PhUSE 2011

Paper IS04

How to Create a Custom Style

Sonia Extremera, PharmaMar, Madrid, Spain
Antonio Nieto, PharmaMar, Madrid, Spain
Javier Gómez, PharmaMar, Madrid, Spain

ABSTRACT

SAS® provide us with a wide range of default styles, however if you cannot find the proper style to satisfy your necessities or if you need to make your SAS reports coincide with your corporate style, you can do it easily using the SAS Template procedure. Proc Template styles give you the power to customize the look of your reports, changing fonts, borders, colors, setting margins, or even adding your corporate logo.

In this paper we will walk through the Template procedure showing, step by step, how to redefine the default style elements and attributes to customize fonts, margins or colors to suit your company style. Also, we will show how to save your new style in the sasuser library or how to share it using the ODS PATH statement. All options and statements shown in the paper apply to ODS destinations, such as ODS RTF or ODS PDF.

INTRODUCTION

When you create a report using the SAS Output Delivery System (ODS), a style template is applied by SAS to control how the results are shown. This includes, but is not limited to, the borders, colors or fonts. SAS has a style template by default which will be used for each output destination.

Before constructing a new style template, all available templates included in the installation can be browsed in order to choose that style which matches better with your company's standards or necessities as the model for the customized style. With the following SAS macro, an example of every style will be stored for the ODS destination used:

```
%macro templates (ods_destination,file);
 goptions reset=all;
 title;footnote;

 data _null_;
  set sashelp.vstyle;
  length name $20.;
  name=tranwrd(style,'.','_');
  call execute ("ODS &ods_destination
file='&file\"||compress(name)||".&ods_destination' style="||compress(style)||";");
  call execute ("title '"||style||"';") ;
  call execute ("Proc contents data=sashelp.lake short;");
  call execute ("ODS &ods_destination close;");
run;
footnote 'Biostatistics department <www.pharmamar.com>';

%mend templates;

*HTML destination;
%templates (HTML,C:\Phuse\SAS templates\HTML);

*PDF destination;
%templates (PDF,C:\Phuse\SAS templates\PDF);

*RTF destination;
%templates (RTF,C:\Phuse\SAS templates\RTF);
```

In the present paper, we will start by working through a series of examples using the HTML destination, however all the techniques shown here will work with all destination that support styles such as RTF or PDF. Moreover, each SAS

procedure has a default template for each output created. The examples exposed below work with general style elements, nevertheless specific style elements can exist for every output depending on the SAS procedure used.

Once the style that fits better with our company's standards has been chosen, we will learn how to display the source code and how to redefine the style elements and attributes to customize fonts, margins, sizes, data justification or colors to suit the desired style. After that, the custom style and template will be saved to a permanent template store location for repeated utilization and to share it with other people.

REDEFINE A STYLE TEMPLATE

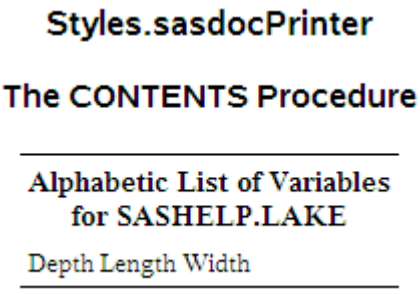
After having chosen the template style that matches better with our necessities, we can modify and customize it using the Proc Template.

HOW TO GET THE TEMPLATE SCRIPT

To get the source code we can either type ODSTEMPLATES into the command bar to open the Template Browser or use the Proc Template procedure. For the first option, double-clicking a template displays its source code. The styles supplied by SAS are located in the styles directory of SASHELP.TMPLMST store. By using Proc Template programming, all templates in styles directory will be listed with the following source code:

```
Proc template;
list styles;
run;
```

For our example, Styles.sasdocPrinter will be selected as default style.



Using the following script, without the File option, the selected template code will be displayed to the log. However, if it is preferred to get the code as a SAS file, the route and name of the file should be specified and the body of the Proc Template will be stored in a SAS code file, therefore Proc Template and Run have to be added at the beginning and end, respectively, of the program:

```
Proc template;
source styles.sasdocPrinter /file='C:\Phuse\SAS templates\ejem1.sas';
run;
```

A long list of source of selected style definitions will be produced. If you skim through the script, you will see that a number of styles are defined such as fonts, tables or headers and footers. Using this code, you will be able to change the attributes of every style in order to make it coincide with your company's standards.

The different parts of the Proc Template have been identified in the script below in order to make it more comprehensible.

```
proc template;
 1 define style Styles.sasdocPrinter;
   2 parent = styles.Printer;
   3 style fonts /
        'TitleFont2' = ("<MTsans-serif>, Helvetica",12pt,bold)
        'TitleFont' = ("<MTsans-serif>, Helvetica",13pt,bold)
        'StrongFont' = ("ITC Bookman, <MTserif>, Times Roman",10pt,bold)
        'EmphasisFont' = ("ITC Bookman, <MTserif>, Times Roman",10pt,italic)
        'FixedEmphasisFont' = ("<MTmonospace>, Courier",9pt,italic)
     4 'FixedStrongFont' = ("<MTmonospace>, Courier",9pt,bold)
        'FixedHeadingFont' = ("<MTmonospace>, Courier",9pt,bold)
        'BatchFixedFont' = ("SAS Monospace, <MTmonospace>, Courier",7pt)
        'FixedFont' = ("<MTmonospace>, Courier",9pt)
        'headingEmphasisFont' = ("ITC Bookman, <MTserif>, Times Roman",11pt,
        bold italic)
```

```
        'headingFont' = ("ITC Bookman, <MTserif>, Times Roman",11pt,bold)
        'docFont' = ("ITC Bookman, <MTserif>, Times Roman",10pt);
    style GraphFonts /
        'GraphDataFont' = ("ITC Bookman, <MTserif>",8pt)
        'GraphUnicodeFont' = ("<MTserif-unicode>",9pt)
        'GraphValueFont' = ("ITC Bookman, <MTserif>",10pt)
        'GraphLabelFont' = ("ITC Bookman, <MTserif>",11pt,bold)
        'GraphFootnoteFont' = ("ITC Bookman, <MTserif>",11pt,bold)
        'GraphTitleFont' = ("ITC Bookman, <MTserif>",12pt,bold)
        'GraphAnnoFont' = ("ITC Bookman, <MTserif>",10pt);
  5 style Table from Output /
        bordercollapse = separate
        backgroundcolor = _undef_
        frame = HSIDES
        padding = 4pt
        borderspacing = 0.75pt
        borderwidth = 0.75pt;
    style HeadersAndFooters from Cell /
        font = fonts('HeadingFont');
  end;
run;
```

STYLE TEMPLATE: BASIC STRUCTURE

Looking at the above code, the following parts have been identified with numbers:

1 It shows the name of the style. If a new style is wanted to be created by modifying the existing style, the style can be renamed here.

2 It specifies a style definition from which style elements are inherited. In this example, the parent style is Styles.Printer. If the template code of Styles.Printer is displayed, you will see it inherits its style elements from Styles.Default.

3 It shows the statements. In this case, it defines a style element with name Font.

4 Attributes for the style element. The attributes from Styles.Printer are also included although they are not shown.

5 This means the style element Table inherits its properties from another style element, Output, however Output element has not been previously defined in this proc template procedure. This is because it may be defined either in the parent style Styles.Printer or in the parent of the parent style Styles.Default.

Therefore, the basic structure of a style template is:

```
Proc template;
 define style style-name;
  <parent=parent-style;>
  ...Statements / Attributes...
run;
```

Where the commonly used statements are:

- **STYLE Statement**: Creates or modifies one or more style elements.
  `style style-element <from parent-style-element> / style-attributes;`
  Define a style element with name *style-element* and attributes *style-attributes*. If specified, the attributes from *parent-style-element* are also included. Attributes of the same name specified in the STYLE statement will override those in the parent.
- **CLASS Statement**: Creates a style element from a like-named style element.
  `class style-element / style-attributes;`
  Define a style element with the name style-element and parent of style-element. This is a shortcut for "style *style-element* from *style-element* / *style-attributes*".

A style element is a collection of style attributes that applies to a particular part of the output. Multiple style elements can be defined simultaneously, by giving a comma-delimited list of style elements in the STYLE/CLASS statement. The main style elements used to modify a style template are:

- **TitlesandFooters:** Controls system page title text and system page footer text.
- **Body:** Controls the body of the output.
- **Font:** Controls the fonts.
- **Colors:** Controls the colors.

- **Table:** Controls overall table style.
- **Cell:** Controls data, header, and footer cells.
- **HeadersAndFooters:** Controls table headers and footers.

For further details, see
http://support.sas.com/documentation/cdl/en/odsug/61723/HTML/default/viewer.htm#a002685136.htm.

A style attribute is a name-value pair that describes a single behavioral or visual aspect of a piece of output. The more commonly used style attributes are:

- **General style attributes:**
  - **Backgroundcolor = *color-name* | *hex-color*:** Specifies the background color of a region.
  - **Backgroundimage = "*path-to-file*":** Specifies an image to put in the background.
  - **Color = *color-name* | *hex-color*:** Specifies the color of the text content.
  - **Height = *dimension* / Width = *dimension*:** Specifies the width and height of an element.
  - **Preimage = "*path-to-file*" / Postimage = "*path-to-file*":** Specifies an image to put before/after the content.
  - **Pretext = "*text*" / Posttext = "*text*":** Specifies text to put before/after the content.
- **Font and text style attributes:**
  - **Fontfamily = "fontfamily-1, fontfamily-2, …":** Specifies a list of font family alternatives for the text content such as Times, Courier, Arial or Helvetica. The first usable font found on the system will be used.
  - **Fontsize = *dimension*:** Specifies the size of the text.
  - **Fontstyle = italic | roman | slant:** Specifies the style of the text.
  - **Fontweight = bold | medium | light:** Specifies whether the text should be bold or not.
- **Table style attributes:**
  - **Borderspacing = *dimension* / Cellspacing = *dimension*:** Specifies the amount of space to put between adjacent table cells.
  - **Frame = box | above | below | hsides | vsides | lhs | rhs | void:** Specifies which borders should appear within the table.
  - **Padding = *dimension* / Cellpadding = *dimension*:** Specifies the amount of space to put between the content and border of the cells.
  - **Rules = all | cols | rows | groups | none:** Specifies which borders should appear within the table.
- **Border style attributes:**
  - **Bordercolor = *color-name* | *hex-color*:** Specifies the color of the border on all sides.
  - **Borderstyle = dashed | dotted | double | groove | hidden | inset | outset | ridge | solid | none:** Specifies the style of the border on all sides.
  - **Borderwidth = *dimension*:** Specifies the width of the border on all sides.

For example, your corporate logo can be added. Using the attribute preimage="path-to-file" within the element Body.

Logo example

	N	*Median*	*Min*	*Max*
Width	315	5.00	0.00	10.00
Length	315	3.50	0.00	7.00
Depth	315	-0.02	-35.39	0.00

PharmaMar Biostatistics department

HOW TO EXPLORE THE OUTPUT ELEMENTS USING ODS MARKUP DESTINATION

There is extensive documentation about styles and attributes in the SAS support web, however, it is easy to get lost with such a number of classes and attributes. Nevertheless, there is a simple way to identify the style elements of an output and to make modifications to style template, thanks to ODS MARKUP destination.
The following code gives you an easy way to see which bits of code control which parts of your output:

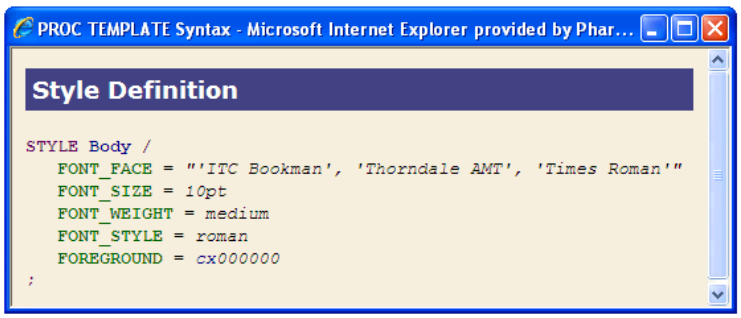
```
ods markup file='example.html' tagset=tagsets.style_popup  style=Styles.sasdocPrinter;
proc tabulate data=sashelp.class;
  var height weight age;
  table (height weight age),(n median min max)/condense ;
  title 'Markup example';
  footnote 'PharmaMar Biostatistics department';
 run;
ods markup close;
```

This code creates Markup example.html output. When you hover over part of the output, its background changes to red color and a label shows the name of the style element that controls that attribute, thanks to the tagset destination Style_popup. In addition, when you click on a part of the output, the actual source code for that style element pops up, but this feature only works with Microsoft Internet Explorer.

Markup example

	N	Median	Min	Max
Width	315	5.00	0.00	10.00
Length	315	3.50	0.00	7.00
Depth	315	-0.02	-35.39	0.00

PharmaMar Biostatistics department



Looking at the figure above, we know that the text in this cell is controlled by the style element Header. If the purpose is to change the header properties, we know what style element we need to modify in the proc template. However, in some occasions, the style element is not displayed in the proc template code. This is because it is inheriting its style attributes and values from other style element or other style parent. In that case, you can use the code in the popup window to create a new custom style. For example, if you prefer the header in italic, you can modify the template by doing the following:

```
Proc template;
define style PhMstyle;
 parent=styles.sasdocPrinter;
   STYLE Header /
       FONT_FACE = "'ITC Bookman', 'Thorndale AMT', 'Times Roman'"
       FONT_SIZE = 11pt
        FONT_WEIGHT = bold
        FONT_STYLE = italic
        FOREGROUND = cx000000;
end;
run;
```

Now, if you apply your new style PhMstyle to your ODS destination, you will see how the Header has been modified. You can do the same with the rest of elements of the style and modify then according to your necessities.

The Style_Popup Tagset destination shows you only the proc tabulate output which was required (in our example: table, title and footnote) but there are many other elements that are not shown up in your output based on the job that you run. Nevertheless, if tagset destination Style_Display is used instead of Style_popup, a sample output for all of the commonly defined elements is created. First of all, the proc tabulate output will be displayed as it was done by the tagset Style_popup destination. After this, the rest of sample outputs are shown. It will help you to discover new style elements that can be useful for your template definition.

SAVE A NEW STYLE AND SHARE IT USING THE ODS PATH STATEMENT

After you edit the template definition, you can submit your PROC TEMPLATE statements as you would any other SAS program. ODS automatically saves the compiled template in the first template library that it can update. By default, it is stored in the library SASUSER.Templat. The style templates provided by SAS are stored in the library SASHELP.Tmplmst. ODS destination searches SASUSER.Templat for templates, and then, if it does not find the requested template in SASUSER.Templat, it searches SASHELP.Tmplmst.

You can see the list of template libraries by submitting the following statement:

```
ods path show;
```

The results are shown in Log screen:

Current ODS PATH list is:

1. SASUSER.TEMPLAT(UPDATE)
2. SASHELP.TMPLMST(READ)

If you want to store modified templates in another template library, you can change the ODS PATH to add template libraries that you have previously created or to change the order in which the libraries are searched. ODS uses the first template that it finds with the requested name. Templates with the same name can exist in more than one template library.

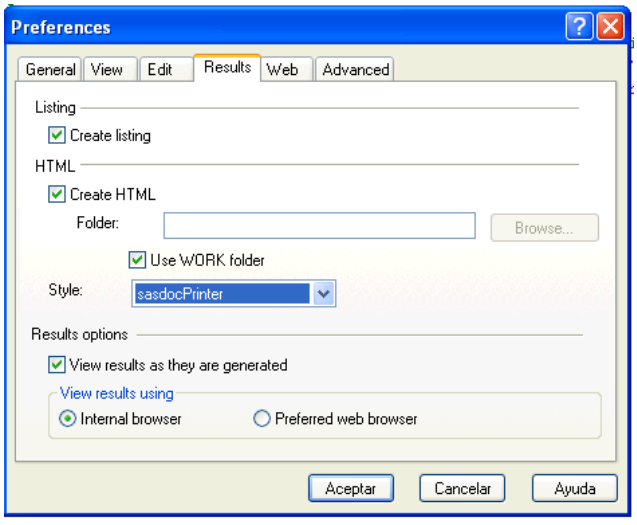
For example, the following statement sets the ODS path so that the template library template.Mytemplate is searched first, followed by SASHELP.Tmplmst:

```
libname template 'C:\Phuse\SAS templates';
ods path template.Mytemplate(update) sashelp.tmplmst(read);
```

The UPDATE option provides update access as well as read access to TEMPLATE.Mytemplate. The READ option provides read-only access to SASHELP.Tmplmst. With this path, the template library SASUSER.Templat is no longer searched.

Therefore, if the new style template is needed to be shared, it must be stored in a shared location. All users must define the SAS library Template in the shared location, with the libname statement. Then, the order in the ODS PATH statement must be changed for all users, setting in the first place, the shared library.

As a useful tip, it is recommended to use an existing name for your new style template and to modify the SAS preferences selecting from any SAS window, Tools > Options > Preferences. Once the Preferences menu is displayed, select the Results tab and scroll through the Style list provided by SAS to select the name of your new style.



CONCLUSION

Proc Template is a powerful tool to customize the look of your reports. In this paper, you have seen how to browse the SAS supplied templates, how to display the template code and how to modify the attributes of the template elements. In addition, ODS markup destination has been presented in order to help you to identify each template element. The way to save and share it, has been also shown. The only thing left is to put into practice all this knowledge and create your own templates.

REFERENCES

SAS OnlineDoc.
Lauren Haworth, “Proc Template: The Basics”, Genentech, Inc., South San Francisco, CA.
Eric Gebhart, “ODS Markup, Tagsets, and Styles! Taming ODS Styles and Tagsets”, SAS Institute Inc., Cary, NC.

CONTACT INFORMATION

Your comments and questions are valued and encouraged.

Sonia Extremera Tenaguillo
Clinical Development. PharmaMar S.A.
Avda. de los Reyes, 1
Polígono Industrial La Mina
28770 Colmenar Viejo. Madrid (SPAIN)
sextremera@pharmamar.com

Antonio Nieto Archilla
Clinical Development. PharmaMar S.A.
Avda. de los Reyes, 1
Polígono Industrial La Mina
28770 Colmenar Viejo. Madrid (SPAIN)
anieto@pharmamar.com

Javier Gómez García
Clinical Development. PharmaMar S.A.
Avda. de los Reyes, 1
Polígono Industrial de la Mina
28770 Colmenar Viejo. Madrid (SPAIN)
jgomez@pharmamar.com