Paper 227-2009

Tiptoe through the Templates

Cynthia L. Zender, SAS Institute, Inc., Cary, NC

ABSTRACT

Are you confused about the difference between style templates, table templates, tagset templates, and graph templates? Do you wonder how they're all used with ODS?

This paper provides an overview of all the different template types and how they're used with the Output Delivery System. From style and table templates, that first appeared with SAS® 7 to the newest graph templates that appeared with SAS® 9.2, this paper will provide an overview and several concrete examples for each template type. Along the way, we'll also discuss the template garden (SASHELP.TMPLMST) where all the templates live, how to transplant your templates to a different garden, how to come up with your own new variety of templates (PROC TEMPLATE), and how to find your way to the new template garden (ODS PATH). New features of PROC TEMPLATE syntax (such as the IMPORT statement) will be highlighted.

INTRODUCTION

I like to think of SAS item stores (where templates live) as being gardens. Gardens are where plants live. The difference between gardens and plants that grow in the wild is that gardens are planned, they have a purpose, they are tended. You might have an herb, vegetable, or kitchen garden by the house, so you can just walk outside and snip some oregano or pull some scallions for dinner. Or you might have a perennial garden in the back yard, so that, over time, the perennials grow and cover the fence between you and the neighbors. You might have a garden of succulents or annual flowers or ornamental grasses in the front yard. Depending on your climate zone, you might even have a cactus garden. If you had a big enough piece of land, you might have fruit trees or melon patches, or strawberry hills. A garden is just a place where plants grow. But the plants also have a gardener, somebody who cultivates, fertilizes, weeds, and waters the vegetables, flowers, shrubs, fruits, herbs. A garden consists of individual plants; some grow from seed, some start from cuttings, and some start from bulbs. A garden could be devoted to one type of plant, or it could contain a pleasing mixture of plants.

OK, before I go overboard with the template/garden metaphor, here's the point: SAS item stores are a proprietary SAS storage structure. An item store doesn't just hold or store one type of SAS item. The SAS registry is contained in an item store. An ODS document is contained in an item store. SAS templates are contained in an item store. An item store is a binary, highly compressed, file structure. If you look at item stores in Windows Explorer, you will see that the item stores, no matter what kind, have a file extension of SAS7BITM.

Item stores live in SAS libraries, but since they're not data sets, you do not see them in the SAS Explorer Window view of a SAS library. Each type of item store has a way for you to see what's inside, and usually, each item store has a procedure that gives you management and update capabilities. The SAS Registry has PROC REGISTRY. ODS DOCUMENT has PROC DOCUMENT. SAS templates have PROC TEMPLATE. And, just as you have to use the right tool for the job in your garden (you wouldn't use a wheat thresher to harvest in your herb garden), you have to use the right procedure for dealing with your item store.

This paper is going to focus on SAS template item stores and the types of templates that are used by the Output Delivery System (ODS). Now that we all know that SAS templates are stored in item stores, I'm going to adopt the following conventions when referring to the item store and the templates in them. The primary item stores that we'll be dealing with will be SASHELP.TMPLMST or SASUSER.TEMPLAT. I'll refer to these item stores by name. These item stores contain the four primary template types: SAS table template definitions (table templates); SAS style template definitions (style templates); SAS tagset template definitions (tagset templates) and SAS graph template definitions (graph templates). There are other types of template definitions that can be contained in our item stores, such as column definitions and header definitions, but these template types are largely "sub-definitions" used by the main template types. I'll discuss them where appropriate.

Before we jump into our item store gardens and start examining the various species of templates, let's look at some PROC TEMPLATE utility code.

DON'T GO INTO THE GARDEN EMPTY-HANDED

My grandmother always kept a small basket in the closet by the back door. It contained her gardening gloves, garden shears, a spade, and a wicked looking fork-like tool to dig out weeds. When we went out to work in the garden, the basket always came with us. Sometimes we'd go out without the basket and just look things over, but when we worked, we never went into the garden empty-handed.

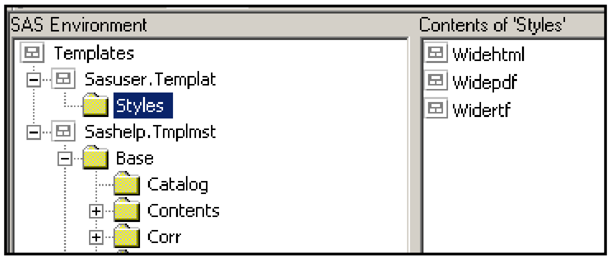
These five PROC TEMPLATE "utility" programs and statements are like my grandmother's garden basket. They will make your life immensely easier as you're investigating and working with templates.

```
** 1) List a particular Template Store;
proc template;
   title '1) List a Particular Template Store';
   list / store=sasuser.templat;
run;

** 2) List a Template Store Folder or Directory;
proc template;
   title '2) List One Template Folder';
   list Base;
   list Styles;
   list Tagsets;
   list StatGraph;
run;

** 3) List a particular Sub-Folder or Template;
proc template;
   title '3) List a particular Sub-Folder or Template';
   list Styles.Default;
   list Base.Contents;
   list Stat.Reg.Graphics;
run;

** 4) Put the Template Source Code in the LOG;
proc template;
   title '4) Put the Template Source Code in the LOG';
   source Styles.Default;
   source Base.Summary;
   source Stat.Reg.Graphics.CooksD;
   source Tagsets.CSV;
run;

** 5) Put the Template Source Code in a file;
proc template;
   title '5) Put the Template Source Code in a file';
   source Styles.Default / file='c:\temp\default.tpl';
   source Base.Summary / file='c:\temp\summary.tpl';
run;
```

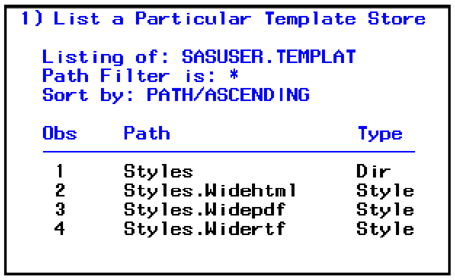
This code generates a *lot* of output. Only partial output is shown in the next series of figures. There are a few comments before we dive into the details. The LIST statement of PROC TEMPLATE shows only the list of items or templates that are stored in the template store. So in step 1 in the code above, the LIST statement would list only what's in the item store that is identified by the STORE= option. Without the STORE= option, the SASHELP.TMPLMST item store would be the one whose contents were listed (in #1).

The contents of the template store will show more than just the template names. Internally, the template store has a directory structure. When you look at the template store in an interactive window, you will see a folder tree structure; but when you look at the template store as a result of the LIST statement, you see a more flattened structure as shown in Figure 1. For your convenience, the LIST results are shown side by side with the Template Browser view in Figure 1. To open the interactive window, you either enter the command 'odstemplates' in the SAS command area or you right-click on the word "Results" in the Results Window and select "Templates" from the drop-down menu to open the Template Browser.

Interactive Window



LISTING Results



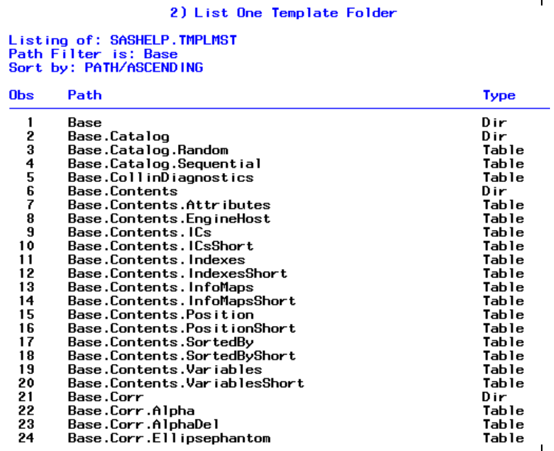
```
 1) List a Particular Template Store

   Listing of: SASUSER.TEMPLAT
   Path Filter is: *
   Sort by: PATH/ASCENDING

   Obs    Path                 Type
   ---------------------------------
    1     Styles               Dir
    2     Styles.Widehtml      Style
    3     Styles.Widepdf       Style
    4     Styles.Widertf       Style
```

Figure 1: Comparison of Interactive Template Browser versus LIST Statement Output

The interactive Template Browser mode is more like the Microsoft Windows interface, but if you're using SAS® Enterprise Guide® or are running in batch mode, the interactive window might not be available to you. In either of these instances, Step 1 code will become very useful to you. You can see how the folder "Styles" on the left is identified as the "Styles" directory on the right. The three individual style templates are all shown as having an item store type of "Style". This means that the three items in the interactive pane (with the rectangular icon) are the style templates.

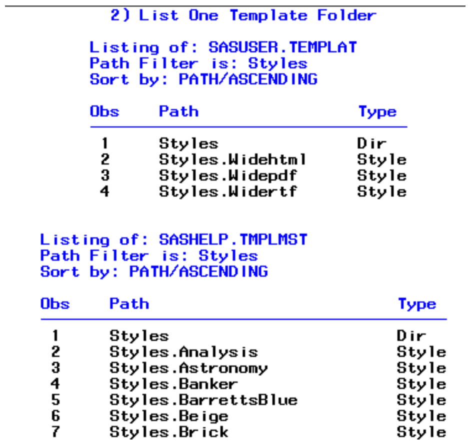
Over in the screen shot of the interactive window, inside SASHELP.TMPLMST, we can see that there is a folder called "Base" and that the Base folder contains subfolders. The code in Step 2 will enable you to list the contents of a specific folder or subfolder in an item store. Without any reference to an item store as in Step 1, the code will produce a list of the items in the folder, if found, from all the item stores in the current list. We can see from the interactive screen shot that only SASHELP.TMPLMST has a Base directory. However, both SASUSER.TEMPLAT and SASHELP.TMPLMST have a STYLES directory, so the contents of both STYLES directories are listed. Figure 2 shows a partial output for each of the four folders listed in Step 2. (Note that your list of SASUSER.TEMPLAT will be different from mine. I created the three style templates shown in my list of the SASUSER.TEMPLAT STYLES directory.)

list Base;



```
                    2) List One Template Folder

 Listing of: SASHELP.TMPLMST
 Path Filter is: Base
 Sort by: PATH/ASCENDING

 Obs    Path                                    Type
 ----------------------------------------------------
   1    Base                                    Dir
   2    Base.Catalog                            Dir
   3    Base.Catalog.Random                     Table
   4    Base.Catalog.Sequential                 Table
   5    Base.CollinDiagnostics                  Table
   6    Base.Contents                           Dir
   7    Base.Contents.Attributes                Table
   8    Base.Contents.EngineHost                Table
   9    Base.Contents.ICs                       Table
  10    Base.Contents.ICsShort                  Table
  11    Base.Contents.Indexes                   Table
  12    Base.Contents.IndexesShort              Table
  13    Base.Contents.InfoMaps                  Table
  14    Base.Contents.InfoMapsShort             Table
  15    Base.Contents.Position                  Table
  16    Base.Contents.PositionShort             Table
  17    Base.Contents.SortedBy                  Table
  18    Base.Contents.SortedByShort             Table
  19    Base.Contents.Variables                 Table
  20    Base.Contents.VariablesShort            Table
  21    Base.Corr                               Dir
  22    Base.Corr.Alpha                         Table
  23    Base.Corr.AlphaDel                      Table
  24    Base.Corr.Ellipsephantom                Table
```

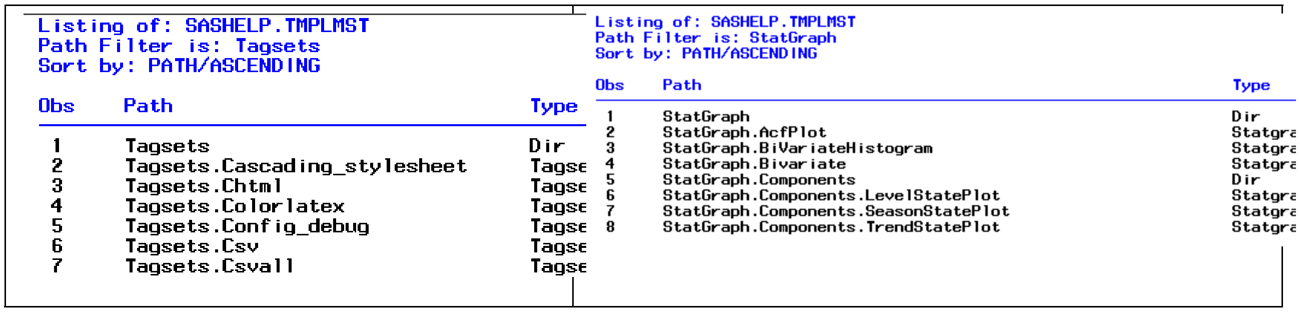
list Styles;



```
          2) List One Template Folder

      Listing of: SASUSER.TEMPLAT
      Path Filter is: Styles
      Sort by: PATH/ASCENDING

      Obs    Path                 Type
      --------------------------------
       1     Styles               Dir
       2     Styles.Widehtml      Style
       3     Styles.Widepdf       Style
       4     Styles.Widertf       Style

 Listing of: SASHELP.TMPLMST
 Path Filter is: Styles
 Sort by: PATH/ASCENDING

 Obs    Path                          Type
 ------------------------------------------
  1     Styles                        Dir
  2     Styles.Analysis               Style
  3     Styles.Astronomy              Style
  4     Styles.Banker                 Style
  5     Styles.BarrettsBlue           Style
  6     Styles.Beige                  Style
  7     Styles.Brick                  Style
```

list Tagsets;

list StatGraph;



```
Listing of: SASHELP.TMPLMST
Path Filter is: Tagsets
Sort by: PATH/ASCENDING

Obs    Path                              Type

 1     Tagsets                           Dir
 2     Tagsets.Cascading_stylesheet      Tagse
 3     Tagsets.Chtml                     Tagse
 4     Tagsets.Colorlatex                Tagse
 5     Tagsets.Config_debug              Tagse
 6     Tagsets.Csv                       Tagse
 7     Tagsets.Csvall                    Tagse
```

```
Listing of: SASHELP.TMPLMST
Path Filter is: StatGraph
Sort by: PATH/ASCENDING

Obs   Path                                        Type

 1    StatGraph                                   Dir
 2    StatGraph.AcfPlot                           Statgra
 3    StatGraph.BiVariateHistogram                Statgra
 4    StatGraph.Bivariate                         Statgra
 5    StatGraph.Components                        Dir
 6    StatGraph.Components.LevelStatePlot         Statgra
 7    StatGraph.Components.SeasonStatePlot        Statgra
 8    StatGraph.Components.TrendStatePlot         Statgra
```

Figure 2: Contents of Particular Folders or Directories in the Template Stores

In the four screen shots, you see the flattened directory structure, but you also see the four major types of templates that are stored in the template stores: Table, Style, Tagset, and StatGraph.

The default list of template stores that are available to you when you start up a SAS session are SASUSER.TEMPLAT and SASHELP.TMPLMST. SASHELP.TMPLMST is the item store that contains the templates that are supplied by SAS. SASUSER.TEMPLAT is the item store that SAS and ODS use, by default, to store any templates that you change or create in a session. You can modify the place where ODS reads and writes templates by using the ODS PATH statement to alter the default item store locations. The ODS PATH statement is very straightforward to use, and we will show its use later.

There's a lot of good usage information in the PROC TEMPLATE documentation. Other useful utility programs (including information about ODS PATH) can also be found in these Tech Support Notes regarding PROC TEMPLATE:

Task	Note URL
Determine template size	http://support.sas.com/kb/23/447.html
Copy a template to another item store	http://support.sas.com/kb/23/448.html http://support.sas.com/kb/23/449.html
Copy an entire item store to another location	http://support.sas.com/kb/24/096.html
Control the path for reading or writing templates	http://support.sas.com/kb/23/446.html http://support.sas.com/kb/23/455.html
Using the STORE= option	http://support.sas.com/kb/23/451.html
Use ODS PATH SHOW	http://support.sas.com/kb/23/445.html
Share a template store among multiple users	http://support.sas.com/kb/23/453.html
Delete a single template from the item store	http://support.sas.com/kb/23/443.html
Delete an entire template store	http://support.sas.com/kb/23/454.html

Table 1: Tech Support Notes That Contain Other "Utility" Information for PROC TEMPLATE

From our utility program above, the Step 3 output looks very much like the Step 2 output, except that it is limited to the folder specified in the LIST statement. Step 4 and Step 5 represent the code that you would use to get a copy of the template code in the SAS log (Step 4) or a copy of the template code into a SAS file. In my examples, I gave the template code a file extension of .TPL so that SAS does not launch if I accidentally double-click on the file that contains the template code. If you download updated template files from the R&D Web site, they, too, will have an extension of .TPL instead of .SAS. We will have the opportunity in upcoming sections to review different types of template code. So in the interest of saving paper here, we'll defer seeing the results of Step 4 and Step 5 in this section. Instead, the next topic of discussion is an overview of how ODS uses templates.

THE GARDEN PLAN: ODS AND TEMPLATE USAGE

ODS templates fall into three categories:

1) "data-centric" templates, which generate either data or graphical output objects. These are the table templates and the graph templates. The output objects created by these templates are tightly linked or bound to the SAS procedures for which they were designed. So, for example, PROC FREQ would never use a table template defined for PROC MEANS and vice versa. PROC CORR would never use a graph template defined for PROC GLM and vice versa. On the other hand, several statistical procedures might reference the **Common.PValue** column template within their table template definitions. For example, PROC UNIVARIATE and PROC ANOVA both calculate *p* values, and their table templates both reference or call this same column definition.

2) "style-centric" templates, which are used to determine the style characteristics of the output objects when the output objects are routed to a destination that supports style. These are the style templates. Style templates define style elements (like Header or Data or GraphFont). A single style element represents a collection of style attributes (foreground color, background color, font, and so on) that set the style characteristics for the style element. Every SAS procedure or process that produces output objects will use these style elements, if and when the output object is routed to a destination that supports setting style in this manner.

3) "destination-centric" templates (or "markup language-centric" templates), which write specific text or markup language tags around SAS procedure or process output, based on the destination. These are the tagset templates, also called markup templates or markup tagsets.

There's a lot that happens between the time you hit 'SUBMIT' for your SAS program and the time that your ODS output file gets created. Most of what happens after your procedure of choice is finished doing analysis has to do with ODS and how it uses the various template types. Figure 3 illustrates how various templates are used at different stages of the process for SAS table templates.

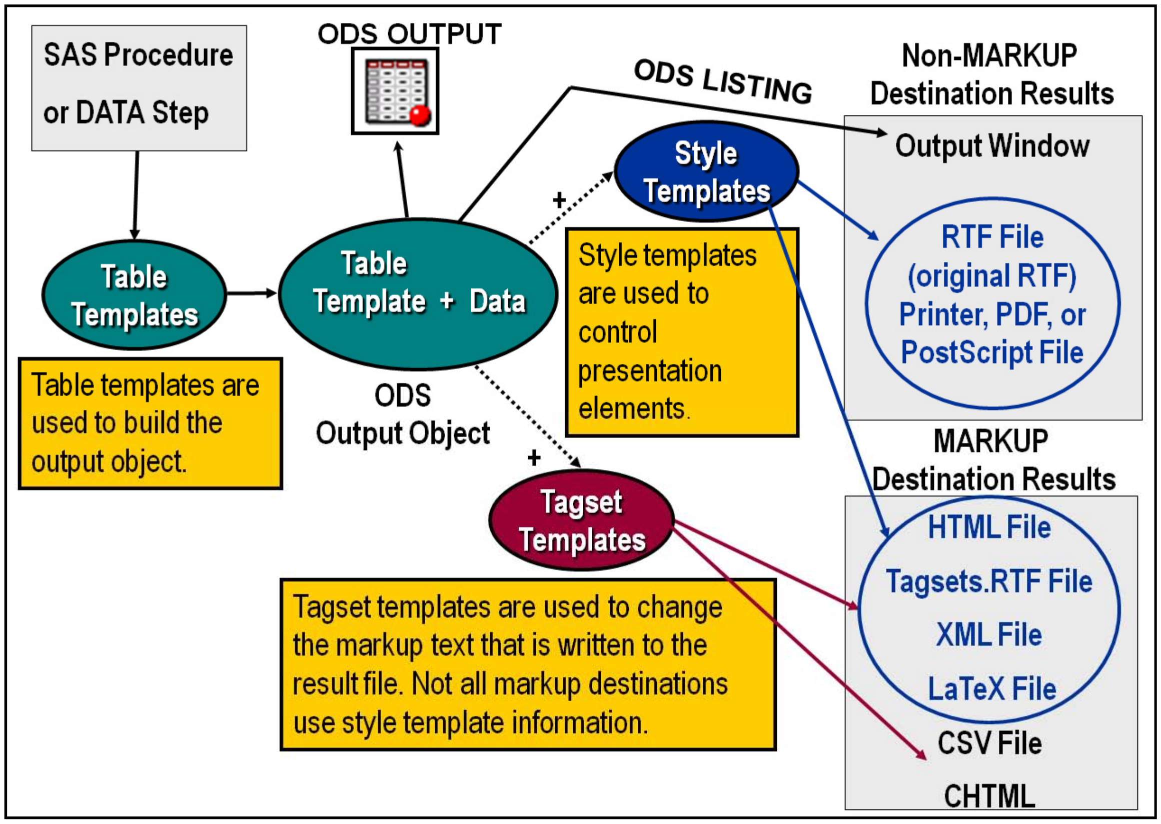

SAS Procedure or DATA Step
ODS OUTPUT
ODS LISTING
Non-MARKUP Destination Results
Output Window
Table Templates
Table Template + Data
Style Templates
RTF File (original RTF) Printer, PDF, or PostScript File
Style templates are used to control presentation elements.
Table templates are used to build the output object.
ODS Output Object
MARKUP Destination Results
Tagset Templates
HTML File
Tagsets.RTF File
XML File
LaTeX File
CSV File
CHTML
Tagset templates are used to change the markup text that is written to the result file. Not all markup destinations use style template information.

Figure 3: Table Templates and the ODS Process

As you can see, not all destinations use style templates and not all destinations use tagset templates. So, if you were creating an output data set or sending your tabular procedure output to the LISTING window, the only template being used would be the table template. But if you were creating tabular PDF output, your output file could be affected by the table template and the style template. If you were creating tabular HTML output, then your output file could be affected by table, style and tagset templates. On the other hand, if you were creating a CSV file, then your output file would be affected only by table and tagset templates. The only procedures that do NOT use table templates are procedures like PROC PRINT, PROC REPORT, and PROC TABULATE. Those procedures create output objects in their own fashion, without an explicit table template. But once their tabular output objects are created, they follow the same rules about whether style templates or tagset templates are used to create output files for the ODS destinations.

Graph templates work a bit differently. Starting in SAS 9.2, the LISTING destination supports styles for graphical output objects only. So the ODS PROCESS is a bit different. In Figure 4 we see that the ODS style template is used in the creation of graphical output. This is because of the underlying ODS Graphics Framework, which will be discussed in more detail in an upcoming section.

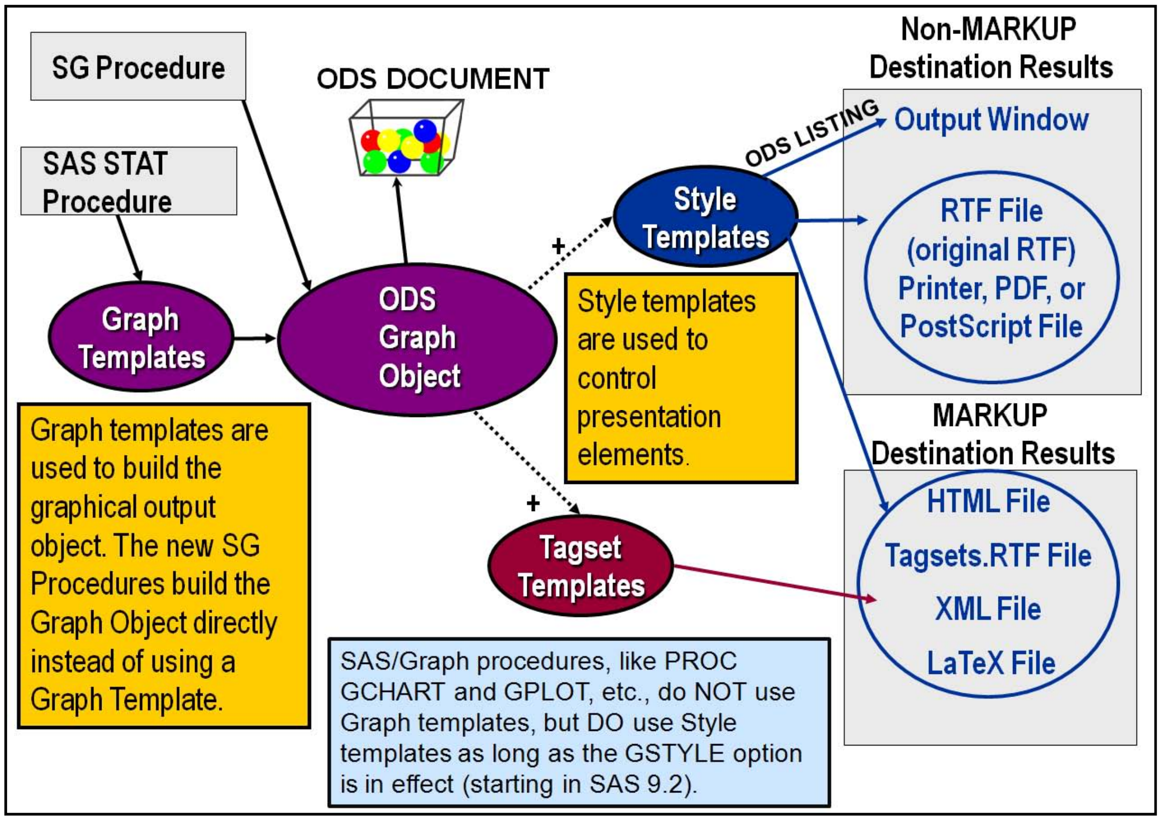

SG Procedure
SAS STAT Procedure
ODS DOCUMENT
Graph Templates
ODS Graph Object
Style Templates
Tagset Templates
ODS LISTING
+
+
Non-MARKUP Destination Results
Output Window
RTF File (original RTF) Printer, PDF, or PostScript File
MARKUP Destination Results
HTML File
Tagsets.RTF File
XML File
LaTeX File
Style templates are used to control presentation elements.
Graph templates are used to build the graphical output object. The new SG Procedures build the Graph Object directly instead of using a Graph Template.
SAS/Graph procedures, like PROC GCHART and GPLOT, etc., do NOT use Graph templates, but DO use Style templates as long as the GSTYLE option is in effect (starting in SAS 9.2).

Figure 4: Graph Templates and the ODS Process

As it says in the callout box in Figure 4, SAS/GRAPH® procedures such as PROC GCHART and PROC GPLOT do not use graph templates, although they do use style templates and in the course of creating the output, tagset templates might be used, too. However, what should be evident from Figure 3 and Figure 4 is not the specifics about individual procedures, but the fact that almost all tabular and most graphical output from SAS and SAS/STAT® use templates of one type or another. Every template type has its own specific PROC TEMPLATE syntax. Some template syntax uses DEFINE blocks, some template syntax uses STYLE or CLASS statements, and some template syntax uses LAYOUT statements. In the upcoming sections, I'll use some simple examples to illustrate how each different template type works. The last example will use all four template types together.

STURDY ROOT VEGETABLES: TABLE TEMPLATES

Table templates don't really grow underground, but they operate mostly behind the scenes of the ODS process. Almost every SAS procedure uses table templates. (The procedures that don't use table templates are called the "gnarly" four: PROC PRINT, PROC REPORT, PROC TABULATE,,and PROC FREQ "multi-way".)

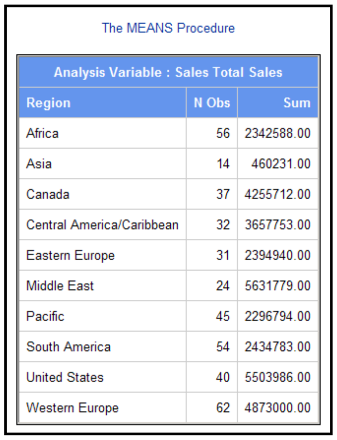
The easy way to find out which table templates are used by your procedures of choice is to either read the documentation for your procedure (the SAS/STAT documentation outlines the names of the output objects that are created by each procedure) or to just add another ODS "sandwich" to your program.

```
ODS TRACE ON / LABEL;
. . . SAS code . . .
ODS TRACE OFF;
```

Tracing ODS output objects reveals information about the output objects that are created by your procedure of choice. The information is specific to a particular invocation of a procedure with specific options. For example, a PROC FREQ with the CHISQ option uses a different table template than a PROC FREQ with the LIST option. All my examples use PROC MEANS, so let's look at the output from the ODS TRACE statement placed around a PROC MEANS program. If you submitted a simple PROC MEANS step with ODS TRACE, you would find the following information written to the SAS log:

```
Output Added:
-------------
Name:       Summary
Label:      Summary statistics
Template:   base.summary
Path:       Means.Summary
Label Path: 'The Means Procedure'.'Summary statistics'
-------------
```

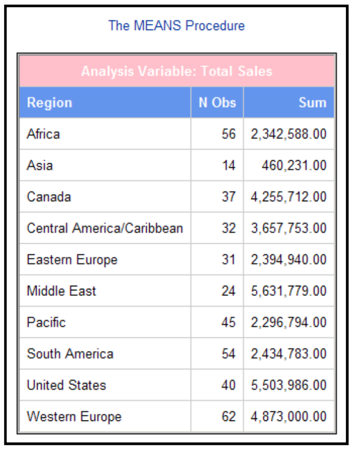
This trace information shows that the name of the output object is "Summary" and that the label of the output object is "Summary statistics", which is the label that you see in the SAS Results window. In addition, the name of the table template that is used to create the output for this step is "base.summary". This means that if you wanted to customize output from base.summary, you'd have to start with the BASE.SUMMARY template in the SASHELP.TMPLMST template item store. A simple PROC MEANS using SASHELP.SHOES produces the output shown in Figure 5:



The MEANS Procedure

Analysis Variable : Sales Total Sales		
Region	N Obs	Sum
Africa	56	2342588.00
Asia	14	460231.00
Canada	37	4255712.00
Central America/Caribbean	32	3657753.00
Eastern Europe	31	2394940.00
Middle East	24	5631779.00
Pacific	45	2296794.00
South America	54	2434783.00
United States	40	5503986.00
Western Europe	62	4873000.00

Figure 5:HTML Output from PROC MEANS



The MEANS Procedure

Analysis Variable: Total Sales		
Region	N Obs	Sum
Africa	56	2,342,588.00
Asia	14	460,231.00
Canada	37	4,255,712.00
Central America/Caribbean	32	3,657,753.00
Eastern Europe	31	2,394,940.00
Middle East	24	5,631,779.00
Pacific	45	2,296,794.00
South America	54	2,434,783.00
United States	40	5,503,986.00
Western Europe	62	4,873,000.00

Figure 6: Desired HTML Output from PROC MEANS

There are a few things in the output that need to change. The header above the three columns needs to change to "Analysis Variable: Total Sales" instead of the header that appears now, with both the variable name and the variable label. Another requirement is the use of the COMMA format for the SUM column. The desired new output from PROC MEANS, based on changes to the table template only, is shown in Figure 6.

The code that produced Figure 6 is shown below, with annotations next to relevant syntax.

Program Code	Notes

`ods path work.sumtemp(update)`
`         sasuser.templat(update)`
`         sashelp.tmplmst(read);`
	Issue an ODS PATH statement to store updated template in the WORK library.
	
	Only changed template statements are highlighted below. Any non-highlighted statements were copied directly from the table template for BASE.SUMMARY.

`proc template;`
`  edit base.summary;`
	Start PROC TEMPLATE by issuing the EDIT statement to create a copy of BASE.SUMMARY

`    header h ;`
`    define h;`
`       text "Analysis Variable: " one_var_label;`
`       space = 1;`
`       just = C;`
`       style=Header{background=pink};`
`       print = one_var;`
`       spill_margin;`
`    end;`
	The first change is to the header. The rest of the header definition is the same as in the original BASE.SUMMARY table template. The TEXT statement changes the header. ONE_VAR_LABEL is the dynamic item sent by PROC MEANS to the table template. This item contains the label being analyzed. The STYLE statement is responsible for changing the style of only this header. Note the override of the background color to PINK.

`    define sum;`
`       header = "Sum";`
`       format=comma14.2;`
`       generic;`
`    end;`
`  end;`
`run;`
	The FORMAT statement is the only statement added to the DEFINE block for the SUM statistic in the table template for BASE.SUMMARY

`ods html path='c:\temp' (url=none)`
`         file='means2.html'`
`         style=styles.sasweb;`
 	
` proc means data=sashelp.shoes sum;`
`  var sales;`
`  class region;`
`run;`
	When this PROC MEANS runs, it will automatically follow the ODS PATH to find the first copy of BASE.SUMMARY in the template stores available. Since WORK.SUMTEMP is the first template store in the ODS PATH, the changed BASE.SUMMARY is used for this PROC MEANS.

`ods _all_ close;`	

It is impossible, given the scope of this paper, to cover the entire syntax of table templates. The syntax for table templates allows style changes, spanning headers, trafficlighting, format changes, and justification changes, and you can even calculate columns by using COMPUTE syntax in a table template.

Not only can you alter a table template that is used with a SAS procedure, you can create a custom table template for use with DATA _NULL_ syntax, as described in my SUGI 30 paper "The Power of TABLE Templates and DATA _NULL_."

The table template is an essential element of ODS, used to produce your procedure output in its final structure. But, unless you are sending output only to the LISTING destination, any time you use ODS to route output to the RTF, PDF, or HTML destinations, a style template is also used. For example, in Figure 5, we saw the output with just the SASWEB style applied, while in Figure 6, we overrode the default blue header style with a pink header, to highlight the header that changed. In the LISTING destination, you would see only the header text change and the format change, but in destinations that support style, you can make style changes in your table template and they will override any style settings that come from the ODS style template. This fact leads to the next discussion on SAS style templates.

FLOWERING BULBS: STYLE TEMPLATES

Flowers that start from bulbs are the delight of the spring and summer garden. Daffodils, tulips, lilies, coral drops, gladioli, begonias, anemones, dahlias, and amaryllis. SAS style templates start behind the scenes of the ODS process. In destinations that do *not* support style attributes, like LISTING and CSV, an ODS style template is dormant and unused, like a tulip bulb sleeping for the winter. But when an ODS destination supports style attributes, SAS style templates add the same color and style to your output that flowering bulbs add to your garden in the spring.

Style templates live in the STYLES folder in the SASHELP.TMPLMST template store, as shown in the output from Figure 2. SAS style templates are verbose. The "grand-daddy" style template, STYLES.DEFAULT, is over 12 pages long if you print it. The syntax for style templates is very different from the syntax for table templates. The paradigm behind ODS style templates is the fact that most of your SAS output is tabular in structure. It doesn't matter whether you use PROC GLM or PROC TABULATE or PROC FREQ or PROC MEANS; if the procedure produces output, the output is in a tabular structure. The tables will all have header cells and data cells. The tabular output might or might not have titles and footnotes, depending on whether your code has a SAS TITLE statement or a SAS FOOTNOTE statement. The way that a style template works is modeled on how cascading style sheets work with HTML pages.

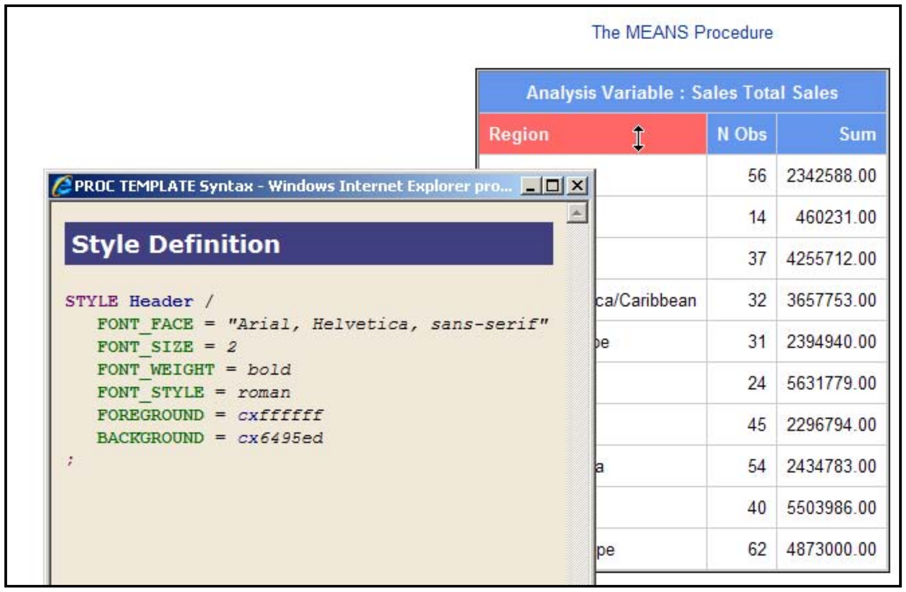
However, since the RTF and PDF destinations did not support style, as specified in the CSS specification, when ODS was first introduced, ODS style templates were designed to provide style attribute information in a way that ODS destinations could use. If the central units of table templates are the table structure of columns and headers, the central units of style templates are style elements and style attributes.

A style element such as SystemTitle or Header is actually a syntax mechanism for referencing a group of style attributes. This is very similar to how a CSS class selector is linked to a group of style property/value pairs. Since SAS procedures know what tabular structure they will produce (based on a procedure's table template), it is possible for SAS style templates to be linked to the tabular structure via style element names.

If you use the STYLE_POPUP tagset template to create output from PROC MEANS, it will help you understand that a style element is a collection of style attributes. This SAS code executes the same PROC MEANS step as the previous example. However, the output has embedded JavaScript, which enables you to review the resolved style attribute inheritance.

```
ods markup type=style_popup path='c:\temp' (url=none)
         file='means_pu.html'
         stylesheet='spu.css'
         style=styles.sasweb;

 proc means data=sashelp.shoes sum;
  var sales;
  class region;
run;

ods _all_ close;
```

Results from PROC MEANS will be the same as the previous output, but the HTML file has some extra information. When you hover your mouse over an output component, such as the header or the procedure title, you will see the output component change color (sort of salmon pink). Then, when you double-click in the highlighted cell, you will see a pop-up window that contains style information, as shown in Figure 7 for the Header style element and in Figure 8 for the ProcTitle style element.



The MEANS Procedure

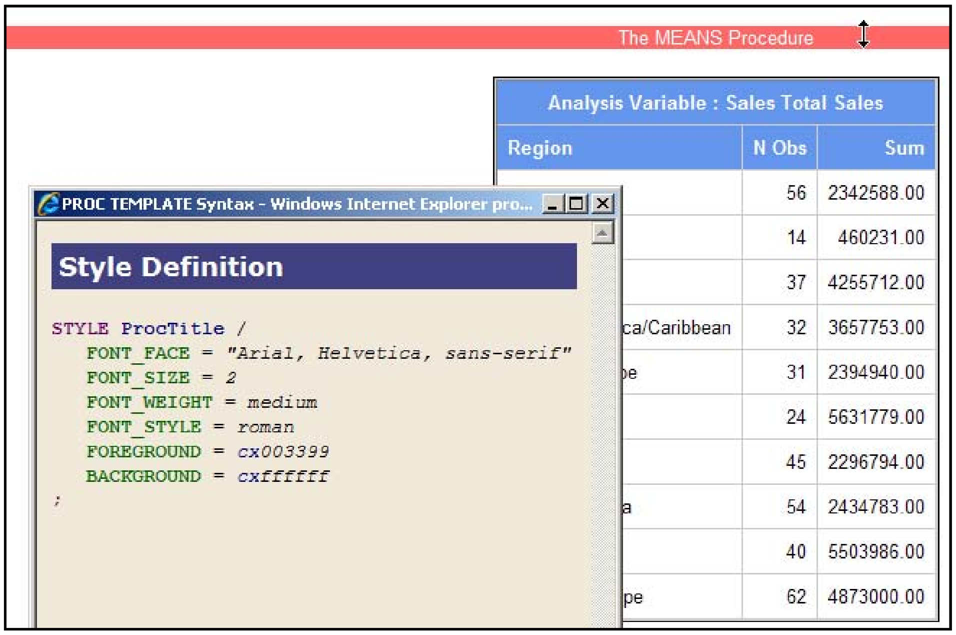
Analysis Variable : Sales Total Sales		
Region	N Obs	Sum
	56	2342588.00
	14	460231.00
	37	4255712.00
ca/Caribbean	32	3657753.00
pe	31	2394940.00
	24	5631779.00
	45	2296794.00
a	54	2434783.00
	40	5503986.00
pe	62	4873000.00

PROC TEMPLATE Syntax - Windows Internet Explorer pro...

Style Definition

```
STYLE Header /
   FONT_FACE = "Arial, Helvetica, sans-serif"
   FONT_SIZE = 2
   FONT_WEIGHT = bold
   FONT_STYLE = roman
   FOREGROUND = cxffffff
   BACKGROUND = cx6495ed
;
```

Figure 7: Header Style Element Pop-up Window



The MEANS Procedure

Analysis Variable : Sales Total Sales		
Region	N Obs	Sum
	56	2342588.00
	14	460231.00
	37	4255712.00
ca/Caribbean	32	3657753.00
pe	31	2394940.00
	24	5631779.00
	45	2296794.00
a	54	2434783.00
	40	5503986.00
pe	62	4873000.00

PROC TEMPLATE Syntax - Windows Internet Explorer pro...

Style Definition

```
STYLE ProcTitle /
   FONT_FACE = "Arial, Helvetica, sans-serif"
   FONT_SIZE = 2
   FONT_WEIGHT = medium
   FONT_STYLE = roman
   FOREGROUND = cx003399
   BACKGROUND = cxffffff
;
```

Figure 8: ProcTitle Style Element Pop-up Window

Notice how the name of the style element (Header or ProcTitle) appears in the pop-up window. The style attributes that are collected under that element name are shown after the slash. In fact, if you want to cut and paste from the keyword 'STYLE' down to the semi-colon from this pop-up window, you have a complete, syntactically correct STYLE statement for use in a STYLE template.

The style for table headers is automatically set to the use the Header style element. So, if the previous program were changed, as shown in the program below, the results would be as shown in Figure 9, where you can see that all the headers use the new style elements.

Program Code	Notes

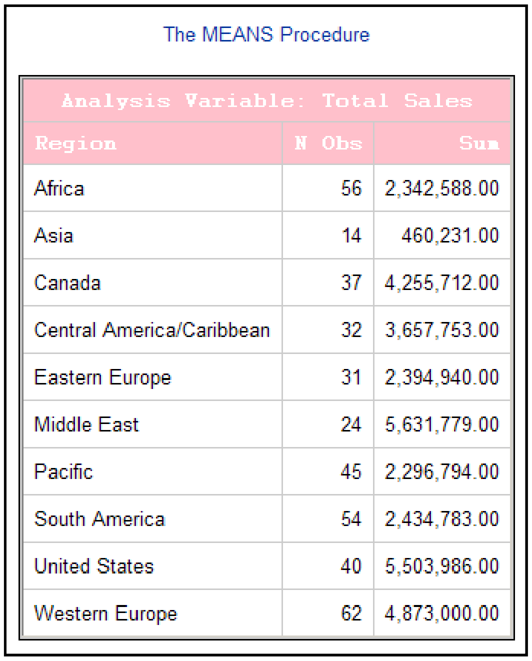
`ods path work.sumtemp(update)`
`         sasuser.templat(update)`
`         sashelp.tmplmst(read);`
 	
`title;`
	Again, issue an ODS PATH statement so that the updated templates are stored in the WORK.SUMTEMP item store.

`proc template;`
`  define style styles.pink;`
`    parent=styles.sasweb;`
`    style Header /`
`      font_face = "Courier"`
`      font_weight = bold`
`      foreground = white`
`      background = pink;`
`  end;`
`run;`
	This style template will inherit all the style elements from STYLES.SASWEB, except the style for the Header element.

`proc template;`
`  edit base.summary;`
`    header h ;`
`    define h;`
`       text "Analysis Variable:  " one_var_label;`
`       space = 1;`
`       just = C;`
`       print = one_var;`
`       spill_margin;`
`    end;`
 	
`    define sum;`
`       header = "Sum";`
`       format=comma14.2;`
`       generic;`
	This version of the table template contains the header text change and the format change for SUM, but does not have the style change of the previous example. This means that the header style element from the style template will control ***all*** the headers in the output.

`    end;`
`  end;`
`run;`
 	
`ods html path='c:\temp' (url=none)`
`         file='means_pink.html'`
`         style=styles.pink;`
 	
` proc means data=sashelp.shoes sum;`
`  var sales;`
`  class region;`
`run;`
 	
`ods _all_ close;`
	The STYLES.PINK style template is used in the STYLE= option of the ODS HTML invocation. The rest of the code is the same as in the previous example.

Note that the style attributes from STYLES.PINK (Courier, bold, pink background, white foreground) are used for all the headers in the output. The rest of the style attributes are set by the style elements in STYLES.SASWEB.



The MEANS Procedure

Analysis Variable: Total Sales		
Region	N Obs	Sum
Africa	56	2,342,588.00
Asia	14	460,231.00
Canada	37	4,255,712.00
Central America/Caribbean	32	3,657,753.00
Eastern Europe	31	2,394,940.00
Middle East	24	5,631,779.00
Pacific	45	2,296,794.00
South America	54	2,434,783.00
United States	40	5,503,986.00
Western Europe	62	4,873,000.00

Figure 9: PROC MEANS Output with STYLES.PINK Style Template Used for Headers

There is no longer any need to distinguish between the STYLE statement and the REPLACE statement, which was one of the trickier aspects of writing style templates in earlier versions of SAS. In addition, the developers have introduced the CLASS statement, which enables you to create a style element from a style element of the same name, if one exists. In the syntax above, we used the STYLE statement. But we could just as easily have used the keyword CLASS instead of the keyword STYLE. The output would have been the same.

In earlier versions of SAS, before SAS 9.2, you had to worry about the style element inheritance. However, SAS 9.2 ushered in a new, more streamlined, form of style template syntax. In addition, using the diagnostic tagset template, STYLE_POPUP, to identify style attributes simplified the whole process of modifying a style template. So, let's turn our attention to the topic of tagset templates.

THORNY ROSE BUSHES: TAGSET TEMPLATES

Rose bushes take a lot of work to maintain and grow. In spite of their thorns, rose bushes are among the most beautiful flowers in the garden. They reward diligent effort with great enjoyment. The thorny issue with tagset templates is that the terminology "tagset" is used to describe HOW output is produced by the ODS MARKUP destination, using Tagset templates; and also describes WHAT is being produced – a set of tags (or tagset) that "marks up" your SAS procedure or process output. It is easy, when starting to study tagset templates, to get lost in what seems to be a circular and confusing instruction to use a tagset or tagset destination in order to produce tagset output.

More precisely, when you generate Markup family destination output, you are using ODS MARKUP, with a specific tagset template definition to produce an ASCII text file that conforms to a particular specification for how text (your SAS output) should be "marked up with a pre-defined set of markup tags" in order to be rendered by a particular software application. For example, when you create ODS HTML output, you are actually using the ODS MARKUP destination, with the HTML4 tagset template definition to produce an HTML ASCII text file that conforms to the W3C specification for an HTML 4.01 file, which can be rendered with any browser that conforms to the HTML 4.01 specification. Or, when you create ODS CSV output, you are using the ODS MARKUP destination, with the CSV tagset template definition, to produce an ASCII text file with comma-separated values, which can be rendered with any software application (Lotus, Excel, Microsoft Access, Harvard Graphics, ClarisWorks) that can open and render a CSV file. Whew! It's like reading the Latin genus and species for a plant.

My SUGI 29 paper " Markup 101: Markup Basics" discusses the ODS MARKUP destination and markup languages and markup basic concepts in great detail. The method by which "marked up" output files are created by ODS is through the use of tagset templates. The key concept behind tagset templates is that as SAS executes a procedure, it sends events to each open destination. If the open destination is controlled by a tagset template, then each event is handled as specified by the DEFINE EVENT block inside the tagset template.

Of course, some destinations, like the original ODS HTML (in SAS® 8) or the original ODS RTF or ODS PDF, are *not* ODS Markup destinations. The method of creating the output files for these destinations is "hard-coded" in the SAS executable file. You can't change how ODS PDF works internally, except through the use of PDF options. When ODS PDF gets a column header, it uses information from the style template to format the header cell. But with HTML-based markup destinations, if you wanted to issue an <H1> tag or a <P> tag or an "on click" JavaScript instruction, you could do that by changing the event that handles the header cell text.

This is why you will hear people say that tagset templates are "event-driven" in addition to being "destination-driven." This means that SAS sends an event (like the System_Title event or the Header event) and each destination, if it is a markup family destination, has the potential to write different markup tags for the value that is carried by the event. A concrete example of how an "event" works for different destinations is the case where a SAS TITLE statement in your program triggers a System_Title event. A program with this statement:

```
title 'My Title';
```

will generate different types of markup output for the same title string, based on how the System_Title event was defined in the tagset template for these ODS Markup destinations:

Destination	Type of Markup Output Written
ODS HTML	`<td class="c SystemTitle">My Title</td>`
ODS LaTeX	`\sassystemtitle[c]{My Title}`
ODS DOCBOOK	`<title>My Title</title>`
ODS TAGSETS.EXCELXP	`<x:Header  ss:StyleID="systemtitle" Data="My Title"/>`

Table 2: How Tagset Templates Use Events to Write Destination-Specific Markup Language Tags

In the above examples, you can see that the text string "My Title" stayed the same in each of the four output examples, but the markup tags that surrounded the title string were all different, as needed by the programs that will render the output files.

If you wanted to change the procedure title, "The Means Procedure", to use the Header style element, you could just change the style template element that is used for the procedure title, the ProcTitle style element (as shown in Figure 8). However, another place where you have a chance to change the output being written is in the tagset template. You can alter the event that handles writing out the procedure title. Not surprisingly, this event is called the PROC_TITLE event (with an underscore to distinguish the event from the style element).

If you examine a tagset template, you will discover that most of the processing in a tagset template involves executing PUT statements to write tags around SAS output. Normally, when ODS HTML writes a ProcTitle, like "The Means Procedure", it writes the following HTML tag:

```
<div class="c ProcTitle">The MEANS Procedure</div>
```

The CLASS= HTML attribute is used to link the text string "The MEANS Procedure" with the ProcTitle style element, which is defined in the style template. If you wanted the Header element to format the procedure title, you could change the style template, as we have seen in the previous example. However, note in the definition for TAGSETS.PT below, how the <DIV> tag in the output was modified to use a different CLASS= attribute in the HTML output file.

Program Code	Notes

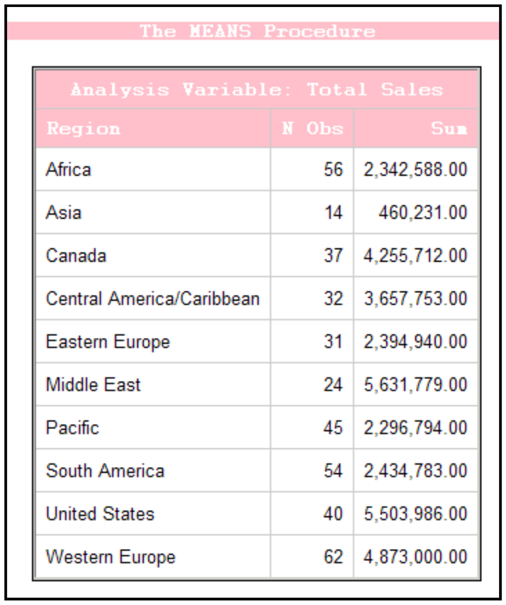
`ods path work.sumtemp(update)`
`         sasuser.templat(update)`
`         sashelp.tmplmst(read);`
	Issue the ODS PATH statement.

`proc template;`
`  define tagset tagsets.PT;`
`    parent=tagsets.html4;`
`    define event proc_title;`
`      put '<div class="c Header"';`
`      put '>';`
`      put VALUE ;`
`      put "</div>" NL;`
`    end;`
`    image_formats = "png,gif,jpg,svg,xml";`
`    output_type = "html";`
`  end;`
`run;`
	Define TAGSETS.PT to handle the PROC_TITLE event differently than it is handled by the HTML4 tagset template. In this tagset template, the PROC_TITLE event will write the type of <DIV> tag that contains a CLASS="Header" attribute.

`proc template;`
`  define style styles.pink;`
`    . . . same code as previous example . . .`
`  end;`
`run;`
	The style template is unchanged from the previous example.

`proc template;`
`  edit base.summary;`
`      . . . same code as previous example. . .`
`end;`
`run;`
	The table template is unchanged from the previous example.

`ods tagsets.pt path='c:\temp' (url=none)`
`         file='means_pt.html'`
`         style=styles.pink;`
 	
` proc means data=sashelp.shoes sum;`
`  var sales;`
`  class region;`
`run;`
 	
`ods _all_ close;`
	The TAGSETS.PT destination is used to create the special HTML output. Because the STYLES.PINK style template is used, the reference to the Header element causes the procedure title to have the same style characteristics as the PROC MEANS table headers, as shown in Figure 10.



The MEANS Procedure

Analysis Variable: Total Sales		
Region	N Obs	Sum
Africa	56	2,342,588.00
Asia	14	460,231.00
Canada	37	4,255,712.00
Central America/Caribbean	32	3,657,753.00
Eastern Europe	31	2,394,940.00
Middle East	24	5,631,779.00
Pacific	45	2,296,794.00
South America	54	2,434,783.00
United States	40	5,503,986.00
Western Europe	62	4,873,000.00

Figure 10: PROC MEANS Output Created with TAGSETS.PT

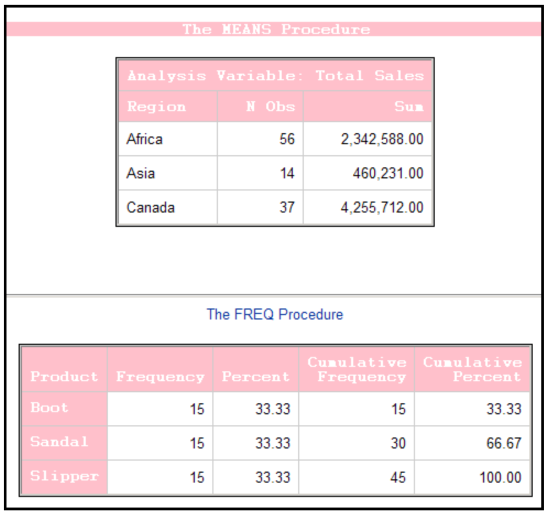
So, no big deal, right? This could have been done with a style template, thank you very much. Can we all leave the paper, now?

What if you wanted to change the procedure title *only* for the MEANS procedure. What if, for any other procedure, you wanted to use the default <DIV> tag with the ProcTitle style element? This is the beauty of tagset templates in practice. You can code conditional logic into your tagset template to control what markup information is written to your output. Suppose that the above tagset template were changed just a bit:

Program Code	Notes

```
proc template;
  define tagset tagsets.PT_alt;
    parent=tagsets.html4;
    define event proc_title;
      do / if index(VALUE,'MEANS') gt 0;
        put '<div class="c Header"';
      else;
        put '<div class="c ProcTitle"';
      done;

      put '>';
      put VALUE ;
      put "</div>" NL;
    end;
    image_formats = "png,gif,jpg,svg,xml";
    output_type = "html";
  end;
run;

ods tagsets.pt_alt path='c:\temp' (url=none)
         file='means_pt_alt.html'
         style=styles.pink;
 proc means data=sashelp.shoes sum;
run;

proc freq data=sashelp.shoes;
run;

ods _all_ close;
```
	The same ODS PATH statement as in the previous examples is still in effect.
	
	This PROC TEMPLATE code defines the PT_alt tagset template. The DO block uses a SAS function to determine whether the event variable, VALUE, for the PROC_TITLE event is carrying the text "The MEANS Procedure", and if so, to write one type of <DIV> tag. If the VALUE event variable is carrying any other procedure title, then the regular <DIV> tag will be written.
	
	Although the construction of the DO block needs the slash and the condition after the keyword DO (unlike a regular DATA step IF statement), the end result of the conditional processing is evident in Figure 11. Different tags were written for the different procedure titles.

Then, if a PROC FREQ step were added to the previous job, you would see the output with a different style for each procedure title, as shown in Figure 11. (In the interests of making both outputs fit in one screen shot, only three regions and three products were used from the data.)


The MEANS Procedure

Analysis Variable: Total Sales		
Region	N Obs	Sum
Africa	56	2,342,588.00
Asia	14	460,231.00
Canada	37	4,255,712.00

The FREQ Procedure

Product	Frequency	Percent	Cumulative Frequency	Cumulative Percent
Boot	15	33.33	15	33.33
Sandal	15	33.33	30	66.67
Slipper	15	33.33	45	100.00

Figure 11: PROC MEANS and PROC FREQ Output Created with TAGSETS.PT_ALT

If you examined the HTML code (by looking at the HTML file in Notepad), you would see these two different <DIV> tags:

```
<div class="c Header">The MEANS Procedure</div>
<div class="c ProcTitle">The FREQ Procedure</div>
```

Although this is a simple example, the implications of its usage, in practice, are huge. Not only can you customize the HTML tags (or other markup tags) that are written for your output, but you can test the value of specific event variables to further control or fine tune what is written out.

Of course, this means you must know your data and the type of events that your procedure of interest will generate, and you must know the syntactically correct markup tags for the conditional change that you want to make. Highlighting this one simple, yet powerful, change, should be all the incentive you need to brave the thorns in the tagset template rose bushes! And, just as there was a diagnostic tagset template to help us understand style elements and style attributes, there are several diagnostic tagset templates to help you understand which events your procedure is sending to ODS and in what order the events are sent and what event variables are carried for each event. In the zip file of downloadable programs for this presentation, there is an example of generating EVENT_MAP output that shows event information for the above PROC MEANS and PROC FREQ (in the Demo4 program).

Last, but certainly not least, we are going to turn our attention to the newest addition in the template garden, graph templates.

AN ABUNDANCE OF FLOWERS: GRAPH TEMPLATES

Graph templates are just one component of the ODS Graphics Framework. If you have SAS/STAT and SAS/GRAPH installed, then you can generate graphical output automatically, using the graph templates that are part of ODS. The general syntax model for using procedures that support ODS Graphics is shown in the code snippet below, using PROC FREQ to produce LISTING output:

```
options gstyle;
ods listing style=ocean;
ODS GRAPHICS ON;
  proc freq data=sashelp.shoes;
    tables product;
  run;
ODS GRAPHICS OFF;
```

What's that, you say? You've never seen the STYLE= option used on an ODS LISTING statement? That's part of the beauty of the ODS Graphics Framework. By default, all graph output (including SAS/GRAPH output) uses SAS style template information, starting in SAS 9.2. The GSTYLE option is the mechanism through which the use of SAS style templates is turned on for automatic graph creation. The NOGSTYLE option would have to be in effect for you to turn off the use of SAS style templates in graph creation.

Output from the above PROC FREQ step is shown in Figure 12 for one of the graph outputs. The output from PROC FREQ can be viewed only in the LISTING window, whereas the output from PROC FREQ can be viewed with the default PNG viewer on your system. On my Windows XP system, the default picture viewer is the Windows Picture and Fax Viewer, as shown in Figure 12.

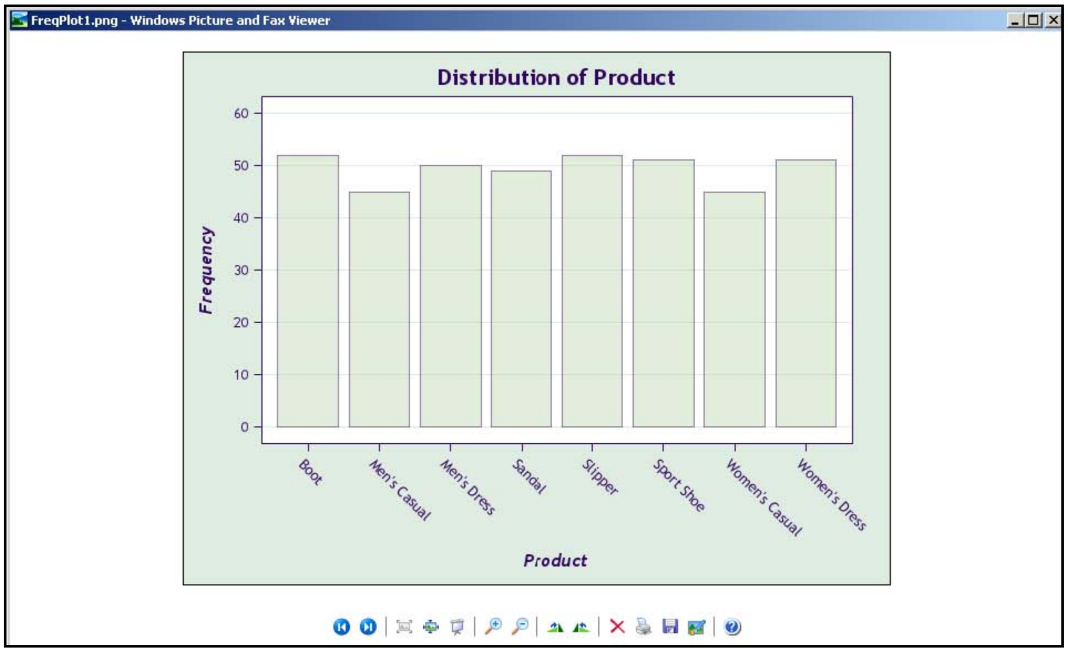


Figure 12: PROC FREQ Graphic Output Viewed with Windows Picture Viewer

It's awkward to look at the tabular output in the LISTING window and the graphical output in the default picture viewer. If you want to see all the output in an ODS output file, then your regular ODS "sandwich" invocation for RTF, PDF, or HTML should be used around your program statements. This code would route the tabular and graphical output objects to the other ODS destinations.

Program Code

```
ods pdf file='c:\temp\demo5_fr.pdf'
        style=journal;
ods rtf path='c:\temp' (url=none)
        file='demo5_fr.rtf'
        style=journal;
ods html path='c:\temp' (url=none)
         gpath='c:\temp' (url=none)
         file='demo5_fr.html'
         style=styles.sasweb;

ODS GRAPHICS ON;
  proc freq data=sashelp.shoes;
    tables product;
  run;

ODS GRAPHICS OFF;

ods _all_ close;
```

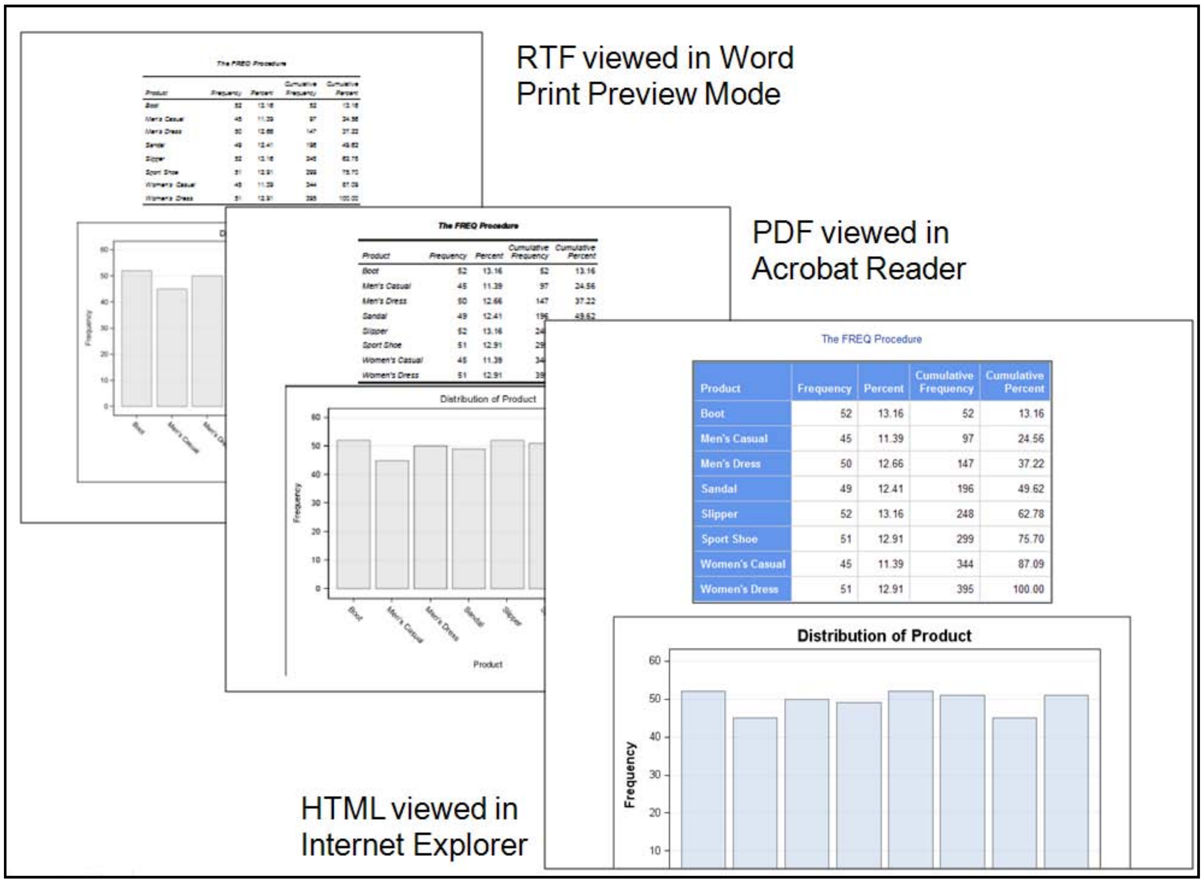
Notes

Use regular ODS invocation statements for any destination to which we want to send the output.

Turn on the ODS Graphics Framework.

Turn off the ODS Graphics Framework.

Close all open destinations.

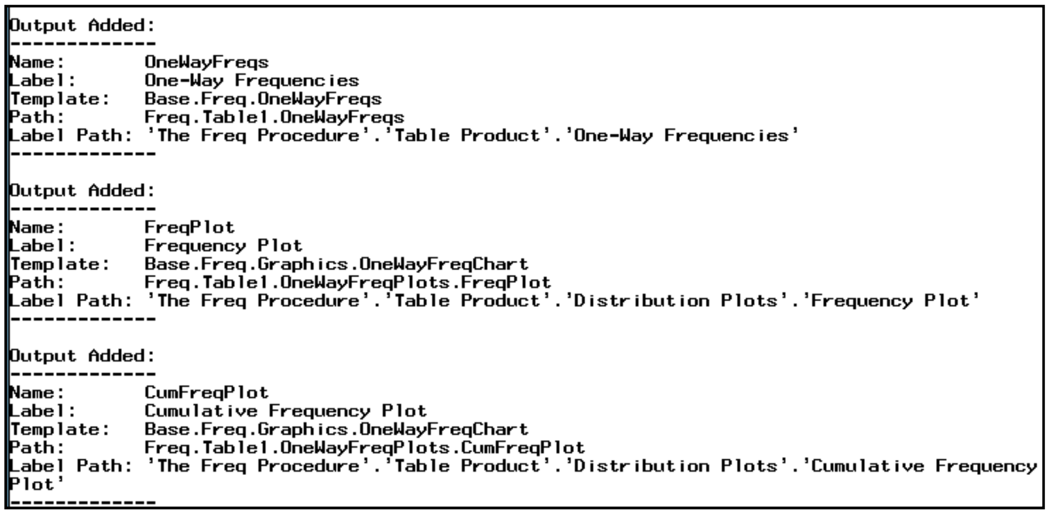


The FREQ Procedure

Product	Frequency	Percent	Cumulative Frequency	Cumulative Percent
Boot	52	13.16	52	13.16
Men's Casual	45	11.39	97	24.56
Men's Dress	50	12.66	147	37.22
Sandal	49	12.41	196	49.62
Slipper	52	13.16	248	62.78
Sport Shoe	51	12.91	299	75.70
Women's Casual	45	11.39	344	87.09
Women's Dress	51	12.91	395	100.00

Figure 13: Tabular and Graphical ODS Output from PROC FREQ

Output from this code, for all destinations, is shown in Figure 13. Note that the RTF and PDF output use the JOURNAL style, and the HTML uses the SASWEB style. If you had turned on an ODS TRACE for the output, you would have seen these output objects in the SAS log, as shown in Figure 14.



```
Output Added:
-------------
Name:       OneWayFreqs
Label:      One-Way Frequencies
Template:   Base.Freq.OneWayFreqs
Path:       Freq.Table1.OneWayFreqs
Label Path: 'The Freq Procedure'.'Table Product'.'One-Way Frequencies'
-------------

Output Added:
-------------
Name:       FreqPlot
Label:      Frequency Plot
Template:   Base.Freq.Graphics.OneWayFreqChart
Path:       Freq.Table1.OneWayFreqPlots.FreqPlot
Label Path: 'The Freq Procedure'.'Table Product'.'Distribution Plots'.'Frequency Plot'
-------------

Output Added:
-------------
Name:       CumFreqPlot
Label:      Cumulative Frequency Plot
Template:   Base.Freq.Graphics.OneWayFreqChart
Path:       Freq.Table1.OneWayFreqPlots.CumFreqPlot
Label Path: 'The Freq Procedure'.'Table Product'.'Distribution Plots'.'Cumulative Frequency
Plot'
-------------
```

Figure 14: ODS TRACE Tabular and Graphical Output Objects

The graph templates and graph objects that you see in the ODS TRACE output are the visible manifestation of the ODS Graphics Framework.

ODS GRAPHICS FRAMEWORK

The ODS Graphics Framework is the infrastructure or umbrella under which different graphical components of the SAS System are created. As we talk about graph templates, it is important to note that not all graphical output created by SAS uses graph templates. Figure 15 shows the four major components of the ODS Graphics Framework.

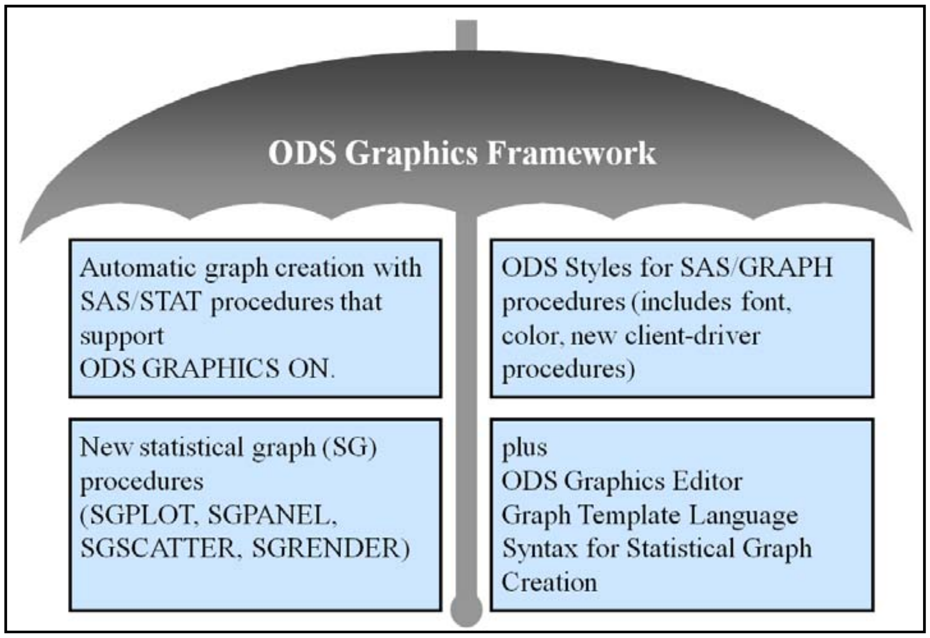

ODS Graphics Framework
Automatic graph creation with SAS/STAT procedures that support ODS GRAPHICS ON.
ODS Styles for SAS/GRAPH procedures (includes font, color, new client-driver procedures)
New statistical graph (SG) procedures (SGPLOT, SGPANEL, SGSCATTER, SGRENDER)
plus ODS Graphics Editor Graph Template Language Syntax for Statistical Graph Creation

Figure 15: ODS Graphics Framework Umbrella

The SAS/STAT procedures that support ODS GRAPHICS, as shown in Table 3, use SAS graph templates and SAS style templates; SAS/GRAPH procedures such as PROC GCHART, GPLOT, and others use only SAS style templates.

Base SAS	SAS/STAT		SAS/QC	SAS/ETS
CORR	ANOVA	MI	ANOM	ARIMA
FREQ	BOXPLOT	MIXED	CAPABILITY	AUTOREG
UNIVARIATE	CALIS	MULTTEST	CUSUM	ENTROPY
	CLUSTER	NPAR1WAY	MACONTROL	EXPAND
	CORRESP	PHREG	PARETO	MODEL
	FACTOR	PLS	RELIABILITY	PANEL
	FREQ	PRINCOMP	SHEWHART	RISK
	GAM	PRINQUAL		SIMILARITY
	GENMOD	PROBIT		SYSLIN
	GLIMMIX	QUANTREG .		TIMESERIES
	GLM	REG .		UCM
	GLMSELECT	ROBUSTREG		VARMAX
	KDE	RSREG		X12
	KRIGE2D	SEQDESIGN		
	LIFEREG	SEQTEST		
	LIFETEST	SIM2D		
	LOESS	TCALIS		
	LOGISTIC	TRANSREG		
	MCMC	TTEST		
	MDS	VARIOGRAM		

Table 3: Procedures in SAS 9.2 That Support the ODS Graphics Framework

The statistical "SG" procedures, SGPLOT, SGSCATTER, and SGPANEL, do not use graph templates, but like SAS/GRAPH procedures do use SAS style templates. The Graph Template Language is the syntax with which all the graph templates were built. That means you can design your own graph template and render it with the SGRENDER procedure and the SAS style template of your choice will be used to style the output. In addition, there is a graph editor that you can use to make non-persistent changes to your graphic output from ODS. Parts of the ODS Graphic Framework are being integrated into SAS Enterprise Guide, as well.

Rather than modify an existing graph template, the next example illustrates how to generate your own graph using the Graph Template Language.

Program Code	Notes

```
ods path work.sumtemp(update)
         sasuser.templat(update)
         sashelp.tmplmst(read);
```
	Use ODS PATH statement to put the graph template in the same item store with the other templates that have been created.

```
proc template;
define statgraph graph.regsales;
  mvar _wanty;
  begingraph;
    entrytitle _WANTY;
    layout overlay /
           xaxisopts=(label="Region")
           yaxisopts=(label=_WANTY)
           cycleattrs=true;
      barchart x=region y=_wanty;
    endlayout;
```
	This PROC TEMPLATE code will put the updated template in the first writeable item store in the ODS PATH.
	The MVAR statement establishes a macro variable name that will be used for column headers and graph variables.
	The BEGINGRAPH//ENDGRAPH statements start the graph, and every statement within this statement block will cause something to be written to the output graph. The ENTRYTITLE statement will use the macro variable at template execution time. Next, the LAYOUT statement specifies OVERLAY and options that should be used for the X and Y axes. This code is generating a simple bar chart,

```
  endgraph;
end;
run;

ods html path='c:\temp' (url=none)
         gpath='c:\temp' (url=none)
         file='demo5.html'
         style=styles.sasweb;

%let _wanty = Sales;
  proc sgrender data=sashelp.shoes
       template=graph.regsales;
  run;

%let _wanty = Returns;
  proc sgrender data=sashelp.shoes
       template=graph.regsales;
  run;

ods _all_ close;
```

where the X axis will always be the REGION variable and the Y axis will be the current macro variable _WANTY. The ENDLAYOUT statement and ENDGRAPH statement close the graph portion of the template, and the END statement closes the beginning DEFINE statement.

For the first invocation of the graph template, the macro variable _WANTY is set to SALES. Then PROC SGRENDER is used to render the graph and to bind it to the data in SASHELP.SHOES.

For the second invocation of the same graph template, the macro variable _WANTY is set to RETURNS. Then PROC SGRENDER will render the graph and will bind it to the data, this time using the RETURNS variable as the value to use for the Y axis.

The partial output from the above program is shown in Figure 16.

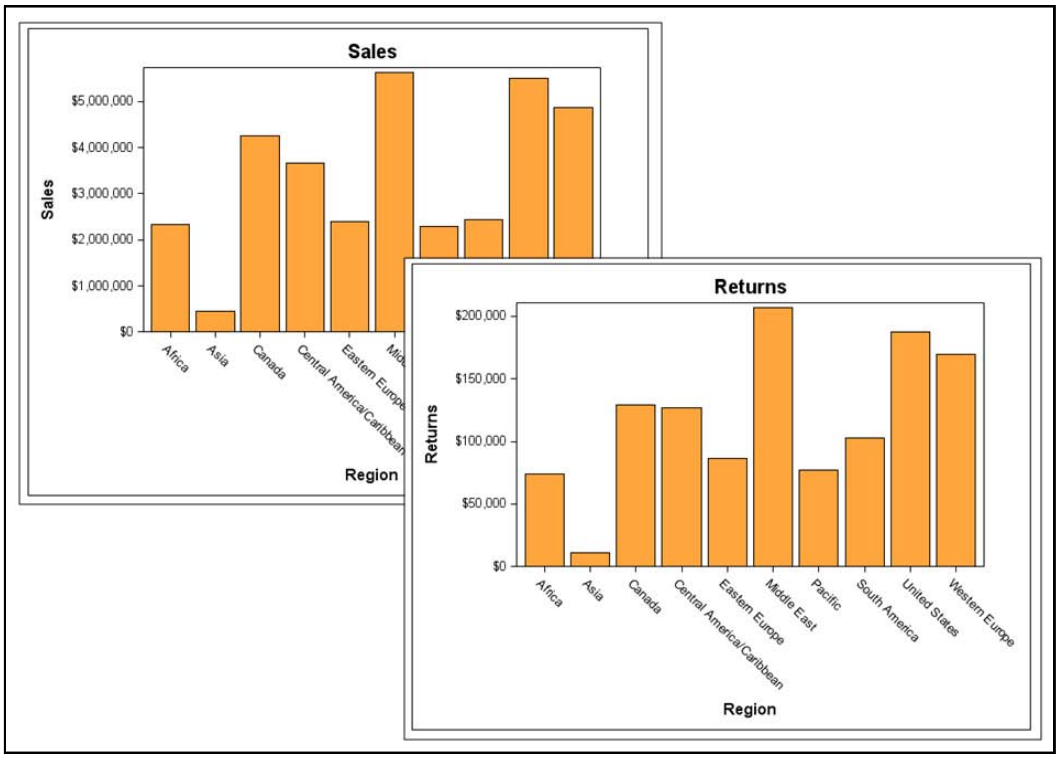


Figure 16: ODS HTML Output Using PROC SGRENDER and a Custom Graph Template

CONCLUSION

The following program illustrates the use of all four template types in one report.

Program Code	Notes

`ods path work.sumtemp(update)`
`         sasuser.templat(update)`
`         sashelp.tmplmst(read);`
	ODS PATH statement points to the item stores.

`proc template;`
`  define tagset tagsets.PT;`
`    parent=tagsets.html4;`
`    define event proc_title;`
`      put '<div class="c RedPT"';`
`      put '>';`
`      put VALUE ;`
`      put "</div>" NL;`
`    end;`
`    image_formats = "png,gif,jpg,svg,xml";`
`    output_type = "html";`
`  end;`
`run;`
	Tagset template (without conditional logic) uses a different class selector (REDPT) for the procedure title.

`proc template;`
`  edit base.summary;`
`     header h ;`
`     define h;`
`      text "Analysis Variable: " one_var_label;`
`      space = 1;`
`      just = C;`
`      style=PinkHdr;`
`      print = one_var;`
`      spill_margin;`
`    end;`
 	
`      define sum;`
`         header = "Sum";`
`         format=comma14.2;`
`         generic;`
`      end;`
`  end;`
`run;`
	Table template for PROC MEANS uses a different class selector (PINKHDR) for the spanning header.

`proc template;`
`  define style styles.special;`
`    parent=styles.sasweb;`
`    import 'chgstyle.css';`
`    class GraphColors /`
`       'gdata12' = cxDDD17E`
`       'gdata11' = cxB7AEF1`
`       'gdata10' = cx336699`
`       'gdata9' = beige`
`       'gdata8' = graycc`
`       'gdata6' = blue`
`       'gdata7' = green`
`       'gdata4' = red`
`       'gdata5' = purple`
`       'gdata3' = yellow`
`       'gdata2' = cyan`
`       'gdata1' = pink;`
`  end;`
	The style template inherits from STYLES.SASWEB, except for the CSS style sheet that is being imported with the IMPORT statement. In addition, the GraphColors style element is being changed so the default color of GDATA1 (pink) will be used for the chart color values.

`run;`	

`proc template;`
`define statgraph graph.regsales;`
`  mvar _wanty;`
`  begingraph;`
`    entrytitle _WANTY;`
`    layout overlay /`
`           xaxisopts=(label="Region")`
`           yaxisopts=(label=_WANTY)`
`           cycleattrs=true;`
`      barchart x=region y=_wanty;`
`    endlayout;`
`  endgraph;`
`end;`
`run;`
	The statgraph template is the same as previously illustrated.

`ods tagsets.PT path='c:\temp' (url=none)`
`         gpath='c:\temp' (url=none)`
`         file='sum_pt.html'`
`         style=styles.special;`
	The new tagset template is used for the ODS invocation statement, and the new style template is specified in the STYLE= option.

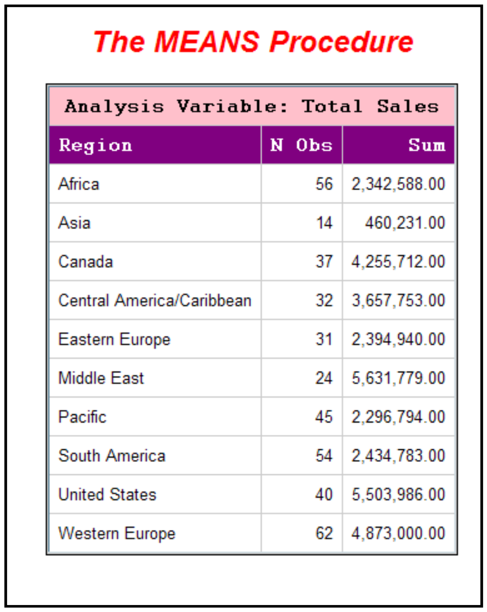
`ods graphics on;`
` proc means data=sashelp.shoes sum;`
`  var sales;`
`  class region;`
`  output out=work.sales sum=sumsale;`
`run;`
	PROC MEANS will automatically retrieve the most current version of BASE.SUMMARY from the SUMTEMP item store.

`%let _wanty = Sales;`
`  proc sgrender data=sashelp.shoes`
`       template=graph.regsales;`
`  run;`
 	
`%let _wanty = Returns;`
`  proc sgrender data=sashelp.shoes`
`       template=graph.regsales;`
`  run;`
`ods graphics off;`
`ods _all_ close;`
	The next two SGRENDER steps are the same as those previously described.

The full text of the CSS file, CHGSTYLE.CSS, is shown below:

```
.PinkHdr{
  font-family: Courier, Courier New, serif;
  font-size: medium;
  font-weight: bold;
  font-style: normal;
  color: black;
  background-color: pink;}
.Header{
  font-family: Courier, Courier New, serif;
  font-size: medium;
  font-weight: bold;
  font-style: normal;
  color: white;
  background-color: purple;}
.RedPT{
  font-family: Arial, Helvetica, sans-serif;
  font-size: large;
  font-weight: bold;
  font-style: italic;
  color: red;
  background-color: white;}
```

Note that the three class selectors use standard CSS syntax to define two new class selectors, which will be turned into SAS style template style elements when this CSS file is imported into a style template. The ".RedPT" class selector was used for "The MEANS Procedure" title, as shown in Figure 17, and the ".PinkHdr" class selector was used for the spanning header, whereas the regular ".Header" style properties were used for the other headers produced by PROC MEANS.



The MEANS Procedure

Analysis Variable: Total Sales		
Region	N Obs	Sum
Africa	56	2,342,588.00
Asia	14	460,231.00
Canada	37	4,255,712.00
Central America/Caribbean	32	3,657,753.00
Eastern Europe	31	2,394,940.00
Middle East	24	5,631,779.00
Pacific	45	2,296,794.00
South America	54	2,434,783.00
United States	40	5,503,986.00
Western Europe	62	4,873,000.00

Figure 17: PROC MEANS Output That Used Table, Style, and Tagset Templates

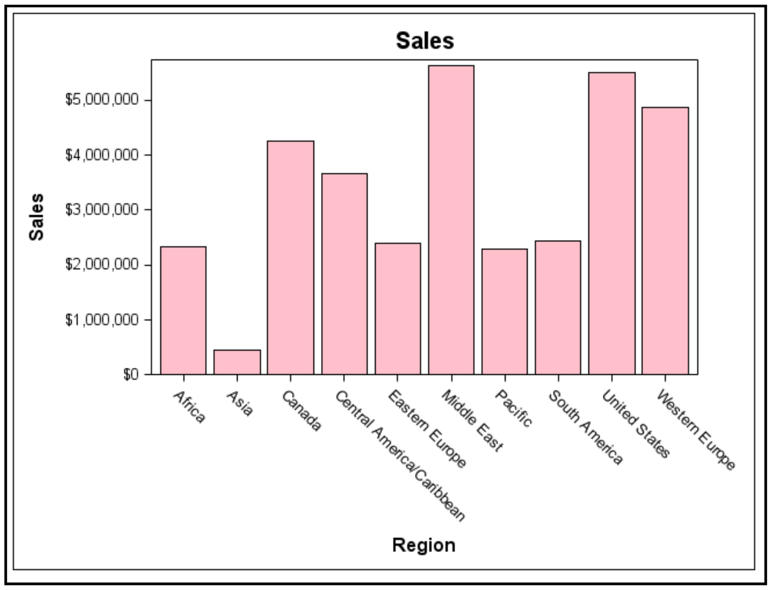


Figure 18: Partial PROC SGRENDER Output That Used Graph and Style Templates

Note that the SGRENDER graph is the same as the one that was produced previously. The only difference is that instead of using orange as the color for the bars on the chart (as with the SASWEB style), this output uses the GDATA1 color of pink from the style template. The chart for Returns from the second SGRENDER also has pink bars and is not shown here for space purposes.

Study of the four template languages might take longer to learn than it took you to learn PROC PRINT. However, as with gardening, your effort is always rewarded either with beautiful flowers to enjoy or healthy fruits and vegetables to eat. Designing and changing templates has gotten easier in SAS 9.2, with the introduction of the ability to import a CSS file into a style template. If you use the SGPLOT, SGSCATTER and SGPANEL procedures, you might not need to use graph templates at all to generate graphical output.

All the programs used in this paper are available for download from the SAS Web site, http://support.sas.com/rnd/papers. The Web site has a section for every major user group's papers and presentations. If you look for the paper title, you will find the download link.

My recommendations, when working with PROC TEMPLATE as a beginner are these:

1) Try only one thing at a time. When that works, move on to the next thing. It is too hard to debug what went wrong if you try to do 10 things in your PROC TEMPLATE code. If you change only one thing at a time, you will always know where and when the problem occurs.

2) Start out working with all your templates in an item store in the WORK library. You will have a lot of mistakes, failed attempts, and bad templates. You don't want them hanging around in SASUSER.TEMPLAT.

3) Remember that the SASUSER library generally gets deleted and recreated when SAS is installed. When you do save permanent templates, I recommend a permanent location other than SASUSER, especially if you are going to share your templates with other people.

4) NEVER (and I really mean it, don't send me e-mail if you corrupt or blow away SASHELP.TMPLMST), NEVER define SASHELP.TMPLMST with an access of UPDATE or WRITE. ALWAYS, ALWAYS, ALWAYS put SASHELP.TMPLMST in your ODS PATH list with an access mode of READ.

5) Know what's possible when SAS is out of the picture before you blame SAS or PROC TEMPLATE for your code not working. For example, HTML files do not have page numbers. Yes, there is a PageNo style element in the Style element list. But if you try to change it for HTML, it will be ignored. There are two reasons that it won't work: 1) The HTML specification, as set by the W3C, does not have any ability to set page numbers on a Web page (remember the "paperless" office); 2) The documentation for the PageNo attribute clearly states that this element controls the style of page numbers for paginated destinations. Here's another example: You want to change the HTML tagset template to do something special with the <DIV> tag. But you are slightly dyslexic and you type the <DVI> tag in the tagset template code. SAS is not going to check whether the <DVI> tag is valid or not. It's just markup text. But the browser will ignore the <DVI> tag. Don't blame SAS that the <DVI> tag doesn't work.

6) Sometimes you don't need templates at all. Sometimes there's more than one way to do something. You want to change the output from PROC FREQ. You don't have a lot of time to study table template syntax. What do you do? Create an output data set from PROC FREQ, manipulate the data set if you need to, and then pass the data set to PROC PRINT or PROC REPORT. You want to do trafficlighting, too: Use the STYLE= override with a user-defined format or with a CALL DEFINE statement in PROC REPORT.

7) You're not sure which template to change, especially when using the TAGSETS.EXCELXP destination. In my experience, very few people ever need to change the tagset template for this destination. Most people really want to a) change something about the style – which points to STYLE= overrides or to style template changes or b) change something about how Excel works that is just not possible in the Spreadsheet ML XML (like inserting graphics into ExcelXP output – which is not possible by Microsoft Design). To help you decide, go back to Figure 3 and Figure 4. If you clearly understand what you get by default from ODS and can articulate what it is that needs to change, you should be able to use these figures to pinpoint where in the process the change needs to occur. Wishful thinking doesn't work. Even if you think that the style template *ought* to be able to rearrange the order of the columns that come from PROC FREQ, it's just not possible. Go back to look at Figure 3. The data from PROC FREQ are bound to the table template for PROC FREQ to construct the output object that will be sent to ODS. By the time PROC FREQ is finished, the order of the columns in the output object has already been determined. There's just not any "column-oriented" syntax in style template definitions.

8) Neatness counts. So do comments. Do pay attention to your indenting and your commenting. It seems needlessly finicky, I know, but even a week after I write a template, I sometimes need a reminder about why I wanted to use an <OL> tag versus an <UL> tag. Skipping a line between DEFINE blocks or STYLE elements makes the template code easier to read.

9) "R the M": When I was first starting out (many years ago, in a galaxy far, far away), the only people in my company who used SAS were the systems programmers (who were doing performance stuff reading SMF records – UGH!). They were not always receptive to questions about how to read data or how to write a report, especially if they'd had system time at 2 a.m. One fellow refused all questions. His only answer to

any question was "RTFM". Finally, one of the other systems guys took pity on me and told me what "RTFM" meant and loaned me his SAS manual (the "M"), which was only one book at the time. And I took the "M" and I read it (the "R"). There's a lot of SAS documentation these days. It's not just one manual. There is good information in the documentation, and there's a lot of good information in the Tech Support notes, which you can search at http://support.sas.com. Use the resources that are available to you. If you can't find code that does exactly what you want in the Template FAQ (http://support.sas.com/rnd/base/ods/templateFAQ/index.html), then look for code that does something similar to what you want, and use that as your starting model.

10) Don't Panic! This paper has attempted to provide you with concrete examples of each of the four template types. We've just tiptoed or strolled through the template garden. We haven't looked under the leaves for mites or dug weeds out of the corn or staked up the tomatoes. You have to become the master gardener of your own template garden. And if you decide that you're not quite ready to roll up your sleeves and dig in the dirt, that's OK, too. When you are ready, you'll have a good idea of where to start.

"*…for every gardener knows that after the digging, after the planting,*
after the long season of tending and growth, the harvest comes."
– "The Seven of Pentacles," by Marge Piercy

REFERENCES

Cartier, J. 2002. "Visual Styles for V9 SAS Output." *Proceedings of the Twenty-Seventh Annual SAS Users Group International Conference.* Cary, NC: SAS Institute Inc. Available at http://support.sas.com/rnd/datavisualization/papers/VisualStyles.pdf.

Cartier, J. 2002. "The Basics of Creating Graphs with SAS/GRAPH Software." *Proceedings of the Twenty-Seventh Annual SAS Users Group International Conference.* Cary, NC: SAS Institute Inc. Available at http://support.sas.com/rnd/datavisualization/papers/GraphBasics.pdf.

Cartier, J. 2003. "It's All in the Presentation." *Proceedings of the Twenty-Eighth Annual SAS Users Group International Conference.* Cary, NC: SAS Institute Inc. Available at http://www2.sas.com/proceedings/sugi28/147-28.pdf.

Cartier, J. 2004. "Graphs In Style." *Proceedings of the Twenty-Ninth Annual SAS Users Group International Conference.* Cary, NC: SAS Institute Inc. Available at http://support.sas.com/rnd/datavisualization/papers/sugi29/ods/GraphsInStyle/GraphsInStyle_files/frame.htm#slide0027.htm.

Gebhart, Eric. 2007. "ODS Markup, Tagsets, and Styles! Taming ODS Styles and Tagsets." *Proceedings of the SAS Global Forum 2007 Conference.* Available at http://www2.sas.com/proceedings/forum2007/225-2007.pdf.

Parker, Chevell. 2003. "Generating Custom Excel Spreadsheets using ODS." *Proceedings of the Twenty-Eighth Annual SAS Users Group International Conference.* Cary, NC: SAS Institute Inc. Available at http://www2.sas.com/proceedings/sugi28/012-28.pdf.

Rodriguez, Robert N. 2008. "Getting Started with ODS Statistical Graphics in SAS 9.2." *Proceedings of the SAS Global Forum 2008 Conference.* Cary, NC: SAS Institute Inc. Available at http://www2.sas.com/proceedings/forum2008/305-2008.pdf.

Smith, Kevin. 2006. "The TEMPLATE Procedure Styles: Evolution and Revolution." *Proceedings of the Thirty-First Annual SAS Users Group International Conference.* Cary, NC: SAS Institute Inc. Available at http://www2.sas.com/proceedings/sugi31/053-31.pdf.

Truxillo, Catherine, and Cynthia Zender. 2005. "Customizing ODS Statistical Graphs." *Proceedings of the Thirtieth Annual SAS Users Group International Conference.* Cary, NC: SAS Institute Inc. Available at http://www2.sas.com/proceedings/sugi30/239-30.pdf.

Zender, Cynthia. 2004. "Markup 101: Markup Basics." *Proceedings of the Twenty-Ninth Annual SAS Users Group International Conference.* Cary, NC: SAS Institute Inc. Available at http://support.sas.com/rnd/papers/sugi29/markup-basics.pdf.

Zender, Cynthia. 2005. "The Power of TABLE Templates and DATA _NULL_." *Proceedings of the Thirtieth Annual SAS Users Group International Conference.* Cary, NC: SAS Institute Inc. Available at http://www2.sas.com/proceedings/sugi30/088-30.pdf.

ACKNOWLEDGMENTS

The author thanks Chevell Parker, Bari Lawhorn, David Kelley, Wayne Hester, Scott Huntley, Eric Gebhart, and Dan O'Connor for graciously answering all ODS and PROC TEMPLATE questions. In addition, the author owes a huge debt of gratitude to the editors at SAS, John Kohl and Amy Wolfe, who worked very hard to make this paper grammatically correct, in spite of her best efforts to the contrary.

CONTACT INFORMATION

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

Cynthia L. Zender
SAS Institute, Inc.
Work Phone: 575-522-3803
E-mail: Cynthia.Zender@sas.com