I'll Have the <u>TABULATEs a la ODS</u> Please, With a <u>Table of Contents</u> On the Side

Ray Pass
Ray Pass Consulting

Abstract

The advent of the Output Delivery System (ODS) in Version 8 (all right, it was really introduced in V7, but I still refer to V7 as V8 beta) of the SAS System is truly monumental in scope. It required rewriting all SAS procedures that produce output (not all do) and splitting out their "data components" from their "table definitions" (presentation format templates). We now have the ability to combine these components into customized "output objects" and to send them to different output "destinations", including casting them as HTML pages. When we do use the HTML destination, we also have the ability to produce our main content output as one component frame of a multi-frame page, the other possible component frames being either procedure or page oriented Tables of Contents. Although there are methods for customizing the contents of the Tables of Contents frames, they still do take up valuable screen real estate. This paper demonstrates a methodology for creating separate stand-alone Tables of Contents for a multi-output ODS-HTML run, with navigational tools included to go back and forth between the Table of Contents and the content pages. Other data-driven techniques are also demonstrated for renaming ODS-HTML generated sequential body file names into more meaningful content-oriented names. The techniques are not difficult when based on the power of ODS and simple SAS programming tools.

Introduction

Starting with Version 7 of the SAS System, the creation of procedure output has been significantly and positively enhanced due to the advent of the Output Delivery System, or ODS to its friends. Waste no time in becoming one of its friends if you have not already done so. ODS is a major undertaking in that it reflects a complete rewrite of all SAS procedures that produce output. Whereas before ODS came along, the creation of a procedure's data was inextricably interwoven with the creation of the display of the output data, these two functions are now totally separate. ODS allows us to take the pure "data components" from SAS procedures, and combine them with various types of "table definition" format templates, to produce customized "output objects" which can be sent to various output "destinations". These "destinations" include standard SAS listing output, HTML formatted output, PostScript ready printer output, SAS data set output and MS Word compatible RTF output (there are also additional experimental output destinations such as LaTeX and CSS stylesheets.)

This paper will be restricted to ODS-HTML output and will present a method for creating an alternate to one of the packaged features of ODS, namely the optional creation of Table of Contents frames to accompany the main "body" frame of the output. If desired, ODS can produce a single frame "body" content output, or it can be instructed to produce the "body" content as one frame in a multi-frame output. If this latter option is chosen, the "body" content frame of the output occupies a large right-hand portion of the page, and the smaller left hand portion of the output page can contain either a procedure-oriented Table of Contents frame, or a page-oriented Table of Pages frame, or both. With simple content output, this can be quite useful in terms of file or page navigation within the total procedure output. In addition, there are methods available to perform limited customization of the Table of Contents and/or Table of Pages. However, when the procedure creates multiple pages of sophisticated output, as can happen with a complex PROC TABULATE, or when the outputs of multiple procedures are to be combined in an output-system, screen real estate becomes much more valuable. It then becomes desirable to dedicate the entire main output screen to the body content, and to build separate stand-alone Tables of Contents, or systems of Tables of Contents, with full built-in screen-to-screen navigability. This can be easily done by enhancing the values of selected data points and/or simple Title statement strings, to contain not only data but hot-link references to other pages as well. The techniques themselves are simple and basic, although the final programs can become quite large and sophisticated.

This paper will first present a sample of the standard ODS system-provided Table of Contents and Table of Pages, and will then present an example of building a system of separate Tables of Contents with full navigational ability. It will also discuss techniques for amplifying (renaming) the sequentially labeled body files created by default by ODS. This technique is totally data-driven and yields meaningful content-specific names for the files.

ODS-Provided Table of Contents

Out-of-the-box, ODS comes with options for creating a multi-frame HTML output as well as a single frame design. If you only specify a `body=` file-specification, you get only a main content body page. You can also however add a `frame=` file-specification to your ODS HTML statement. If you do so, you must add either a `contents=` file-specification, a `page=` file-specification, or both. These give you respectively, a Table of Contents frame on your output page displaying all the various procedure outputs that you create, or a Table of Pages

frame displaying all the various *pages* of output that you create. These are often identical, however there are many instances wherein a procedure creates many pages of output. The following diagram represents a typical ODS standard output screen containing a multi-frame output as described above. *(All diagrams in this paper are NOT true screenshots, but rather are constructed facsimiles.)*

The SAS System

The UNIVARIATE Procedure
Variable: INPDOL

Moments			
N	1319	Sum Weights	1319
Mean	289.777582	Sum Observations	382216.63
Std Deviation	1268.59913	Variance	1609343.75
Skewness	7.65461147	Kurtosis	74.4715424
Uncorrected SS	2231872876	Corrected SS	2121115065
Coeff Variation	437.783739	Std Error Mean	34.9302755

Basic Statistical Measures			
Location		Variability	
Mean	289.7776	Std Deviation	1269
Median	0.0000	Variance	1609344
Mode	0.0000	Range	18880
		Interquartile Range	0

ODS does give you the ability to resize all the frames in the standard display, and there are some customization features available for the Table of Contents in which you can do some re-labeling of output descriptions and some font customization. When dealing with a large and complex reporting system as the one used in the application described in this paper however, screen real estate is quite important and the limited amount of shared space allotted to the Tables of Contents and the body output do neither justice. It is just not possible to optimally include enough output content information on the page as well as a meaningful Table of Contents

Alternate ODS Table of Contents
This paper reports on work done on a set of business reports to transform them from a SAS V6 solution to a SAS V8 solution. Originally, SAS and Excel were used to create a series of Excel spreadsheets which were then distributed as either files or hardcopy. With the SAS V8 solution, the reports are now created as HTML pages and are distributed on a company intranet. The final reports are complex in that they are the result of a multi-platform set of raw data sources, and because they are multi-dimensional in structure.

All of the needed tables are created in a series of data-driven macros. At the most granular level, each report is created via a lengthy PROC TABULATE run resulting in a table looking something like the following (the tables presented here are only partial representations of the actual tables, and contain randomly generated data):

Month Table of Contents   *Year Table of Contents*

		YE99 N	YE99 %	MAR00 N	MAR00 %	APR00 N	APR00 %	MAY00 N	MAY00 %	YTD00 N	YTD00 %
Vehicle Tier	Premier	20,000	57.1%	2,000	58.7%	1,500	57.5%	1,000	55.4%	10,000	49.9%
	Preferred	10,000	28.6%	1,000	29.3%	800	30.7%	600	33.2%	6,000	29.9%
	Standard	5,000	14.3%	400	11.7%	300	11.5%	200	11.1%	4,000	19.9%
	Unknown	100	0.3%	10	0.3%	10	0.4%	5	0.3%	60	0.3%
		35,100	100.0%	3,410	100.0%	2,610	100.0%	1,805	100.0%	20,060	100.0%
Blended Policies	Blended	10,000	11.1%	1,000	9.1%	800	11.8%	600	10.7%	5,000	7.7%
	Straight	80,000	88.9%	10,000	91.0%	6,000	88.2%	5,000	89.3%	60,000	92.3%
		90,000	100.0%	11,000	100.0%	6,800	100.0%	5,600	100.0%	65,000	100.0%
Policies with Restricted Vehicles	Restricted	5,000	5.6%	9,500	86.4%	800	11.8%	1,000	17.9%	5,500	8.5%
	Not Restricted	85,000	94.4%	1,500	13.7%	6,000	88.2%	4,600	82.1%	59,500	91.5%
		90,000	100.0%	11,000	100.0%	6,800	100.0%	5,600	100.0%	65,000	100.0%
Many more report variables	...	...	...	...	...	...	...	...	...	...	...
	...	...	...	...	...	...	...	...	...	...	...
	...	...	...	...	...	...	...	...	...	...	...
	...	...	...	...	...	...	...	...	...	...	...

The above table is the result of a two-dimensional PROC TABULATE. The actual application includes this "OVERALL" table (note the top level Title line) as well as a series of tables breaking the data down further by using a third "page"dimension. Note also that the above table is for "ALL STATES". Similar tables are created for each individual state represented in the data. In addition, sets of tables are created each month (this one is for MAY2000 – 2nd Title line), with the months contained in the report rollling forward with each new monthly run. So, the final application is actually "5-dimensional" (month, state and the three dimensions contained in each PROC TABULATE.) Since PROC TABULATE can create a maximum of three reporting dimensions (page, row, column), and since the reports are quite wide in terms of screen space, it was decided to create separate stand-alone Tables of Contents with full navigational features.

The list of third TABULATE page dimensions (including the OVERALL level – collapsing all pages), as well as the states reported on (including "ALL STATES"), is contained in a generated Month Table of Contents (one per month), and the list of sets of monthly reports is contained in a generated Year Table of Contents. Each Table of Contents is tabular in structure with the cells containing hot links to lower level pages. The Year Table of Contents contains links to various Month Tables of Contents, and each Month Table of Contents contains links to data tables. Each page also contains hot links back to the higher level Tables of Contents in the Title lines. The Tables of Contents look something like this (once again, these are only partial renditions):

AUTOMOBILE Book of Business ---YEAR TABLE OF CONTENTS---

New **QUOTES** as of: Jun 30, 2000

*** Click on any Month-Year to go to reports for that Month-Year. ***

Obs	Month and Year of Reports
1	May, 2000
2	April, 2000
3	March, 2000
4	February, 2000
5	January, 1999
6	December, 1999
7	November, 1999
8	October, 1999
9	September, 1999
10	August, 1999
11	July, 1999
12	June, 1999

AUTOMOBILE Book of Business ---MONTH TABLE OF CONTENTS---

New **QUOTES** for MAY2000 as of: Jun 30, 2000

*** Click on any state-category combination to go to that page. ***

	Overall	by Comp/ NonComp	by Tier	by Business	by Blend	by Single/ Multiple	by Seg/ Unseg
ALL STATES	ALL-OVER	ALL-CNC	ALL-TIER	ALL-BUS	ALL-BLEND	ALL-SM	ALL-SUS
AL	AL-OVER	AL-CNC	AL-TIER	AL-BUS	AL-BLEND	AL-SM	AL-SUS
AR	AR-OVER	AR-CNC	AR-TIER	AR-BUS	AR-BLEND	AR-SM	AR-SUS
AZ	AZ-OVER	AZ-CNC	AZ-TIER	AZ-BUS	AZ-BLEND	AZ-SM	AZ-SUS
CO	CO-OVER	CO-CNC	CO-TIER	CO-BUS	CO-BLEND	CO-SM	CO-SUS
CT	CT-OVER	CT-CNC	CT-TIER	CT-BUS	CT-BLEND	CT-SM	CT-SUS
DC	DC-OVER	DC-CNC	DC-TIER	DC-BUS	DC-BLEND	DC-SM	DC-SUS
Etc	…	…	…	…	…	…	…

From the viewpoint of the user, the application begins with the Year Table of Contents (this is actually the page that appears when the user decides to view these reports by clicking on a hot link on a higher overall system-wide Table of Contents.) After choosing a specific month to view, all states and "by" reports are viewable (clickable). Navigation to other "by" reports, or other monthly reports are always available.

Each Table of Contents is created as a simple PROC PRINT output. When a PROC PRINT output is sent to HTML via ODS, the default ODS template highlights the column headers and the Obs column as seen in the Year Table of Contents diagram. If any report variable is declared as an ID variable, the template then highlights the column for that variable, as shown for **state** in the Month Table of Contents diagram (the Obs column is not printed when an ID variable is present.) The Month Table of Contents also uses an HREF enriched Title statement to provide a link back to the Year Table of Contents.

Creation of the Year Table of Contents
The independent Tables of Contents are created each month along with the sets of data reports. The Year Table of Contents is updated so as to only include twelve months of reports, and each new Month Table of Contents is totally data-driven so as to only include those states for which there are data. At present, all states are used each month, but the system is flexible enough to handle exceptions, should they occur.

Throughout this paper, pieces of code will be given to demonstrate certain features of the system. These are merely excerpts of the overall program, which is quite large and highly intra-connected within itself. The code is also adjusted herein to conform to the columnar restrictions of these proceedings. The following code is part of that used to create the Year Table of Contents:

```
%let t_sp  =
    %nrstr(     );
%let today = %sysfunc(today(),worddate12.);
*-----------------------------------------;
ods html body = "_yeartoc.htm";
*-----------------------------------------;
title1 "<H3>AUTOMOBILE Book of Business
        &t_sp --YEAR TABLE OF CONTENTS---
        </H3>";
title2 "<H3>New **QUOTES** &t_sp as of:
        &today</H3>";
title3 "<H4>*** Click on any Month-Year to
        go to reports for that Month-Year.
        ***</H4>";
run;
*-----------------------------------------;
data yeartoc;
   set yeartoc;
```

```
   label monyrhtm = 'Month and Year of
                     Reports';

   if _n_ = 1 then do;
      output;
      monyr    = input("&my3",monyy7.);
      monyrexp = put(monyr,monname9.)||','
                 ||put(monyr,year4.);
      monyrhtm = "<A HREF='&my3d/_toc.htm'>"
                 ||monyrexp||"</a>";
      output;
   end;

   else output;
run;
*-----------------------------------------;
proc sort data=yeartoc nodup;
   by descending monyr;
run;
*-----------------------------------------;
data yeartoc;
   set yeartoc(obs=12);
run;
*-----------------------------------------;
proc print data=yeartoc label;
   var monyrhtm;
run;
*-----------------------------------------;
```

The system starts with an input parameter for the current month (in mmmyyyy format) which ends up in a macro variable called **my3**. This is used above to create a new data set variable **monyr**, which is then transformed into **monyrexp**, and finally into a string contained in variable **monyrhtm**. This string is constructed as an HTML Anchor tag which displays as a hot link when viewed in a web browser. Each month, a new observation is added to the permanent SAS data set called **yeartoc**. The data set is sorted (in descending order), unduplicated (to account for update runs *within* a month) and then culled back to only 12 observations. The final data set is the run through ODS and PROC PRINT and its output is saved as an HTML file called "_yeartoc.htm". This is the initial data set of the process and is referred to in the Title statements for all other files.

Creation of the Month Table of Contents and the TABULATE Report Tables
The Month Table of Contents is also created via a PROC PRINT output, but it is much more data-driven. Before we look at the code for it's creation however, we must take a short tangential trip to a naming issue in ODS-HTML-land, and also a longer trip to the production of the actual TABULATE output. Much of what happens here is intermingled and can not be separated out in a neat sequential modular manner. Eventually it all does come together however.

All SAS output goes through ODS. Even if you do not use any ODS statements at all, the output is still being routed to the default ODS LISTING destination. If the ODS HTML destination is opened via an `ods html` statement, then all output is routed there until an `ods html close` statement is issued. When ODS creates a series of "body" files deriving from one `ods html` statement, it names the files sequentially by appending sequential numbers to the given `body=` name. For example, given the following opening ODS code snippet,

```
*----------------------------------------;
ods html body = bodyname.htm;
*----------------------------------------;
```

files created after this code would be named: "bodyname.htm", "bodyname1.htm", "bodyname2.htm", etc. This is as it should be because ODS has no way of knowing what you *really* want to call the files, unless you tell it. You could continually open and close the ODS HTML destination, but that takes away the possibility of automating the job through data-driven processing.

Data-driven techniques are those in which code is system-created, usually through macro processing, which reflects characteristics of the data being processed. In the present situation, default sequentially named ODS HTML output files are renamed with meaningful state labels, depending on the states that are present in the input data. This is accomplished in steps. In the sequence that follows, we first set up some macro variables needed for the renaming, then we create the TABULATE output, and then we do the actual file renaming. At that point, we create the Month Table of Contents.

First, three macro variables are created from the input data as follows:

```
*----------------------------------------;
proc sql;
   create table states as
      select   distinct state
      from     source
      order by state;
quit;
*----------------------------------------;
data states;
   set states;
   num = _n_;
run;
*----------------------------------------;
proc sql noprint;
   select count(*),
          state,
          num
   into   :mcount,
          :mstates separated by '#',
          :mnums   separated by '#'
   from   states;
quit;
*----------------------------------------;
```

The first thing that happens above is that we create data set `states` containing an ordered unduplicated list of those states that are contained in the data. A counter variable, `num` is then added to the data set which assigns a sequential number to each state. We then create three macro variables as follows:

`mcount` - contains the number of states
`mstates` - contains a #-delimited list of the states
`mnums` - contains a #-delimited list of numbers corresponding to the values in `mstates`

After these macro variables are created, all of the ODS HTML files are created via a series of TABULATE runs. The code for these will not be included in any detail at all here because it is not germane to the techniques under discussion. Skeletally, this is what happens.

A few Title statement macro variables are created as follows:

```
*----------------------------------------;
%let t4a = <H4><A HREF='_toc.htm'> Month
           Table of Contents</A>;
%let t4b = <A HREF='../../_yeartoc.htm'>
           Year Table of Contents</A></H4>;
*----------------------------------------;
```

The source data set is sorted by `state` in preparation for later PROC TABULATEs to be run using `state` as a byvar. A macro is then built and run which executes a PROC TABULATE for "ALL STATES". This macro, and the next to be discussed, has many input parameters, the most important of which is `&body`. The value of this parameter is passed into the macros to be used as the "page" dimension variable in the PROC TABULATEs. The code looks something like this:

```
*----------------------------------------;
%macro taball(body=, …);
   ods html newfile   = proc
            body       = "&body..htm"
            stylesheet = "_tab.css"
                         (url="_tab.css");
   *--------------------------------------;
   title1 "<H3>AUTOMOBILE Book of Business
           &t_sp ALL STATES (&by1)</H3>";
   title2 "<H3>New **QUOTES** for &my3
           &t_sp as of: &today</H3>";
   title3 "<H4>(all pages should be printed
           in landscape format)<H4>";
   title4 "&t4a &t_sp &t4b";
   *--------------------------------------;
   proc tabulate data=source missing
                 format=comma9.0
                 style=[font_size=1];
      class &body …
      table &body,
      (rest of TABULATE code)
   run;
%mend;
*----------------------------------------;
```

Next, another macro is built and run which executes a TABULATE procedure using `state` as a byvar to create a separate table for each individual state contained in the data. The code looks something like this:

```
*---------------------------------------;
%macro tabxst(body=, …);
   ods html newfile   = bygroup
            body       = "&body1..htm"
            stylesheet = (url="_tab.css");
   *---------------------------------------;
   title1 "<H3>AUTOMOBILE Book of Business
           &t_sp STATE: #byval(state)
           (&by1)</H3>";
   title2 "<H3>New **QUOTES** for &my3
           &t_sp as of: &today</H3>";
   title3 "<H4>(all pages should be printed
           printed in landscape format)
           </H4>";
   title4 "&t4a &t_sp &t4b";
   *---------------------------------------;
   proc tabulate data=source missing
                 format=comma9.0
                 style=[font_size=1];
     by state;

     class &body …
     table &body,

     (rest of TABULATE code)
   run;
%mend;
*---------------------------------------;
```

The first macro executes a PROC TABULATE for all the states combined. The second macro runs the TABULATE by `state`, creating separate tables, *and* separate HMTL output files, for each state. Note that in the first macro, a `newfile=proc` option is used, creating one HTML file for the whole TABULATE output (all states combined), and in the second macro, a `newfile=bygroup` option is used, creating a new HTML file for each individual `state` byvar value.

The macros are called once for each "by" group as follows:

```
*-- ALL STATES COMBINED ------------------;
%taball(body=over,  …)
%taball(body=cnc,   …)
%taball(body=tier,  …)
%taball(body=bus,   …)
%taball(body=blend, …)
%taball(body=sm,    …)
%taball(body=sus,   …)

*-- BY STATE ----------------------------;
%tabxst(body=over,  …)
%tabxst(body=cnc,   …)
%tabxst(body=tier,  …)
%tabxst(body=bus,   …)
%tabxst(body=blend, …)
%tabxst(body=sm,    …)
%tabxst(body=sus,   …)
```

At this point, HTML files exist with names like:

```
over.htm,  over1.htm,  over2.htm …
cnc.htm,   cnc1.htm,   cnc2.htm  …
tier.htm,  tier1.htm,  tier2.htm …
etc.
```

The final file renaming is accomplished through a macro loop containing operating system delete and rename commands. This system is run on a UNIX platform so the `rm` and `mv` commands are used. Different commands would be used with different operating systems. The code looks something like the following:

```
*---------------------------------------;
%macro names;
   %do m=1 %to &mcount;
      %let nm = %scan(&mnums  ,&m,#);
      %let st = %scan(&mstates,&m,#);
      %if %sysfunc(fileexist(over&nm..htm))
      %then %do;
         %sysexec rm over&st..htm;
         %sysexec rm cnc&st..htm;
         %sysexec rm tier&st..htm;
         %sysexec rm bus&st..htm;
         %sysexec rm blend&st..htm;
         %sysexec rm sm&st..htm;
         %sysexec rm sus&st..htm;

         %sysexec mv over&nm..htm
                     over&st..htm;
         %sysexec mv cnc&nm..htm
                     cnc&st..htm;
         %sysexec mv tier&nm..htm
                     tier&st..htm;
         %sysexec mv bus&nm..htm
                     bus&st..htm;
         %sysexec mv blend&nm..htm
                     blend&st..htm;
         %sysexec mv sm&nm..htm
                     sm&st..htm;
         %sysexec mv sus&nm..htm
                     sus&st..htm;
      %end;
   %end;
%mend names;
*---------------------------------------;
```

This macro first checks for the existence of the "over.htm" file. If this file exists, the macro then deletes previous copies of the state-named files such as "overAL.htm", "overAR.htm", etc. It then renames each file from its numeric version to its corresponding state-named version.

```
over1.htm  becomes  overAL.htm,
over2.htm  becomes  overAR.htm,
etc.
```

The correspondence between numbers and states was accomplished earlier in the process when the `mstates` and `mnums` macro variables were established.

Now we are finally ready to create the Month Table of Contents. Here is the code:

```
*---------------------------------------;
proc template;
   define style styles.lhead;
      parent=styles.default;
         style header from
                headersandfooters
                / just=left;
   end;
run;
*---------------------------------------;
ods html body  = "_toc.htm"
         style = lhead ;
*---------------------------------------;
title1 "<H3>AUTOMOBILE Book of Business
        &t_sp--MONTH TABLE OF CONTENTS---
        </H3>";
title2 "<H3>New **QUOTES** for &my3 &t_sp
        as of: &today</H3>";
title3 "<H4>*** Click on any state-category
        combination to go to that page. ***
        </H4>";
title4 "<H4><A HREF='../_yeartoc.htm'> Year
        Table of Contents</A></H4>";
run;
*---------------------------------------;

data toc;
  length state over cnc tier bus blend sm
         sus $ 100;

  label state = '00'x
        over  = '\\Overall'
        cnc   = 'by\Comp/\NonComp'
        tier  = '\by\Tier'
        bus   = '\by\Business'
        blend = '\by\Blend'
        sm    = 'by\Single/\Multiple'
        sus   = 'by\Seg/\Unseg';

  state = 'ALL STATES';
  over  = "<A HREF='over.htm'> ALL-OVER
           </A>";
  cnc   = "<A HREF='cnc.htm'> ALL-CNC
           </A>";
  tier  = "<A HREF='tier.htm'> ALL-TIER
           </A>";
  bus   = "<A HREF='bus.htm'> ALL-BUS
           </A>";
  blend = "<A HREF='blend.htm'> ALL-BLEND
           </A>";
  sm    = "<A HREF='sm.htm'> ALL-SM
           </A>";
  sus   = "<A HREF='sus.htm'> ALL-SUS
           </A>";
  output;

  %do m=1 %to &mcount;
     state = "%scan(&mstates,&m,#)";
     over  = "<A HREF='over" || trim(state)
               || ".htm'> "  || trim(state)
               || "-OVER </A>";
     cnc   = "<A HREF='cnc"  || trim(state)
               || ".htm'> "  || trim(state)
```

```
               || "-CNC  </A>";
     tier  = "<A HREF='tier" || trim(state)
               || ".htm'> "  || trim(state)
               || "-TIER </A>";
     bus   = "<A HREF='bus"  || trim(state)
               || ".htm'> "  || trim(state)
               || "-BUS </A>";
     blend = "<A HREF='blend"|| trim(state)
               || ".htm'> "  || trim(state)
               || "-BLEND </A>";
     sm    = "<A HREF='sm"   || trim(state)
               || ".htm'> "  || trim(state)
               || "-SM   </A>";
     sus   = "<A HREF='sus"  || trim(state)
               || ".htm'> "  || trim(state)
               || "-SUS </A>";
     output;
  %end;
run;
*---------------------------------------;
proc print data=toc split='\';
   id state;
run;
*---------------------------------------;
```

The process is to basically create a SAS data set and then PROC PRINT it, but let's go over some code anyway. First of all, we had to use a small PROC TEMPLATE to create a style to overcome a minor bug known to SI about headers in PRINT output via ODS HTML. Sometimes they just don't align properly. We wanted them all left-justified, and that is exactly what the **lhead** style does. It is created in the TEMPLATE code and is used in the ODS code. Simple. This is not the proper forum to go into PROC TEMPLATE, so suffice it to say that it is an extremely powerful new formatting tool in SAS. Spend some time with it.

Data set **toc** is created with the TABULATE "page" dimensions to serve as variables, labels to serve as column headers, and observation cell values to serve as the hot links in the display. The data set is run through a PROC PRINT with ODS directing the output to the file "_toc.htm". The first observation output is for "ALL STATES". Next, an observation is created for each state present in the source data by looping through the state macro variables created earlier. Each cell value is built as an HTML hot link. When it is displayed and clicked on, the browser links to the called page. As mentioned earlier, this PROC PRINT uses state as an ID variable. The default display template used for PROC PRINT automatically highlights ID variables. Since there is no Obs column present when ID variables are used, the display, with row and column headers highlighted, further enhances the tabular Table of Contents display.

That's basically it.

Conclusion

Although ODS is a monumental undertaking resulting in a new world of display options and power, as is usually the case, there is always room for improvement. In particular, Table of Contents features of the new system leave something to be desired when the created output is more than a simple display. The technique of creating HTML tag-enriched data values in separate stand-alone Table of Contents data sets, coupled with similarly enriched titles, provides a method of developing whole systems of sophisticated, fully navigable Tables of Contents to accompany and enhance the very powerful reporting display systems that can be built with ODS. The tools presented here are simple, and yet powerful. You owe it to yourself, your clients, your organizations, whatever, to spend some time using them to developing your own powerful and productive systems.

Author Contact Information

The author of this paper can be contacted as follows:

Ray Pass
Ray Pass Consulting
5 Sinclair Place
Hartsdale, NY 10530

Voice: (914) 693-5553
Fax: (914) 206-3780
e-mail: raypass@att.net