Paper 125-29

SAS® with Style: Creating your own ODS Style Template for RTF Output

Lauren Haworth, Genentech, Inc., South San Francisco, CA

➤ ABSTRACT

Once you've started using the Output Delivery System, you'll quickly discover that your taste in output design probably doesn't coincide with the built in ODS styles shipped with SAS software. While you can edit your RTF output in Word to improve its appearance, a better approach is to create your own style template. This workshop will take you step by step through the process of creating a custom style for your RTF output.

You'll learn how to make minor modifications, and how to give your output a complete makeover. If you'd like all of your SAS output to be in hot pink with a gigantic script font, this workshop will show you how! Or, if you'd just like to use fonts and borders that coordinate with your corporate style guidelines, you can do that too. The workshop will also provide tips and tricks for taking advantage of the RTF destination, including the generation of custom page numbers and page breaks.

The workshop will walk through the TEMPLATE procedure, showing how you can redefine the default style elements and attributes to customize fonts, colors, and borders to suit your own personal or corporate style. You'll be given a basic style template that you can customize during the workshop and then take home to try out on your ODS output. While many of the techniques in this workshop apply to other ODS destinations, the focus will be on RTF. The workshop is aimed at beginning to intermediate ODS users, and is based on SAS versions 8.2 and 9.

➤ INTRODUCTION

ODS styles are a huge topic, and there's no way to cover them in depth in this format. This workshop will take a fast-track approach. We'll cover just enough of the syntax to get you started. Then, we'll use a sample style template that can be edited to modify color, fonts, spacing, rules, and borders. This will allow you customize most aspects of your output. However, to truly control the look of your output, plan on taking a much more in-depth course.

➤ USING THE STYLE= OPTION

You may not have realized it, but whenever you issue an ODS command you are using a style definition. By default, ODS uses a standard style for each output destination. When you issue an ODS statement like:

```
ods rtf body='sample.rtf';
```

You're really issuing the following:

```
ods rtf body='sample. rtf'
        style=Default;
```

So if you wish to switch to another style, all you have to do is add a STYLE= option and specify the name of a different style. However, the only choices you have are the standard styles shipped with your SAS software.

➤ PROC TEMPLATE

To truly change the look of your output, you need to create your own style. This is done by using the TEMPLATE procedure. This new procedure has statements that allow you to define every aspect of a style. However, if we had to specify every aspect of every new style, we'd spend all of our time typing PROC TEMPLATE code. A complete style definition could run to hundreds of lines of code. To make our life easier, we have the PARENT statement. It allows a new style to be based on an existing style. Then you can add lines of code for only those things you want to change.

➤ THE EXAMPLE PROGRAM

Rather than try to explain all of the statements and syntax available for PROC TEMPLATE, let's just look at our example program (Appendix A). This program creates a new custom style. The first section of code sets up the name of the style (Custom) and indicates that it will be based on the Default style.

```
proc template;
  define style Styles.Custom;
  parent = Styles.RTF;
```

The next section of code sets up a list of font names and assigns them characteristics. This list is used later in the program as a shorthand way to specify fonts.

```
replace fonts /
  'TitleFont' = ("Times Roman",13pt,Bold Italic)   /* Titles from TITLE statements */
  'TitleFont2' = ("Times Roman",12pt,Bold Italic) /* Procedure titles ("The _____ Procedure")*/
  'StrongFont' = ("Times Roman",10pt,Bold)
  'EmphasisFont' = ("Times Roman",10pt,Italic)
  'headingEmphasisFont' = ("Times Roman",11pt,Bold Italic)
  'headingFont' = ("Times Roman",11pt,Bold)        /* Table column and row headings */
  'docFont' = ("Times Roman",10pt)                 /* Data in table cells */
  'footFont' = ("Times Roman",13pt)                /* Footnotes from FOOTNOTE statements */
  'FixedEmphasisFont' = ("Courier",9pt,Italic)
  'FixedStrongFont' = ("Courier",9pt,Bold)
  'FixedHeadingFont' = ("Courier",9pt,Bold)
  'BatchFixedFont' = ("Courier",6.7pt)
  'FixedFont' = ("Courier",9pt);
```

This style statement is used to supply attributes to the style element called "fonts". By using the "replace" syntax, this code will overwrite the existing fonts style element. In this case, we are setting up seven font names and their characteristics. See Appendix B for a reference on how and where each font name is used. Each attribute includes three characteristics in parentheses. Commas separate each characteristic. The first thing we specify is the typeface. The next item is the font size. The final item is the font weight.

The next section of code is very similar to the font style element. Instead of a list of font names, this one is a list of font colors. In this case a replace statement is again used since we're going to replace the entire list. The cryptic color names like 'fg' and 'bg' are used by the style definition to apply these colors to various parts of the output.

```
replace color_list /
  'link' = blue              /* links */
  'bgH' = grayBB             /* row and column header background */
  'fg' = black               /* text color */
  'bg' = white;              /* page background color */
```

The next section of code sets up the style element that controls the page margins.

```
replace Body from Document /
  bottommargin = 0.25in
  topmargin = 0.25in
  rightmargin = 0.25in
  leftmargin = 0.25in;
```

With ODS, it is possible to set different widths for each of the four page margins. In this example, the margins are set to .25 inches, which is the default for RTF output. Unlike the previous sections of code, this one is not a list of names to be used elsewhere. This element lists the actual style attributes and applies settings.

The next section of code sets up the style element that controls rules, borders, and spacing for all tables. Since virtually all ODS output is in the form of tables, this is an important style element.

```
replace Table from Output /
  frame = box        /* outside borders: void, box, above/below, vsides/hsides, lhs/rhs */
  rules = all    /* internal borders: none, all, cols, rows, groups */
  cellpadding = 3pt  /* the space between table cell contents and the cell border */
  cellspacing = 0pt  /* the space between table cells, allows background to show */
  borderwidth = .75pt    /* the width of the borders and rules */;
```

The next sections of code will not be covered in this workshop. It sets up some additional font characteristics that we will not be modifying. In addition to this last section that modifies some style elements, there are dozens of other style statements that are "included" in our style. Those elements are part of the RTF style, and are included by way of the PARENT statement at the beginning of our PROC TEMPLATE. (If you'd like to see the full Default style, issue a PROC TEMPLATE with a single statement: "source styles.RTF;" and a RUN. This will dump the full definition to the log. For the purposes of this workshop, you don't need to understand the last section of code, or the code in the RTF style. We're just going to work with the top parts.

At the end of the example PROC TEMPLATE are two more lines of code. These end the style definition that began with the DEFINE STYLE statement, and run the procedure.

```
    end;
  run;
```

After the PROC TEMPLATE, the example program includes some code to run a sample procedure so we can see what our style looks like. This code starts with some options settings.

```
  options nodate nonumber;
  ods noptitle;
  ods proclabel 'Frequencies';
```

The OPTIONS statement gets rid of dates and page numbers. The first ODS statement turns off the standard procedure titles ("The FREQ Procedure") so they don't clutter up our output. The second ODS statement is used to control the procedure labels in the HTML table of contents. Instead of the default title "The FREQ Procedure", our table of contents will use "Frequencies".
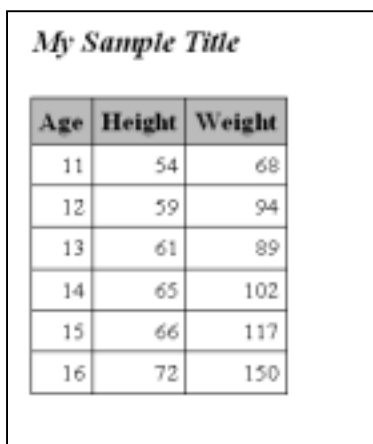
The remaining lines of example code are a simple PROC REPORT, and the ODS statements needed to create RTF output. This example produces web output for ease of review during this workshop. This same code will work for HTML or PDF output as well, with a simple change to the ODS calls before and after the PROC REPORT.

```
  ods rtf file='c:\sample.html' style=Custom;
    title 'My Sample Title';
    footnote 'My Sample Footnote';
    proc report data=sashelp.class nowd;
       column age height weight;
       define age / group;
       define height / mean f=8.;
       define weight / mean f=8.;
    run;
  ods rtf close;
```

The only important thing to note here is the style=Custom option on the ODS statement. This calls our newly created style and applies it to the results. That's it for the sample program. It's a very simple example of customizing a style, but it can be very powerful, as you'll see later.

➢ **RUNNING THE EXAMPLE PROGRAM**

Before going any further, try running this sample program. Open the output file to see how the style looks right now.

If you've used the RTF style before, you'll realize that right now the Custom style doesn't look very different from the default RTF style. The remainder of this workshop will be devoted to customizing the style. Warning: as the workshop proceeds, it is important that you close the RTF file you generate each time. Otherwise, when you re-submit the program to make changes, you'll get an error message about the file being open. An RTF file cannot be regenerated while it is still open in Word.

➢ FIXING THE MARGINS

The first thing we will learn how to modify is the margins. For other ODS destinations, it is possible to modify the margins using the system options TOPMARGIN, BOTTOMMARGIN, LEFTMARGIN, and RIGHTMARGIN. However, for the RTF destination, the margin settings are hard-coded into the default RTF template, and cannot be modified via an OPTIONS statement.

To change the margins, we could use our custom template to pick new hard-coded values. If you plan to always use the same margins, then just edit over the 0.25-inch settings in the PROC TEMPLATE code. For example, to get 1-inch margins at the top and bottom, and 1.25-inch margins on the sides, the code would be:

```
replace Body from Document /
  bottommargin = 1in
  topmargin = 1in
  rightmargin = 1.25in
  leftmargin = 1.25in;
```
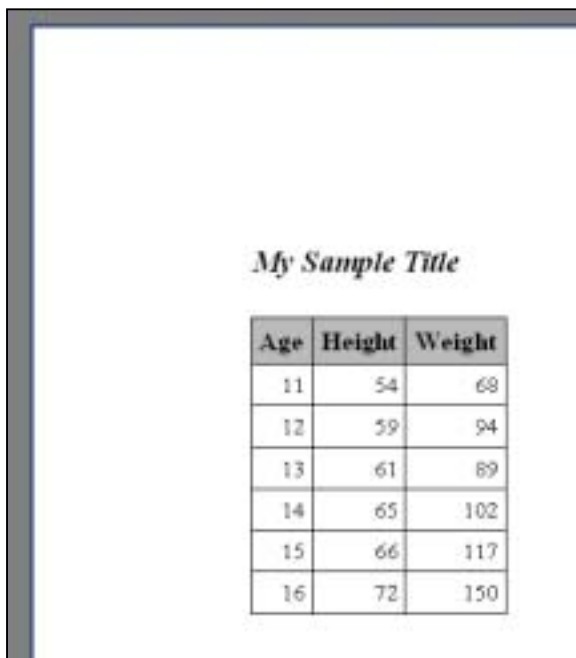
However, if you'd like your style to allow flexibility to set margins differently for each job, then the better solution is to set the margins to undefined in the PROC TEMPLATE, and then set them separately in each SAS program using the system options TOPMARGIN, BOTTOMMARGIN, LEFTMARGIN, and RIGHTMARGIN. The PROC TEMPLATE code for this is:

```
replace Body from Document /
  bottommargin = _undef_
  topmargin = _undef_
  rightmargin = _undef_
  leftmargin = _undef_;
```

With this code in place in the style template, the code to set the margins within the SAS program would be:

```
options  bottommargin = 1in
         topmargin = 1in
         rightmargin = 1.25in
         leftmargin = 1.25in;
```

Using the sample program, try changing the margin settings used in the fonts style element. You can either hard-code new settings, or set them to undefined and use the system options. Before doing this, you may want to save the sample program using a different name so that you can go back if you make a mistake. Try changing the margin settings and then re-running the program. With the new settings shown above, the resulting output is as follows:

➢ FIXING THE TITLES

When you first open your RTF output in Word, you may have noticed that the titles show up in pale gray. However, if you print preview or print the document, they show up in the expected black. The reason for this is that ODS puts the titles into the Word document header, and the footnotes into the Word document footer.

If you don't like the way this looks, or if you need to use the Word header and footer for other titles, then you can turn this behavior off. On the ODS RTF statement, you can add a BODYTITLE option to request that the titles and footnotes be made part of the body of the document. The syntax is as follows:

```
ods rtf file='c:\sample.rtf' style=Custom bodytitle;
```

Try adding the BODYTITLE option to your code and rerunning the program. The resulting output is shown below. It looks similar to the previous output samples, as those were shown in Print Preview mode. However, with this example, the output would look the same even in the normal views.

Notice that in this view, the footnote appears right below the table, instead of at the bottom of the page.

➢ CHANGING THE TYPEFACES

The next thing we will learn how to modify is the typefaces. We'll be working with the fonts style element. To change the font in part of your output, all you have to do is use PROC TEMPLATE to modify the font definition that applies to that part of your output. Appendix B lists each of the font names and where they apply.

For each font name, we can modify three characteristics. The first is the typeface. To make a change, simply replace the typefaces listed between quotes with typefaces of your choice. Keep in mind that the person receiving your output will need to have the same fonts in order to view the RTF file properly. If you are delivering only hard-copy output, then you can use any font available on your system. If you do need to pick fonts that are commonly available, Appendix C lists some good fonts.

Using the sample program, try changing the typefaces used in the fonts style element. View the **RTF** file again to see how the change affected the output. A sample modification:

```
replace fonts /
  'TitleFont' = ("Arial",13pt,Bold Italic)    /* Titles from TITLE statements */
  'TitleFont2' = ("Arial",12pt,Bold Italic) /* Procedure titles ("The _____ Procedure")*/
  'StrongFont' = ("Arial",10pt,Bold)
  'EmphasisFont' = ("Arial",10pt,Italic)
  'headingEmphasisFont' = ("Arial",11pt,Bold Italic)
  'headingFont' = ("Arial",11pt,Bold)        /* Table column and row headings */
  'docFont' = ("Arial",10pt)                 /* Data in table cells */
  'footFont' = ("Arial",13pt)                /* Footnotes from FOOTNOTE statements */
  'FixedEmphasisFont' = ("Courier",9pt,Italic)
  'FixedStrongFont' = ("Courier",9pt,Bold)
  'FixedHeadingFont' = ("Courier",9pt,Bold)
  'BatchFixedFont' = ("Courier",6.7pt)
  'FixedFont' = ("Courier",9pt);
```

The resulting output:

> **CHANGING THE FONT SIZES**

Now that we've got the typefaces we want, we can turn to the font sizes. The default RTF output from ODS uses font sizes ranging from 10 point to 13 point. The default RTF style uses the same font specification for titles and footnotes. Our example style uses a different setting for each, so that you can have large titles and small footnotes. Otherwise, the example style has been set up with fonts similar to those in the default RTF style.

Try making some of the fonts bigger or smaller and see how this affects the output. As a sample modification, the code below reduces the titles to 12 points, and the footnotes to 8 points. Also, the emphasized heading fonts are reduced to 10 points.

```
replace fonts /
  'TitleFont' = ("Arial",12pt,Bold Italic)    /* Titles from TITLE statements */
  'TitleFont2' = ("Arial ",12pt,Bold Italic) /* Procedure titles ("The _____ Procedure")*/
  'StrongFont' = ("Arial ",10pt,Bold)
  'EmphasisFont' = ("Arial ",10pt,Italic)
  'headingEmphasisFont' = ("Arial ",10pt,Bold Italic)
  'headingFont' = ("Arial ",10pt,Bold)        /* Table column and row headings */
  'docFont' = ("Arial ",10pt)                 /* Data in table cells */
  'footFont' = ("Arial ",8pt)                 /* Footnotes from FOOTNOTE statements */
  'FixedEmphasisFont' = ("Courier",9pt,Italic)
  'FixedStrongFont' = ("Courier",9pt,Bold)
  'FixedHeadingFont' = ("Courier",9pt,Bold)
  'BatchFixedFont' = ("Courier",6.7pt)
  'FixedFont' = ("Courier",9pt);
```

The resulting output:

➢ **SPECIAL NOTE: 10 POINT FONTS**

If you are using a version of SAS prior to 9.1, then you will encounter a bug[1] related to font sizes when creating RTF output. If you have any of your font style attributes set to 10 points, they will end up in the Word document as 9.5 points. You can see this if you highlight a portion of the text in the example table.

To fix this, list the font size as 10.1 points in the fonts style element. When you open the RTF file in Word, the fonts will show up as 10 points. In the example program, the fix would be applied like this:

```
replace fonts /
  'TitleFont' = ("Arial",12pt,Bold Italic)    /* Titles from TITLE statements */
  'TitleFont2' = ("Arial ",12pt,Bold Italic) /* Procedure titles ("The _____ Procedure")*/
  'StrongFont' = ("Arial ",10.1pt,Bold)
  'EmphasisFont' = ("Arial ",10.1pt,Italic)
  'headingEmphasisFont' = ("Arial ",10.1pt,Bold Italic)
  'headingFont' = ("Arial ",10.1pt,Bold)         /* Table column and row headings */
  'docFont' = ("Arial ",10.1pt)                  /* Data in table cells */
  'footFont' = ("Arial ",8pt)                 /* Footnotes from FOOTNOTE statements */
  'FixedEmphasisFont' = ("Courier",9pt,Italic)
  'FixedStrongFont' = ("Courier",9pt,Bold)
  'FixedHeadingFont' = ("Courier",9pt,Bold)
  'BatchFixedFont' = ("Courier",6.7pt)
  'FixedFont' = ("Courier",9pt);
```

And the resulting output would look much the same. However, if you highlight the text, you will see that the fonts are now 10 point instead of 9.5 point.

➢ **CHANGING THE WEIGHTS AND STYLES**

The other thing you can change about fonts is the font weight (Medium, Bold, Light) and the font style (Italic, Roman, Slant). You may also be able to control the font width, though few fonts honor settings like Compressed or Expanded. To use any of these settings, just list the appropriate keyword(s) after the font size specification. Generally, the only two settings that you'll want to add are Bold and/or Italic. If you leave this setting blank, the fonts are set to Medium Roman.

---

[1] See SAS Note SN-006436 for more information.

Try changing some of these settings to see what happens. A sample modification is shown below. This code removes the italics from the titles, and adds italics to the footnotes.

```
replace fonts /
  'TitleFont' = ("Arial",12pt,Bold)    /* Titles from TITLE statements */
  'TitleFont2' = ("Arial ",12pt,Bold) /* Procedure titles ("The _____ Procedure")*/
  'StrongFont' = ("Arial ",10pt,Bold)
  'EmphasisFont' = ("Arial ",10pt,Italic)
  'headingEmphasisFont' = ("Arial ",10pt,Bold Italic)
  'headingFont' = ("Arial ",10pt,Bold)       /* Table column and row headings */
  'docFont' = ("Arial ",10pt)                /* Data in table cells */
  'footFont' = ("Arial ",8pt, Italic)              /* Footnotes from FOOTNOTE statements */
  'FixedEmphasisFont' = ("Courier",9pt,Italic)
  'FixedStrongFont' = ("Courier",9pt,Bold)
  'FixedHeadingFont' = ("Courier",9pt,Bold)
  'BatchFixedFont' = ("Courier",6.7pt)
  'FixedFont' = ("Courier",9pt);
```

The resulting output:

My Sample Title

Age	Height	Weight
11	54	68
12	59	94
13	61	89
14	65	102
15	66	117
16	72	150

My Sample Footnote

➢ **CHANGING THE TABLE RULES AND BORDERS**

The next thing we will modify is the table rules and borders. The lines around the table and between rows and columns are controlled by two style attributes: rules and frame.

The frame attribute specifies whether there will be any lines around the outside of your tables. The frame is currently set to box, which means there will be a line around the entire table. Another setting to try is void, which removes all of the borders around the table. There are also settings that let you have borders top and bottom, both sides, or any individual edge.

Try changing the frame setting. Don't worry about the line width right now; we'll get to that later. A sample modification:

```
replace Table from Output /
  frame = hsides         /* outside borders: void, box, above/below, vsides/hsides, lhs/rhs */
  rules = all       /* internal borders: none, all, cols, rows, groups */
  cellpadding = 3pt     /* the space between table cell contents and the cell border */
  cellspacing = 0pt     /* the space between table cells, allows background to show */
  borderwidth = .75pt       /* the width of the borders and rules */;
```

8

The resulting output is below. Notice that now the frame is only at the top and bottom of the table. The sides have no borders.

The rules attribute controls the lines that appear inside your tables. This attribute is currently set to none, so there are no lines at all. Other settings to try are all and group. All turns on all possible lines, creating a table grid. Groups puts a border between row and column headers and footers and the rest of the table body. Other settings include rows and cols, which include only row dividers or column dividers.

Try changing the rules setting. You may also want to experiment with combinations of frame and rules settings. A sample modification:

```
replace Table from Output /
  frame = hsides         /* outside borders: void, box, above/below, vsides/hsides, lhs/rhs */
  rules = groups         /* internal borders: none, all, cols, rows, groups */
  cellpadding = 3pt      /* the space between table cell contents and the cell border */
  cellspacing = 0pt      /* the space between table cells, allows background to show */
  borderwidth = .75pt       /* the width of the borders and rules */;
```

The resulting output:

Now that you have all of the lines you want, we can look at the width for those lines. This is controlled by the borderwidth attribute. Don't forget that changing this setting will not have any effect unless you've specified some lines for your table. With frame=void and rules=none, the border width is irrelevant.

Borderwidth is simply the line width. It affects the width of the table border, but not the rules. Use a number followed by "pt" to set the width in points. Try experimenting with the border width. If your custom style will not have any lines, go ahead and turn on frame=box and rules=all so that you can at least see how it works. You can reset frame and rules later.

A sample modification:

```
replace Table from Output /
  frame = hsides         /* outside borders: void, box, above/below, vsides/hsides, lhs/rhs */
  rules = groups         /* internal borders: none, all, cols, rows, groups */
  cellpadding = 3pt      /* the space between table cell contents and the cell border */
  cellspacing = 0pt      /* the space between table cells, allows background to show */
  borderwidth = 2pt      /* the width of the borders and rules */;
```

The resulting output:

> **CHANGING THE TABLE SPACING**

The final thing we will modify is the table spacing. This is the amount of space that is left between table cell contents (your results) and the top, bottom, left, and right sides of the cell. To make your table readable, you want a large value. But to squeeze more information on the page, you probably want a smaller value. The attribute that controls this spacing is called cellpadding. The example program uses a value of 3, which is about as small as you can go without losing readability. Experiment with various values to see what you like. One thing to note here is that you have to have the same amount of space on all sides. A sample modification:

```
replace Table from Output /
  frame = hsides         /* outside borders: void, box, above/below, vsides/hsides, lhs/rhs */
  rules = groups         /* internal borders: none, all, cols, rows, groups */
  cellpadding = 5pt      /* the space between table cell contents and the cell border */
  cellspacing = 0pt      /* the space between table cells, allows background to show */
  borderwidth = 2pt      /* the width of the borders and rules */;
```

The resulting output is shown below. If you look closely, you can see the change from the previous output.

➢ **CHANGING THE COLORS**

Changes to the fonts and lines are fairly subtle. This next section lets you make big bold changes to your output. This section considers the color scheme.

ODS allows you to set the foreground (text) colors and background colors of every part of your output. These colors are set by defining a color scheme in the colors style element.

In the example program, each color is identified by name. Appendix D lists the color names you can use. This gives you a palette of 216 colors. This is a list of web-safe colors.

You also have the option of specifying custom colors by using their RGB values given in hexadecimal. For example, white would be cxFFFFFF, and black would be cx000000 (the "cx" tells SAS that the following value is a hexadecimal color). For the purposes of this workshop, let's stick to the named colors.

When you modify these colors, notice that some of the names start in "fg" and represent foreground colors. Others start in "bg" and represent background colors. These colors work in pairs, and you need to be sure that you pick pairs of colors that will be readable. For example, pink foreground text on a red background would be a problem.

Try creating a new color scheme. See how it looks. A sample modification:

```
replace color_list /
  'link' = blue            /* links */
  'bgH' = lime             /* row and column header background */
  'fg' = black             /* text color */
  'bg' = white;            /* page background color */;
```

The resulting output:

This example modification uses color. If your company primarily produces black and white output, then you may wish to limit your choices to black, white, and various shades of gray. Here's another sample modification:

```
replace color_list /
  'link' = blue            /* links */
  'bgH' = white            /* row and column header background */
  'fg' = black             /* text color */
  'bg' = white;            /* page background color */;
```

11

The resulting output:

My Sample Title

Age	Height	Weight
11	54	68
12	59	94
13	61	89
14	65	102
15	66	117
16	72	150

My Sample Footnote

If you're not very creative, there's a web site that will help you design an attractive color scheme. Go to
http://www.colorschemer.com/online/ and click on a color that you like. The web site will generate a group of 16 related
colors that create an attractive color scheme. You can then copy down the hex codes for these colors and use them in your
style. Another way to pick colors for your scheme is to use colors from your corporate logo. Ask your graphics department
for the correct color codes. They should be able to give you the RGB values (you can find an RGB/hex converter on the
web).

➢ **RTF TWEAKS: DECIMAL ALIGNMENT**

One of the advantages of producing RTF output is that you can customize your output by inserting RTF tags. For example, in
version 9, SAS added support for decimal alignment in table columns[2]. However, you can get this same result in version 8 by
using RTF tags. To see how this works, we need to create a table with varying decimal places. In the sample program,
changing the formats to best6. will accomplish this. The table now looks like this:

My Sample Title

Age	Height	Weight
11	54.4	67.75
12	59.44	94.4
13	61.433	88.667
14	64.9	101.88
15	65.625	117.38
16	72	150

My Sample Footnote

To use RTF tags to create decimal alignment for a column, we need to add a STYLE option to the DEFINE statement for that
column.

```
proc report data=sashelp.class nowd;
   column age height weight;
   define age / group;
   define height / mean f=best6. style(column)=[protectspecialchars=off
                                                pretext="\qj\tqdec\tx500 "
                                                cellwidth=.75in];
   define weight / mean f=best6. style(column)=[protectspecialchars=off
                                                pretext="\qj\tqdec\tx500 "
                                                cellwidth=.75in];
run;
```

---

[2] To add decimal alignment in version 9, add the following code to the end of each column's DEFINE statement: `style(column)=[just=d];`

There are three parameters that need to be set. First, PROTECTSPECIALCHARS is set to OFF. This tells SAS to ignore the special RTF tags and pass them through to the output file. Then, the PRETEXT parameter is set to the RTF tags that generate a decimal-aligned tab. The "tx500" refers to a distance of 500 twips[3]: Depending on the number of decimal places you want to display, and the size of your numbers, this value may need to be adjusted. The rest of the RTF tag does not need to be modified. It is important to get the tag entered exactly as shown, including the blank space at the end. The final parameter in the STYLE option is the cell width. This is set to .75 inches, which allows enough room for the numbers to fit without wrapping. You'll find that you need to experiment a bit with the tab location (in twips) and the cell width (in inches).

Here is the revised output:

This is just one simple example of how you can use RTF tags to enhance your output. In the Resources section at the end of this paper, there is a link to the full RTF specification, which lists all of the RTF tags. If the full specification is too confusing, another way to find the RTF tags you need is to create a Word document that contains only the feature you're trying to replicate. Save the file as RTF, and then open the file in Notepad. The RTF files are large, but if you look at a version before and after the change you're trying to make, you should be able to spot the tag you need.

Other things you can do with RTF tags include: inserting Word symbols, applying Word styles, adding a hyperlink, or creating a custom table of contents.

➢ RTF TWEAKS: PAGE NUMBERS

By default, ODS puts a large page number at the top right of each page of your output. Using the example program, the page number looks like this:

---

[3] A twip is a unit of measurement used by Word. There are 1440 twips to the inch.

This page number is built in, and there is no way to remove it. However, you can make it invisible. To do this, we need to add another style element to our PROC TEMPLATE. The following code can be added anywhere in the style definition before the END statement.

```
style PageNo from PageNo/
      font_size=0.1pt
      background=white
      foreground=white;
```

This code reduces the size of the page number to a tiny 0.1 points, and then colors the text and the background both to white. The number is still there in your output, but it's tiny and invisible.

You can choose to leave your output without page numbers. Or, once you've hidden the standard page number, you can create a custom page number in the format and location you prefer. For example, to create a centered page number at the bottom of the page in the format "Page X of Y", the code to add is:

```
style SystemFooter from SystemFooter /
    font = fonts("footFont")
    protectspecialchars=off
    posttext='\par page }{\field{\*\fldinst {\cs17  PAGE }}{\fldrslt
{\cs17\lang1024 1}}}{\cs17  of }{\field{\*\fldinst {\cs17
NUMPAGES }}{\fldrslt {\cs17\lang1024 1}}}{\cs17  ';
```

The long string for the POSTTEXT attribute is the RTF tags needed to generate the page numbers. To use this code you have to get the tags typed in exactly right, including all of the spaces. Also, there cannot be any returns in the string, the text just wraps from line to line.

The other thing you need to add is to request that the footnote be centered.

```
footnote j=c 'My Sample Footnote';
```

Here is the resulting page footer:

Notice that both the footnote and the page number are centered. With page numbering like this, you may prefer to separate the two, putting the footnote directly below the table, and leaving only the page number in the footer. To do that, remove the footnote text from the FOOTNOTE statement:

```
footnote j=c ' ';
```

And then add the following line of code between the RUN statement for the PROC REPORT and the ODS RTF CLOSE statement. This

```
ods rtf text='{\i\fs18 My Sample Footnote }';
```

This inserts the text and RTF tags right below the table. The RTF tags request an italicized 9-point font. Here is the new footnote:

And here is the page number at the bottom:

➢ RTF TWEAKS: PAGE BREAKS

By default, ODS puts a page break after each procedure's output. If you run two PROC REPORTs, the RTF file will have the first table, then a page break, and then the second table. For small tables, this can waste a lot of space. You can prevent the page breaks from being generated by adding a STARTPAGE option. The syntax is:

```
ods rtf startpage=never;
```

This code should be added between the two PROC REPORT calls. You can try this in the example program by copying the PROC REPORT code so it runs twice, and then inserting the STARTPAGE call between the two. The new output will have the two tables on the same page.

To turn page breaks back on, you can issue another statement with STARTPAGE=YES. Or, to insert a single page break but keep the automatic breaks turned off, use STARTPAGE=NOW.

➤ SAVING YOUR STYLE

Once you've created your custom style, you can save the program that creates the style. This will allow you to regenerate it at any time. But you don't need to run this PROC TEMPLATE every time you want to use your new style. SAS has saved the style for you in the sasuser library.

If this style is for you alone, this will work just fine. But if you want to share your style, you will need to make a couple of changes. First, set up a libname for your custom style in a commonly accessible area. Then, you'll need to learn about the ODS PATH statement, which you can use to route your custom style to this libname. Other users can set up the same ODS PATH statement in their programs to reference this libname and access your style.

➤ CONCLUSIONS

This workshop has been a short cut to using some basic style functionality. If you just need to quickly modify a style, this may be enough.

However, this template only allows you to modify certain aspects of your output. You may find that you want to control other aspects. To do that, you're going to have to learn a lot more about PROC TEMPLATE syntax.

➢ **RESOURCES**

PROC TEMPLATE documentation is in the References chapter of:

Guide to the Output Delivery System in SAS Online Doc, version 8, ©1999, SAS Institute Inc., Cary, NC, USA.

Preliminary documentation of new features and sample programs can be found at:

http://www.sas.com/rnd/base/
index-ods-resources.html.

Documentation of the RTF Specifications (useful for looking up RTF tags):

http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnrtfspec/html/rtfspec.asp

My book on ODS has a number of chapters on modifying ODS styles:

Haworth, Lauren, *Output Delivery System: The Basics*, ©2001, SAS Institute Inc., Cary, NC, USA.

➢ **ACKNOWLEDGEMENTS**

SAS is a registered trademark or trademark of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

➢ **CONTACTING THE AUTHOR**

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

APPENDIX A

```
proc template;

  define style Styles.Custom;

  parent = Styles.RTF;

    replace fonts /
        'TitleFont' = ("Times Roman",13pt,Bold Italic)   /* Titles from TITLE statements */
        'TitleFont2' = ("Times Roman",12pt,Bold Italic) /* Procedure titles ("The _____ Procedure")*/
        'StrongFont' = ("Times Roman",10pt,Bold)
        'EmphasisFont' = ("Times Roman",10pt,Italic)
        'headingEmphasisFont' = ("Times Roman",11pt,Bold Italic)
        'headingFont' = ("Times Roman",11pt,Bold)        /* Table column and row headings */
        'docFont' = ("Times Roman",10pt)                 /* Data in table cells */
        'footFont' = ("Times Roman",13pt)                /* Footnotes from FOOTNOTE statements */
        'FixedEmphasisFont' = ("Courier",9pt,Italic)
        'FixedStrongFont' = ("Courier",9pt,Bold)
        'FixedHeadingFont' = ("Courier",9pt,Bold)
        'BatchFixedFont' = ("Courier",6.7pt)
        'FixedFont' = ("Courier",9pt);

    replace color_list /
        'link' = blue               /* links */
        'bgH' = grayBB              /* row and column header background */
        'fg' = black               /* text color */
        'bg' = white;              /* page background color */;

    replace Body from Document /
        bottommargin = 0.25in
        topmargin = 0.25in
        rightmargin = 0.25in
        leftmargin = 0.25in;

    replace Table from Output /
        frame = box         /* outside borders: void, box, above/below, vsides/hsides, lhs/rhs */
        rules = all    /* internal borders: none, all, cols, rows, groups */
        cellpadding = 3pt  /* the space between table cell contents and the cell border */
        cellspacing = 0pt  /* the space between table cells, allows background to show */
        borderwidth = .75pt    /* the width of the borders and rules */;

    * Leave code below this line alone ;
    style SystemFooter from SystemFooter /
        font = fonts("footFont");

  end;

run;

options nodate nonumber;
ods noptitle;
ods proclabel 'Frequencies';
ods rtf file='c:\sample.html' style=Custom;
  title 'My Sample Title';
  footnote 'My Sample Footnote';
  proc report data=sashelp.class nowd;
    column age height weight;
    define age / group;
    define height / mean f=8.;
    define weight / mean f=8.;
  run;
ods rtf close;
```

18

APPENDIX B

Font Style	Portion of Output it Controls
TitleFont	Titles generated with TITLE statement
TitleFont2	Titles for procedures ("The _____ Procedure")
StrongFont	Strong (more emphasized) table headings and footers, page numbers
EmphasisFont	Titles for table of contents and table of pages, emphasized table headings and footers
headingFont	Table column and row headings and footers, by-group headings
docFont	Data in table cells
footFont	Footnotes generated with FOOTNOTE statement

APPENDIX C

"Safe" fonts[4,5]
Times Roman
Arial
Arial Black
Book Antigua
Comic Sans MS
Verdana
Impact
Georgia
News Gothic MT
Tahoma
Trebuchet MS

---

[4] This list is based on standard Windows fonts. If you have a lot of Mac users, you may want to use Mac fonts like Chicago, Geneva, Helvetica, Monaco, New York, Times and Palatino as alternatives.
[5] Two very unsafe fonts are SAS Monospace and SAS Monospace Bold.

APPENDIX D

White	Cornsilk	Antiquewhite	Seashell	Linen	Ivory	Floralwhite
Snow	Azure	Mintcream	Ghostwhite	Honeydew	Aliceblue	Beige
Oldlace	Bisque	Moccasin	Wheat	Navajowhite	Blanchedalmond	Tan
Gray	Lightgrey	Darkgray	Dimgray	Gainsboro	Silver	Whitesmoke
Black	Darkslategray	Slategray	Lightslategray	Lemonchiffon	Khaki	Darkkhaki
Brown	Sienna	Chocolate	Saddlebrown	Sandybrown	Burlywood	Peru
Red	Tomato	Darkred	Indianred	Mistyrose	Lavenderblush	Firebrick
Crimson	Maroon	Peachpuff	Goldenrod	Darkgoldenrod	Palegoldenrod	Lavender
Orange	Darkorange	Orangered	Forestgreen	Greenyellow	Lime	Lightgoldenrodyellow
Yellow	Lightyellow	Gold	Springgreen	Darkolivegreen	Olive	Limegreen
Green	Lightgreen	Darkgreen	Mediumseagreen	Mediumspringgreen	Palegreen	Olivedrab
Lawngreen	Chartreuse	Yellowgreen	Paleturquoise	Darkseagreen	Aquamarine	Mediumaquamarine
Teal	Lightseagreen	Seagreen	Darkblue	Mediumturquoise	Turquoise	Darkturquoise
Darkcyan	Cyan	Lightcyan	Mediumslateblue	Lightskyblue	Skyblue	Deepskyblue
Blue	Lightblue	Mediumblue	Steelblue	Darkslateblue	Powderblue	Cornflowerblue
Royalblue	Dodgerblue	Slateblue	Plum	Cadetblue	Mediumorchid	Darkorchid
Navy	Midnightblue	Lightsteelblue	Mediumvioletred	Orchid	Thistle	Rosybrown
Purple	Mediumpurple	Indigo	Fuchsia	Palevioletred	Magenta	Darkmagenta
Violet	Darkviolet	Blueviolet	Salmon	Deeppink	Coral	Lightcoral
Pink	Lightpink	Hotpink	Lightsalmon	Darksalmon		