A Soft Landing to ODS 9.1

Overview

This document hopes to provide some information on the changes that occurred with the ODS destinations between SAS 8.2 and SAS 9.1. This is not intended as a "What's New" in ODS , but to point out changes between the releases that might affect your output in SAS 9.1. The HTML destination is one of the major changes to SAS 9.1. The HTML destination in SAS 9.1 is now a part of the ODS Markup family. When the HTML destination is opened in SAS 9.1, the HTML4 tagset is used to render the output. The beauty of this is that the HTML that is generated has been taken out of the C code and added to the tagsets. What this means to you is that you now have 100% flexibility in modifying output generated. In SAS 8.2, PROC TEMPLATE allowed modification of style information, however, it did not allow complete flexibility such as the ability to change the tags generated, change the flow of the output, or the ability to remove a tag such as the anchor tag. You now have the power to do this and quite a bit more. With the new tagset enhancements for SAS 9.1, you have the power to do anything that you are creative enough to think of.

The HTML destination keeping up with the industry changes and standards has updated the HTML generated from HTML 3.2 to HTML 4.0. As the version of HTML changes, keeping up with the industry demands so will the HTML generated with the ODS HTML destination. In the near future, some of those HTML validation programs such as the W3C's will no longer run cleanly using the HTML 3.2 output which was generated by the HTML destination by default prior to SAS 9.1. Accessibility issues were handled using the accessibility tagset for SAS 8.2. This tagset was downloaded from our web site and included such items as a summary, scope, alt and other tags. The new HTML destination incorporates all of this functionality by default without the need for this separate tagset.

File sizes are also smaller with the new HTML destination for 9.1. This is because internal CSS style sheets are used for formatting, which is a W3C mandate for HTML 4.0. All of the fluff have been removed from this HTML file such as the now deprecated alignment and font tags. Beginning in SAS 7 when the HTML destination was introduced, much of this was needed to render the output correctly on the majority of the browsers. Now the industry standards are pointing to Cascading Style Sheets as the formatting of choice. One benefit of the reduction of the file size is that the file does not take as long to load into the browser. There is new functionality and enhancements to all of the other destinations such as the PDF destination with compression being a default and the ability to embed fonts.

If you are new to the ODS destination for SAS 9.1, then you will think that ODS is the greatest enhancement to SAS since "sliced bread". If you have been using the ODS destination since SAS 7, then you will still think the same unless you are on the Atkins diet. If you have been using ODS for some time, you might incur a few growing pains and hopefully this document will get you over the hump.

As mentioned, the HTML generates HTML 4.0 output by default. All 4 below statements produce the same output in SAS 9.1.

```
ODS HTML file='foo.html'
ODS HTML  STYLESHEET file='foo.html'
ODS MARKUP  tagset=html4  file='foo.html'
ODS HTML4  file='foo.html'
```

Getting Started

If you have run your first program using ODS HTML in SAS 9.1 and the output looks still the same as it did in SAS 8.2, all is well. Then again, you might have run your program and saw some differences; the

output does not look quite the same as it did in SAS 8.2. The ODS development team has given you a way to go back to the HTML 3.2 output that you are use to. If desired, there are 2 ways to do this..

1) You can use the ODS HTML3 destination which will allow you to create the old HTML 3.2 output that was the default prior to SAS 9.1. Using this method, you have to modify all of your programs changing the syntax from ODS HTML to ODS HTML3. A tagset does not exist for the ODS HTML3 destination. (To prevent modifying your programs, let's look at the second method.)

```
ods html3 file="temp.html";
 proc print data=sashelp.class;
 run;
ods html3 close;
```

2) There is an entry in the registry which will allow you to use the HTML3 destination by default, eliminating the need to modify your programs. This can be done interactively or programmatically. The ODS HTML destination will use HTML4 by default. To modify this setting, follow the below instructions.

   1. From the TASK bar, select REGEDIT.
   2. Go to the ODS key and click
   3. "Default HTML Version"="<HTML3"|"HTML4>" (You will insert HTML3)

3) The above can also be done programmatically by exporting the appropriate registry keys out to a file and using PROC REGISTRY to import the keys.

```
data _null_;
  file "c:\oldhtml.sasxreg";
    put'[ODS]';
    put '"Default HTML Version" = "HTML3"';
run;

proc registry import="c:\oldhtml.sasxreg";
run;


ods preferences;
```

Differences

You will probably want to know where to expect differences in the ODS destinations in Version 9.1 that are different from what you received in SAS 8.2. I hope to explain the differences in this section and how to prepare for it.

HTML

STYLES

How styles are handled is one of the major differences from the prior release with the HTML destination. When PROC TEMPLATE was used to create styles , many of the now deprecated tags in the HTML file were added or modified. The tags in the HTML file included such things such as the fonts and alignments. These redundant tags helped inflate the size of the HTML file. The W3C mandates that the data and the style information be separated. By default, an internal cascading style sheet is generated with the default style information in SAS 9.1. If another style is applied other than the default, this template information is used to convert the PROC TEMPLATE definition to the appropriate internal CSS style sheet. This information is added between the <head> and </head> sections of the HTML file.

As the file gets larger, this greatly reduces the size of the file by associating the formatting instruction using the CLASS= attribute within the tags. Below are some of the differences that may result with this change.

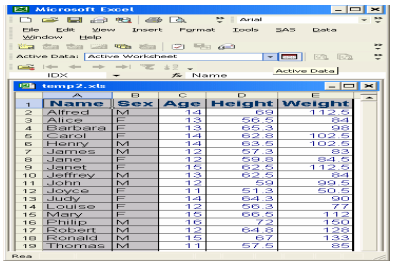
- When you specify a font size in SAS 9.1 using the font_size= attribute, the size of the font is generated in a relative size rather than absolute. The relative font sizes are specified as xx-small x-small, small, medium, large, and xx-large. These sizes are relative and interpreted by the browser to determine how they are displayed. If you run a job which modified the font size in 8.2 and run this job in SAS 9.1, the size of the component modified might not appear to be the same when viewed by the browser. For instance, the below PROC TEMPLATE code specifies a font size of 3 however, the internal CSS style property font-Size specified the relative size "small" . You will want to stay away from using a dimension such as a percentage. The relative sizes work well for the browser, which allows it to resize appropriately when modifying the size from the browser. If you specify a unit of dimension, then we will use absolute sizes which will be converted to points.

```
proc template;
  define style styles.test;
    parent=styles.default;
        style data from data /
           font_size=3;
end;
run;

ods html file='temp.html' style=styles.test;
proc print data=sashelp.class;
run;
ods html close;
```

Portion of the internal CSS file

```
.data
{
  font-family: Arial, Helvetica, sans-serif;
  font-size: small;
  font-weight: normal;
  font-style: normal;
  color: #000000;
  background-color: #D3D3D3;
}
```

- Styles with Excel do not work the same with the HTML destination in 9.1. This is because Excel does not fully support HTML 4.0 which SAS 9.1 produces by default. The output generated using the HTML destination may have some style problems , whether it's fonts or colors when viewed in Excel. There are a couple of target tagsets we suggest using with Excel in 9.1 which are the MSOffice2k and the ExcelXp tagsets for users of Excel 2002 and greater. For more information on these changes with Excel and Office, see the following link below.
http://support.sas.com/rnd/base/topics/templateFAQ/office91.pdf

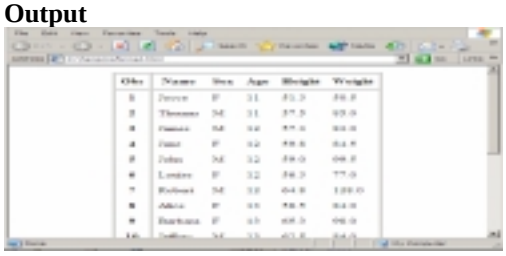


	Name	Sex	Age	Height	Weight
2	Alfred	M	14	69	112.5
3	Alice	F	13	56.5	84
4	Barbara	F	13	65.3	98
5	Carol	F	14	62.8	102.5
6	Henry	M	14	63.5	102.5
7	James	M	12	57.3	83
8	Jane	F	12	59.8	84.5
9	Janet	F	15	62.5	112.5
10	Jeffrey	M	13	62.5	84
11	John	M	12	59	99.5
12	Joyce	F	11	51.3	50.5
13	Judy	F	14	64.3	90
14	Louise	F	12	56.3	77
15	Mary	F	15	66.5	112
16	Philip	M	16	72	150
17	Robert	M	12	64.8	128
18	Ronald	M	15	67	133
19	Thomas	M	11	57.5	85

- Appending to the beginning of the HTML file using the NOTOP HTML sub-option will now prevent any style information from being generated. This is because the HTML destination in SAS 9.1 generates an internal CSS, which is stored in the header of the HTML file. Specifying the NOTOP option removes this information from the file. The CLASS= HTML attribute is used to apply the information from the CSS file to the current tag. If the NOTOP option is specified, the output will not have any formatting. For instance, if you are appending to the beginning of a HTML like the example below, all the formatting will be lost.

```
Filename temp ‘c:\temp.html’ mod;

Data _null_;
   File  temp;
   Put “<html>”;
   Put “<head>”;
   Put “<title>Browser title</title>”;
   Put “</head>”;
   Put “<body>”;
Run;

Ods html file=temp(notop);
Proc print data=sashelp.class;
Run;
Ods html close;
```

Output



The workaround for appending to the beginning of the file and keep the formatting is append the <link> tag to include the CSS file which has the formatting properties. The CSS file below is created in the first step to add with the <link> tag.

```
ods html path=”c:\”(url=none)  stylesheet="temp.css";
Filename temp 'c:\temp.html' mod;

Data _null_;
 File  temp ;
  Put "<html>";
  Put "<head>";
  Put "<link rel=""stylesheet"" type=""text/css"" href=""temp.css"">";
  Put “<head>”;
  Put "<body class=""body"">";
Run;

Ods html file=temp(notop) stylesheet="temp.css";
 Proc print data=sashelp.class;
 Run;
Ods html close;
```

- There has been a change in behavior to the STYLESHEET= option beginning with SAS 9.1.2 when pointing to a CSS file using the URL = sub-option. If you specify a path that includes spaces, the CSS that you attempted to use will not be found and you will receive black and white output like the table above. The enhancement to SAS 9.1 allows you to specify multiple CSS styles sheets separated by spaces. Therefore, if the path specified has a space, a separate link tag is generated for each part of the path where the space occurs. This creates and incorrect link to the CSS file. If the path to the CSS file has spaces, then the %20 needs to be specified which is an encoded space in the path where the spaces occur to prevent this path from breaking.

Incorrect

```
ods html file="temp.html" stylesheet=(url="c:\program files\sas
 institute\egdefault.css");
```

<link rel="stylesheet" href="c:\program
<link rel="stylesheet" href="files
<link rel="stylesheet" href="sas
<link rel="stylesheet" href="institute
<link rel="stylesheet" href="egdefault.css

Correct

```
ods html file="temp.html" stylesheet=(url="c:\program%20files\sas%20
institute\egdefault.css");
```

The enhancement is significant which allows you to specify multiple CSS style sheets.
This opens the door to doing such things as applying a style sheet for the various media types such as the viewed screen or printed output.

- Cascading Style Sheets are widely accepted by the major browsers since this is the formatting of the future with HTML, however, the level of support can vastly differ from browser to browser. For instance, OPERA supports landscape printing which not many other browsers support this. See the below chart for browser compatibility. You will probably want to familiarize yourself with CSS as this is the formatting of the future of HTML.
  http://www.corecss.com/properties/full-chart.php

- There are some template attributes that do not have CSS equivalents such as the BORDERCOLORDARK and BORDERCOLORLIGHT attributes. These attributes only worked for I.E which gave the 3D effect in SAS 8.2. You can use the new experimental style attributes that affect the individual borders of the table to mimic the BorderColorDark and BorderColorLight attributes.

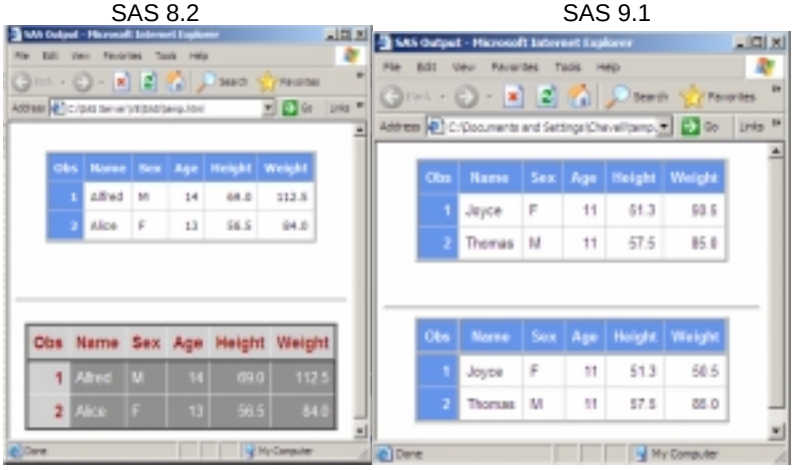
```
proc template;
 define style styles.test;
  parent=styles.minimal;
       style table from table /
         bordertopcolor=blue
         borderleftcolor=blue
         borderrightcolor=dark blue
         borderwidth=8;
   end;
run;

ods html file="temp.html" style=styles.test ;

proc print data=sashelp.class(obs=5);
title;
run;
ods html close;
```

Output

- We are no longer allowed to change the style information of tables and cells using the ODS HTML statement with the STYLE= option while the file is currently open. This is because we are using CSS for the formatting rather than HTML 3.2 tagging. Whether the CSS information is embedded within the file or stored externally, it is specified only once and each cell points to it using the CLASS= HTML attribute. To specify separate formatting for the second table, use the HTMLCLASS= style attribute to reference other style properties either in the external CSS file, internal CSS file, or with the HEADTEXT= option. The style can also be redefined by adding the various style attributes with the STYLE= option for the procedures that support this. This requires that each part of the output be formatted rather than the ability to specify an entire style definition which we could do at SAS 8.2. Note, when you use the STYLE= option to redefine the style, in-line style information is added to the HTML file.

```
Ods html file="temp.html" style=styles.sasweb ;
 Proc print data=sashelp.class;
 Run;

Ods html style=styles.brick;  <_-----------
 Proc print data=sashelp.class;
 Run;
Ods html close;
```

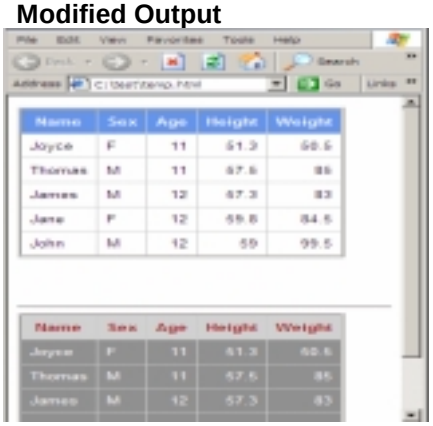
SAS 8.2 SAS 9.1



SAS 9.1 Workaround

```
/* This example redefines the colors for the foreground and the background */
/* of the headers and the cell values of the second table.                 */

Ods html file="temp.html" style=styles.sasweb;

 proc report data=sashelp.class nowd;
 Run;

 Proc report data=sashelp.class nowd
    style(header)={foreground=#A80000 background=#D2D2D2}  /* Header Style */
     style(column)={background=#909090 foreground=#ffffff}; /* Cell value  */
 Run;
Ods html close;
```

Modified Output



- Many more styles and style elements were created to affect graphics output for SAS 9.1. The below uses the Replace statement for the GraphFonts style element allowing the replacement of all of the fonts using PROC Template. All of the colors and many other style properties can be modified as well using PROC TEMPLATE. See the documentation for more information on this.

```
Proc template;
  Define style styles.test;
    Parent=styles.default;
     replace GraphFonts
       "Fonts used in graph styles" /
       'GraphDataFont' = ("Arial",8pt)
       'GraphValueFont' = ("Arial",10pt)
       'GraphLabelFont' = ("Arial",12pt,Bold)
       'GraphFootnoteFont' = ("Arial",12pt,Bold)
       'GraphTitleFont' = ("Arial",14pt,Bold);
End;
Run;
```

- The OUTPUTWIDTH= attribute works on the cell level in this release. In the past releases, this only worked on the Table element. Cellwidth and Outputwidth are aliases for one another, however, Cellwidth will be deprecated in this release even though it will still function the same. You will see this documented as Outputwidth in the future.

- There are new experimental style attributes such as: BorderBottomColor,BorderBottomStyle, BorderBottomWidth, BorderLeftColor, BorderLeftStyle, BorderLeftWidth, BorderRightColor, BorderRightStyle, BorderRightWidth, BorderStyle, BorderTopColor, BorderTopStyle, BorderTopWidth, Text_decoration and Text_indent. In SAS 8.2, you had the ability to modify these properties, but it had to be done with CSS style sheet properties within the HTMLSTYLE or CSSSTYLE attributes. By specifying style attributes , we can now supply style information for each cell individually using ODS style attributes for the borders. The Text_decoration style attribute allows you to strike-through or underline text, while the Text_indent attribute allows you to indent a value.

```
ods html file="temp.html" style=styles.minimal;

 proc print data=sashelp.class ;
 var age /    style={borderbottomcolor=red bordertopcolor=green};
 var height / style={borderbottomcolor=green bordertopcolor=blue};
 run;

 ods html close;
```

Obs	Age	Height
1	11	51.3
2	11	57.8
3	12	57.3
4	12	59.8
5	12	59.0
6	12	56.3

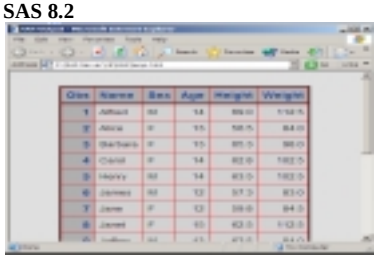
- The ODS Markup language affords you the opportunity to do anything that you are creative enough to imagine. This new language will revolutionize the way that you do things. This is because you have access to all pieces of the output that are generated, therefore you have the ability to generate new formats, or modify existing tagsets and formats. You now have complete control over the output that you generate. SAS 9.1 introduced Data step functions to the ODS Markup language.

- The HEADTEXT= option does not allow style overrides as it did with SAS 8.2. In SAS 8.2 and below, this option could be used to add style information that would override HTML tags without using PROC TEMPLATE. There are a couple of reasons why this does not work the same in SAS 9.1. Prior to SAS 9.1, we used HTML 3.2 which modified the tagging directly in the HTML file. We now use CSS for the formatting in 9.1. The order of importance in the application of styles is; the closer the style to the data is the style that is used. Because we use CSS, the CLASS= attribute is the closest to the data and what is used. The below code would override the table tag in 8.2 to modify the background color, but will not work in 9.1. This syntax can be modified a couple of ways to get this to work correctly in SAS 9.1. The !important CSS rule can be specified after the CSS style property to override what has been specified in the CSS file, otherwise what has been specified in the style sheet will override. An alternate method is to specify the CLASS name which is the same as the style element except it is preceded by a ".". This will also allow you to override the style element as well.

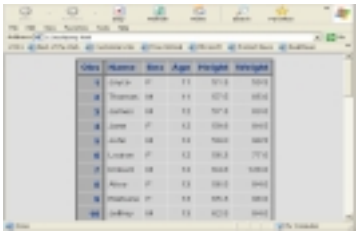
  SAS 8.2

```
ods html file="temp.html" headtext="<style> table {background-color:red}</style>";

proc print data=sashelp.class;
run;

ods html close;
```

SAS 8.2



SAS 9.1



SAS 9.1 workarounds

```
/* Example uses the !important rule */
ods html file="temp.html" headtext="<style>
                           table {background-color:red ! important}</style>";

proc print data=sashelp.class;
run;
```

```
ods html close;

/* Example uses class names which are the same as the element names */
 ods html file="temp.html" headtext="<style> .table {background-color:red}</style>";

 proc print data=sashelp.class;
 run;

 ods html close;
```

- When using the Java and Activex Devices, the graph is output in XML with SAS 9.1.

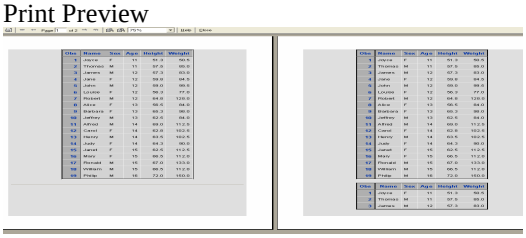
- If you specify the NOBYLINE system option, the By-values in the table of contents will not display the value. The label “BygroupN” is displayed instead which is different from the previous release.

- The TEXT= option has been added to the HTML destination for SAS 9.1. The TEXT= option has been there for the RTF and PDF destination since SAS 8.2 which allows text to be written to the output at the point the TEXT= option is specified. This option is great for adding both text and HTML code in-stream. This can be used in place of the PUT statement in the Data step to write HTML or text. There are many uses for this statement in the HTML destination. One such usage is the ability to add page breaks wherever you want them in the document by placing the TEXT= option where you want the page break to occur. What you see on the screen as a page break is the <HR> (Horizontal rule) tag, however, this tag does not cause a physical page break when you send the output to the printer. This is handled by the CSS style property Page-break-after:always or page-break-before:always. We could turn off the page break as a whole for the document using PROC TEMPLATE and selectively add the page break information. The below example generates a page break after the first table and combines the second and third table on the same page. The output is a screen shot taken from print preview.

  /* Turn off page breaks for the entire document */

```
 proc template;
 define style styles.test;
   parent=styles.default;
        style body from body /
         pagebreakhtml=_undef_;
   end;
run;

ods html file="temp.html" style=styles.test;

proc print data=sashelp.class;
run;

ods html text="<hr/><p style=""page-break-after:always"">";

 proc print data=sashelp.class;
run;

proc print data=sashelp.class(obs=3);
run;

ods html close;
```

  Print Preview



- SAS 9.0 introduced the ODS NOUSEGOPT statement. This statement tells ODS not to use the goptions specified for titles and footnotes. This was a bit of a problem in SAS 8.2 when the goptions such as FTEXT= was specified with a hardware font such as Swissb. ODS has no idea of this hardware fonts and generated a warning and used the default font. If you want the goptions specified to override the font specified in the default style, use the ODS USEGOPT statement which will restore the old behavior. The ODS NOUSEGOPT statement is the default. The below code generates a title with a courier font and foreground text of red as it did in SAS 8.2.

```
ods usegopt;
goptions ftext=courier ctext=red;

ods html file="temp.html";

proc print data=sashelp.class(obs=5);
title "this is a new title";
run;

ods html close;
```

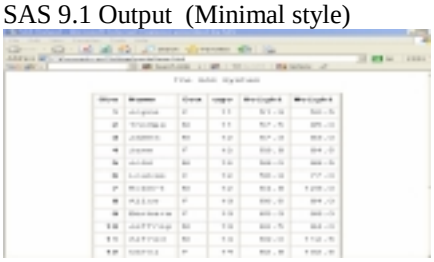
- If you've used the non-breaking space character ( ) with the HTML destination in 8.2 to add spaces, then you might see a difference when using this in SAS 9.1. If you specify the non-breaking space character to add spaces in 9.1, this HTML entity is protected and you will see the literal text " " rather than the space which this character represents in HTML. The HTML generated will protect this character and it becomes   in the HTML source. To make sure that this character is not protected in SAS 9.1, you can set the attribute PROTECTSPECIALCHARS= to the value OFF, or make sure that the first and last characters of the string is an HTML tag. See the below example. In SAS 8.2, the HTML blocking tags such as the   tags or the PROTECTSPECIALCHARS= attribute were not needed to add the non-breaking space character to a string. If you need embedded spaces, also look at the NOBREAKSPACE= style attribute as a workaround to adding the non-breaking space tags.

/* The label for age has 3 embedded spaces */

```
ods html file="temp.html";

proc print data=sashelp.class label;
label age="<div>this   date</div>";
run;

ods html close;
```

- Because the HTML 4.01 specifications are different from HTML 3.2, the layout of the file in SAS 9.1 is different than was created in SAS 8.2 by default. The below URL should help you understand the layout of the HTML file in SAS 9.1. For instance, if you used the Minimal style in SAS 8.2, you will see that the output does not look quite the same in SAS 9.1. You will see that the headers and the row headers appear to be bolded. The header and row header appear to be bolded because we now use the <TH> tags for the headers and row headers rather than the <td> tag and has an appearance of being bolded. This in fact is following the guideline for SAS 4..01. There are also some specifics that I detail below., but to understand the structure of the file, see the below link.

SAS 9.1 Output (Minimal style)

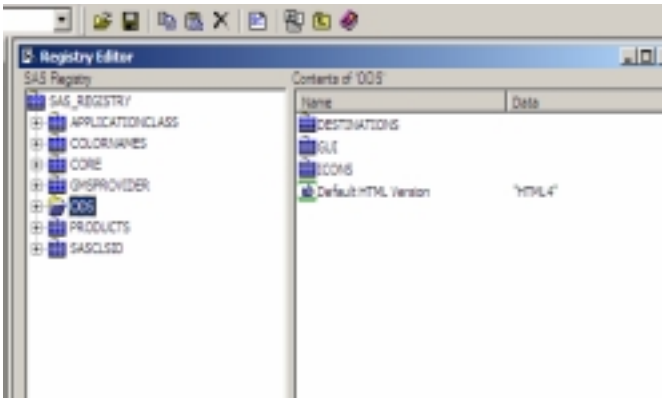


The Results Window and the SAS Registry

Some of the ODS keys in the SAS/Registry have been changed for SAS 9.1. The Preferences key in the SAS registry has a new path. The key now resides in a new location which is [ODS\GUI\RESULTS] rather than the earlier location which was [ODS\Preferences\] .

Because the registry path has changed, if you try preventing the Results Viewer from popping up programmatically, you will need to change the keys and the value for SAS 9.1.

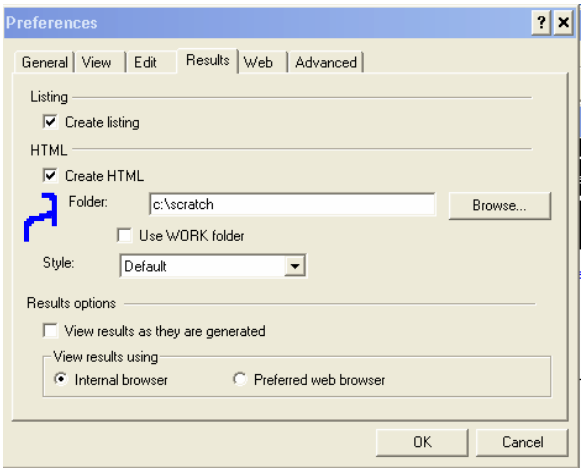
```
data _null_;
  file "c:\oldreg.sasxreg";
     put'[ODS\GUI\RESULTS]';
     put '"Auto Navigate" = "Off"';
run;


proc registry import="c:\oldreg.sasxreg";
run;


ods preferences;
```



Setting a Default Path for HTML

You can set a default path for both the local files that get generated with ODS and well as the HTML files that get generated as a result of generating HTML by default. This is done by going to the Results tab (Tools->Options->Preferences->Results) and selecting HTML. The Folder box is used to either select or specify the folder that you want your HTML files to appear. Note: this selects the location for the default HTML files created and HTML file created with the ODS HTML statement. If you create HTML using ODS HTML and you specify a path for the file, an error will error. This is because we prepend the path specified for the folder with the path specified with ODS. To specify a different path than indicated in the folder of the Results tab, use the ODS HTML PATH= option with the URL=NONE sub-option. Below is the error that occurs if this is not done.

```
1    ods html file="c:\temp\temp.html";
NOTE: Writing HTML Body file: c:\temp\temp.html
ERROR: A component of c:\test\c:\temp\temp.html is not a directory.
WARNING: No body file. HTML output will not be created.
NOTE: Writing HTML Body file: sashtml1.htm
run;
```



The OUTPUT destination

When it comes to the output destination, there has been a major change from the prior releases. In releases prior to 9.1, the MATCH_ALL= option was used to create a macro variable with all of the data sets created from the procedure. The macro variable could then be used to concatenate all of the data sets created into a single data set. Beginning with SAS 9.1, this is no longer necessary because the data sets will get appended by default without having to use the MATCH_ALL option on the ODS OUTPUT statement. This option is still available, however, it has been deprecated. If you use this option, you will also get a note in the log mentioning this.

In this release, there are some table structures that have changes since SAS 8.2. Unfortunately, I do not have all of the changes at my disposal. From release to release, there is a possibility of changes made to the underlying template. As these differences are known, we will attempt to document them through SAS Notes.

ODS OBJECTS

The behavior of ODS Objects have changed since SAS 8.2 as it pertains to the PATH of tables. The path of tables in the trace record generated with the ODS Trace statement addresses tables as TABLEn rather than by "variable1_by_variable2" for crossings or simply field name as in the prior releases. This is more dynamic and you don't have to worry about the 32 character truncation limit anymore. Looking at the below PROC FREQ statement, the table path specifies each object generated in the path using the TABLEn syntax.

For instance, the below PROC FREQ generates 3 tables with the paths:

```
ods trace on;
Ods output onewayfreqs=test;

Proc freq data=sashelp.class;
   Table age  sex*height  weight;
Run;
```

```
Output Added:
-------------
Name:       OneWayFreqs
Label:      One-Way Frequencies
Template:   Base.Freq.OneWayFreqs
Path:       Freq.Table1.OneWayFreqs
-------------
Output Added:
-------------
Name:       CrossTabFreqs
Label:      Cross-Tabular Freq Table
Data Name:
Path:       Freq.Table2.CrossTabFreqs
-------------

Output Added:
-------------
Name:       OneWayFreqs
Label:      One-Way Frequencies
Template:   Base.Freq.OneWayFreqs
Path:       Freq.Table3.OneWayFreqs
-------------
```

NEW Path	OLD Path
Freq.Table1.Onewayfreqs.	Freq.Age.OneWayFreqs
Freq.Table2.Crosstabs	Freq.Sex_by_Height.CrossTabs
Freq.Table3.Onewayfreqs	Freq.Weight.OneWayFreqs

ODS TRACE

When you have excluded objects, it no longer shows the records in the trace record as in the previous releases. For instance, consider that we exclude the MOMENTS output object from PROC UNIVARIATE, it does not show the MOMENTS in the trace records. Beginning in SAS 9.1.2, excluded objects are shown only when the Excluded option is added to the ODS Trace statement.

```
Ods trace on;
Ods exclude moments;

Proc univariate data=sashelp.class;
Run;
```

PDF

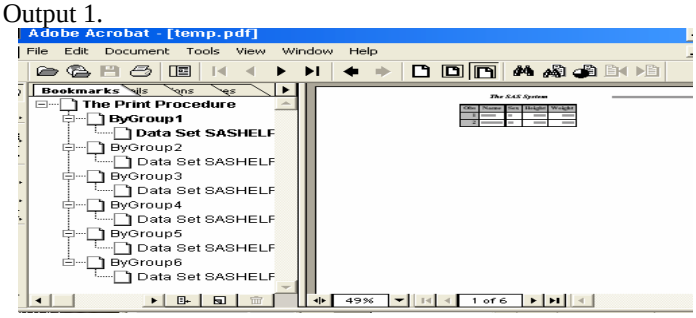
Creating PDF (batch) output such as Data _null_, PROC FORMS, PLOT, CHART, and PROC COMPARE embeds the SAS Monospace font rather than the Courier font as it did with SAS 8.2. This output looks a little different than it did in SAS 8.2. We can change the font to restore the old behavior by modifying the Batch style element of PROC TEMPLATE and supplying the desired font.

```
Proc template;
   Define style styles.newstyles;
       Parent=styles.printer;
          Style batch from batch /
              Font_face=courier;
End;
Run;
```

Another issue that you might want to be aware of is Adobe no longer ships the PDF Writer driver with Adobe 6.0. If you use the PRINTER= option with this driver to distill the output within SAS, it will not find the driver. Adobe does still ship the Acrobat Distiller which you can still use. Now that we have compression and embed Fonts with SAS 9.1, there should not be as great a need to distill the output as in past releases.

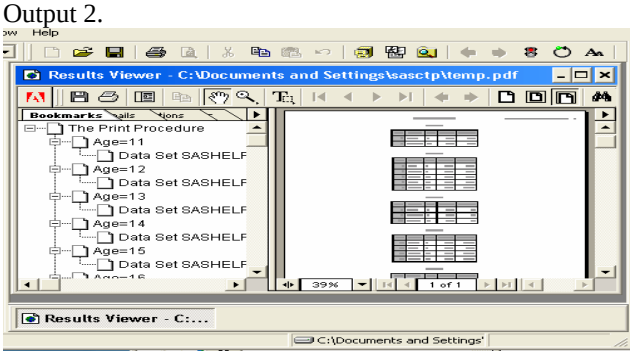
If you use the NOBYLINE system option to get rid of the BYLINE information in the PDF/HTML file, the entry in the table of contents does not include the By-value, but the string "BygroupN". The only way around this currently is to remove the NOBYLINE option. See output 1 for an example of the output.

Output 1.



The PROC PRINT procedure now adds the label information in the bookmarks when By-groups are created. In SAS 8.2 the label information was not present. To get rid of this so that it will look like the 8.2 output, you can use the CONTENTS= option with a null string on the PROC PRINT statement.

Output 2.



There is the ability to create GIF files with the style information from procedure and data step output. This is very helpful when you want to do such things as import output into PowerPoint. This is done by using the ODS PRINTER statement and the PRINTER= option specified with the value GIF. This is considered experimental for SAS 9.1. This will allow you to maintain style information that you create with ODS Printer and only requires Base SAS.

```
Ods printer SAS file="c:\temp.gif" printer=gif ;

Proc print data=sashelp.class style(column)={background=yellow}
style(header obs obsheader)={background=brown foreground=yellow} ;

Run;

Ods printer close;
```

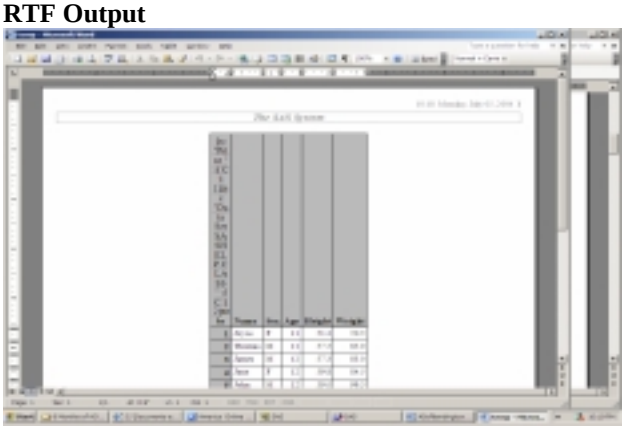
GIF file created

Obs	Name	Sex	Age	Height	Weight
1	Joyce	F	11	51.0	50.5
2	Thomas	M	11	57.5	85.0
3	James	M	12	57.3	83.0
4	Jane	F	12	59.8	84.5
5	John	M	12	59.0	99.5
6	Louise	F	12	56.3	77.0
7	Robert	M	12	64.8	128.0
8	Alice	F	13	56.5	84.0
9	Barbara	F	13	65.3	98.0
10	Jeffrey	M	13	62.5	84.0
11	Alfred	M	14	69.0	112.5
12	Carol	F	14	62.8	102.5
13	Henry	M	14	63.5	102.5
14	Judy	F	14	64.3	90.0
15	Janet	F	15	62.5	112.5
16	Mary	F	15	66.5	112.0
17	Ronald	M	15	67.0	133.0
18	William	M	15	66.5	112.0
19	Philip	M	16	72.0	150.0

RTF

If you use the Paragraph option in WORD and use the TEXT= option, or have the “Show hidden Text” or “Print hidden text” options specified, stray information will be printed or shown in the header of column 1. This is because we now have Table of Content information produced by default which gets picked up by word. In SAS 9.13 SP2, there is a NOTOC_DATA ODS RTF option to prevent writing this information. This information is generated in accordance with the RTF 1.6 specification.

RTF Output



Using the Minimal style with the RTF destination does not work the same as it did in SAS 8.2. The Minimal style does not contain any style information other than the rules for the table and the border . In SAS 8.2 when the RTF file was opened in Word, because the file did not contain any style information such as fonts, it used Word’s default which is Times New Roman. In SAS 9.1, the host adds a default font if none is found unlike SAS 8.2. On windows the SAS Monospace font is added to the font table in SAS 9.1 if no font is found, therefore, the default Times New Roman is not used when the RTF file is opened in Word. The Courier font is added to the font table on the non-windows platforms when no font is found. On the windows platform you can specify the SYSPRINTFONT= option specifiy the font to add to the font table for all of the output while still using the Minimal style.

SAS 8.2 RTF font table

```
{\fonttbl
{\f1\fnil\fprq0\fcharset0 ;}
```

SAS 9.1 RTF font table on Windows

```
{\fonttbl
{\f1\fnil\fprq0\fcharset0 SAS Monospace;}
```

XML Engine

The option XMLSCHEMA= has been changed from XMLSCHEMA= in SAS 8.2 to XMLMETA= for SAS 9. The XML MAP technology is production beginning in SAS 9.0, therefore you will not need to download the experimental modules as you did in Version 8.2. Also, SAS 8.2 now has a production image of the XML Engine which is used for reading non rectangular files.

If you generate map files, you will want to install the XMLMapper. This is a GUI Java application which allows you to build the map without much effort. This was introduced in Version 9.0 as the XML Atlas, but has since been renamed the XMLMapper.

SAS 9.1.3 added the new CDISC procedure. This has a also been added via a production hotfix for SAS 8.2. The CDISC procedure imports and exports XML documents that are in CDISC ODM 1.2 format.

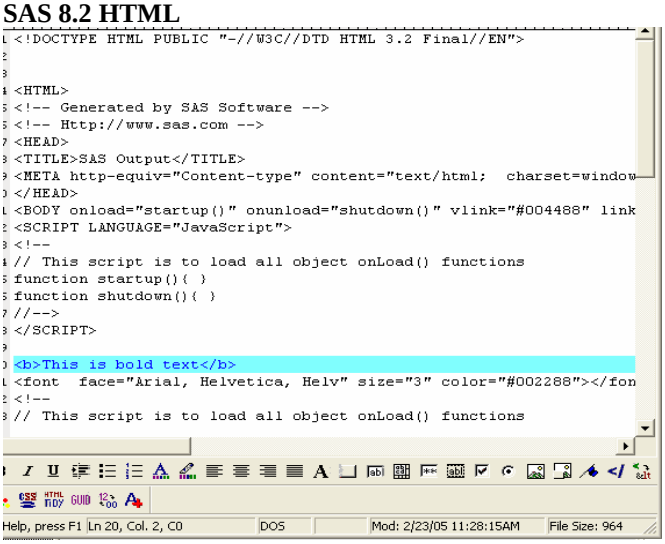
SAS/IntrNet &ODS

Beginning in SAS 9.1.2, you can no longer add _WEBOUT on a FILE statement of the Data step while the ODS destination is already open with _WEBOUT. This was allowed in prior releases but did not always work correctly because we were writing to the same location. For example, the Data step below would write the bolded text "**the is bold text**" prior to SAS 9.1.2. However, beginning with SAS 9.1.2 no text is written to the output. If you view the HTML file, the text is output, but not in a location that can be rendered. Writing to _WEBOUT while the file is open is no longer allowed. The HTML generated from the below code shows the output written to the body section of the HTML file where it can be rendered in 8.2. If you look at the HTML generated in the SAS 9.1 you will see that the text is output to the header-style section of the HTML file which is why it is not rendered. If you look at the log, there is 1 record output, just not in a location where it can be rendered.. Your legacy code might have used this format to create forms, selection boxes and various other forms of HTML in the prior release, which default appended to you generated ODS output such as your procedures. The next paragraph will discuss the workaround for this new behavior.

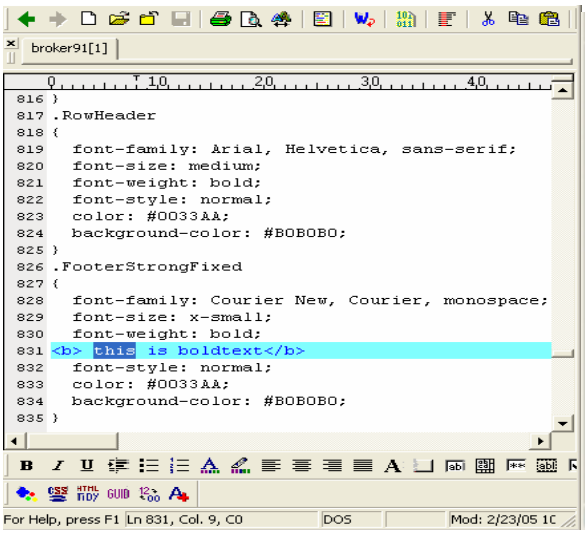
Generates no viewable output in SAS 9.1

```
ods html file=_webout;

data _null_;
   file _webout;
       put "<b>this is bold text</b>";
run;
ods html close;
```

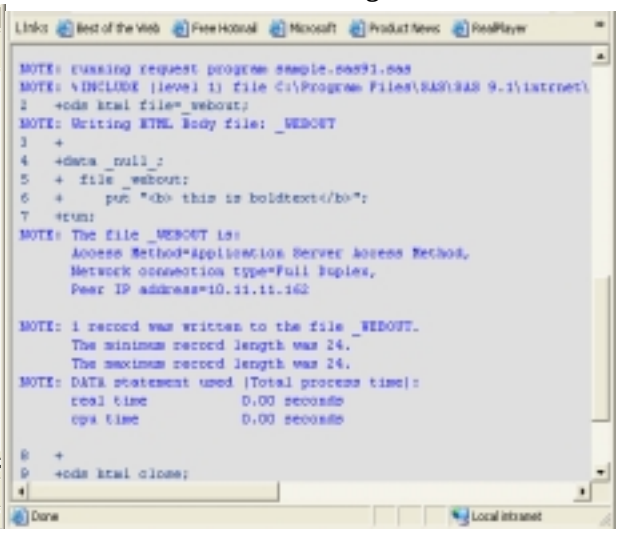
SAS 8.2 HTML



SAS 9.1 HTML



SAS 9.1 OUTPUT with Debug 131



The workaround for this new behavior is to use the new TEXT= option of the ODS HTML statement. This option is new to the HTML destination for SAS 9.1 which was discussed in an earlier section. This option will allow you add HTML and text in stream. When the TEXT= option is specified, the argument is placed within the HTML paragraph tags( ). The benefit of this is that you do not have to worry about the tags in this argument getting protected and displayed as text. This is because as long as the first and last characters are HTML tags, ODS turns off the level of protection. If you are displaying the value of SAS variables, then you will need to use the PUT statement rather than the TEXT= option. However, the PUT statement would need to be appended before of after the ODS HTML statement using the NOTOP or NOBOT HTML options. Prior to SAS 9.1.2, this information was appended by default. You can also use the ODS HTML3 statement to create the old behavior. This will generate the HTML 3.2 output that was generated prior to 9.1.

Generates output in SAS 9.1

```
ods html file=_webout;

data _null_;
   ods html text=”<b> This is bold text</b>”;
run;
ods html close;
```

Defects

Because of a defect, omission of the ODS HTML close statement will generate a Read access violation error. This is fixed in 9.13. See SAS Note 11595.

Templates

Migration

If you have style templates that were generated in SAS 8.2, they can still be read and written to and from the same template store in SAS 9.1 on the same operating system. This is not true if you are on one of the UNIX boxes in SAS 9.1 which is 64 bit. The 64 bit versions of SAS cannot read templates generated with a 32 bit version. Templates created on a 32 bit OS are not recognized, therefore, the only error you get is that the template is not found. If you are running on a 64 bit version of SAS, you will need to rebuild the templates by recompiling the template’s source code. This can be done by using PROC TEMPLATE with the SOURCE statement along with the FILE= option. The below macro does this dynamically for an entire template store.

The below macro creates a file with the source code to rebuild the templates based on the template store that you specify with the ODS PATH statement. You will need to add the beginning PROC TEMPLATE statement and the ending RUN statement to the generated code. The macro has 2 parameters; file=(the filename the we use to store the source code) and OUTDS= (the output data set containing the name of all of the styles converted.).

```
libname temp ‘c:\templib’;
Ods path(prepend) temp.templat(read);     /* Specify the template store that you want to access */


%macro test(file=c:\new.txt,outds=xxx);
ods output stats(match_all=&outds)=&outds(where=(type="Style"));

proc template;
list;
run;

proc sql noprint;
 select count(path) into:total
  from &outds;
 select path into:path1-:path%left(&total)
  from &outds;
  run;
  quit;

  filename file "&file" mod;

  proc template;
   %do i=1 %to &total;
      source &&path&i / file=file;
   %end;
  run;
%mend;
%test()
```

Tagsets

Tagsets created with the experimental ODS Markup language in SAS 8.2 might not generate the same output in SAS 9.1. The reason that the output generated might not be the same as the experimental 8.2 version is because of all of the new enhancements to the ODS Markup language. The events that you used in SAS 8.2 are still there, but there have been quite a few more added with this release. Also, tagsets are the window into the procedures and ODS. If the underlying code changes for the fore-mentioned, then the tagsets would need to change to reflect the updates. Some of the tags in the various destination have been updated requiring that the tagsets change. Because you have complete control with the ODS Markup language, you can still generate the same output that you created in SAS 8.2, however, you might have to update your tagset.

E-mail

The HTML file generated with SAS 9.1 is a little different then the file generated with SAS 8.2. This is because we generate an internal CSS file, separating the formatting from the data. You will need to be aware of this change when using the FILENAME statement with the E-mail access method.

- To send an e-mail attachment with records that contain more than 256 characters, the ATTACH email-option in the FILENAME, EMAIL statement now supports the LRECL and RECFM options.

- If you have output that creates stacked tables (more that one value per cell) and the job worked in 8.2 and is getting truncated in SAS 9.1 this could be due to a defect in SAS 9.1. This will happen if you have a lot of variables in procedures such as PROC MEANS, PROC UNIVARIATE which creates stacked tables. In SAS 8.2, this added a line break after each variable in the HTML file, however, in SAS 9.1, this streams across the page and will truncate or wrap after a certain length. This is also true when using multiple LINE statements with PROC REPORT. See SAS Notes 13467 for more details.

MVS

- When specifying the TRANTAB=ASCII option with ODS in SAS 9.1, you will see a new note in the log.This new note is because of an enhancement with the LOCALE= system option. In SAS 9.1, the LOCALE= is set for English_UnitedStates with the English version as it is set for the other languages. This is a difference from SAS 8.2 where the default was “English”, therefore, the LOCALE= option did not need to be set. This means that the first table listed with the TRANTAB= overrides the ASCII table. If you run using the NLSCOMPATMODE option, you will see a different list of tables used when the NONLSCOMPATMODE option is specified which will generate incorrect output because a different translation table is used.. You will want to run with the NONLSCOMPATMODE option when using the TRANTAB=ASCII option with ODS. For more information on this or restoring the 8.2 behavior, please see the below SAS Note.
  http://support.sas.com/techsup/unotes/SN/013/013513.html

  NOTE: TRANTAB=ASCII interpreted as TRANTAB=EOL1WLT1 via Locale Specific Setup.

- If you’ve created EBCDIC output on MVS with the ODS HTML destination, you will need to make sure to use the RS=NONE ODS HTML sub-option. This is because with the additional CSS information in the header of the HTML file that streams will cause the output to truncate or wrap. This internal CSS file was not there in by default prior to SAS 9.1. The RS=NONE option allows ODS to

write one observation at a time. You will be able to determine if the output has been truncated or wrapped because you will see stray characters where the tags have been split or truncated or incorrect style information.

Miscellaneous Errors

If you receive Java Plug-in errors when using an ODS destination in 9.1, see the below SAS Note which will explain how to avoid this error. A dialog box will pop-up with the below.
http://support.sas.com/techsup/unotes/SN/013/013594.html

```
Register Failk

Cannot load class sun/plugin/JavaRunTime.
```

The Document Facility

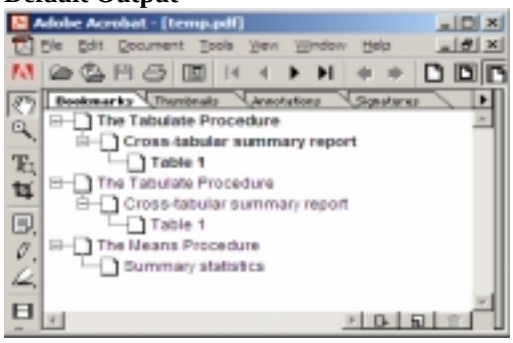
The Document facility became production in SAS 9.0. This facility allows you to store output in a new ODS destination called an ODS Document. Saving output in this format prevents from having to save output in third party formats, but rather save the file in this file type called a document. This document can then be replayed to any of the other ODS destinations. The document facility also allows you to replay a portion of the document, re-order the objects and delete objects. This also allows you to fully customize the bookmarks with ODS PDF or the TOC with the HTML destination. Take the below example which manipulates the order of the objects to create a less busy looking bookmark list.

In the below output, the first 2 PROC TABULATE procedures add 3 nodes in the bookmark list each. Each of the 3 levels point to the same part of the output within the main window, therefore you might desire to control the hierarchy or nesting level. Instead of having 3 nodes for each of the PROC TABULATE procedures, and two levels for PROC MEANS, we can create a single header and a single entry for each piece of output. To do this we first need to locate the paths of the objects that you would like to manipulate. You can find this specifying the LIST statement of PROC DOCUMENT.
On the LIST statement you can define the levels of output that you would like displayed. After looking at the levels, you can see exactly how the output is structured. If you routinely use the same procedures, you will most likely not have to view the paths after a while. This can all be done interactively using the Document window.

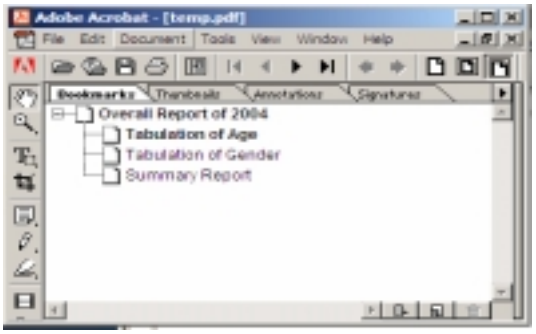
```
ods document name=test(write);

proc tabulate data=sashelp.class;
class age;
table age;
run;

proc tabulate data=sashelp.class;
class sex;
table sex;
run;

proc means data=sashelp.class;
run;

ods document close;


PROC DOCUMENT NAME=TEST;
LIST/ LEVELS=ALL;
```

```
Run;
QUIT;


ods pdf file="temp.pdf";

proc document name=test;
/* Set working directory that we use to copy objects */

dir \Tabulate#1;

/* Set the label of the objects */

Setlabel \Tabulate#1                  “Overall Report of 2004”;
setlabel \Tabulate#1\Report#1\Table#1 "Tabulation of Age";
setlabel \Tabulate#2\Report#1\Table#1 "Tabulation of Gender";
setlabel \Means#1\Summary#1           "Summary Report";

/* Copy objects to working directory */

copy \Tabulate#1\Report#1\Table#1 to ^;
copy \Tabulate#2\Report#1\Table#1 to ^;
copy \Means#1\Summary#1 to ^;

/* Delete directory paths not needed */

delete \Tabulate#1\Report#1;
delete \Tabulate#2;
delete \Means#1;

replay;

run;
quit;
ods pdf close;
```

Default Output



Restructured Output



Conclusion

There are more items that will surface which I will attempt continue to update this document with. Please visit the Technical Support home page and search the SAS Notes for related defects.

Contact Information

You can contact me at Chevell.Parker@sas.com. Also visit the Base R&D web site which is located at : http://support.sas.com/rnd/base/ for a host of other useful ODS and BASE SAS topics.