Using Styles and Templates to Customize SAS® ODS Output

Sunil K. Gupta, Gupta Programming, Simi Valley, CA

ABSTRACT

SAS's new Output Delivery System (ODS) feature enables the creation of various new file types including Rich Text Format (RTF), PostScript and HTML. The standard default style used to control the output is generally well organized and sufficient for most purposes. When the default SAS ODS output report will no longer suffice, SAS users can take advantage of ODS advanced features to customize the output report with styles and templates. By having greater control over the report details such as color, font, size, justification, order and labels, the format and quality of the report will be enhanced.

This paper introduces the concepts necessary to understand and apply the advanced features of SAS's ODS. Issues in defining and selecting output destinations, selecting output objects and creating customized output files will be discussed. The focus is on the creation of custom styles and templates with PROC TEMPLATE for visually appealing output.

INTRODUCTION

SAS's new ODS features offer significant improvements in the presentation of reports and files. Some of ODS's advantages include the following:
- Creation of output objects from most all procedures
- Creation of RTF, HTML and PostScript files
- Creation of data sets from output objects.

The advanced features of ODS enable greater control and flexibility for the presentation of the data. Some of ODS's advanced functions include the following:
- Control of format and style of report – font, color, etc.
- Support for web site development and management
- Creation of a navigational system.

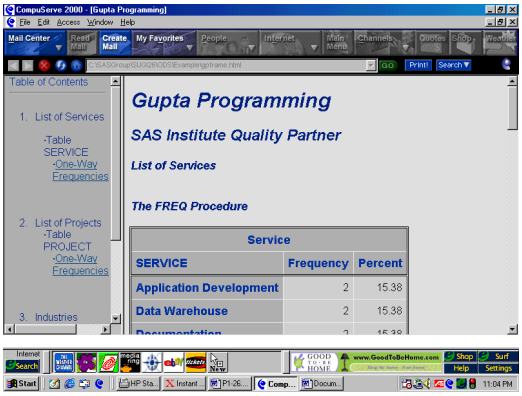
This paper will show how to apply the advanced features of ODS to create customized HTML files for a company's intranet. By using style and table templates, programmers have greater control on report details such as colors, fonts, size, data justification, order and label to meet the company's standards. This paper will show the steps involved in converting the default style and table template to a customized style and table template. The custom style and table template will be saved to a permanent template store location for multi-use access and repeated utilization.

To understand how styles and table templates affect the output file, it is helpful to consider the output file as a composite of global attributes defined by the style template and table and cell attributes defined by the table template. These customized styles and table templates need to be predefined before they can be accessed.

Before constructing a new style template, all available styles included in the installation should be considered. The closest existing standard style to the company's standard should be used as the model for the customized style. The collection of existing styles includes styles such as D3D, default and brown that can be directly accessed.

This paper will demonstrate how to customize the following items for a company's style & table templates:
1) Replace default color_list style element
2) Replace colors style element
3) Replace fonts style element
4) Change output margin
5) Change content file title
6) Change page file title
7) Create custom rowheader style element
8) Apply custom rowheader style element
9) Change header label
10) Change data format

FROM: DEFAULT STYLE & TEMPLATE



TO: CUSTOM STYLE & TEMPLATE

ODS BASICS AND PROCESSES

The sequence of steps to follow in using ODS for report generation include the following:
1. Defining Output Destinations

2. Selecting Output Objects
3. Creating Output Files
4. Using Styles and Templates to Customize Output

When working in the ODS environment, it is helpful to consider the following definitions:
1. DESTINATION – "Final File Type"
Where you want to be?
(List, HTML, RTF, Printer, PDF, Data set)

2. OBJECT – "A Non-Physical Item"
What you have to work with?
(Select or Exclude Output Objects)

3. STANDARD REPORT – "As-Is Final End Product"
How you will reach your destination?
(Default Attributes defined in Default Style & Template)

4. CUSTOM REPORT – "Focused Final End Product"
How you can control your destination?
(Custom Style & Template)

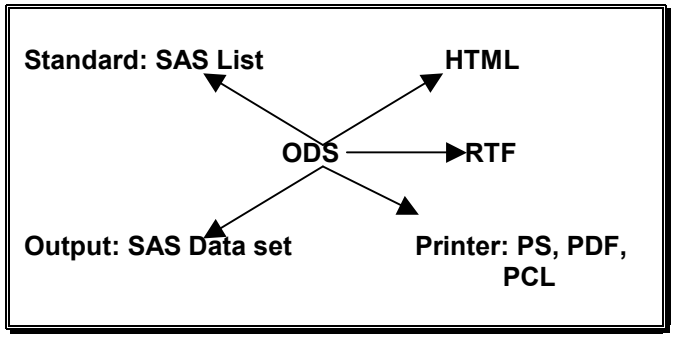
Defining Output Destinations

In defining the destinations to use, ODS needs to open and then close the output destination. More than one destination may be defined.

Currently, the available output destinations include:
1. Standard SAS List
2. HTML
3. RTF
4. Output - SAS Data Set
5. Printer – PostScript, PDF, PCL

Styles set at the ODS statement will remain in effect until changed to another style or until the destination is closed.

OUTPUT DESTINATIONS

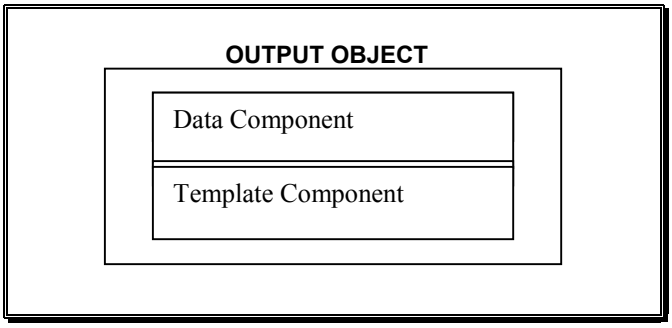


Selecting Output Objects

Before table templates can be customized, it is important to first identify the table templates that will be utilized. Understanding how objects work, identifying objects from SAS Log, and selecting objects to include in output files will need to be done.

ODS creates output objects from the execution of each procedure. Procedure output is divided into one or more output objects. Each output object has a set of attributes such as name and label.

Each output object has two components.
1) Data component (raw numbers and characters)
2) Template component (description of format and arrangement instructions)



Each procedure has a default template for each output object created. The template component can be customized and saved.

The method used to identify objects from the SAS Log is the TRACE option. Use the ODS TRACE statement to write a record of each output object that is created to the SAS log.

PARTIAL SAS LOG OF TRACE OPTION

```
Output Added:
-------------
Name:       OneWayFreqs
Label:      One-Way Frequencies
Template:   Base.Freq.OneWayFreqs
Path:       Freq.SERVICE.OneWayFreqs
-------------
```

OUTPUT OBJECT ATTRIBUTES

ATTRIBUTE	DESCRIPTION
NAME	Output object name
DATA	Data component used to create output object (Note: Appears only if different from name of output object)
LABEL	Contents of the output object
TEMPLATE	Template component used to format the object
PATH	Path of the output object

Selecting objects to include in the output file is accomplished with the select option. Use the ODS select option to select several objects (name1 name2 … namen, where name1, 2, n are the object's name).

Below is an object reference table for selected procedures.

OBJECT REFERENCE BY PROCEDURE

PROCEDURE	NAME	TEMPLATE	PATH
Proc_name;	Obj_name	Base.Proc_name. Obj_name	Proc_name. X.Obj_name
Var X;			
Run;			
Freq	OneWay Freqs	Base.Freq. OneWayFreqs	Freq.X. OneWayFreqs
Means	Summary	Base.Summary	Means. Summary
SQL	SQL_ Results	Base.SQL	SQL.SQL_ Results
Univariate	Moments	Base.Univariate. Moments	Univariate.X. Moments
	Basic Measures	Base.Univariate. Measures	Univariate.X. Measures
	TestsFor Location	Base.Univariate. Location	Univariate.X. Location
	TestsFor Normal.	Base.Univariate. Normality	Univariate.X. Normality
	Quantilies	Base.Univariate. Quantiles	Univariate.X. Quantiles
	Extreme.	Base.Univariate. ExtObs	Univariate.X. ExtObs
	Frequency	Base.Univariate. Frequency	Univariate.X. Frequency
	Plots	N/A	Univariate.X. Plots

Creating Output Files

Not all destinations support all templates available. The table and column templates are supported by all destinations because the templates are internal to the output object. The style templates, however, are supported by destinations that support report details such as color, font and size. The HTML destination will be reviewed as an example.

TEMPLATE SUPPORT BY DESTINATION

DESTINATION	SUPPORT TABLE/COLUMN	SUPPORT STYLE
Listing	Yes	No
Printer	Yes	Yes
RTF	Yes	Yes
Data Set	Yes	No
HTML	Yes	Yes

The two options for creating reports are to use the default style or to specify a different style. The style specified must first be defined and accessible. To send output to a destination file, use the file = option with the appropriate file type.

ODS *DESTINATION* FILE="file-spec.ext"
style = style-spec;

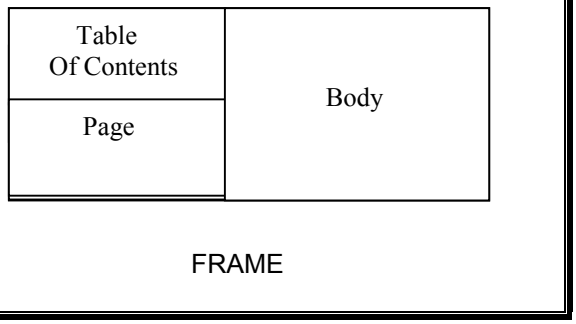
Where *DESTINATION* is one of the following: LIST, RTF, Printer or HTML.

```
* Example: Create RTF file with SAS supplied style;

ODS RTF
FILE='c:\sasgroup\sugi26\ods\example\profile.rtf'
STYLE=barrettsblue;
```

Creating HTML Files

HTML FILES



The HTML file destination saves the content into logical files. The style = option is used to specify the style.

```
ODS HTML
  PATH = 'PATH-spec'          (folder for html files)
  BODY = 'HTML-FILE-spec' (html filename)
  CONTENTS = 'TOC-spec'   (links to pieces in body)
  PAGE = 'PAGE-spec'        (individual pages)
  FRAME = 'FRAME-spec'    (integrate toc, body &
                              pages)
  STYLE = 'STYLE-spec'      (style type);
```

```
ODS HTML
PATH = 'c:\sasgroup\sugi26\ods\example\' (url=none)
          BODY = 'gpbody.html'
          CONTENTS = 'gptoc.html'
          FRAME = 'gpframe.html'
          STYLE = styles.gpstyle;
```

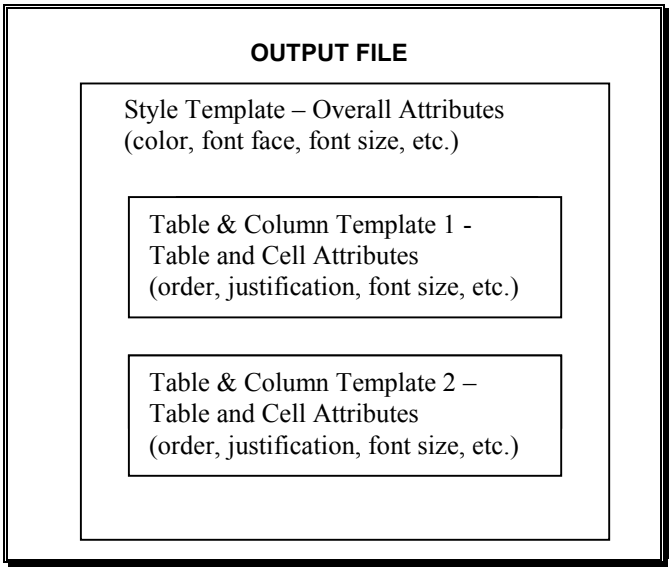
Note that it is possible to change the style within the same HTML file by issuing another ODS HTML STYLE = without a new HTML filename. If this code is placed before the next procedure, then the next procedure will utilize the second style.

USING STYLES AND TEMPLATES TO CUSTOMIZE OUTPUT

To understand how styles and table templates affect the output file, it is helpful to consider the output file as a composite of global attributes defined by the style template and table and cell attributes defined by the table template. Style templates define the overall attributes of the output file such as color, font face and size. Table

and column templates define the specific table and cell attributes such as order, justification and font size. The report style defines the overall look of the output.

UNDERSTANDING HOW TEMPLATES WORK



DEFAULT STYLE AND TEMPLATE

Without specifying any style in the ODS statement, SAS uses the default style and template to present the data. The default templates supplied by SAS Institute are stored in a template store SASHELP.TMPLMST.

To access templates from the Display Manager, follow these steps:

1. Select View in the top menu bar – left click
2. Select Results in the drop-down menu – left click
3. Select Results in the results window – right click
4. Select Templates in the drop-down menu – left click
5. Expand the Sashelp.Tmplmst item by clicking its +
6. Display the contents of folder in the right hand window
7. Double click on any template in the right window

By default, custom templates are saved in the SASUSER library. The custom templates override the default settings of the default templates.

EXISTING STYLES

The easiest method to construct a new style template is to use an existing style included in the installation that most closely matches the requirements. Note that the default style for the LISTING and the HTML destinations is different from the default style for the PRINTER and the RTF destinations. Be sure to use the appropriate default style for the defined destination.

Below is a collection of existing styles that can be directly accessed.

STANDARD STYLES

NAME	DESCRIPTION
BarrettsBlue	Blue header background, light table background
Beige	Beige header text, white text in table
Brick	Brick color header text, white text in table
Brown	Brown title, black header, light table background
D3D	White header, bold table border
Default	Dark blue header, shade table background (Default for LISTING and HTML Destinations)
Minimal	No color, light text in table
NoFontDefault	Black header text, white background table
Printer	Printer Style (Default for PRINTER Destination)
RTF	RTF Style (Default for RTF Destination)
Statdoc	Blue header, black text in table
Theme	Dark header, dark table
FancyPrinter	Printer Style
SansPrinter	Printer Style
SasdocPrinter	Printer Style
SerifPrinter	Printer Style

UNDERSTANDING CUSTOM STYLES AND TEMPLATES

You can control essentially every aspect of your report using templates. The template procedure is used to create and modify styles and table definition templates. Custom styles are similar to custom templates, since styles are stored within a template store and are created by the PROC TEMPLATE. Styles differ from templates in that the code for a style may include a parent statement. This defines all the attributes of the parent style.

The table below shows the level of customization that can be achieved. Any combination of the templates can be defined and utilized for customization.

LEVEL OF CUSTOMIZATION: TEMPLATE TYPES

TEMPLATE	SCOPE	ELEMENT	ATTRIBUTES
Style	SAS Job	Position on Report	Color, Font face, Font size
Table	Object	Position on Table	Column ordering, Table header order
Column	Object	Position on Table	Cell formats - font, Cell justification
Header	Object	Position on Table	Label
Footer	Object	Position on Table	Text
Tree	Object	Position on Table	Equations and functions

The style template component controls the following:

Background image and color, left and right margins, header and labels for table of contents and table of pages.

STYLE DEFINITION: COLLECTION OF STYLE ELEMENTS (Position on Report)

STYLE ELEMENT TYPE	COLLECTION OF STYLE ATTRIBUTES
Layout	Background Image, Color, Left & Right Margin, Font Style
Table of Contents Section	Header, Label
Table of Pages Section	Header, Label
Titles	Color – background, foreground, Font, Size
Footnotes	Color – background, foreground, Font, Size
Tables	Color – background, foreground, Font, Size, Spacing

The table template component controls the following: the order of the columns, text and order of column, headers, formats for data and font sizes.

TABLE DEFINITION: COLLECTION OF TABLE ELEMENTS (Data Format and Position on Table)

TABLE ELEMENT NAME	COLLECTION OF TABLE ATTRIBUTES
Header	Order of headers Default style element = header
Column	Order of variables Default style element = data
Define Block	Label, Cell Format, Justification, Spacing, Font Style
Footer	Label Default style element = footer

The styles behave as a hierarchy in which the lower detail level or child template can inherit or override aspects of the higher or parent template attributes.

The concept of inheritance is utilized to help minimize the amount of redundant code required for each style element and to help enforce consistency and organization across all related style elements. A change in the attribute of a parent style element will also affect all related child style elements. A reference to an existing style element with the from clause searches the current child first for the original SAS code and then the parent definition if it is not found in the current child style element.

The table below shows the level of dependency between the style elements. Level 1 defines the core list of attribute values. Level 2 defines the core list of original style elements. Level 3 defines the style elements that are child dependents of the level 2 style elements.

LEVEL	STYLE ELEMENTS
1	Attribute Values: Fonts, color_list, colors, HTML, text, etc.
2	Original Style Elements: example - Style Container
3	Style Definition inheritance with Style Statement: example - Style Document from Container

The Fonts style element establishes fonts to be used in items such as titles and headers. The font definition consists of the following items: font face, font size, font weight, font style, font width. The fonts used for the majority of the report sections include TitleFont, HeadingFont and DocFont.

Examples of the font face include the following: **times, courier, arial, Helvetica.**

FONT STYLE ELEMENT

FONT WEIGHT	FONT STYLE	FONT WIDTH
Medium	Italic	Normal
Bold	Roman	Compressed
Demi_bold	Slant	Extra_compressed
Extra_bold		Narrow
Light		Wide

The color_list and colors style elements establish colors to be used in the report. The color value is a string that identifies the color. Any color name supported by SAS/GRAPH can be used.

COLOR STYLE ELEMENT

COLOR TYPE	EXAMPLE
Simple color	Blue
Complex color	Light Blue
red/green/blue (RBG) value	CX70DB93 or #70DB93
hue/light/saturation (HLS) value	H14E162D
grapy-scale value	GRAYBB

UNDERSTANDING HOW TO CUSTOMIZE STYLE AND TEMPLATE

There are two methods to customize the output:
1. Create a new template.
   (ex. STYLES.GPSTYLE)
2. Modify an existing template.
   (ex. BASE.FREQ.ONEWAY.FREQS)

There are two options to customize the output:
A. For the entire SAS job, use the style definition.
B. For a single output object, use the table definition.

A. To customize by Style, follow these general steps:

1) Set Style Definition
   (Collection of Style Elements: color, font face, font size, etc.)

2) Set Style Elements
   (Collection of Style Attributes that apply to a specific part of the output)

3) Save to Standard Style Element Name or to new

Style Element Name

B. To customize by Table Object, follow these general steps:

1) Set Table Definition
(Order of Table Headers, Footers and Columns)

2) Set Table Elements
(Collection of Table Attributes for a specific column, header or footer)

3) Apply customized standard element names automatically with STYLE= in the ODS statement line or set STYLE = new element name or set STYLE = style attributes to directly apply customization

The following guidelines can be used to customize styles and templates:

1. Select closest existing standard style definition to meet requirements.

2. Save SAS code for the selected style template into a file.

```
PROC TEMPLATE;
    SOURCE STYLES.DEFAULT /
    FILE='C:\SAS\TEMPLATE\STYLE_GP.SAS';
RUN;
```

3. Edit the **STYLE_GP.SAS** program to customize style elements as needed. Programmer can a) customize an existing style element name or b) create a new style element. The customization made at this level is global.

a) Customize page file title

```
style PagesTitle from IndexTitle
      "Controls the title of the Pages file." /
      pretext = "Pages";
```

b) Create new custom rowheader style element

```
style gprowheader from header
      "Controls row headers." /
      font=fonts('Emphasisfont');
```

4. Identify output object with **TRACE** option.

5. Select output object with **SELECT** option.

6. Save SAS code for the selected output object template into a file.

```
PROC TEMPLATE;
    SOURCE  BASE.FREQ.ONEWAYFREQS/
    FILE='C:\SAS\TEMPLATE\FREQ_GP.SAS';
RUN;
```

7. Edit the **FREQ_GP.SAS** program to customize table elements as needed. Programmers can a) apply the custom style attributes directly or b) use the new style element in table elements. Options a) and b) show two different methods to customize style at the table template level. When customizing directly at the table definition level, be sure to apply appropriate style elements and style attribute settings to the table template.

a) Apply custom style attributes directly

```
define Variable;
      just = varjust;
      style = {font=fonts('Emphasisfont')};
      id;
     generic;
end;
```

b) Apply new custom rowheader style element

```
define Variable;
      just = varjust;
      style = gprowheader;
      id;
     generic;
end;
```

8. To apply custom style elements at the SAS job level, the new style must be defined in the ODS statement. In the statement below, the custom style elements are defined in the GPSTYLE style definition.

```
*Apply Customized Style Definition;
ODS HTML STYLE = STYLES.GPSTYLE;
```

CREATING CUSTOM STYLE AND TEMPLATE

Using custom styles requires setting the search path to include the location of the template store. The search path specifies which locations to search for definitions that were created by PROC TEMPLATE along with the order in which to search for them. Any changes and creation of templates from the current SAS session update the SASUSER.TEMPLATE library. All templates are read from the SASHELP.TMPLMST library.

SAS PROGRAM FOR DEFAULT STYLE AND SETTING

The default search path setting is:

```
ODS PATH SASUSER.TEMPLAT (UPDATE)
SASHELP.TMPLMST (READ);
```

Below is the partial SAS program (STYLE_DEFAULT.SAS) to display the default style elements and default attribute values of the default style definition.

```
************************************************************;
* PROGRAM: STYLE_DEFAULT.SAS;
* DATE: January 27, 2001;
* PURPOSE: To display the style definition of the style;
*                STYLES.DEFAULT;
************************************************************;

PROC TEMPLATE;

 define style styles.default;

 style body from document
          "Controls the Body file." /;

 style color_list
```

```
            "Colors used in the default style" /
            'fgB2' = cx0066AA
            'fgB1' = cx004488
            'fgA4' = cxAAFFAA
            'bgA4' = cx880000;

 style colors
       "Abstract colors used in the default style" /
        'headerfgemph' = color_list('fgA2')
        'headerbgemph' = color_list('bgA2')
        'headerfgstrong' = color_list('fgA2')
        'headerbgstrong' = color_list('bgA2')
        'headerfg' = color_list('fgA2')
        'headerbg' = color_list('bgA2') ;

 style fonts
  "Fonts used in the default style" /
  'TitleFont2' = ("Arial, Helvetica, Helv", 4, Bold)
  'TitleFont' = ("Arial, Helvetica, Helv", 5, Bold)
  'StrongFont' = ("Arial, Helvetica, Helv", 4, Bold);

 style Body from Document  /
        leftmargin = 8;

 style SysTitleAndFooterContainer from Container /
        outputwidth = 100%
        cellpadding = 1
        cellspacing = 1
        borderwidth = 0;

 style ContentTitle from IndexTitle
        "Controls the title of the Contents file." /
        pretext = text('content title');

 style PagesTitle from IndexTitle
        "Controls the title of the Pages file." /
        pretext = text('pages title');

     end;

RUN;
```

Below is the partial SAS program (FREQ_DEFAULT.SAS) to display the default attributes of the default FREQ table template.

```
************************************************************;
* PROGRAM: FREQ_DEFAULT.SAS;
* DATE: January 27, 2001;
* PURPOSE: To display the default FREQ table template;
************************************************************;

PROC TEMPLATE;

Define table Base.Freq.OneWayFreqs;
 Parent = Base.Freq.OneWayList;
 notes "One-Way Frequency table";
end;

Define table Base.Freq.OneWayList;
 notes "Parent for One-Way Frequency table and LIST
 table";
 dynamic page needlines plabel varlabel lw varjust gluef
 gluep;
 column Line Variable ListVariable Frequency
TestFrequency Percent  TestPercent CumFrequency
CumPercent;

header h1;
     translate _val_=._ into "";

define h1;
        text varlabel;
        space = 1;
        split = "";
        spill_margin;
        highlight;
end;

define Variable;
        just = varjust;
        style = rowheader;
        id;
       generic;
end;

define TestFrequency;
        header = "\   Test  \Frequency\";
        glue = 4;
        format = BEST8.;
        just = c;
end;

define TestPercent;
        header = "\   Test\ Percent\";
        glue = 3;
        format = 6.2;
        just = c;
end;

RUN;
```

SAS PROGRAM FOR CUSTOM STYLE AND SETTING

To create new templates in a separate location, use this setting to first search your location:

```
LIBNAME MYLIB
'C:\SASGROUP\SUGI26\ODS\EXAMPLE\';

ODS PATH MYLIB.MYSTORE (UPDATE)
SASHELP.TMPLMST (READ);
```

Below is the partial SAS program (STYLE_GP.SAS) to customize the style template and save as the new style GPSTYLE. Items in bold identify the changes made from the default style definition.

```
************************************************************;
* PROGRAM: STYLE_GP.SAS;
* DATE: January 27, 2001;
* PURPOSE: To create custom style STYLES.GPSTYLE;
*                 from the style STYLES.DEFAULT;
************************************************************;

PROC TEMPLATE;

 define style styles.gpstyle;

   parent = styles.default;
```

```
style body from document /
          backgroundimage = "gp_bkgrd.bmp";
```

1) Replace default color list style element

```
replace color_list /
          "purple" = cx663399
          "green" = cx66cccc
          "lgreen" = cxbbeeee
          "red" = cx990066
          "white" = cxffffff
          "black" = cx000000
           'fgB2' = cx0066AA ;
```

2) Replace colors style element

```
replace colors  /
       'headerfgemph' = color_list('red')
       'headerbgemph' = color_list('lgreen')
       'headerfgstrong' = color_list('red')
       'headerbgstrong' = color_list('lgreen')
       'headerfg' = color_list('red')
       'headerbg' = color_list('lgreen') ;
```

3) Replace fonts style element

```
replace fonts  /
 'TitleFont2' = ("Verdana, Arial, Helvetica, Helv",4,Bold)
 'TitleFont' = ("Verdana, Arial, Helvetica, Helv",5,Bold)
 'StrongFont' = ("Verdana, Arial, Helvetica, Helv",3,Bold);
```

4) Change output margin

```
style Body from Document  /
       rightmargin = 5
       leftmargin = 95 ;

style SysTitleAndFooterContainer from Container /
       outputwidth = 98%
       cellpadding = 4
       cellspacing = 2
       borderwidth = 0;
```

5) Change content file title

```
style ContentTitle from IndexTitle
       "Controls the title of the Contents file." /
       pretext = "Web Site Contents";
```

6) Change page file title

```
style PagesTitle from IndexTitle
       "Controls the title of the Pages file." /
       pretext = "Pages";
```

7) Create custom rowheader style element

```
style gprowheader from header
      “Controls row headers.” /
      font=fonts(‘Emphasisfont’);

end;

RUN;
```

Below is the partial SAS program (FREQ_GP.SAS) to customize the FREQ table template. Items in bold identify the changes made from the default table template.

```
************************************************************;
* PROGRAM: FREQ_GP.SAS;
* DATE: January 27, 2001;
* PURPOSE: To create custom FREQ table template ;
*                  from default FREQ table template;
************************************************************;

PROC TEMPLATE;

edit Base.Freq.OneWayFreqs;
 notes "Parent for One-Way Frequency table and LIST
 table";

dynamic page needlines plabel varlabel lw varjust gluef
gluep;

* Column defines the order of the variables in which they
appear;

column Line Variable ListVariable Frequency
TestFrequency Percent  TestPercent CumFrequency
CumPercent;

* Header defines the order in which the template uses the
headers;

header h1;
     translate _val_=._ into "";

* Define blocks define the headers - format, label,
justification, font style;

define h1;
        text varlabel;
        space = 1;
        split = "";
        spill_margin;
        highlight;
end;

define Variable;
        just = varjust;
```

8) Apply custom rowheader style element

```
       style = gprowheader;
       id;
      generic;
end;

define TestFrequency;
```

9) Change header label

```
        header = "\   Test  \Count\";
        glue = 4;
        format = BEST4.;
        just = c;
end;


define TestPercent;
        header = "\   Test\ %\";
        glue = 3;
```

10) Change data format

```
        format = 4.0;
        just = c;
end;
```

RUN;

USING CUSTOM STYLE AND TEMPLATE

In the code below, since the ODS path statement lists the company's template store before the SASHELP template store, all custom styles and table templates will be located and utilized prior to the default styles and table templates. For each destination defined, the STYLE = STYLES.GPSTYLE must be included in the ODS statement. All style elements defined in the GPSTYLE style will be utilized. The PROC FREQ output object will utilize the custom table template.

Below is the partial SAS program required to utilize the new style GPSTYLE and the custom PROC FREQ table template.

```
LIBNAME MYLIB
'C:\SASGROUP\SUGI26\ODS\EXAMPLE\';

ODS PATH MYLIB.MYSTORE (READ)
SASHELP.TMPLMST (READ);

ods html
path = 'c:\sasgroup\sugi26\ods\example\' (url=none)
   body = 'gpbody.html'
   contents = 'gptoc.html'
   frame = 'gpframe.html' (title = 'Gupta Programming')
   style = styles.gpstyle;

ods proclabel 'List of Services';

proc freq data = mylib.client;
 tables service/nocol norow nocum;
run;

ods html close;
```

SUMMARY

SAS's new Output Delivery System (ODS) feature enables the creation of various new file types including Rich Text Format (RTF), PostScript and HTML. When the default SAS ODS output report will no longer suffice, SAS users can take advantage of ODS advanced features to customize the output report with styles and templates. By having greater control over the report details such as color, font, size, justification, order and labels, the format and quality of the report will be enhanced. The creation of custom styles and templates with PROC TEMPLATE facilitates more visually appealing output.

TRADEMARK INFORMATION

WEB SITES

http://www.sas.com
http://www.sas.com/rnd/base/

SUGI PAPERS

Fehlner, William, "Making the Output Delivery System (ODS) Work for You", SAS Institute (Canada) Inc., Toronto, Ontario

Gupta, Sunil, "Customized Reports with SAS ODS", WUSS 2000 Class, Gupta Programming, Simi Valley, CA

Haworth, Lauren, "HTML for the SAS Programmer", WUSS 2000, Ischemia Research & Education Foundation, San Francisco, CA

Lafler, Kirk Paul, "Creating HTML Output with Output Delivery System", SUGI 25, Software Intelligence Corporation, Spring Valley, CA

Lafler, Kirk Paul, "Delivering Results with the Output Delivery System", SUGI 24, Software Intelligence Corporation, Spring Valley, CA

McNeill, Sandy, "ODS for Dummies", WUSS 2000, SAS Institute Inc., Cary, NC

Muller, Sally, Bryant, Lara, Pass, Ray, "ODS, YES! Odious, NO! – An Introduction to the SAS Output Delivery System", SUGI 25, University of North Carolina at Chapel Hill, Chapel Hill, NC, Ray Pass Consulting, Hartsdale, NY

Olinger, Chris, "ODS for Dummies", SUGI 25, SAS Institute Inc., Cary, NC

Olinger, Chris, "Twisty Little Passages, All Alike – ODS Templates Exposed", PharmaSUG '99, SAS Institute Inc., Cary, NC

Olinger, Chris, Tobias, Randall D., "ODS for Data Analysis: Output As-You-Like-It in Version 7", SAS Institute Inc., Cary, NC

ABOUT THE AUTHOR

The author welcomes your comments & suggestions.

Sunil K. Gupta
Gupta Programming
SAS Institute Quality Partner™
213 Goldenwood Circle, Simi Valley, CA 93065
Phone: (805)-577-8877
E-mail: Sunil@GuptaProgramming.com
http://www.GuptaProgramming.com

Sunil is a senior consultant at Gupta Programming. He specializes in SAS/BASE®, SAS/AF®, SAS/FSP®, SAS/STAT® and SAS/GRAPH®. His consulting projects with pharmaceutical companies include the development of a Clinical Study Data Entry System, a Macro-Based Application for Report Generation, and customized plots and charts with SAS/GRAPH®. He has been using SAS® software for over 10 years and is a SAS Institute Quality

Partner™.

ACKNOWLEDGEMENTS

The author would like to thank Sandy McNeill for her invaluable assistance in the preparation and technical review of this paper.