

ODS for PRINT, REPORT and TABULATE

Lauren Haworth, Genentech, Inc., San Francisco

➤ ABSTRACT

For most procedures in the SAS system, the only way to change the appearance of the output is to change or modify the ODS style definition. There are three exceptions: REPORT, TABULATE, and PRINT. These procedures allow you to change the output style attributes on the fly when the output is generated.

With these three procedures, you can create almost any type of tabular report. Add in the extra control over style attributes, and you have a reporting powerhouse.

This paper will show how to change the fonts, colors, and alignment of your output. You will also learn how to use formats to highlight key results in special colors and use images in table headings.

For convenience, all of the examples are shown as HTML output. Except where noted, the examples all work for RTF or printer output as well.

The examples in this paper are based on SAS version 8.2. The PRINT examples will only work with version 8.2 or later. Most of the REPORT and TABULATE examples will also work with versions 8.0 and 8.1.

➤ INTRODUCTION

This first series of examples will show how you can use style attributes to modify the appearance of the result table headings. Later examples will show techniques for rows and table values.

The STYLE= option is a new option available for just the PRINT, REPORT, and TABULATE procedures. What it allows you to do is to specify a number of different style attributes for specific parts of your output. These style attributes control things like typefaces, foreground and background colors, text alignment, and table borders.

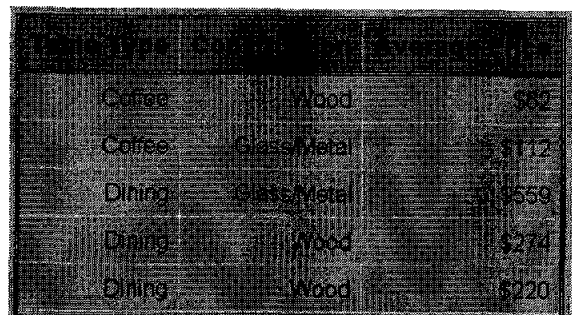
➤ EXAMPLE #1: CHANGING TABLE HEADING STYLES FOR PROC PRINT

Applying style attributes to PROC PRINT output is very simple. All you have to do is add STYLE= options to whichever part of the output you wish to change.

There are a number of ways you can use STYLE= in the PRINT procedure. To keep things simple, we'll just look at one technique: adding a STYLE= option to the PROC PRINT statement. We'll start with the following code. It produces a listing of three variables.

```
ODS HTML FILE='tables.htm';
proc print data=tables noobs label;
  var Type Material Price;
run;
ODS HTML CLOSE;
```

This code produces the output shown below. It's a basic table in the Default style definition (only the first few rows of the table are shown).



Type	Material	Price
Coffee	Wood	\$82
Coffee	Glass/Metal	\$112
Dining	Glass/Metal	\$559
Dining	Wood	\$274
Dining	Wood	\$220

The typeface used in the headings and table body is Arial. As an example, we will change this typeface to Arial Narrow, which will make the headings narrower. This is useful if you have a table that is too wide.

We do this by adding a STYLE= option to the PROC PRINT statement. The STYLE keyword is followed by the name of the style element we wish to modify. In this case, the element is called Header. A list of the elements you can modify is in the *Online Documentation for the PRINT procedure*.

The style element is listed in parentheses. Following the name of the style element, we use an equal sign and then list the attribute we wish to change. Detailed documentation on these attributes and their settings can be found in the "Guide to the Output Delivery System" in the Online Documentation.

The attribute must be contained between square brackets "[]" or curly brackets "{ }". The code below changes the font typeface to Arial Narrow. No-

tice the use of quotes around the typeface name. Because 'Arial Narrow' contains spaces, these quote marks are necessary.

```
ODS HTML BODY='tables.htm';
proc print data=tables noobs label
  STYLE(Header)=[FONT_FACE='Arial Narrow'];
  var Type Material Price;
run;
ODS HTML CLOSE;
```

The results are shown below. Now each of the table headings takes up less width. This makes the overall table narrower, allowing you to fit more information on the page (or screen). This technique can be used to change any of the column heading attributes.

Type	Material	Price
Coffee	Wood	\$62
Coffee	Glass/Metal	\$112
Dining	Glass/Metal	\$589
Dining	Wood	\$274
Dining	Wood	\$220

➤ EXAMPLE #2: CHANGING TABLE HEADING STYLES FOR PROC REPORT

Next, we'll run through the same column heading modification for PROC REPORT output. We'll start with a table similar to the previous example.

```
ODS HTML BODY='tables.htm';
proc report data=tables nowd;
  column Type Material Price;
  define Type / group;
  define Material / group;
  define Price / analysis mean;
run;
ODS HTML CLOSE;
```

This code produces the output shown in below. It's a basic table using the Default style definition, and is very similar to the PROC PRINT table we modified earlier.

Type	Material	Price
Coffee	Glass/Metal	\$103
	Wood	\$59
Dining	Glass/Metal	\$404
	Wood	\$196
End	Glass/Metal	\$107
	Wood	\$81

Once again, we're going to change the heading font. We do this with the exact same technique as we used for PROC PRINT. We add a STYLE(Header)= option to the PROC REPORT statement.

```
ODS HTML BODY='tables.htm';
proc report data=tables nowd
  STYLE(Header)=[FONT_FACE='Arial Narrow'];
  column Type Material Price;
  define Type / group;
  define Material / group;
  define Price / analysis mean;
run;
ODS HTML CLOSE;
```

The new results are shown below. Again, we get a table with narrower headings. This technique can be used to change any of the column heading attributes.

Type	Material	Price
Coffee	Glass/Metal	\$103
	Wood	\$59
Dining	Glass/Metal	\$404
	Wood	\$196
End	Glass/Metal	\$107
	Wood	\$81

➤ EXAMPLE #3: CHANGING TABLE HEADING STYLES FOR PROC TABULATE

To round out these examples, we'll run through making the same change for a table produced by PROC TABULATE.

Once again, we will add a `STYLE=` option to the part of the output we wish to change. We'll start with a table similar to the previous examples.

```
ODS HTML BODY='tables.htm';
proc tabulate data=tables f=dollar8.;
  class Type Material;
  var Price;
  table Type*Material,
        Price*Mean=" ";
run;
ODS HTML CLOSE;
```

This code produces the output shown below. It's a basic table using the Default style definition, and is very similar to the previous tables created with `PRINT` and `REPORT`.

Type	Material	Price*Mean
1	WOOD	\$198
1	GLASS	\$404
2	WOOD	\$69
2	GLASS	\$103
3	WOOD	\$61
3	GLASS	\$187

This time, instead of adding a `STYLE(Header)=` option to the main procedure call, we need to do things variable by variable. With `TABULATE`, you specify styles for the row and column headings in the `VAR` and `CLASS` statements that identify the variables.

```
ODS HTML BODY='tables.htm';
proc tabulate data=tables f=dollar8.;
  class Type Material /
    STYLE=[FONT_FACE="Arial Narrow"];
  var Price /
    STYLE=[FONT_FACE="Arial Narrow"];
  table Type*Material,
        Price*Mean=" ";
run;
ODS HTML CLOSE;
```

The results are shown below. Notice how the three top headings now show the new narrower font. However, the row headings do not show the same

change. This is because these are the class level *value* headings, not the class *variable* headings.

Type	Material	Price*Mean
1	WOOD	\$198
1	GLASS	\$404
2	WOOD	\$69
2	GLASS	\$103
3	WOOD	\$61
3	GLASS	\$187

To make all of the headings match, we need to add one more statement to our code. The `CLASSLEV` statement is used to apply options to class level values. You can use the `STYLE=` option on the `CLASSLEV` statement to change the fonts to match the top headings.

```
ODS HTML BODY='tables.htm';
proc tabulate data=tables f=dollar8.;
  class Type Material /
    STYLE=[FONT_FACE="Arial Narrow"];
  classlev Type Material /
    STYLE=[FONT_FACE="Arial Narrow"];
  var Price /
    STYLE=[FONT_FACE="Arial Narrow"];
  table Type*Material,
        Price*Mean=" ";
run;
ODS HTML CLOSE;
```

The new results are shown below.

Type	Construction	Price
Dining	Wood	\$186
Coffee	Glass/Metal	\$404
Dining	Glass/Metal	\$59
Dining	Wood	\$103
Dining	Wood	\$81
Dining	Wood	\$107

This technique can be used to change any of the row or column headings. You can also use different style attributes for each classification variable, by creating multiple CLASS and CLASSLEV statements, one set for "Type" and one set for "Material". Then you could use a different STYLE= option for each variable. The same technique also works when you have two analysis variables, you can use two VAR statements with two different STYLE= options.

➤ EXAMPLE #4: APPLYING TRAFFIC LIGHTING TO PROC PRINT RESULTS

What do traffic lights have to do with SAS Output? The answer is that the familiar red, yellow, and green lights can be used to highlight results in your output. You use red for bad results, yellow for neutral results, and green for good results. If you're creating a large table, this technique is great for focusing the reader's attention on the key results.

In this example, we will take our familiar table and use traffic lighting to mark low prices in green, high prices in red, and prices in between in yellow.

The first step is to set up a format with the colors we'd like to use. The three color settings are RGB values for red, yellow, and green.

```
proc format;
  value traffic low-100='cx006600'
               100<-300='cxFF9900'
               300<-high='cxCC0000';
run;
```

The next step is to set up the code for the table, and then apply the format. The code is shown below.

```
ODS HTML FILE='tables.htm';
proc print data=tables noobs label;
  var Type Material;
  var Price /
    STYLE=[BACKGROUND=traffic.];
run;
ODS HTML CLOSE;
```

It's the same code from our previous example. The only difference is that the VAR statement has been split into two statements. That's so the STYLE= option can be applied to just the variable Price. The style attribute we are modifying is BACKGROUND, which controls the background color.

You could just specify a single background color here, such as "Red" or "cxCC0000", which would make the entire column of results red. Instead, we will use our format. This allows us to have a conditional background color. Depending on the value of each table cell, the background will be set as green, yellow or red. The results are shown below.

Type	Construction	Price
Coffee	Wood	\$186
Coffee	Glass/Metal	\$404
Dining	Glass/Metal	\$59
Dining	Wood	\$103
Dining	Wood	\$81

Now the low prices show up in green, the high prices in red, and the ones in between are in yellow. However, with these dark backgrounds, the price values get a bit hard to read. The default font weight is a bit light for these intense backgrounds. To fix that, we'll make the fonts bold, as shown in the code below.

```
ODS HTML FILE='tables.htm';
proc print data=tables noobs label;
  var Type Material /
    STYLE=[FONT_WEIGHT=BOLD];
  var Price /
    STYLE=[BACKGROUND=traffic.
            FONT_WEIGHT=BOLD];
run;
ODS HTML CLOSE;
```

The font is changed to bold in both VAR statements so that the table will look consistent. The new output is shown below.

Coffee	Wood	
Coffee	Glass/Metal	
Dining	Glass/Metal	
Dining	Wood	
Dining	Wood	

The resulting table is shown below. It looks much like the PROC PRINT results.

Coffee	Glass/Metal	
	Wood	
Dining	Glass/Metal	
	Wood	
End	Glass/Metal	
	Wood	

By the way, you can also do traffic lighting by changing the foreground values. Instead of making the cell background red, yellow, or green, you can make the cell text red, yellow, or green. The code is the same, except instead of using `BACKGROUND=` in the `STYLE=` option, you use `FOREGROUND=`.

One additional note regarding this example: you may have noticed that the price amounts were not formatted as dollar amounts. That is because there is a bug in version 8.2 that prevents the use of a format in this way on a variable that is already formatted. The problem affects only the PRINT procedure, and not the REPORT or TABULATE procedures.

► EXAMPLE #5: APPLYING TRAFFIC LIGHTING TO PROC REPORT RESULTS

For PROC REPORT, the technique for doing traffic lighting is almost the same.

You add the `STYLE=` options to the `DEFINE` statements that set up each table column. For the Price column, the background color is set to the format we defined previously. For all of the columns, the font weights are set to bold to increase readability. The code is shown below.

```
ODS HTML BODY='tables.htm';
proc report data=tables nowd;
  column Type Material Price;
  define Type / group
    STYLE=[FONT_WEIGHT=BOLD];
  define Material / group
    STYLE=[FONT_WEIGHT=BOLD];
  define Price / analysis mean
    STYLE=[BACKGROUND=traffic.
      FONT_WEIGHT=BOLD];
run;
ODS HTML CLOSE;
```

► EXAMPLE #6: APPLYING TRAFFIC LIGHTING TO PROC TABULATE RESULTS

For PROC TABULATE, the technique for doing traffic lighting is a bit different. The `STYLE=` option settings are the same as the previous two examples, but the way to apply the `STYLE=` option is different.

You might think that to apply a style to the values of Price, you would add the `STYLE=` option to the VAR statement for Price. However, that would only affect the heading for Price.

To change the appearance of values within a table, you need to apply the `STYLE=` option in the `TABLE` statement. The following code shows how this is done.

```
ODS HTML BODY='tables.htm';
proc tabulate data=tables f=dollar8.;
  class Type Material;
  var Price;
  table Type*Material,
    Price*Mean=" "
    [STYLE=[BACKGROUND=traffic.
      FONT_WEIGHT=BOLD]]];
run;
ODS HTML CLOSE;
```

The `STYLE=` option is added to the `TABLE` statement by using an asterisk operator. Notice that it is added to specification for the variable Price in the column dimension. This can be a little confusing to see, because the statistic MEAN is also applied to the variable Price. Another thing to notice is that this time the `STYLE=` option uses two sets of brackets. The additional brackets surround the entire `STYLE=` option. These are required when adding `STYLE=` options within a `TABLE` statement.

The resulting table is shown below.

Type	Material	Price
Coffee	Glass/Metal	\$103
Dining	Glass/Metal	\$404
End	Glass/Metal	\$107

➤ EXAMPLE #7: CREATING ALTERNATE ROW SHADING FOR PROC REPORT RESULTS

While most everything you can do in PRINT and TABULATE can be done in REPORT, a few tricks work best in PROC REPORT. This is because you can use a COMPUTE block to calculate the output formatting.

A good example of this is creating a table with alternate row shading. This means that the rows of data alternate light and dark colors, making them easier to read. The code for doing this is shown below.

```
ODS HTML BODY='tables.htm';
proc report data=tables nowd;
  column Type Material Price;
  define Type / group;
  define Material / group;
  define Price / analysis mean;
  compute Material;
    count+1;
    if (mod(count,2)) then do;
      CALL DEFINE(_ROW_, "STYLE",
        "STYLE=[BACKGROUND=white]");
    end;
  endcomp;
run;
ODS HTML CLOSE;
```

What this code does is figure out whether each row is an even or odd number. For odd numbered rows, the background color is assigned to white (the default color is gray).

Instead of using STYLE=, we will use a CALL DEFINE statement. This allows us to call up the row style element, and modify its attributes. The CALL DEFINE has three parameters: the name of the element being modified (_ROW_), the description of what is being modified (STYLE), and the STYLE= code to make the change.

The resulting table is shown below.

Type	Material	Price
Coffee	Glass/Metal	\$103
Dining	Glass/Metal	\$404
End	Glass/Metal	\$107

➤ EXAMPLE #8: ADDING A LOGO TO A PROC REPORT TABLE

Changing fonts and colors goes a long way to livening up your output, but adding graphics takes your results to a new level. This example will show how to add a heading with a logo to your PROC REPORT results.

The first thing you need to do is set up the heading. This is done with a COMPUTE BEFORE _PAGE_ statement and a LINE statement.

```
ODS HTML BODY='tables.htm';
proc report data=tables nowd;
  column Type Material Price;
  define Type / group;
  define Material / group;
  define Price / analysis mean;
  compute before _page_ / LEFT;
    LINE "Totally Tables, Inc.";
  endcomp;
run;
ODS HTML CLOSE;
```

The result is that a new row is added to the top of the table, and it contains the text given in the LINE statement, as shown in the output below. Unfortunately, the row is created using the default table cell style attributes, which makes it rather faint in appearance.

Totally Tables, Inc.		
Type	Material	Price
Coffee	Glass/Metal	\$103
	Wood	\$59
Dining	Glass/Metal	\$404
	Wood	\$196
End	Glass/Metal	\$107
	Wood	\$61

To fix the appearance of the title and add a graphic, we can use the **STYLE=** option on the **COMPUTE** statement. First, we'll add the graphic using the **PREIMAGE** attribute. This example assumes that the graphic is stored in the same file location as the HTML page we are creating. The second fix we need to make is to make the font bolder, bigger, and in a brown color that coordinates with the graphic.

```
compute before _page_ / LEFT
  STYLE=[PREIMAGE='table.gif'
        FONT_WEIGHT=BOLD
        FONT_SIZE=5
        FOREGROUND=cx993300];
  LINE "Totally Tables, Inc.";
endcomp;
```

The new results are shown below.

Totally Tables, Inc.		
Type	Material	Price
Coffee	Glass/Metal	\$103
	Wood	\$59
Dining	Glass/Metal	\$404
	Wood	\$196
End	Glass/Metal	\$107
	Wood	\$61

Note: This example is set up for HTML output, and gives the font size using a number which represents an HTML font size. If you are creating RTF or printer output, list the font size in points (for example, "14pt").

► EXAMPLE #8: ADDING A LOGO TO A PROC TABULATE TABLE

Adding a logo is even easier using **PROC TABULATE**. You can easily place a graphic image in the top left corner of the table using the **BOX=** option.

The style attribute information is added with a **STYLE=** suboption on the **BOX=** option. The **PREIMAGE** style attribute allows you to name an image that precedes the text in the cell.

In this example, the title is added using a **LABEL=** option, and then is followed by the **STYLE=** option. Note the unusual syntax. In order to have both a **LABEL=** and a **STYLE=** suboption, both need to be enclosed in brackets following the **BOX=** option.

```
ODS HTML BODY='tables.htm';
proc tabulate data=tables f=dollar8.;
  class Type Material;
  var Price;
  table Type*Material,
        Price*Mean=" "
        / BOX=[LABEL='Totally Tables, Inc.'
              STYLE=[PREIMAGE='dining.gif']];
run;
```

ODS HTML CLOSE;

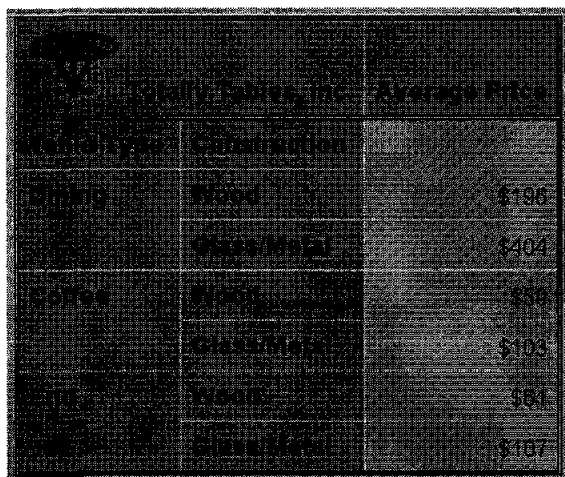
This code produces the following table.

Totally Tables, Inc.		
Type	Material	Price
Coffee	Glass/Metal	\$103
	Wood	\$59
Dining	Glass/Metal	\$404
	Wood	\$196
End	Glass/Metal	\$107
	Wood	\$61

Now our table contains the logo and title in the top left corner. However, the headings look funny because the title is sitting at the bottom of the table cell, but the heading "Average Price" is in the middle of the cell. We can use another `STYLE=` option to fix this.

```
ODS HTML BODY='tables.htm';
proc tabulate data=tables f=dollar8.;
  class Type Material;
  var Price / STYLE=[VJUST=B];
  table Type*Material,
    Price*Mean=" "
    / BOX=[LABEL='Totally Tables, Inc.'
      STYLE=[PREIMAGE='dining.gif']];
run;
ODS HTML CLOSE;
```

The `VJUST=B` style attribute added to the `VAR` statement for `Price` causes the heading for this variable to be vertically justified to the bottom of the table cell. The new results are shown below.



Totally Tables, Inc.		Average Price
Type	Material	
Dining	Wood	\$196
Dining	Glass/Metal	\$404
Coffee	Wood	\$103
Coffee	Glass/Metal	\$103
Tea	Wood	\$61
Tea	Glass/Metal	\$107

➤ OTHER STYLE ATTRIBUTES YOU CAN MODIFY

These few examples have only scratched the surface as far as what you can do with ODS and the `PRINT`, `REPORT`, and `TABULATE` procedures. There are dozens of style attributes you can modify, and many places to use them within each procedure's output.

In general, the same style attributes are available for use with the `STYLE=` options for these three procedures. The table in Appendix A lists the attributes and what they do.

The examples in this chapter have shown how to modify individual cells, rows, columns, or headings. More examples of this type of modification are in-

cluded in the "Guide to the SAS Output Delivery System" in the Online Documentation.

This paper has not covered how to apply a style attribute to the entire table or report. The syntax is actually quite simple. For `PROC REPORT`, use the following code:

```
PROC REPORT DATA=dataset
  STYLE=[style-attribute(s)];
```

For `PROC PRINT`, the syntax is very similar:

```
PROC PRINT DATA=dataset
  STYLE=[style-attribute(s)];
```

For `PROC TABULATE`, the syntax is different. You use a `STYLE=` option at the end of the `TABLE` statement:

```
TABLE <<page-definition,>
  row-definition,>
  column-definition
  / STYLE=[style-attribute(s)];
```

➤ CONCLUSION

I hope this paper has encouraged you to start experimenting with the new `STYLE=` options available for the `PRINT`, `REPORT`, and `TABULATE` procedures. The possibilities for output enhancement are endless. Have fun with these techniques.

➤ ACKNOWLEDGEMENTS

SAS is a registered trademark 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.

➤ CONTACTING THE AUTHOR

Please direct any questions or feedback to the author at: info@laurenhaworth.com

APPENDIX: ODS STYLE ATTRIBUTES

Attribute	What it affects	Can be used for individual cells, columns, rows, headings?	Can be used for entire table?
ASIS	leading/trailing spaces and line breaks	✓	✓
BACKGROUND=	background color	✓	✓
BACKGROUNDIMAGE=	background image	✓	✓
BORDERCOLOR=	border color	✓	✓
BORDERCOLORDARK=	dark border color for 3-D borders	✓	✓
BORDERCOLORLIGHT=	light border color for 3-D borders	✓	✓
BORDERWIDTH=	width of border	✓	✓
CELLHEIGHT=	height of table cell	✓	
CELLWIDTH=	width of table cell	✓	
CELLPADDING=	space between cell text and borders	✓	
CELLSPACING=	space between cells	✓	
FLYOVER	text to display for mouse over (HTML)	✓	
FONT=	font definition (face, size, weight/style)	✓	✓*
FONT_FACE=	font typeface	✓	✓*
FONT_SIZE=	font size	✓	✓*
FONT_STYLE=	font style	✓	✓*
FONT_WEIGHT=	font weight	✓	✓*
FONT_WIDTH=	font width	✓	✓*
FOREGROUND=	text color	✓	✓*
FRAME=	table frame type	✓	
HREFTARGET=	window or frame to open for link	✓	
HTMLCLASS=	name of stylesheet to use	✓	✓
JUST=	horizontal justification	✓	✓
NOBREAKSPACE=	handling of spaces at line breaks	✓	
OUTPUTWIDTH=	width of table	✓	
POSTHTML=	HTML code to add at end of item	✓	✓
POSTIMAGE=	image to display at end of item	✓	✓
POSTTEXT=	text to display at end of item	✓	✓
PREHTML=	HTML code to add at beginning of item	✓	✓
PREIMAGE=	image to display at beginning of item	✓	✓
PRETEXT=	text to display at beginning of item	✓	✓
PROTECTSPECIALCHARS=	handling of <, >, and & characters	✓	
RULES=	lines between table cells	✓	
TAGATTR=	string to insert in HTML tag for the item	✓	
URL=	URL to link to when item is clicked	✓	
VJUST=	vertical justification	✓	