Paper 251-2011

Add Style to ODS Output by Stretching Your Inheritance in Version 9.2 SAS®

Perry Watts, Stakana Analytics, Elkins Park, PA

ABSTRACT

Inheritance in ODS style templates has been simplified in Version 9.2 SAS. Now you don't need a FROM clause in a CLASS statement when you work with defined lineages in the Styles.Default template. However, lineages as well as attributes can be changed when new style templates are created in ODS. This means that you need an ODS lineage tracer that links inheritance from Base.Template.Style to associated attribute settings in the Styles.Default template. Only by viewing attribute settings at their assigned locations in a lineage, is it possible to know for certain when STYLE … FROM can safely be discarded in favor of the CLASS statement.

This paper shows by example how to use the lineage tracer plus additional tools to develop new styles[1] from the Styles.Default template. By studying the examples presented, you will be able stretch ODS inheritance to get the output you want.

INHERITANCE, LINEAGE AND FAMILY TREES

Since inheritance plays a pivotal role in ODS style definitions, terms need to be defined from the outset. The added underlined text in the following definitions taken from *The Concise Oxford Dictionary* [2] shows that *inheritance*, *lineage*, and *family tree* are closely related:

inheritance *n.* **1** a thing that is inherited. **2** the act of inheriting.

inherit *v.* **2** *tr.* to derive (a quality or characteristic) genetically from one's parents or ancestors .

lineage *n.* lineal descent; ancestry, pedigree.

lineal *adj.* **1** in the direct line of descent or ancestry. **2** linear; of or in lines.

family tree ***n***. a chart showing relationships and lines of descent.

From the definitions above, it can be seen that the non-linear tree is the graphic of choice for representing relationships among lineages. However, different motivations exist for graphing family and ODS trees. When it comes to a family tree, you don't actually inherit anything from your Aunt Suzie, but Aunt Suzie may express a different trait in the gene-pool passed to both of you from your grandparents who are Aunt Suzie's parents. Therefore, you will get a more complete picture of what you will pass on to your descendents if (a) you have a lot of relatives and (b) you can build a family tree that expresses their characteristics.

Inheritance is also restricted to the lineage when new style elements are defined in ODS. However, the composition of a lineage can be altered in a new style definition. This means that the revised lineage can transform the equivalent of an Aunt Suzie style element into a parent. With such flexibility, you need to have access to a tree-structure that represents the parent template in its entirety. In short, you need to know your roots!

LINEAGES ARE DEFINED BY APPLYING RECURSION TO BASE.TEMPLATE.STYLE

Base.Template.Style is not organized by lineage, since style elements appear only once in the file. For example in Figure 1, style elements 1 to 101 from the template must be scanned recursively to locate the six ancestors for ROWHEADEREMPHASISFIXED. A similar search for the ancestors of ROWHEADERFIXED from a different lineage also yields style elements 1, 76, 87, 91, and 98, since ROWHEADER is a parent to ROWHEADERFIXED and a grandparent to ROWHEADEREMPHASISFIXED. This relationship means lineage listings always involve redundancy.

Figure 1. A lineage derived from Base.Template.Style is listed in Element Number order. CONTAINER has no ancestors (the FROM clause is missing) and ROWHEADEREMPHASISFIXED has no descendents. Therefore, the lineage is complete.

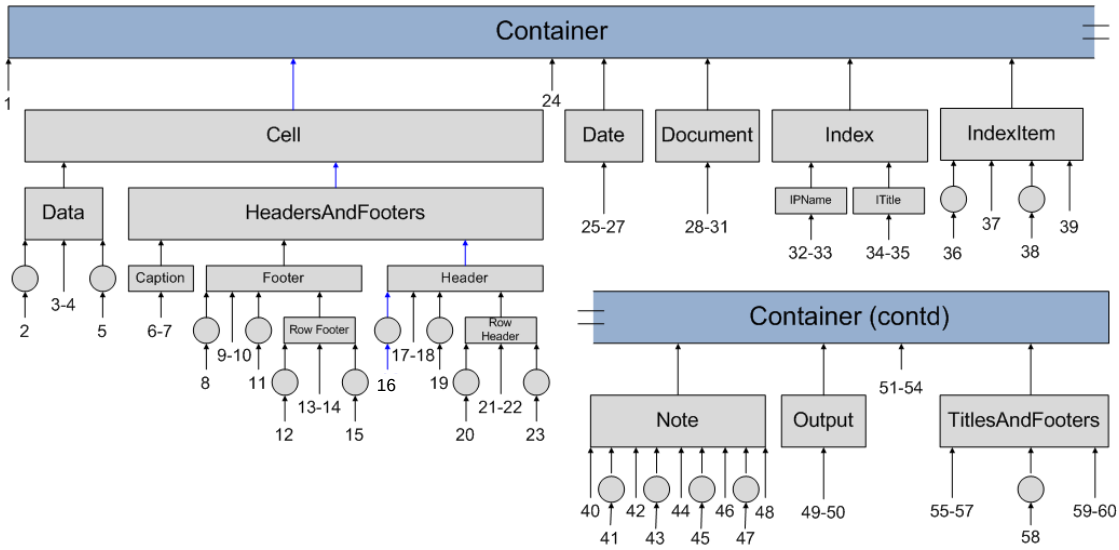
```
  1    style Container
...
 76    style Cell from Container
...
 87    style HeadersAndFooters from Cell
...
 91    style Header from HeadersAndFooters
...
 98    style RowHeader from Header
...
101    style RowHeaderEmphasis from RowHeader

102    style RowHeaderEmphasisFixed from RowHeaderEmphasis
```

[1] In ODS, the terms *style* and *template* are synonymous. The STYLE statements in Figure 1 define *style elements* not a style.

Lineages are typically displayed in an upside-down tree with the root node for the common ancestor at the top and leaf nodes with no progeny at the base. This format is used in Figure 2 to display the CONTAINER lineages from the Styles.Default.Template in 9.1.3 SAS. The number of lineages in the tree is equivalent to the number of leaf nodes that are displayed. The tree originally appeared in the NESUG paper, *Using Recursion to Trace Lineages in the SAS® ODS Styles.Default Template* presented in 2010 [8].

Figure 2. The CONTAINER tree is wide and shallow. Nodes are sometimes represented by numbers or circles instead of names to save space. The number of CONTAINER lineages accessible to the Styles.Default.Template increases from 60 to 66 in 9.2 SAS.



The existence of circles and numbers in Figure 2 illustrates a major problem with the top-down tree: there is not enough space to label all the style elements by name. The problem is solved when the tree is rotated 90 degrees. In Figure 3, a rotated tree derived from Base.Template.Style for the Styles.Default template in 9.2 SAS becomes the home page for an updated lineage tracer.

Each row in the new tracer is reserved for a single, complete lineage. When rows are read left-to-right, the domain of the examined style element is successively diminished. For example CONTAINER in Lineage #22 defines default font and color settings for all tabular output, whereas only the font assigned to a row header is changed in ROWHEADERFIXED. Along with the successive reduction in dominion comes a corresponding reduction in redundancy. CONTAINER is always assigned to Element #1 in the Styles.Default lineage tracer whereas ROWHEADERFIXED in Lineage #22 is not found anywhere else in the table.

Figure 3. Only CONTAINER style elements are included in the new lineage tracer updated for 9.2 SAS. Graphics style elements have been omitted, and settings for FONTS and COLORS can be found in drill-down pages that list lineage-associated attributes. The title is off-center, because the maximum number of style elements in a single lineage is 12.

66 Container Lineages In or Accessible to the ODS Styles.Default Template
(Style Elements Derived Exclusively In BASE.TEMPLATE.STYLE are in BLUE)

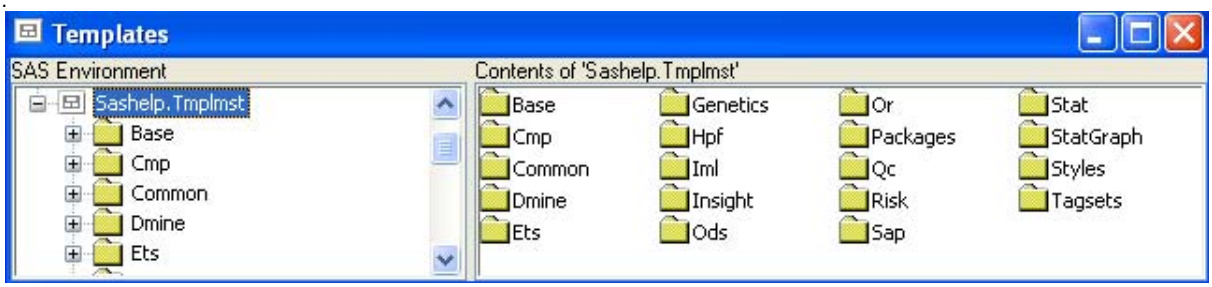
Lineage#	Element#1	Element#2	Element#3	Element#4	Element#5	Element#6	Element#7
1	Container	Byline Container					
2	Container	Cell	Data	Data Emphasis	Data Emphasis Fixed		
→ 3	Container	Cell	Data	Data Empty			
. . .							
20	Container	Cell	Headers And Footers	Header	Row Header	Row Header Emphasis	Row Header Emphasis Fixed
21	Container	Cell	Headers And Footers	Header	Row Header	Row Header Empty	
→ 22	Container	Cell	Headers And Footers	Header	Row Header	Row Header Fixed	
23	Container	Cell	Headers And Footers	Header	Row Header	Row Header Strong	Row Header Strong Fixed
24	Container	Column Group					
25	Container	Date	Body Date				

CONNECTING LINEAGE TO ATTRIBUTE

While lineages of style elements in a template define the paths for inheritance, still missing in the discussion is information about *what* actually is being inherited. In the case of ODS style templates attribute settings that control the outward appearance of a report constitute the *what*. Attributes are the building blocks for the style elements, and they can be modified in accordance with the rules for inheritance. Since attributes and lineages are so tightly bound, it makes sense to access both in a single lineage tracer. However, knowledge of file structure is required, because lineage and attribute information for the Styles.Default template is stored separately in Sashelp.Tmplmst. In Figure 4 access to the separate sources for lineage and attribute information is provided.

Figure 4. To access Sashelp.Tmplmstr first make the *Results* window active in your SAS session. Next press VIEW then TEMPLATES and select SasHelp.Tmplmstr.

.



To get to Base.Template.Style, select the **BASE** folder in Sashelp.Tmplmst. Next, press **TEMPLATE** followed by **STYLE** to bring up PROC TEMPLATE for Base.Template.Style. The presence of FROM in the listing shows that Base.Template.Style provides information about inheritance, not only for the default template but for all 53 templates that are in the **STYLES** folder.

```
proc template;
   define style Base.Template.Style;
      notes "Implicit parent for all style templates"
      ...
      style Container
         "Controls all container oriented elements." /
         abstract =| on;
      style Document from Container
         "Controls the various document bodies."
      style Body from Document
         "Controls the Body/Frame/Contents/Page file.";
```

To get to the Styles.Default template where style element definitions include attribute settings, select the **STYLES** folder from SAShelp.Tmplmst. Next, press **DEFAULT** to bring up PROC TEMPLATE for Styles.Default. The Styles.Default template is the only template in the STYLES folder that identifies all member style elements with a CLASS statement. What this says about inheritance is discussed later in the paper.

```
proc template;
   define style Styles.Default;
     ...
     class Container /
         font = Fonts('DocFont')
         color = colors('docfg')
         backgroundcolor = colors('docbg');
     class Index /
         color = colors('contentfg')
         backgroundcolor = colors('contentbg');
```

While Base.Template.Style is used to create the home page in the lineage tracer, detailed information about attribute settings for each of the 66 CONTAINER lineages is supplied by the Styles.Default template. The detailed information is made available by hyperlink. Pressing #3 in Figure 3, for example, brings up a listing about Lineage #3 that is reproduced in Figure 5.

Figure 5. Attributes, if set, are always affiliated with style elements in a lineage. The lineage tracer is created by formatting PROC REPORT in ODS and sending the output to HTML. Since the Styles.Default template is being visually described in the lineage tracer, STYLE=STYLES.DEFAULT is used in its construction. The algorithm for inserting internal hyperlinks comes from *Carpenter's Complete Guide to the SAS® Report Procedure* **[1, p. 257-260]**.

*Lineage # 3: Container * Cell * Data * DataEmpty*

Style Element	Default Assignment: (ATTRIBUTE = Value)	Font or Color Code
Container	Abstract: Controls all container oriented elements.	
	FONT = Fonts('DocFont')	"<sans-serif>,Helvetica,sans-serif",3
	COLOR = colors('docfg')	cx002288
	BACKGROUNDCOLOR = colors('docbg');	cxE0E0E0
Cell	Controls general cells.	
Data	Default style for data cells in columns.	
	COLOR = colors('datafg')	cx000000
	BACKGROUNDCOLOR = colors('databg');	cxD3D3D3
DataEmpty	Controls empty data cells in columns.	
Return to Lineage List		

SAS data sets used for creating the lineage tracer come from an application of the ODS SOURCE command to the Base.Template.Style and Styles.Default templates. For drill-down displays such as the one shown in Figure 5, PROC SQL is used for joining STYLE elements originating in Base.Template.Style to their counterpart CLASS elements from Styles.Default. The SQL join is needed for the Figure 5 display, because style element descriptions and lineage order are pulled from Base.Template.Style whereas the attribute settings plus font and color codes come from Styles.Default.

Let's go through the listing in Figure 5 to see what it tells us about ODS inheritance. First we see that FONT, COLOR (for text), and BACKGROUNDCOLOR are assigned to the CONTAINER style element. CELL inherits these settings and passes them on to DATA. DATA keeps FONT but changes COLOR and BACKGROUNDCOLOR. Finally, DATAEMPTY inherits FONT from CONTAINER and the color settings from DATA. Take a look at the table in Figure 12 that uses DATAEMPTY to highlight missing values.

Additional enhancements have been made to fonts and colors in the lineage tracer, since they account for more than 50 percent of all the attribute settings in the Styles.Default template. The enhancements are made possibly by processing three non-CONTAINER style elements in the Styles.Default template: FONTS, COLORS, and COLOR_LIST. Table 1 below shows how CONTAINER fonts and colors listed in column #2 of the table in Figure 5 are mapped to their actual codes.

Table 1. The FONTS, COLORS, and COLOR_LIST style elements are used to map the assigned font and color attribute settings in the CONTAINER style element to their actual codes. Mapping is achieved by creating input control data sets that are operated on by PROC FORMAT. The text that is transferred to the lineage tracer is highlighted in blue.

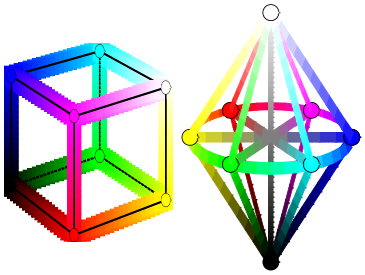
Default Assignment	Attribute Name	FONTS or COLORS Mapping	COLOR_LIST Mapping
FONT = Fonts('DocFont')	DocFont	'docFont' = **("<sans-serif>, Helvetica, sans-serif",3);**	
COLOR = colors('docfg')	docfg	'docfg' = color_list('fgA')	'fgA' = **cx002288**
BACKGROUNDCOLOR = colors('docbg');	docbg	'docbg' = color_list('bgA')	'bgA' = **cxE0E0E0**

Color and font codes are not only listed in the drill-down pages of the lineage tracer, they are also *applied* to the relevant cells in column #2 that list default settings. Thus BACKGROUNDCOLOR for CONTAINER at cxE0E0E0 in Figure 5 is a lighter gray than the background color used for DATA (cxD3D3D3).

Color codes formatted as cx**RRGGBB** in SAS use hexadecimals in the RGB (red| green| blue) color coding system. Each color component ranges from 00 – FF which translates to 0 – 255 in decimal. Light colors are generated when code components contain high values. Codes with low component values map to dark colors. In the case of Lineage #3 displayed in Figure 5, dark colors (cx002288 and cx000000) are defined as foreground colors for text. Since dark text is more visible on a light background, the background colors are much lighter (cxE0E0E0 and cxD3D3D3). Note that the background colors in Lineage #3 have the same values for each of the code components. That means they map to a shade of gray which is devoid of hue. Black at cx000000 and white at cxFFFFFF are also devoid of hue. Finally, when components in an RGB code have different values, hue will be determined by the component with the highest value. Therefore, we can know that cx002288 for the COLOR attribute in CONTAINER will be a dark blue by just looking at the RGB code. For additional information about color, please see papers cited at the end of the paper and stored online at the author's screencast.com account. Examples of color *spaces*, *scales*, and *charts* can also be found at screencast.com. What is meant by each of these terms is defined visually in *Advanced Programming Techniques for Working with Color in SAS® Software* [7] and reproduced in Figure 6 below.

Figure 6. Pictorial definitions for color *spaces*, *charts* and *scales*.

Color spaces are 3-D structures that show how a gamut of codes is mapped in a color system. The RGB space is a cube whereas the HLS space is double-ended cone.

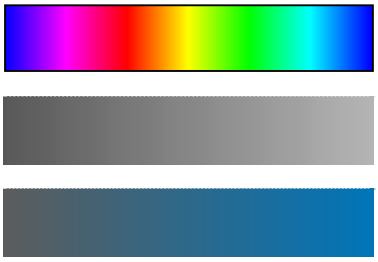


Color charts show colors as discrete entities in tabular format. Here are the unique SAS predefined colors and a digitized listing of a shade of blue.

CX00FFFF	CX3883A8	CX4F4F4F	CX700000
CXC0FF81	CXE06090	CXE0A860	CXE8D898
CXFF0055	CXFF00FF	CXFF6060	CXFFAA00

CX001928	CX002032	CX00263C	CX002C46	CX003350	CX00395A
CX003F64	CX00466E	CX004C78	CX005282	CX00598C	CX005F96
CX0065A0	CX006CAA	CX0072B4	CX0078BE	CX007FC8	CX0085D2
CX008BDC	CX0092E6	CX0098F0	CX009EFA	CX05A3FF	CX0FA7FF
CX19ABFF	CX23AEFF	CX2DB2FF	CX37B6FF	CX41B9FF	CX4BBDFF
CX55C1FF	CX5FC4FF	CX69C8FF	CX73CCFF	CX7DCFFF	CX87D3FF
CX91D7FF	CX9BDAFF	CXA5DEFF	CXAFE2FF	CXB9E5FF	CXC3E9FF
CXCDEDFF	CXD7F0FF	CXE1F4FF	CXEBF8FF	CXF5FBFF	CXFFFFFF

Color scales represent colors along a continuum. Here are **h**ue, **l**ightness and **s**aturation scales built with the HLS color coding system.



We are not completely finished with our discussion about Figure 5. At this point, you may be wondering why nothing stands out in the cells where the default color assignments are listed for the DATA style element. Foreground and background colors in these cells are the same as those in surrounding cells. The lack of uniqueness can be attributed to the fact that the Styles.Default template is used for generating the lineage tracer. DATA style element settings in the template are used to format every cell in the DATA region of a PROC REPORT table.

You can download a copy of the ODS|HTML lineage tracer from the author's web accounts. How it is used is demonstrated when instructions for constructing new style templates are presented later in the paper. Also available by download is the ODS|HTML attribute descriptor with definitions originating in the 9.1 and 9.2 ODS User's Guides [11(p.296-300) 12(p.480-487)].

ADDITIONAL LEVELS OF INHERITANCE IN ODS

Inheritance at the Style Template Level

Inheritance exists among the Sashelp.Tmplmst style templates. This means 44 lineages of the 53 style templates can be arranged in a tree. The lineage tracer for style templates is reproduced in part in Figure 7. An enhanced version of the complete tree can be found in the Appendix.

Figure 7. A lineage tracer for Sashelp.Tmplmst style templates. While Base.Template.Style is defined as the "implicit parent for all style templates" in the note from Figure 3, inheritance among style templates is traced by the PARENT statement in a template definition. For example, `Parent=styles.journal` defines the immediate predecessor in the Styles.Journal2 template.

Lineage#	Implicit Parent	Style #1	Style #2	Style #3	Style #4
1	Base.Template.Style	styles.minimal			
2	Base.Template.Style	styles.normal	styles.festival	styles.festivalprinter	
...					
23	Base.Template.Style	styles.default	styles.journal	styles.journal2	styles.journal3
24	Base.Template.Style	styles.default	styles.listing		
25	Base.Template.Style	styles.default	styles.magnify		
26	Base.Template.Style	styles.default	styles.money		

Inheritance for Font Definitions

A type of inheritance also exists at the attribute level for font definitions. In the case of the Styles.Default template, all font references are in FONT format where four font components are combined into a single value assignment. Here is the syntax from the attribute descriptor that is included in the Appendix as a screen snapshot:

```
FONT= [face(s), size, keywords]
```
where keywords reference font *weight, style* and *width*.

Examples of font settings taken from the FONT class statement in the Styles:Default template include:

```
'TitleFont' = ("<sans-serif>, Helvetica, sans-serif", 5, bold italic)
'FixedHeadingFont' = ("<monospace>, Courier, monospace",2)
```

For FixedHeadingFont font *weight, style* and *width* are set by default to MEDIUM, ROMAN and NORMAL.

To increase the font size in TITLEFONT, just enter:

```
FONTSIZE=6;
```

What happens is that FONTSIZE is effectively integrated into a new FONT definition as:

```
'TitleFont' = ("<sans-serif>, Helvetica, sans-serif", 6, bold italic)
```

DEFINING NEW STYLE TEMPLATES IN ODS

Style templates are easier to define in Version 9.2 SAS. Now it is possible to ignore inheritance with the new CLASS statement, because underlying lineage definitions are imported intact from the parent template. In 9.1.3 SAS ABSTRACT style elements also had to be identified, since they could only be changed with a REPLACE statement. Since REPLACE is not supported in 9.2 SAS, there is no longer any need to distinguish ABSTRACT style elements from regular ones. Inheritance works just the same regardless of style element type.

Despite the convenience of the new CLASS statement, STYLE and STYLE … FROM are still needed for changing lineage definitions in a new style template. That STYLE continues to play an important role in ODS can be confirmed by observing that 45 out of the 53 styles in Sashelp.Tmplmst rely exclusively on the STYLE keyword to define member style elements. CLASS, on the other hand, is used exclusively only in the Styles.Default template. For a complete listing of keyword status by template, see the color coded version of the Sashelp.Template lineage tracer in the Appendix. Because STYLE is still used in 9.2 SAS, detailed reviews for all three element definition types: CLASS, STYLE, and STYLE … FROM are included in separate sub-sections below.

Here is how the sashelp.shoes data set has been changed to support all the examples in this section:

```
data shoes2(keep=product subsidiary stores rename=(stores=nstores));
  set sashelp.shoes;
  if product in ('Boot','Sandal','Slipper','Sport Shoe');
run;
```

In Example #1, no new style is being defined. Instead the Styles.Default template is used to format output sent to HTML. The source code in Example #1 illustrates Cynthia Zender's ODS "sandwich" metaphor where PROC FREQ is surrounded by ODS commands [10]. In subsequent examples only PROC TEMPLATE source code for new style definitions is displayed alongside screen snapshots of HTML output.

1) PROC FREQ output is formatted by the Default template in ODS.

Code

```
ods path work.temp(update)
     sasuser.templat(update)
     sashelp.tmplmst(read);
ods html path="&htmPath" (url=none)
  file='default.html' style=STYLES.DEFAULT;
  title 'Default Template Output';
  proc freq data=styleApp.shoes2;
    weight nstores;
    tables product;
  run;
ods _all_ close;
```

HTML Output

Default Template Output

The FREQ Procedure

Product	Frequency	Percent	Cumulative Frequency	Cumulative Percent
Boot	864	30.44	864	30.44
Sandal	564	19.87	1428	50.32
Slipper	794	27.98	2222	78.29
Sport Shoe	616	21.71	2838	100.00

To validate colors and fonts displayed in the screen snapshot from Example #1, check lineages #2(DATA), #20(HEADER|ROWHEADER), #64(PROCTITLE), and #66(SYSTEMTITLE) in the lineage tracer.

USING THE CLASS STATEMENT IN STYLE DEFINITIONS

The CLASS statement automatically preserves the lineage structure defined in Base.Template.Style that is passed down to the Styles.Default template. By looking at the lineage trace in Example #2 it is easy to see that STYLE

`header from header` and `CLASS Header` produce identical results. In the examples that follow, altered child style elements in the lineage traces are denoted in **red** or **blue**. Attribute settings defined in the child-elements prevail and are passed down to the descendents (also colored in **red** or **blue**).

2) Change a single attribute in HEADER.

Code

```
proc template;
  define style styles.ChangeHeader1;
  parent=styles.default;
  class header /
    font=(arial, 14pt, bold);
  end;
run;
```

HTML Output

Font Size for HEADER and ROWHEADER Increases with a CLASS statement

The FREQ Procedure

Product	Frequency	Percent	Cumulative Frequency	Cumulative Percent
Boot	864	30.44	864	30.44
Sandal	564	19.87	1428	50.32
Slipper	794	27.98	2222	78.29
Sport Shoe	616	21.71	2838	100.00

Adapted From **(#20 in the Lineage Tracer)**: Container > Cell > HeadersAndFooters > Header > Header > RowHeader

In Example #3, settings for non-lineage attributes BORDERWIDTH and BORDERCOLOR are changed in HEADER with a CLASS statement. The assigned COLOR for HEADERSANDFOOTERS is copied from the Default lineage tracer and pasted directly into SAS thus insuring that border and header text colors match perfectly. As expected, the lineage trace from Example #2 (not shown) remains unchanged. A Header cell is selected with the STYLE_POPUP tagset template to track the non-lineage attributes. POPUP results are also displayed in Example #3.

3) Change Non-Lineage Attributes in HEADER.

Code

```
proc template;
  define style styles.NonLineageAttrs;
  parent=styles.default;
  class Header/
    font=(arial, 14pt, bold)
    borderwidth=3;
    bordercolor=colors('headerfg');
  end;
run;
```

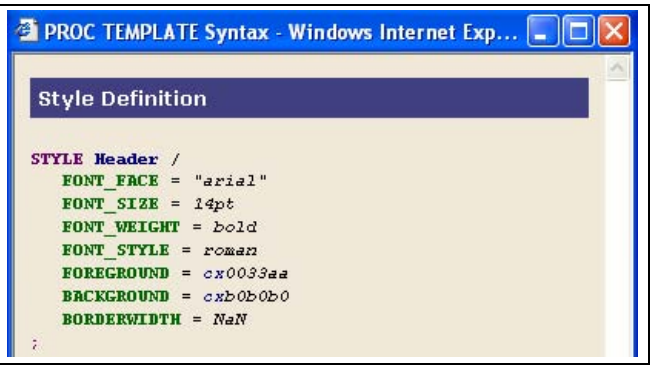
HTML Output

Non-lineage attributes BORDERWIDTH and BORDERCOLOR are added to HEADER

The FREQ Procedure

Product	Frequency	Percent	Cumulative Frequency	Cumulative Percent
Boot	864	30.44	864	30.44
Sandal	564	19.87	1428	50.32
Slipper	794	27.98	2222	78.29
Sport Shoe	616	21.71	2838	100.00

```
ODS TAGSETS.STYLE_POPUP path="&htmpath"
  FILE='StyleDiag.HTML'
  style=NonLineageAttrs;
  proc freq data=styleApp.shoes2;
    weight nstores;
    tables product;
run;
ods _all_ close;
```



Even though the non-lineage attributes are inserted properly into the ODS output, the pop-up window doesn't do a very good job announcing their arrival. BORDERWIDTH is registered but no value is displayed. BORDERCOLOR is not listed at all.

In Example #4, attributes for two style elements in the same lineage are changed. In #4a the order of the CLASS statements is the same as the listing for lineage #20 in the lineage tracer. The desired output is produced: a larger font size for row and column headers, red text for HEADER, and blue text for ROWHEADER. In #4b the same output is generated even though the order of the CLASS statements in the new style definition is reversed.

4 a) Change Two Style Elements in the Same Lineage.

Code

```
proc template;
  define style styles.TwoClassesA;
  parent=styles.default;
  class header /
    font=(arial, 14pt, bold)
    color=red;
  class rowheader /
    color=blue;
  end;
run;
```

HTML Output

Correct Lineage Order: HEADER -> ROWHEADER

The FREQ Procedure

Product	Frequency	Percent	Cumulative Frequency	Cumulative Percent
Boot	864	30.44	864	30.44
Sandal	564	19.87	1428	50.32
Slipper	794	27.98	2222	78.29
Sport Shoe	616	21.71	2838	100.00

Adapted From **#20 in the Lineage Tracer**: Container > Cell > HeadersAndFooters > Header > Header > RowHeader

Adapted From **the first lineage defined**: Container > Cell > HeadersAndFooters > Header > RowHeader > RowHeader

If the CLASS statement did not trigger Base.Template.Style to preserve lineage definitions in a new style template, ROWHEADER in #4b would have red text just like HEADER.

4 b) Change Two Style Elements in the Same Lineage

Code

```
proc template;
  define style styles.TwoClassesB;
  parent=styles.default;
  class rowheader /
    color=blue;
  class header /
    font=(arial, 14pt, bold)
    color=red;
  end;
run;
```

HTML Output

Wrong Lineage Order: ROWHEADER -> HEADER is Corrected

The FREQ Procedure

Product	Frequency	Percent	Cumulative Frequency	Cumulative Percent
Boot	864	30.44	864	30.44
Sandal	564	19.87	1428	50.32
Slipper	794	27.98	2222	78.29
Sport Shoe	616	21.71	2838	100.00

Adapted From **#20 in the Lineage Tracer**: Container > Cell > HeadersAndFooters > Header > RowHeader > RowHeader

Adapted From **the first lineage**: Container > Cell > HeadersAndFooters > Header > Header > RowHeader

The screen snapshot of the lineage tracer in Figure 8 now includes SYSTEMTITLE*n* elements highlighted in blue to indicate that they reside only in Base.Template.Style. Despite their origin, inheritance works just the same as they had homes in both the Base.Template.Style and Styles.Default templates. This claim is supported in Example #5 where SYSTEMTITLE2 and SYSTEMTITLE3 are processed alongside HEADERSANDFOOTERS.

Figure 8. From the screen snapshot of the lineage tracer, it can be seen that TITLESANDFOOTERS resides in both the Base.Template.Style and Styles.Default templates. On the other hand, SYSTEMTITLE2 and SYSTEMTITLE3 are only found in Base.Template.Style.

66 Container Lineages In or Accessible to the ODS Styles.Default Template
(Style Elements Derived Exclusively In BASE.TEMPLATE.STYLE are in BLUE)

Lineage#	Element#1	Element#2	Element#3	Element#4	Element#5	Element#6	Element#7
1	Container	BylineContainer					
2	Container	Cell	Data	DataEmphasis	DataEmphasisFixed		
3	Container	Cell	Data	DataEmpty			
...							
20	Container	Cell	HeadersAndFooters	Header	RowHeader	RowHeaderEmphasis	RowHeaderEmphasisFixed
...							
65	Container	TitlesAndFooters	SystemFooter	SystemFooter2	SystemFooter3	SystemFooter4	SystemFooter5
66	Container	TitlesAndFooters	SystemTitle	SystemTitle2	SystemTitle3	SystemTitle4	SystemTitle5

When a new style template is being created, it is not important to know if a given style element originates in the parent or grandparent template. CLASS processing works the same regardless of origin.

5) Define a Style that includes Style Elements that originate in Base.Template.Style only

Code

```
proc template;
  define style styles.BaseOnlyClass;
  parent=styles.default;
  class HeadersAndFooters/
    font=(arial, 12pt, bold)
    borderwidth=2
    bordercolor=colors('headerfg');
  class systemTitle2 /
    foreground=black
    fontStyle=roman
    fontSize=12pt;
  class systemTitle3 / fontSize=10pt;
  end;
run;
```

HTML Output

The SYSTEMTITLEn Classes are NOT in the ODS Styles.Default Template
Instead, they are defined in Base.Template.Style
Base.Template.Style is "the implicit parent for all style templates"

The FREQ Procedure

Product	Frequency	Percent	Cumulative Frequency	Cumulative Percent
Boot	864	30.44	864	30.44
Sandal	564	19.87	1428	50.32
Slipper	794	27.98	2222	78.29
Sport Shoe	616	21.71	2838	100.00

Adapted from **#20 in the Lineage Tracer**: Container > Cell > HeadersAndFooters > HeadersAndFooters > Header > RowHeader

Adapted from **#66 in the Lineage Tracer**: Container > TitlesAndFooters > SystemTitle > SystemTitle2 > SystemTitle2

Adapted from **the second lineage**: Container > TitlesAndFooters > SystemTitle > SystemTitle2 > SystemTitle3 > SystemTitle3

One of the most useful applications of the CLASS statement involves updating the FONTS style element. Until recently, changing a single font setting involved first copying and pasting the entire FONTS style element into a new style definition and then making the desired change. Without care being taken to document the source code, it would impossible later on to determine exactly what had been changed. The CLASS statement in Example #6 provides the perfect solution to this problem. As with all CLASS statements, child-FONTS inherits attribute settings from the parent-FONTS. Notice from the lineage trace in Example #6 that FONTS does not descend from CONTAINER. Instead, FONTS is the sole member of a different lineage.

6) Change the FONTS Style Element with a CLASS statement

Code

```
proc template;
  define style styles.OneFontChange;
        parent=styles.default;
  CLASS fonts /
    'docFont'=("Courier",3,Bold);
  end;
run;
```

HTML Output

Only DOCFONT affecting the DATA Style Element has been Changed
Titles and Headers Retain their Default Fonts

The FREQ Procedure

Product	Frequency	Percent	Cumulative Frequency	Cumulative Percent
Boot	864	30.44	864	30.44
Sandal	564	19.87	1428	50.32
Slipper	794	27.98	2222	78.29
Sport Shoe	616	21.71	2838	100.00

Not in the Lineage Tracer: Fonts > Fonts

STARTING A NEW LINEAGE WITH *STYLE* WITHOUT *FROM*

In this section and the next, the CLASS statement is replaced by STYLE or STYLE…FROM. In the ODS User's Guide for 9.2 SAS, the implications of omitting the FROM option in a STYLE statement are described as follows:

> If there is a like-named style element within the child style definition that *does not* have a FROM option specified, then the style element from the child style definition overrides the style element from the parent style definition [12, p. 521].

In "lineage-speak" this means that a new lineage is defined when an unadorned STYLE statement is issued. Nothing is inherited from the parent and, by extension, from any ancestor. Nevertheless, the newly defined style element passes attribute settings on to its descendents. In Example #7, output changes dramatically when CLASS FONTS becomes STYLE FONTS. All the text strings in the output have the same font: Courier, 3, bold.

7) Change the FONTS Style Element with a STYLE statement

Code

```
proc template;
  define style styles.OneFontChangeWstyle;
        parent=styles.default;
  STYLE fonts /
   'docFont'=("Courier",3,Bold);
  end;
run;
```

HTML Output

DOCFONT is Used by All Style Elements in PROC FREQ Output
(Including Titles and Headers)

The FREQ Procedure

Product	Frequency	Percent	Cumulative Frequency	Cumulative Percent
Boot	864	30.44	864	30.44
Sandal	564	19.87	1428	50.32
Slipper	794	27.98	2222	78.29
Sport Shoe	616	21.71	2838	100.00

Fonts The FONTS style element now has no ancestors or descendents

In Base.Template.Style, CONTAINER is defined as a common ancestor, meaning that like FONTS it has no ancestor, but in contrast to FONTS, it has many descendents. In Example #8 below, all attributes listed in parent-CONTAINER are assigned values in child-CONTAINER. TEXTDECORATION, new in 9.2 SAS, is also assigned to the mix. The addition of TEXTDECORATION with a value of UNDERLINE proves that the newly defined child-CONTAINER passes attribute settings to its descendents. However, the descendents of CONTAINER are not being included in the lineage panel at the base of Figure 8, because the panel would then have to accommodate the entire CONTAINER tree from the lineage tracer!

A description of TEXTDECORATION can be found in AttributeDescriptor92.html available by download. Attributes are defined and described by example in the descriptor. Aliases to version 9.1.3 SAS are also provided, and if an attribute is new to 9.2 SAS, it is highlighted in blue. (See also Appendix 3).

8) Change CONTAINER with a STYLE statement

Code

```
proc template;
  define style styles.ChangeContainer1;
    parent=styles.default;
    STYLE Container /
      FONT = Fonts('DocFont')
      COLOR = colors('docfg')
      BACKGROUNDCOLOR = cxEFF3FF
      TEXTDECORATION = underline;
  end;
run;
```

HTML Output

Change CONTAINER with a STYLE Statement
(Original attributes are assigned values. TEXTDECORATION is added.)

The FREQ Procedure

Product	Frequency	Percent	Cumulative Frequency	Cumulative Percent
Boot	864	30.44	864	30.44
Sandal	564	19.87	1428	50.32
Slipper	794	27.98	2222	78.29
Sport Shoe	616	21.71	2838	100.00

Container Does Not Define a Lineage. It defines a tree.

All text in the HTML output is now underlined thanks to the addition of TEXTDECORATION to CONTAINER. You also may be wondering how or why cxCDD9FF is chosen as the new background color, and why the original CONTAINER gray in the SYSTEM and PROC titles is not erased. To find out about the persistence of gray in the output, check out lineage #64 in the lineage tracer. There you will see that cxE0E0E0 is assigned as a background color to both TITLESANDFOOTERS and PROCTITLE. Since these style elements come after CONTAINER in the lineage, their color settings prevail.

The decision to go with cxCDD9FF is more involved. Working with color involves restraint. As Jenny Preece points out in *Human-Computer Interaction*:

> Colour coding provides many opportunities for coding and structuring information at the interface as well as making it pleasant and enjoyable to look at. However, excessive use of colour can result in **colour pollution**, particularly when highly saturated colours such as 'full' red and a 'deep' blue are used **[6,p.89].**

A good way to lessen color pollution is to populate ODS output with related hues. cxEFF3FF is just a lighter variation of cx002288 that is assigned to COLOR via 'docfg' in Example #8. Obtaining the RGB code for a lighter shade of blue involves executing the macro function, GETALIGHTER_DARKERCOLOR also available for download. What the macro does is to take a user-supplied RGB code and convert it to HLS described in Figure 6. Next, a lightness percent also supplied by the user is used to calculate a new value for 'L' in the HLS code. Finally, the new HLS value is translated back into RGB and delivered as output to the macro function call.

In Example #8a, the gray title backgrounds are fixed with CLASS statements, and in Example #16 cx00269A, also derived from cx002288 is added to the mix.

8a) Improve appearances with a couple of CLASS Statements

Code

```
proc template;
  define style styles.ChangeContainer1a;
    parent=styles.default;
    STYLE Container /
      FONT = Fonts('DocFont')
      COLOR = colors('docfg')
      BACKGROUNDCOLOR = cxEFF3FF
      TEXTDECORATION = underline;
    Class TitlesAndFooters /
      BACKGROUNDCOLOR = cxEFF3FF;
    Class ProcTitle /
      BACKGROUNDCOLOR = cxEFF3FF;
  end;
run;
```

HTML Output

Change CONTAINER with a STYLE Statement
The Background Color of the Title and PROC Areas Now Match CONTAINER

The FREQ Procedure

Product	Frequency	Percent	Cumulative Frequency	Cumulative Percent
Boot	864	30.44	864	30.44
Sandal	564	19.87	1428	50.32
Slipper	794	27.98	2222	78.29
Sport Shoe	616	21.71	2838	100.00

In Example #9 all attributes are removed in the definition of CONTAINER. The only visible effect of their removal is that the background color changes from light gray to white. White is the default background color in HTML. Inheritance from CONTAINER *descendents* is preserved. That means table and titles retain their default settings.

9) Remove all Attributes from the Definition for CONTAINER

Code

```
proc template;
   define style styles.ChangeContainer2;
     parent=styles.default;
     STYLE Container;
    end;
 run;
```

HTML Output

Change CONTAINER with a STYLE Statement
NO Attributes are Specified for CONTAINER

The FREQ Procedure

Product	Frequency	Percent	Cumulative Frequency	Cumulative Percent
Boot	864	30.44	864	30.44
Sandal	564	19.87	1428	50.32
Slipper	794	27.98	2222	78.29
Sport Shoe	616	21.71	2838	100.00

As a final example of STYLE with no FROM, attributes are modified for HEADER. In Example #10, no background color is defined for HEADER. HEADER can't inherit from CONTAINER, because HEADER now defines a new lineage. Thus the default HTML white background is assigned to HEADER and to its descendent, ROWHEADER.

10) Change the HEADER Style Element with a STYLE statement

Code

```
proc template;
   define style styles.ChangeHeader;
     parent=styles.default;
     STYLE Header /
       FONT = fonts('HeadingFont')
       COLOR = BLACK;
    end;
 run;
```

HTML Output

Change HEADER with a STYLE Statement
A Background Color is NOT Specified.

The FREQ Procedure

Product	Frequency	Percent	Cumulative Frequency	Cumulative Percent
Boot	864	30.44	864	30.44
Sandal	564	19.87	1428	50.32
Slipper	794	27.98	2222	78.29
Sport Shoe	616	21.71	2838	100.00

New Style Lineage Header → RowHeader

CHANGING LINEAGE DEFINITIONS WITH *STYLE … FROM*

STYLE … FROM, saved for last, is the most complicated system of inheritance in ODS. From the 9.2 User's Guide:

> If there is a like-named style element within the child style definition that *does* have the FROM option specified, then the child style element absorbs the style attributes from the parent style element. If there are like-named style attributes in the two style elements, then the style attributes from the child style element are used **[12, p. 521]**.

In Example #11, STYLE … FROM is used to restore defaults to ROWHEADER after HEADER is changed in a new style definition. This example is adapted from *SAS® Style Templates: Always in Fashion* by Cynthia Zender **[9]**. ROWHEADER stays black in the lineage trace, because it is not changed. However, since HEADER has been removed in the second style element definition, the connecting arrow is labeled with a summing junction (⊗).

11) Use Two Lineages to Restore Default Settings to ROWHEADER with STYLE…FROM

Code

```
proc template;
  define style styles.ChangeHeaderOnly;
  parent=styles.default;
  class header /
    font=(arial, 14pt, bold)
    foreground=blue;
  style rowHeader FROM headersAndFooters;
  end;
run;
```

HTML Output

Restore Defaults for ROWHEADER by Changing ATTRIBUTES

The FREQ Procedure

Product	Frequency	Percent	Cumulative Frequency	Cumulative Percent
Boot	864	30.44	864	30.44
Sandal	564	19.87	1428	50.32
Slipper	794	27.98	2222	78.29
Sport Shoe	616	21.71	2838	100.00

Adapted From **#20 in the Lineage Tracer**: Container → Cell → HeadersAndFooters → Header → Header

Adapted From **#20 in the Lineage Tracer**: Container → Cell → HeadersAndFooters ⊗ RowHeader

Let's replace the HTML white background from Example #10 with the default background color from CONTAINER in Example #12. Since changes are made to HEADER, it is colored in **red**.

12) Inherit from a Distant Ancestor with STYLE…FROM

Code

```
proc template;
  define style styles.HeaderFromContainer;
   parent=styles.default;
   style HEADER from CONTAINER /
    FONT = fonts('HeadingFont')
    COLOR = BLACK;
  end;
run;
```

HTML Output

Inherit from a Distant Ancestor: CONTAINER -> HEADER (Bypass CELL and HEADERSANDFOOTERS)

The FREQ Procedure

Product	Frequency	Percent	Cumulative Frequency	Cumulative Percent
Boot	864	30.44	864	30.44
Sandal	564	19.87	1428	50.32
Slipper	794	27.98	2222	78.29
Sport Shoe	616	21.71	2838	100.00

Adapted From **#20 in the Lineage Tracer**: Container ⊗ Header

In Example #13, two STYLE statements are issued so that HEADER can inherit from ROWHEADER.

13) Inherit from a Descendent with STYLE…FROM

Code

```
proc template;
  define style styles.HeaderFromRowHeader;
  parent=styles.default;
  style rowheader from headersAndFooters /
    font = (arial,12pt,bold)
    foreground=blue;
  style header from rowheader /
    foreground=red;
  end;
run;
```

HTML Output

INHERIT FROM A DESCENDENT HEADERSANDFOOTERS -> ROWHEADER -> HEADER

The FREQ Procedure

Product	Frequency	Percent	Cumulative Frequency	Cumulative Percent
Boot	864	30.44	864	30.44
Sandal	564	19.87	1428	50.32
Slipper	794	27.98	2222	78.29
Sport Shoe	616	21.71	2838	100.00

Adapted From **#20 in the Lineage Tracer**: Container → Cell → HeadersAndFooters ⊗ RowHeader → Header

Example #14 is similar to Example #11. In both, two lineages are created to get the desired output. In Example #11, the second lineage was defined to restore defaults, whereas in Example #14 it is used to transfer CONTAINER settings to both HEADER and DATA. In Examples #14 and #16 below, the PROC title is being removed with ODS NOPTITLE in order to conserve space. Additional information about NOPTITLE can be found in *Output Delivery System: The Basics and Beyond* by Lauren Haworth, Cynthia Zender and Michelle Burlew[5, p. 436].

14) Create Uniform Tabular Output from Two Lineages with STYLE…FROM

Code

```
proc template;
  define style styles.HdrDataFromContainer;
  parent=styles.default;
  class CONTAINER /
    font=(arial, 14pt, bold)
    foreground=black;
  style HEADER from CONTAINER;
  style DATA from CONTAINER;
  end;
run;
```

HTML Output

Inherit from a Distant Ancestor: CONTAINER -> HEADER
Then from the same Ancestor to a Different Lineage: CONTAINER -> DATA

Product	Frequency	Percent	Cumulative Frequency	Cumulative Percent
Boot	864	30.44	864	30.44
Sandal	564	19.87	1428	50.32
Slipper	794	27.98	2222	78.29
Sport Shoe	616	21.71	2838	100.00

Adapted From **#20 in the Lineage Tracer**: Container → Container → Header

Adapted From **Previous lineage**: Container → Data

Finally, in Example #15 cross-lineage inheritance is used to generate uniform output for both titles and table. Use the Lineage Tracer to find out how and why the code works. Why do you think attribute settings in SYSTEMTITLE had to be changed? Remove them to verify your answer.

15) Create Uniform Output with Cross-Lineage Inheritance Using STYLE … FROM

Code

```
proc template;
  define style styles.Output2HandF2Data;
   parent=styles.default;
   class output /
     backgroundcolor=colors('docbg')
     bordercolor=colors('docfg')
     borderwidth=2
     borderspacing=2;
   style headersAndFooters from output;
   style data from headersAndFooters;
   style systemTitle from data /
     padding=0
     borderwidth=0
     borderspacing=0;
   style procTitle from systemTitle;
  end;
```

HTML Output

Get Uniform Output with Cross-Lineage Inheritance
OUTPUT -> HEADERSANDFOOTERS -> DATA -> SYSTEMTITLE -> PROCTITLE

The FREQ Procedure

Product	Frequency	Percent	Cumulative Frequency	Cumulative Percent
Boot	864	30.44	864	30.44
Sandal	564	19.87	1428	50.32
Slipper	794	27.98	2222	78.29
Sport Shoe	616	21.71	2838	100.00

Lineage Number of first appearance: Container #1 → Output #52 → HeadersAndFooters #6 → Data #2 → SystemTitle #66 → ProcTitle #64

CREATE A STYLE THAT EMBODIES COLOR PRINCIPLES DESCRIBED IN THE PAPER

Example #16 completes our discussion about styles. Three related shades of blue are used to color the entire output. Since output regions are naturally distinguished by different font settings, lineage-hopping for uniform output with STYLE…FROM is being replaced with CLASS statements where inheritance is restricted to a single lineage.

16) Colors for HEADERSANDFOOTERS and DATA are almost inverses of each other. Different font settings used for region identification are preserved with CLASS statements.

```
proc template;
define style styles.BlueOutput;
 parent=styles.default;
 class Container /
  backgroundcolor = cxEFF3FF;
 class HeadersAndFooters /
  color=cxEFF3FF fontsize=12pt
  backgroundcolor=cx00269A;
 class Output /
  backgroundcolor= cxEFF3FF
  bordercolor=colors('docfg')
  borderwidth=2 borderspacing=2;
 style Data from Output /
  color=colors('docfg')
  backgroundcolor=cxEFF3FF;
 class SystemTitle /
  backgroundcolor = cxEFF3FF
  fontStyle=roman fontSize=14pt;
 class SystemTitle2 / fontSize=12pt;
end; run;
```

Final Output uses Three Shades of the Same Blue
Only One STYLE...FROM Statement is Required

Product	Frequency	Percent	Cumulative Frequency	Cumulative Percent
Boot	864	30.44	864	30.44
Sandal	564	19.87	1428	50.32
Slipper	794	27.98	2222	78.29
Sport Shoe	616	21.71	2838	100.00

MAPPING STYLE TEMPLATES TO SAS PROCS: MEANS, FREQ, PRINT AND REPORT

Once styles are defined, they are mapped to SAS procedures that make up the "fixings" of the ODS "sandwich". Mapping occurs automatically if standard names such as HEADER and ROWHEADER are used in style definitions.

While automatic, mapping is not uniform among the various SAS PROCs. As we have seen, PROC FREQ automatically generates HEADERS, ROWHEADERS, and DATA. However, ROWHEADERS are not formatted in PROC MEANS or in PROC REPORT. From Delwiche and Slaugther [4,p. 167] we learn that output from PROC PRINT can change depending on which options are set for the procedure.

ODS output contained in the figures for this section use the Styles.Default template. To show that no style elements are named in ODS, source code for the first two PROC invocations in Figure 8 is listed below:

```
ods html path="&htmPath" (url=none)
file='defaultOtherProcs.html' style=STYLES.Default;
  title 'PROC MEANS with Automatic Style Element Mapping';
  title2 '(Only HEADER is Used by Default)';
  proc means data=sashelp.heart maxdec=3;
    var Height Weight Diastolic Systolic;
  run;
  title 'PROC PRINT with Automatic Style Element Mapping';
  title2 '(HEADER and ROWHEADER are Used by Default)';
  proc print data=styleApp.shoes2(obs=7) label;
  run;
  ...
ods _all_ close;
ods listing;
```

Figure 8. Output generated from the Default style for the MEANS and PRINT procedures shows that variation exists when it comes to automatic mapping. Nevertheless, SYSTEMTITLE, SYSTEMTITLE2, and DATA are the same in all four mappings.

PROC MEANS with Automatic Style Element Mapping
(Only HEADER is Used by Default)

The MEANS Procedure

Variable	N	Mean	Std Dev	Minimum	Maximum
Height	5203	64.813	3.583	51.500	76.500
Weight	5203	153.087	28.915	67.000	300.000
Diastolic	5209	85.359	12.973	50.000	160.000
Systolic	5209	136.910	23.740	82.000	300.000

PROC PRINT with Automatic Style Element Mapping
(HEADER and ROWHEADER are Used by Default)

Obs	Product	Subsidiary	Number of Stores
1	Boot	Addis Ababa	12
2	Boot	Al-Khobar	10
3	Boot	Algiers	21
4	Boot	Auckland	12
5	Boot	Bangkok	1
6	Boot	Bogota	19
7	Boot	Budapest	22

PROC PRINT with Automatic Style Element Mapping
(ROWHEADER is Erased with NOOBS)

Product	Subsidiary	Number of Stores
Boot	Addis Ababa	12
Boot	Al-Khobar	10
Boot	Algiers	21
Boot	Auckland	12
Boot	Bangkok	1
Boot	Bogota	19
Boot	Budapest	22

PROC PRINT with Automatic Style Element Mapping
(ROWHEADER is Restored with ID)

Name	Sex	Age	Height	Weight
Alfred	M	14	69.0	112.5
Alice	F	13	56.5	84.0
Barbara	F	13	65.3	98.0
Carol	F	14	62.8	102.5
Henry	M	14	63.5	102.5
James	M	12	57.3	83.0
Jane	F	12	59.8	84.5

Occasionally conflicts arise between ODS and the SAS PROC that is being exercised. For example, observe in Figure 9 that the ODS settings for the JUST attribute are overruled in PROC FREQ.

Figure 9. Headers in PROC FREQ are always right-justified. This makes sense, since all the DATA cells display numbers. Row headers, on the other hand, are left-justified to accommodate character strings. Note that PRODUCT is defined as a row header, not a header in PROC FREQ.

```
proc template;
  define style styles.FreqJust;
  parent=styles.blueOutput;
  class header/
    cellwidth=1in
    just=left;  /* GET RIGHT */
  class rowheader/
    cellheight=0.5in
    just=right; /* GET LEFT */
  end;
run;
```

JUST Fails in PROC FREQ for HEADER and ROWHEADER

GET LEFT _____ *GET RIGHT*

The FREQ Procedure

Product	Frequency	Percent	Cumulative Frequency	Cumulative Percent
Boot	864	30.44	864	30.44
Sandal	564	19.87	1428	50.32
Slipper	794	27.98	2222	78.29
Sport Shoe	616	21.71	2838	100.00

THE SPECIAL CASE OF PROC REPORT

While PROC REPORT has the same limited default mapping capabilities as PROC MEANS, it is very easy as Vincent DelGobbo points out to format columns in the REPORT, TABULATE and PRINT procedures [3]. In Figure 10, default mappings in PROC REPORT are displayed first. Then in Figure 11, style element overrides are applied to the first two columns in PROC REPORT. Finally in Figure 12, overrides are applied to user-named style elements that are abbreviations of the last four elements in Lineage #20. Also in Figure 12, the missing value in the "Total Returns" column is formatted with the DATAEMPTY style element that was introduced in the discussion about lineage #3 in Figure 5. The characters 'Missing' are inserted into the cell along with the altered colors by exercising RETFMT referenced in the DEFINE statement of PROC REPORT. RETFMT works well, because it maps all non-missing values to the embedded format, DOLLAR8.

Figure 10. The HEADER style element is mapped by default in PROC REPORT.

```
ods html …  style=STYLES.Default;
proc report data=shoes4 nowindows;
column
Product Subsidiary Nstores Sales Returns;
 Define Product    / ORDER;
 Define Subsidiary / ORDER;
 Define nStores / display '# Stores';
 Define Sales   / display format=dollar8.;
 Define Returns / display format=dollar8.;
run;
ods HTML close;
```

PROC REPORT with an Automatic Mapping for the HEADER

Product	Subsidiary	# Stores	Total Sales	Total Returns
Boot	Cairo	20	$4,846	$229
	Montreal	25	$40,213	.
	Moscow	23	$67,476	$3,142
	New York	18	$97,151	$3,983
Sandal	Cairo	9	$10,532	$598
	Montreal	7	$3,002	$122
	New York	1	$554	$23

Figure 11. Style element overrides are implemented with **STYLE(COLUMN)=STYLE-ELEMENT NAME** from Delgobbo [3].

```
ods html … style=STYLES.Default;
proc report data=shoes4 nowindows
split='*';
column
Product Subsidiary Nstores Sales Returns;
 Define Product    / ORDER
        'Product*(ROWHEADER)'
         STYLE(COLUMN)=ROWHEADER;
 Define Subsidiary / ORDER
        'Subsidiary*(ROWHEADEREMPHASIS)'
         STYLE(COLUMN)=ROWHEADEREMPHASIS;
 Define nStores / display '# Stores';
 Define Sales   / display format=dollar8.;
 Define Returns / display format=dollar8.;
run;
ods HTML close;
```

Two Columns are Formatted with Style Element Overrides

Product ROWHEADER	Subsidiary ROWHEADEREMPHASIS	# Stores	Total Sales	Total Returns
Boot	*Cairo*	20	$4,846	$229
	Montreal	25	$40,213	.
	Moscow	23	$67,476	$3,142
	New York	18	$97,151	$3,983
Sandal	*Cairo*	9	$10,532	$598
	Montreal	7	$3,002	$122
	New York	1	$554	$23

Figure 12. Style elements are renamed in PROC TEMPLATE with STYLE ... FROM statements to save space. The Missing value in RETURNS is highlighted by the assignment of DATAEMPTY in a compute block and by the application of RETFMT.

```
proc template;
define style styles.ColumnsOutput;
parent=styles.default;
 style Hdr from Header;
 style RowHdr from RowHeader;
 style RowHdrEmph from RowHeaderEmphasis;
 style RowHdrEmphFx from RowHeaderEmphasisFixed;
 class SystemTitle2 / fontSize=10pt;
 class DATAEMPTY / backgroundcolor=colors('headerfg') color=colors('headerbg');
end;
run;

proc format;
  value RetFmt .='Missing' other=[dollar8.];
run;

ods html path="&htmPath" (url=none)
body='ColumnsOutput.HTML'
style=ColumnsOutput;

proc report
 data=shoes4wMissVal nowindows nowindows split='*';
  column Product Subsidiary nstores Sales Returns;
  Define Product    / ORDER 'Product*(Hdr)'                 STYLE(COLUMN)=Hdr;
  Define Subsidiary / ORDER 'Subsidiary*(RowHdr)'           STYLE(COLUMN)=RowHdr;
  Define nStores    / display '# Stores*(RowHdrEmph)'       STYLE(COLUMN)=RowHdrEmph ;
  Define Sales      / display 'Total Sales*(RowHdrEmphFx)' STYLE(COLUMN)=RowHdrEmphFx
                      format=dollar8.;
  Define Returns    / display 'Total Returns*DATA(default)*OR*DATAEMPTY(missing)'
                      format=RetFmt.;
  compute Returns;
    if Returns eq . then
      call define ('_c5_', "style", "style=DATAEMPTY");
  endcomp;
run;

ods HTML close;
```

PROC REPORT with Abbreviated Hdr...RowHdrEmphFx for Lineage #20

#20 Trace: Hdr->RowHdr->RowHdrEmph->RowHdrEmphFx

Colors for the MISSING Value in RETURNS come from DATAEMPTY

Product (Hdr)	Subsidiary (RowHdr)	# Stores (RowHdrEmph)	Total Sales (RowHdrEmphFx)	Total Returns DATA(default) OR DATAEMPTY(missing)
Boot	Cairo	20	$4,846	$229
	Montreal	25	$40,213	Missing
	Moscow	23	$67,476	$3,142
	New York	18	$97,151	$3,983
Sandal	Cairo	9	$10,532	$598
	Montreal	7	$3,002	$122
	New York	1	$554	$23

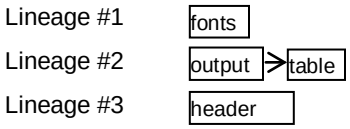
CONCLUSION: USE *LINEAGE* TO UNDERSTAND HOW STYLES ARE CONSTRUCTED IN ODS

Lineage is the major construct presented in this paper. By understanding that *lineage* always defines a style element's ancestors and descendents, it becomes possible to communicate *visually* how inheritance works in ODS. The visual tool is the *lineage tracer* also used for modeling individual traces found in many of the examples presented in the paper.

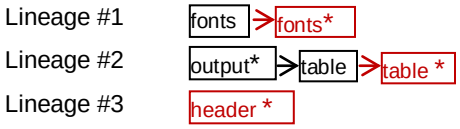
It should be noted that the tables of *ODS Style Elements* in Appendix 4 of the User's Guide are not substitutes for the *lineage tracer*. Multiple, partial lineages are listed in each of the tables that comprise Appendix 4. Again, what you need is a tracer that lists all the lineages in a style template separately and completely.

As a final exercise, take a look at the two styles, STYLE1 and STYLE2 on page 521 in the 9.2 ODS User's Guide. STYLE2 inherits from STYLE1. Before reviewing *Example Code 10.5*, on the following page, see if you can predict with greater ease what the final attribute settings will be from the following traces:

From STYLE1:



From STYLE2:



As in the previous examples, the black style elements in STYLE2 come from STYLE1. Attribute settings only have to be calculated for style elements that have asterisks appended to their names in STYLE2. Remember if the same attribute is set in a new style element with the same name as the old one (e.g. table from table), the attribute setting in new definition prevails; i.e. take the final attribute setting from table. If you complete this exercise and learn what is going on in each of the examples presented in the paper, you will be able to stretch your inheritance to get the output you want when you create a new style template.

COPYRIGHT STATEMENT

REFERENCES

[1] Carpenter, Art. *Carpenter's Complete Guide to the SAS® REPORT Procedure.* Cary, NC: SAS Institute Inc., 2007.

[2] *The Concise Oxford Dictionary of Current English: Ninth Edition.* Edited by Della Thompson. Oxford, England: Oxford University Press., 1995.

[3] Delgobbo, Vincent. *Traffic Lighting Your Multi-Sheet Microsoft Excel Workbooks the Easy Way with SAS®.* Proceedings of the 23rd Annual Northeast SAS Users Group Conference. Baltimore, MD 2010, paper #HW01.

[4] Delwiche, Lora D. and Susan J. Slaughter. *The Little SAS Book: A Primer, Fourth Edition.* Cary, NC: SAS Institute Inc., 2008.

[5] Haworth, Lauren E., Cynthia L. Zender, and Michele M. Burlew. *Output Delivery System: The Basics and Beyond.* Cary, NC: SAS Institute Inc., 2009.

[6] Preece, Jenny, et al. *Human-Computer Interaction.* Harlow, England: Addison-Wesley, 1994.

[7] Watts, Perry. *Advanced Programming Techniques for Working with Color in SAS® Software*. Proceedings of the Twenty-Ninth SAS® User Group International Conference, Cary, NC: SAS Institute, **2004,** paper #091.

[8] Watts, Perry. *Using Recursion to Trace Lineages in the SAS® ODS Styles.Default.Template*. Proceedings of the 23rd Annual Northeast SAS Users Group Conference. Baltimore, MD 2010, paper #BB13.

[9] Zender, Cynthia L. *SAS® Style Templates: Always in Fashion*. Proceedings of the SAS® Global Forum 2010 Conference. Seattle, WA, 2010, paper #033-2010.

[10] Zender, Cynthia L. *Tiptoe through the Templates*. Proceedings of the SAS® Global Forum 2009 Conference. Seattle, WA, 2009, paper #227-2009.

SAS INSTITUTE REFERENCES:

[11] SAS Institute Inc. *SAS 9.1 Output Delivery System: User's Guide,* Cary NC: SAS Institute Inc., 2004.

[12] SAS Institute Inc. *SAS 9.2 Output Delivery System: User's Guide,* Cary NC: SAS Institute Inc., 2008.

HOW TO LOCATE 251-2011.ZIP WHERE RELATED FILES ARE STORED:

AT SCREENCAST.COM

1) Go to http://www.screencast.com/users/PerryWatts
2) Click on README for a quick guide to website navigation.
3) Click on the folder icon labeled *ODS Style Inheritance*. This will take you to a subdirectory that contains abstracts for related papers. Click on *Add Style to ODS Output by Stretching your Inheritance in Version 9.2 SAS*, then click on "Attachments".

AT SASCOMMUNITY.ORG

1) Go to http://www.sascommunity.org
2) Follow sidebar links: Navigate>Popular Links>Presentations>SGF2011 Presentations

FILES ON 251-2011.ZIP:

1) The SHOES2 and SHOES3 data sets
2) SAS COLOR MACROS
   - HLStoRGB.
   - RGBDec.sas
   - RGBHex.sas
   - RGBtoHLS.sas
   - RGBtoHUE.sas
   - RGBtoLUM.sas
   - RGBtoSAT.sas

   The calling program:
   - GetALighter_DarkerColor.sas
3) Programs for Figures and Examples
   - UseCLASSstmt.sas
   - UseStyleWITHfrom.sas
   - UseSTYLEwNOfrom.sas
   - UsingOtherProcsByDefault.sas
   - UsingProcReport.sas
4) HTML Files (See Appendix for screen snapshots of all except Container92Lineages.HTML , shown in Figures 3,5, and 8)
   - AttributeDescriptor92.HTML
   - Container92Lineages.HTML
   - Normal92Lineages.HTML
   - Style92TemplateLineagesHighlighted.HTML

PAPERS IN THE COLOR FOLDER ON SCREENCAST.COM

Watts, Perry. *Using ODS and the Macro Facility to Construct Color Charts and Scales for SAS® Software Applications*. Proceedings of the Twenty-Seventh SAS® User Group International Conference, Cary, NC: SAS Institute, 2002, paper #125.

Watts, Perry. *Working with RGB and HLS Color Coding Systems in SAS® Software*. Proceedings of the Twenty-Eighth SAS® User Group International Conference, Seattle, WA, 2003, paper #136.

Watts, Perry. *Advanced Programming Techniques for Working with Color in SAS® Software*. Proceedings of the Twenty-Ninth SAS® User Group International Conference, Cary, NC: SAS Institute, 2004, paper #091.

Watts, Perry. *New Palettes for SAS® Color Utility Macros*. Proceedings of the Twenty-Ninth SAS® User Group International Conference, Cary, NC: SAS Institute, 2004, paper #162.

TRADEMARK CITATION

CONTACT INFORMATION

The author welcomes feedback via email at perryWatts@comcast.net

APPENDIX:

1) LINEAGE TRACER FOR SASHELP.TMPLMST STYLES (Style92TemplateLineagesHighlighted.HTML)

44 Lineages of 53 Sashelp.Tmplmst Styles

Color Coded to Reflect Naming Conventions for Style Element Definitions:

CLASS only (n=1), STYLE only (n=45), CLASS and STYLE (n=5), No CLASS or STYLE (n=2)

Lineage#	Implicit Parent	Style #1	Style #2	Style #3	Style #4
1	Base.Template.Style	styles.minimal			
2	Base.Template.Style	styles.normal	styles.festival	styles.festivalprinter	
3	Base.Template.Style	styles.normal	styles.meadow	styles.meadowprinter	
4	Base.Template.Style	styles.normal	styles.normalprinter	styles.seasideprinter	
5	Base.Template.Style	styles.normal	styles.plateau		
6	Base.Template.Style	styles.normal	styles.seaside		
7	Base.Template.Style	styles.sasweb			
8	Base.Template.Style	styles.default	styles.analysis		
9	Base.Template.Style	styles.default	styles.astronomy		
10	Base.Template.Style	styles.default	styles.banker		
11	Base.Template.Style	styles.default	styles.barrettsblue		
12	Base.Template.Style	styles.default	styles.beige	styles.d3d	
13	Base.Template.Style	styles.default	styles.blockprint		
14	Base.Template.Style	styles.default	styles.brick		
15	Base.Template.Style	styles.default	styles.brown		
16	Base.Template.Style	styles.default	styles.curve		
17	Base.Template.Style	styles.default	styles.education		
18	Base.Template.Style	styles.default	styles.egdefault		
19	Base.Template.Style	styles.default	styles.electronics		
20	Base.Template.Style	styles.default	styles.gears		
21	Base.Template.Style	styles.default	styles.harvest		
22	Base.Template.Style	styles.default	styles.highcontrast		
23	Base.Template.Style	styles.default	styles.journal	styles.journal2	styles.journal3
24	Base.Template.Style	styles.default	styles.listing		
25	Base.Template.Style	styles.default	styles.magnify		
26	Base.Template.Style	styles.default	styles.money		
27	Base.Template.Style	styles.default	styles.nofontdefault		
28	Base.Template.Style	styles.default	styles.ocean		
29	Base.Template.Style	styles.default	styles.printer	styles.fancyprinter	
30	Base.Template.Style	styles.default	styles.printer	styles.grayscaleprinter	
31	Base.Template.Style	styles.default	styles.printer	styles.monochromeprinter	
32	Base.Template.Style	styles.default	styles.printer	styles.rtf	
33	Base.Template.Style	styles.default	styles.printer	styles.sansprinter	
34	Base.Template.Style	styles.default	styles.printer	styles.sasdocprinter	
35	Base.Template.Style	styles.default	styles.printer	styles.serifprinter	
36	Base.Template.Style	styles.default	styles.rsvp		
37	Base.Template.Style	styles.default	styles.science		
38	Base.Template.Style	styles.default	styles.sketch		
39	Base.Template.Style	styles.default	styles.solutions		
40	Base.Template.Style	styles.default	styles.statdoc		
41	Base.Template.Style	styles.default	styles.statistical		
42	Base.Template.Style	styles.default	styles.theme		
43	Base.Template.Style	styles.default	styles.torn		
44	Base.Template.Style	styles.default	styles.watercolor		

2) PART OF THE LINEAGE TRACER FOR THE NORMAL STYLE (Normal92Lineages.HTML)

Part of the Styles.Normal template is being included here to show how much it differs in structure from the Styles.Default template. CONTAINER is pretty much the common ancestor for all non-graphics style elements in Styles.Default, whereas there is no corresponding common ancestor in Styles.Normal. DEFAULT listed at the head of lineages 18 - 54 is the common ancestor for only about half the lineages.

From Appendix #1 it can also be seen that both the Styles.Default and styles.Normal templates inherit from Base.Template.Style. However, Base.Template.Style does not dovetail with Styles.Normal. The lack of fit becomes evident when Styles.Normal HEADER attributes are displayed in a POPUP window. Most from the template are missing in the output.

72 Non-Graphics Lineages in the ODS Styles.Normal Template

Class Statements are in BLUE

Ambiguous or Erroneous Entries are in RED

Lineage#	Element#1	Element#2	Element#3	Element#4	Element#5	Element#6	Element#7	Element#8
1	bodydate							
2	bycontentfolder							
3	byline							
4	bylinecontainer							
5	caption	aftercaption						
6	caption	beforecaption						

...

18	default	aligncontents						
19	default	body						
20	default	container						

...

53	default	table	column	header	rowheader	rowheaderfixed	rowheaderstrongfixed	
54	default	table	column	header	rowheader	rowheaderstrong		
55	document							
56	fonts							
57	frame							
58	pageno							
59	pagesitem							
60	shutdownfunction							
61	startupfunction							
62	systemfooter							
63	systemtitle	normaltext	bodytext					
64	systemtitle	normaltext	drilldetaildata					
65	systemtitle	normaltext	drilldetailhover					
66	systemtitle	normaltext	heading1					
67	systemtitle	normaltext	heading2					
68	systemtitle	normaltext	heading3					
69	systitleandfootercontainer							
70	tablefootercontainer							
71	tableheadercontainer							
72	titleandnotecontainer							

3) PART OF THE ATTRIBUTE DESCRIPTOR (AttributeDescriptor92.HTML)

Non-Graphics Attributes from the 9.2 ODS User's Guide
(Blue Attributes are New for the 9.2 Release of SAS Software)

Attribute (9.2)	Alias (9.1.3)	Attribute Defintion	Legitimate Values
ACTIVELINKCOLOR		Color for active links	Any valid SAS color
ASIS		How leading spaces and line breaks are handled	ON or OFF. If OFF then leading spaces are trimmed, line breaks ignored.
BACKGROUNDCOLOR	BACKGROUND	Background color	Any valid SAS color
BACKGROUNDIMAGE		Background image	'string' where string is the name of a GIF or JPEG file identified with a simple file name, a complete path, or a URL. Easiest approach: use a simple filename and place all image files in the local directory.
BACKGROUNDREPEAT		Background Image repeat	NO_REPEAT, REPEAT (horizontally & vertically), REPEAT_X (horizontal), REPEAT_Y (vertical)
BODYSCROLLBAR		Scroll bar in the body file frame	YES | NO | AUTO, where AUTO specifies "only if needed".
BODYSIZE		Frame width for body file in HTML	nonnegative number + a unit of measurement (e.g. IN, CM, MM, PT) --OR-- integer% of entire display
BORDERBOTTOMCOLOR		Bottom border color of a table	Any valid SAS color
BORDERBOTTOMSTYLE		Bottom border line style of a table	A named line style (DASHED DOTTED DOUBLE GROOVE HIDDEN INSET OUTSET RIDGE SOLID)
BORDERBOTTOMWIDTH		Bottom border line width of a table	nonnegative number + a unit of measurement (e.g. IN, CM, MM, PT)
BORDERCOLOR		Single border color	Any valid SAS color.
BORDERCOLORDARK		Darker color used in a 3-D 2-color border.	Any valid SAS color.
BORDERCOLORLIGHT		Lighter color used in a 3-D 2-color border.	Any valid SAS color.
BORDERLEFTCOLOR		Left border color of a table	Any valid SAS color.
BORDERLEFTSTYLE		Left border line style of a table	A named line style (DASHED DOTTED DOUBLE GROOVE HIDDEN INSET OUTSET RIDGE SOLID)
BORDERLEFTWIDTH		Left border line width of a table	nonnegative number + a unit of measurement (e.g. IN, CM, MM, PT)

. . .

FONT		FONT= [face(s), size, keywords]. Keywords= font weight and font style. The destination device uses the first installed font	Syntax: "font-face-1 <... , font-face-n>", font-size, keyword-list myfont=("arial, helvetica", 4, bold roman) myfont=(Arial, 2pt, medium italic)
FONTFAMILY	FONT_FACE	Font Face	font-face-1 <... , font-face-n>
FONTSIZE	FONT_SIZE	Font size	A nonnegative number + a unit of measurement OR relative size (range=1-7)
FONTSTYLE	FONT_STYLE	Font style	ITALIC | ROMAN |SLANT
FONTWEIGHT	FONT_WEIGHT	Font weight	MEDIUM | BOLD |DEMI_BOLD|EXTRA_BOLD|LIGHT|DEMI_LIGHT|EXTRA_LIGHT
FONTWIDTH	FONT_WIDTH	Font width	Relative Width: NORMAL|COMPRESSED|EXTRA_COMPRESSED| NARROW|WIDE|EXPANDED

. . .

RULES		Lines within a table	ALL=between all rows and columns. COLS between all columns. GROUPS between the table header and the table and between the table and the table footer, if there is one. NONE no rules anywhere. ROWS between all rows
TAGATTR		Text inserted in the HTML code	'string'
TEXTALIGN	JUST	Justification in tables, cells, graphs	CENTER | DEC | LEFT | RIGHT (DEC means at decimal points)
TEXTDECORATION		Change visual presentation of text	BLINK|LINE_THROUGH|OVERLINE|UNDERLINE
TEXTINDENT	INDENT	Number of spaces to indent first line of text output	n=number of spaces. Default: 2 for XML, 0 everything else
URL		Specify target URL	'Uniform-Resource-Locator'
VERTICALALIGN	VJUST	Vertical justification	BOTTOM | MIDDLE | TOP
VISITEDLINKCOLOR		Visited link color	Any valid SAS color
WATERMARK		Translate the target for BACKGROUNDIMAGE into a "watermark. A watermark appears in a fixed position as the window is scrolled	ON|OFF
WIDTH	CELLWIDTH OUTPUTWIDTH	Width of a cell, table, line or graph	nonnegative number + a unit of measurement (e.g. IN, CM, MM, PT,PX) --OR-- integer% of table width