]

Where style-attribute-specification(s) is:

```
Style-attribute-name = Style-attribute-value
```

Here are tables which show the values that can be used for the LOCATION-NAMES and also the default style element that is in effect if you do not specify any STYLE-ELEMENT-NAME. These tables are broken down by what is valid for different statements, so there is a table for the Proc Print statement, a table for the VAR statement, a table for the ID statement, and a table for the SUM statement.

Location-Names and Default-Style-Element-Names for the PROC PRINT Statement

Location Name(s)	Affects	Default Style Element
DATA COLUMN COL	Default for all data cells of all columns	Data
HEADER HEAD HDR	Default for all header cells of all columns	Header
OBS OBSDATA OBSCOLUM N OBSCOL	Data cells of OBS column	RowHeader
OBSHEADER OBSHEAD OBSHDR	Header of OBS column	Header
TOTAL TOT BYSUMLINE BYLINE BYSUM	The SUM line containing totals for each BY group	Header
BYLABEL BYSUMLABEL BYLBL BYSUMLBL	The label for the by-variable on the line containing SUM totals	Header
GRANDTOTAL GRANDTOT GRAND GTOTAL GTOT	The SUM line containing the grand totals for the whole report	Header
TABLE REPORT	Output data table (not byline)	Table

Location-Names and Default-Style-Element-Names for the VAR Statement

Location Name(s)	Affects	Default Style Element
DATA COLUMN COL	Data Cells	Data
HEADER HEAD HDR	The column header cell	Header

Location-Names and Default-Style-Element-Names for the ID Statement

Location Name(s)	Affects	Default Style Element
DATA COLUMN COL	Data Cells	RowHeader
HEADER HEAD HDR	The column header cell	Header

Location-Names and Default-Style-Element-Names for the SUM Statement

Location Name(s)	Affects	Default Style Element

DATA COLUMN COL	Data cells	Data
HEADER HEAD HDR	The column header cell	Header
TOTAL TOT BYSUMLINE BYLINE BYSUM	Data cell containing sum	Header
GRANDTOTAL GRANDTOT GRAND GTOTAL GTOT	Data cell containing sum over whole report	Header

Some of the more widely used style-attribute-names are:

- Font
- Font_face
- Font_size
- Font_style
- Font_width
- Foreground
- Background
- URL
- Just
- Preimage
- Postimage

A complete list of the style-attribute-names is documented in "The Complete Guide to the SAS Output Delivery System", specifically within the Style Statement section under The TEMPLATE Procedure. These style-attribute-names are the same attribute names used wherever you might see the STYLE statement.

So what does all this mean? How do you use it? Better to explain this with examples. I am going to use the HTML destination for these examples.

PROC PRINT STYLE EXAMPLE 1

Make the background of the OBS column and the background of the column headers the color blue.

```
ods html file='PrintStyleEx1.htm';
Proc Print data=sashelp.class
   style(OBS HEADER OBSHEADER) =
            {background=blue};
Run;
Ods html close;
```

PROC PRINT STYLE EXAMPLE 2

A little fancier. Shows output using by and pageby.

```
proc sort data=sashelp.class out=class;
     by sex age;
run;

ods html file='PrintStyleEx2.htm';

proc print data=class n='Number of
observations for the sex and age '
     style(N)={foreground=black
               font_weight=bold}
     style(OBS HEADER OBSHEADER) =
        {background = mediumred
         foreground = black}
        /*TOTAL makes the whole TOTAL line red*/
        style(TOTAL) = {background = white
                        foreground = mediumred}
        style(GRANDTOTAL) = {background = BLACK
                             foreground = white}
        style(BYSUMLABEL)= {background=mediumred
                            foreground = white
                            font_weight = bold};
var name height weight ;
sum height weight;
by sex age ;
pageby sex;
run;
ods html close;
```

PROC PRINT STYLE EXAMPLE 3

Since the by-line is generated outside of Proc Print, the style of the by-line is not controllable from Proc Print. You would have to modify the style element ByLine in the style template that you are using. In these HTML examples, the default style template is styles.default. However, a way around this with Proc Print is to use the ID statement and use the same variables on the ID statement as on your BY statement.

Use ID statement, which gets rid of the byline. Notice that I removed the style for the OBS and the OBSHEADER and placed a style on the ID statement.

```
proc sort data=sashelp.class out=class;
     by sex age;
run;
ods html file='PrintStyleEx3.htm';
proc print data=class n='Number of
observations for the sex and age '
     style(N)={foreground=black
               font_weight=bold}
     style(HEADER)={background=mediumred
                    foreground=black}
/* TOTAL makes the whole TOTAL line red */
     style(TOTAL)={background=white
                   foreground=mediumred}
     style(BYSUMLABEL)={background=mediumred
                        foreground=white
                       font_weight=bold};
var name height weight ;
sum height weight;
by sex age ;
id sex age / style(HEADER DATA) =
     {background=mediumred
      foreground=black};
pageby sex;
run;
ods html close;
```

PROC PRINT STYLE EXAMPLE 4

This last example shows how to do traffic lighting with Proc Print. This capability is possible now through the use of the style statement and user-defined formats. Once the formats are defined to bind a particular style (in this case a background) to a range of values, then the format is used as the attribute's value.

In this example, we have four user-defined formats:

Revfmt – format for revenue numbers
Expfmt -- format for expense numbers
Revfly – format for revenue flyover (popup msg when hovering over the cell)
Expfly – format for the revenue flyover

Their color schemes are just the opposite from each other: the

lower revenue numbers are shaded red to point out the fact that the revenues were not so good, but the lower expense numbers are indicated as green.

```
data exprev;
   input Region $ State $ Month monyy5.
         Expenses Revenues;
   format month monyy5.;
   datalines;
Southern GA JAN95 2000  8000
Southern GA FEB95 1200  6000
Southern FL FEB95 8500 11000
Northern NY FEB95 3000  4000
Northern NY MAR95 6000  5000
Southern FL MAR95 9800 13500
Northern MA MAR95 1500  1000
;

options nodate pageno=1 linesize=70
    pagesize=60 nobyline;
proc sort data=exprev;
   by region;
run;

proc format;
 value revfmt
    low-5000    = 'light red'
     5000-10000 = 'yellow'
     other      = 'green';
 value revfly
    low-5000    = 'Your revenues are
                   dangerously low'
    5000-10000  = 'Your revenues are all
                   right'
    other       = 'GREAT JOB! Keep up the good
                   work';
 value expfmt
    low-5000    = 'green'
    5000-8000   = 'yellow'
    other       = 'red';
 value expfly
    low-5000    = 'Great job controlling those
                   expenses'
    5000-8000   = 'You had better start
                   controlling expenses'
    other       = 'I''m bringing in the
                   comptroller';
run;

ods html file='PrintStyleEx4.htm';
proc print data=exprev noobs
   n='Number of observations for the state: '
 'Number of observations for the data set: ';
var revenues /
    style(COLUMN) = {background=revfmt.
                     flyover=revfly.};
var expenses /
    style(COLUMN) = {background=expfmt.
                     flyover=expfly.};
sum expenses revenues ;
by region;
id region;
run;

ods html close;
```

I will admit that there is a small defect in Version 8.2 with using the user-defined formats as style attribute values. If you use a format as a style attribute value, then you cannot define another format for that variable. For instance, in the example above where I use the format revfmt, you could not also have a format statement for revenues, such as:

```
Format revenues comma9.;
```

This problem will definitely be fixed at Version 9.

CONCLUSION

I hope that this paper has given you an insight into some of the new features awaiting your use in SAS Version 8.2. ODS continues to grow from input from YOU, the valued users, as we strive to make SAS an invaluable tool to help you in your jobs.

REFERENCES

SAS Institute, Inc., *The Complete Guide to the SAS® Output Delivery System, Version 8,* Cary, NC: SAS Institute, Inc., 1999. 310 pp.

Schellenberger, Brian., *Presentation-Quality Tabular Output via ODS.,* SAS Institute, Inc., 2000. http://www.sas.com/rnd/base/topics/odsprinter/qual.pdf

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Mrs. Sandy McNeill
SAS Institute, Inc.
Cary, NC 27513
Email: Sandy.McNeill@sas.com
ODS specific questions can be directed to:
ods@sas.com