Paper 252-2011

Let's Give 'Em Something to TOC about: Transforming the Table of Contents of Your PDF File

Bari Lawhorn, SAS Institute Inc., Cary, NC

ABSTRACT

In PDF files, the table of contents provides a map that helps your audience to navigate the document easily. However, the default table of contents that is generated by the SAS® Output Delivery System (ODS) destination, while informative, is fairly utilitarian. Your procedures and DATA steps generate tables and graphs that have meaning to you and your audience. Likewise, the table of contents should also be as meaningful as possible by clarifying the contents of your PDF. This paper explains and demonstrates step by step how to use the following statement, options, and procedures to customize your table of contents:

- the ODS PROCLABEL statement
- the CONTENTS= and the DESCRIPTION= options
- the DOCUMENT destination and procedure
- the TEMPLATE procedure

These tools provide you with the flexibility and the power to customize your table of contents so that you really leave your audience with something to TOC about!

INTRODUCTION

The ODS PDF destination was introduced in SAS® 8.2 to provide SAS programmers with a convenient way to obtain printer-friendly results that are also easy to access on the Web. A PDF file that is generated by SAS automatically creates a nifty table of contents (TOC) that enables you to click and link directly to your point of interest. For every table or graph that was generated by a SAS procedure or a DATA step, the TOC contains two or more nodes. The default TOC, however, is not always informative. Consider the following nodes in a default TOC: **`The PRINT Procedure`** and **`The UNIVARIATE Procedure`**. These entries are not helpful because they do not provide any information about the data tables or graphs that are included in the PDF files that you create with the ODS PDF destination. Even an experienced SAS programmer wants to know what, specifically, the TOC entries have to offer.

This paper helps you create descriptive TOC entries for your tables and graphs so that readers of your PDF files can access information quickly and easily. The paper discusses statements and options that you can include to make seamless changes to your TOC entries. In addition, tips and techniques are provided for more complicated processing to transform and expound on the information that is delivered by the TOC in your PDF files.

SHHH, NO TOC'ING!

There are times when you do not want any table of contents for your PDF file. If all you want to do is remove the TOC from a PDF file that is created with ODS, you can use the NOTOC option in the ODS PDF statement, as shown here:

```
ods pdf file="file.pdf" notoc;
```

If removing the TOC were the only task that you wanted to do, this discussion could stop here. However, most people want some sort of navigation tool for their PDF file, so the following sections explore possible options for customizing the TOC.

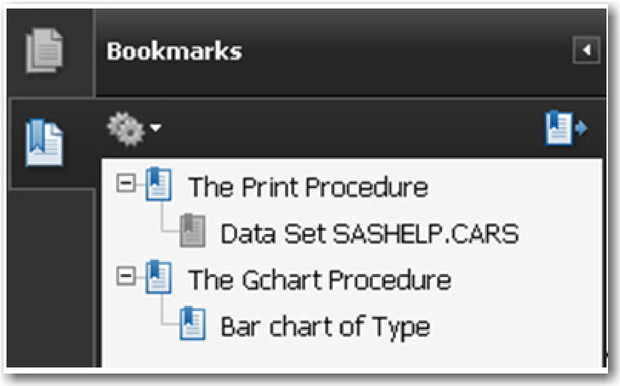
THE FIRST-LEVEL NODE: CHANGE IT

This first code sample creates a default TOC that includes node text that is generated by two simple procedures (the PRINT and GCHART procedures).

```
ods pdf file="display1.pdf";
proc print noobs data=sashelp.cars;
run;
```

```
proc gchart data=sashelp.cars;
   vbar type / sumvar=horsepower;
run;
quit;

ods pdf close;
```

Each procedure in this example creates at least two nodes in the TOC. As shown in Display 1, the first-level node (*parent* node) for each entry names the procedure that generates the corresponding output in the body of the file. The second-level node (*child* node) for each entry attempts to describe the output found in the body of the PDF file.



The Print Procedure
Data Set SASHELP.CARS
The Gchart Procedure
Bar chart of Type

Display 1. A Default Table of Contents That Is Created with Two SAS Procedures

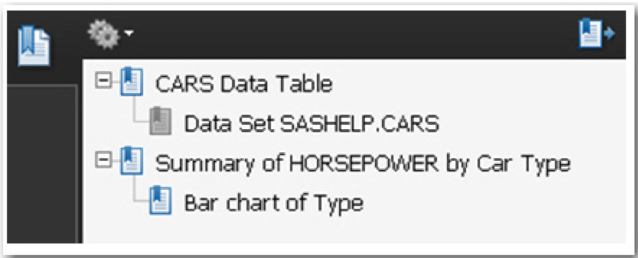
While the text strings **`The PRINT Procedure`** and **`The GCHART Procedure`** mean something to a SAS programmer, the PDF file is most likely intended to be shared with a broader and more general audience. Their concern is not so much which SAS procedure created the table or graph in the PDF file. Most likely, what the readers want is a short description of the table or graph that is in the body of the PDF file.

Fortunately, SAS provides a way to clarify the contents of this file. You can control the first node in the TOC with the ODS PROCLABEL statement. The syntax for this statement can be written in two ways:

ODS PROCLABEL '*string*';
ODS PROCLABEL='*string*';

The ODS PROCLABEL statement remains in effect for one step only. It resets at a step boundary (for example, a procedure statement). The following code sample uses the ODS PROCLABEL statement to change the text of the first-level nodes (Display 2):

```
ods pdf file="display2.pdf";

ods proclabel="CARS Data Table";
proc print noobs data=sashelp.cars;
   var type horsepower msrp;
run;

ods proclabel="Summary of HORSEPOWER by Car Type";
proc gchart data=sashelp.cars;
   vbar type / sumvar=horsepower;
run;
quit;

ods pdf close;
```


CARS Data Table
Data Set SASHELP.CARS
Summary of HORSEPOWER by Car Type
Bar chart of Type

Display 2. Using the ODS PROCLABEL Statement to Change the First-Level Nodes

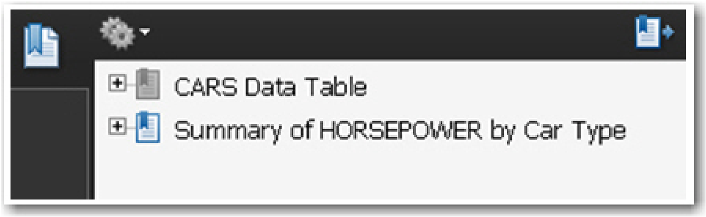
THE SECOND-LEVEL NODE: HIDE IT, CHANGE IT, OR REMOVE IT

While the first-level nodes now have added value as a result of using the ODS PROCLABEL text, the second-level node still does not provide much useful information about the PROC PRINT and PROC GCHART output. Options are available for use in both procedures to modify the text in the second node. In addition, the PDF destination provides syntax that enables you to hide and remove the second-level through *n*-level nodes. The following sections illustrate how to use these options.

HIDING THE SECOND-LEVEL THROUGH *N*-LEVEL NODES

Suppose you want to display **only** the first-level nodes in your TOC. You can hide the other levels by using the PDFTOC= option in the ODS PDF statement (Display 3), as in the following ODF PDF statement.

```
ods pdf file="display3.pdf" pdftoc=1;
```



Display 3. Using the Option PDFTOC=1 to Display Only First-Level Nodes

The PDFTOC= option does not remove or change the nodes. It simply controls which levels are expanded when the PDF file opens. The option PDFTOC=*n* controls the levels of expansion for the TOC in a PDF file. In the syntax, *n* specifies the expansion level that you want to show. In the Display 3 example, the TOC shows only the first-level nodes (PDFTOC=1). To show all nodes, use the default value, PDFTOC=0.

CHANGING THE SECOND-LEVEL THROUGH *N*-LEVEL NODES

To change the text of the second-level through *n*-level nodes, you can use other options that are specific to certain procedures. For example, the CONTENTS= option is valid in the PROC PRINT statement as well as in the following statements:

- the TABLES statement in the FREQ procedure. The CONTENTS= option is valid in the TABLES statement only when you request a two-way table (for example, X*Y).
- the PROC REPORT statement.
- the BREAK statement in the REPORT procedure. You can use the CONTENTS= option in the BREAK statement when you specify the PAGE option.
- the DEFINE statement in PROC REPORT. You can use the CONTENTS= option when the PAGE option is specified.
- The RBREAK BEFORE statement in PROC REPORT. You can use the CONTENTS= option in this statement when you specify the SUMMARIZE and PAGE options.
- the PROC TABULATE statement.
- the TABLE statement in the TABULATE procedure.

Traditional SAS/GRAPH® procedures (for example, the GCHART, GPLOT, and GMAP procedures) also enable you to change the text of the second-level node. For example, the following GCHART procedure uses the DESCRIPTION= option in the VBAR statement to change the text description of the second-level node (Display 4):

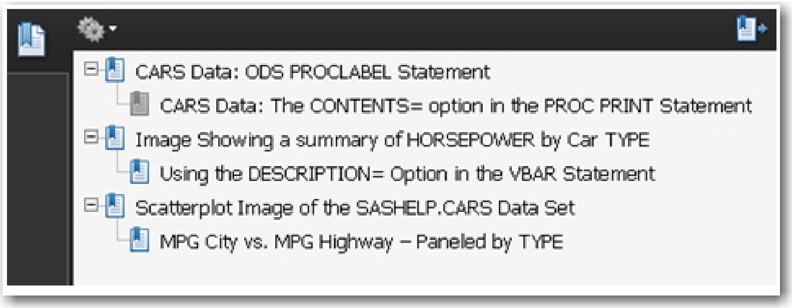
```
proc gchart data=sashelp.cars;
   vbar type / sumvar=horsepower
               description="Using the DESCRIPTION= Option in the VBAR
                            Statement";
run;
quit;
```

With the new statistical graphics procedures, use the DESCRIPTION= option in the PROC statement, as shown in this example:

```
proc sgpanel data=sashelp.cars
             description="Using the DESCRIPTION= Option in the PROC SGPANEL
                          Statement";
   ...more statements...
```

Combining these options with your knowledge of the ODS PROCLABEL statement, you can highly customize your TOC. The following code generates a TOC with modified second-level nodes (Display 4):

```
ods pdf file="display4.pdf" ;

ods proclabel="CARS Data: ODS PROCLABEL Statement";
proc print noobs data=sashelp.cars
                 contents="CARS Data: The CONTENTS= option in the PROC PRINT
                           Statement";
   var type horsepower msrp;
run;

ods proclabel="Image Showing a summary of HORSEPOWER by Car TYPE";
proc gchart data=sashelp.cars;
   vbar type / sumvar=horsepower
               description=Using the "DESCRIPTION= Option in the VBAR
                                      Statement";
run;
quit;

ods proclabel="Scatterplot Image of the SASHELP.CARS Data Set";
proc sgpanel data=sashelp.cars
             description="MPG City vs. MPG Highway – Paneled by TYPE";
   panelby type / rows=2 columns=3;
   scatter X=Mpg_City Y=Mpg_Highway;
run;

ods pdf close;
```


CARS Data: ODS PROCLABEL Statement
CARS Data: The CONTENTS= option in the PROC PRINT Statement
Image Showing a summary of HORSEPOWER by Car TYPE
Using the DESCRIPTION= Option in the VBAR Statement
Scatterplot Image of the SASHELP.CARS Data Set
MPG City vs. MPG Highway – Paneled by TYPE

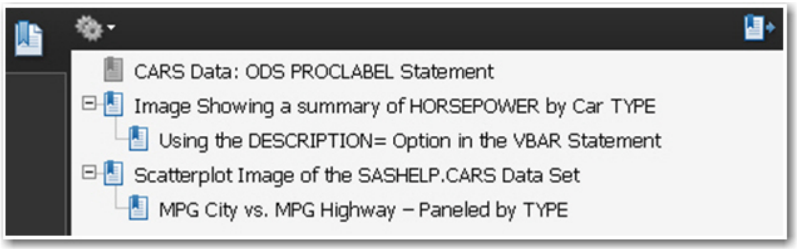
Display 4. Using the CONTENTS= and DESCRIPTION= Options to Modify the Second-Level Nodes

REMOVING THE SECOND-LEVEL THROUGH *N*-LEVEL NODES

The CONTENTS= option is especially helpful because you can also use it to completely remove a node. For example, suppose you include the following PRINT procedure in the previous code example:

```
proc print noobs data=sashelp.cars contents="";
```

Specifying a null value for CONTENTS=”” in that same code example generates output that no longer contains the second-level node for the first item in the TOC (Display 5). **Note**: A null value means that there is no space between the quotation marks.


CARS Data: ODS PROCLABEL Statement
Image Showing a summary of HORSEPOWER by Car TYPE
Using the DESCRIPTION= Option in the VBAR Statement
Scatterplot Image of the SASHELP.CARS Data Set
MPG City vs. MPG Highway – Paneled by TYPE

Display 5. Removing a Second-Level Node by Specifying a Null Value for the CONTENTS= Option

This raises a question: If you can eliminate this node by using CONTENTS="", can you also eliminate it by using the DESCRIPTION="" option?

At this time, SAS/GRAPH procedures do not support the elimination of the second node directly. You can use the PROC GREPLAY procedure to manipulate these nodes for traditional SAS/GRAPH procedures such as PROC GMAP, PROC GCHART, and PROC GPLOT.

Note: A later section, "Absolute Control over the TOC: ODS and the DOCUMENT Procedure" shows you another way to manipulate the nodes. That section describes PROC DOCUMENT, which also encompasses the following new SAS/GRAPH procedures: PROC SGPANEL, PROC SGPLOT, PROC SGRENDER, and PROC SGSCATTER.

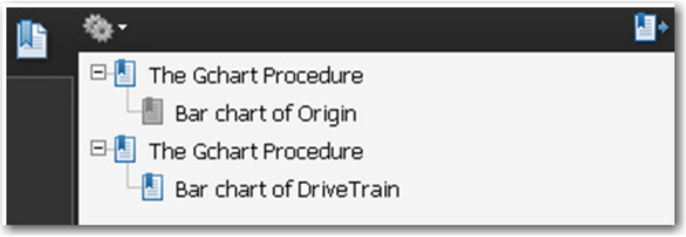
In the next example, PROC GREPLAY serves two purposes; it simplifies and combines graphs on a single page using the H2 graph template. The default TOC that is created by PROC GREPLAY displays only two nodes in the final PDF.

Using PROC GREPLAY to manipulate the nodes is a two-part process. First, use two PROC GCHART steps with the NODISPLAY graphics option to save your graphs to the WORK.GSEG catalog. Then, PROC GREPLAY creates the PDF file with a simplified TOC. :

The following code handles the first part of the process (saving your graphs to the WORK.GSEG catalog):

```
goptions reset=all nodisplay;

proc gchart data=sashelp.cars;
   vbar origin / sumvar=mpg_city;
run;
quit;

proc gchart data=sashelp.cars;
   vbar drivetrain / sumvar=mpg_city;
run;
quit;
```

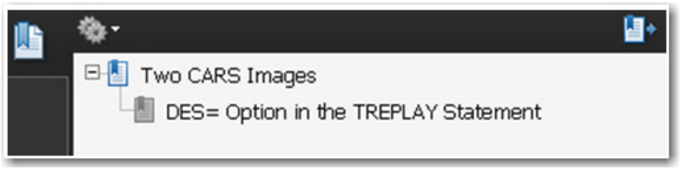
If you use ODS PDF statements around the previous code, along with the DISPLAY option in the GOPTIONS statement, then the TOC that is generated appears as follows:


The Gchart Procedure
Bar chart of Origin
The Gchart Procedure
Bar chart of DriveTrain

Display 6. The Default TOC That Is Created by Using Two GCHART Procedures

In the second part of the process, PROC GREPLAY combines graphs on a single page (using the H2 graph template) and creates a PDF file that contains only two nodes (a first-level node and a second-level node, shown in Display 7). This code also uses the previously discussed PROCLABEL statement and the DES= option.

```
ods pdf file="display7.pdf";

ods proclabel="Two CARS Images";
proc greplay tc=sashelp.templt nofs igout=work.gseg template=h2;
   treplay 1:gchart 2:gchart1
           des="DES= Option in the TREPLAY Statement";
run;
quit;

ods pdf close;
```


Two CARS Images
DES= Option in the TREPLAY Statement

Display 7. Using PROC GREPLAY to Combine the Output and to Display Only One Set of Nodes

WHAT IF I DON'T USE ANY OF THE PROCEDURES YOU'VE TOC'D ABOUT?

If the procedures that you use work with table templates, you have control over some TOC entry text. You can manipulate the contents using the TEMPLATE procedure. If you do not know whether your procedure uses a table template or if you are not sure of the name of your table template, submit the following statement with your code:

```
ods trace on;
```

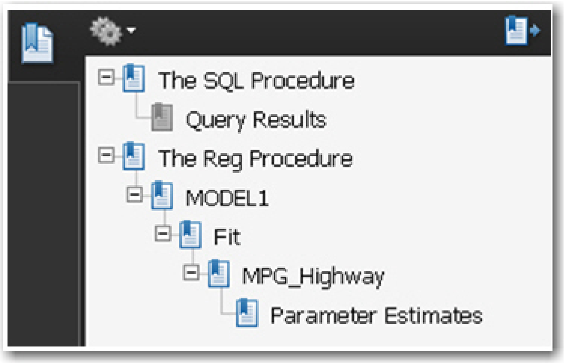
Running your code with this statement generates trace information in your log. From this information, you can determine whether your procedure uses a table template and what the name of the table is.

The next code sample demonstrates the SQL and REG procedures, the latter of which generates numerous tables. For demonstration purposes, the table that is used in the code that follows below is the ParameterEstimates table. This table is listed in the EDIT statement in PROC TEMPLATE and in the ODS SELECT statement. When you run the ODS TRACE ON statement with the code sample, the SAS log includes the following information:

```
Output Added:
-------------
Name:       SQL_Results
Label:      Query Results
Template:   Base.SQL
Path:       SQL.SQL_Results
-------------
Output Added:
-------------
Name:       ParameterEstimates
Label:      Second node:Contents_label for REG
Template:   Stat.REG.ParameterEstimates
Path:       Reg.MODEL1.Fit.MPG_Highway.ParameterEstimates
-------------
```

The following code creates a TOC (Display 8) from PROC SQL and the PROC REG ParameterEstimates table.

```
ods pdf file="display8.pdf";
proc sql;
   select * from sashelp.cars(obs=3);
run;
quit;

ods select parameterestimates;

proc reg data=sashelp.cars;
    model mpg_highway=mpg_city;
run;
quit;

ods pdf close;
```



```
The SQL Procedure
   Query Results
The Reg Procedure
   MODEL1
      Fit
         MPG_Highway
            Parameter Estimates
```

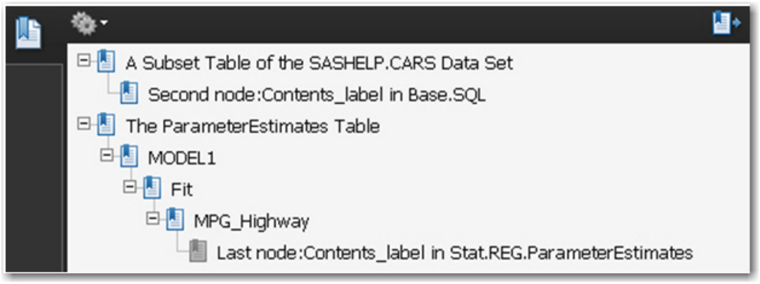
Display 8. Default TOC That Is Created by PROC SQL and the MODEL Statement in PROC REG

The text of this default TOC might not be as descriptive as you would like it to be. The next example illustrates how you can make some of the nodes more descriptive.

As you can see from the trace output shown earlier in this section, the table template that is used by PROC REG is called Stat.REG.ParameterEstimates. The table that is used by PROC SQL is called BASE.SQL. These names indicate that these tables are stored in SAS item stores that reside in, respectively, the STAT and BASE template paths.

In the following code sample (and resulting output, Display 9), PROC TEMPLATE modifies these table templates using the EDIT statement, and the procedure defines a macro variable with the MVAR statement. SAS uses this macro variable to populate the CONTENTS_LABEL table attribute.

```
proc template;
   edit base.sql;
      mvar sqlcl;
      contents_label=SQLcl;
   end;
   edit Stat.REG.ParameterEstimates;
      mvar regcl;
      contents_label=REGcl;
   end;
run;

%let SQLcl=Second node:Contents_label in Base.SQL;
%let REGcl=Last node:Contents_label in Stat.REG.ParameterEstimates;
ods trace on;
ods pdf file="display9.pdf";
ods proclabel="A Subset Table of the SASHELP.CARS Data Set";

proc sql;
   select * from sashelp.cars(obs=3);
run;
quit;

ods proclabel="The ParameterEstimates Table ";
ods select parameterestimates;

proc reg data=sashelp.cars;
   model mpg_highway=mpg_city;
run;
quit;

ods pdf close;
```


A Subset Table of the SASHELP.CARS Data Set
Second node:Contents_label in Base.SQL
The ParameterEstimates Table
MODEL1
Fit
MPG_Highway
Last node:Contents_label in Stat.REG.ParameterEstimates

Display 9. Using PROC TEMPLATE and Table Templates to Control Nodes

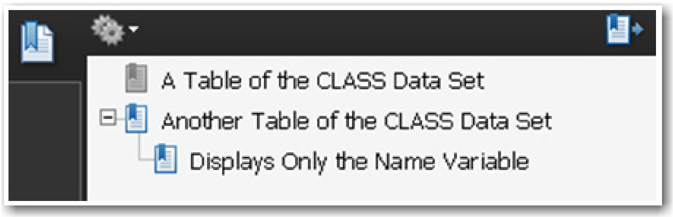
The new table templates are stored, by default, in the SASUSER.TEMPLAT item store. This table-template definition does not have any effect on the body of the PDF file. If the macro variables SQLCL or REGCL do not exist or are null in subsequent SAS sessions, the TOC entries revert to their default text.

Because the PROC REG step creates many nodes in the TOC, you might find this method a bit overwhelming.. Not to worry! Stay tuned for a way to eliminate some or all of these nodes. (See the section "Absolute Control over the TOC: ODS and the DOCUMENT Procedure.")

WHAT IF I USE DATA STEPS?

What about those faithful DATA step coders? Again, no need to worry. You can still change or eliminate the second node in the TOC by using the full realm of styles and table control options that the SAS Output Delivery System offers. In this case, you can use a FILE statement that contains the reserved fileref PRINT along with the ODS= option and the OBJECTLABEL= suboption. This suboption enables you to change or eliminate the second node in the TOC. (As always, the ODS PROCLABEL statement controls the first node.) When you use a FILE statement with the PRINT fileref, ODS is not in control of the nodes. Therefore, you can use a TITLE statement to change the second node, as shown in this code sample and in the output that is shown in Display 10:

```
ods pdf file="display10.pdf" ;
ods proclabel="A Table of the CLASS Data Set";

data _null_;
   file print ods=(objectlabel="");
   set sashelp.class;
   put _ods_;
run;

ods proclabel="Another Table of the CLASS Data Set";

title "Displays Only the Name Variable";

data _null_;
   file print;
   set sashelp.class;
   put name;
run;

ods pdf close;
```


A Table of the CLASS Data Set
Another Table of the CLASS Data Set
Displays Only the Name Variable

Display 10. Using DATA Steps to Control the Second-Level Node

ABSOLUTE CONTROL OVER THE TOC: ODS AND THE DOCUMENT PROCEDURE

Introduced in SAS® 9.0, the ODS DOCUMENT destination enables you to store output in an item store. PROC DOCUMENT provides the flexibility to replay that item store as output to any destination at a later date. PROC DOCUMENT also gives you absolute control over the TOC.

Note: Throughout the rest of this paper, the term *DOCUMENT facility* refers to the use of the ODS DOCUMENT destination in conjunction with PROC DOCUMENT.

Regardless of the procedure that you use, PROC DOCUMENT can help you manipulate and control your TOC. This section illustrates that ability with several examples.

EXAMPLE 1: COLLAPSING TWO LEVELS OF NODES INTO SINGLE NODES

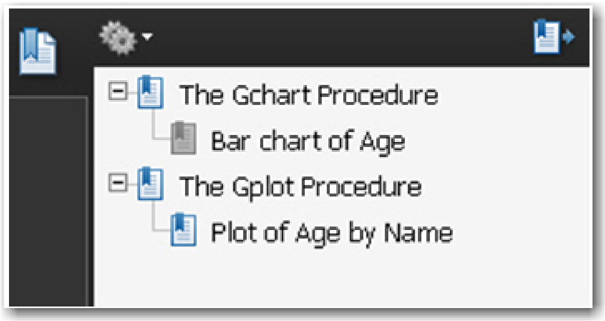
Part 1: Creating an Item Store with the ODS DOCUMENT Destination

If you want your PDF to show only two nodes, you should store your procedure output in an item store with PROC DOCUMENT. In the following example, the item store is named TOCtest. This example creates a PDF file with the default text and nodes generated by PROC GCHART and PROC GPLOT (see Display 11).

```
ods listing close;
ods pdf file="before.pdf";
ods document name=TOCtest(write);

proc gchart data=sashelp.class;
   vbar age / sumvar=height;
run;
quit;
```

```
proc gplot data=sashelp.class;
   plot age*name;
run;
quit;

ods document close;
ods pdf close;
```

The PDF file does not need to be created in this instance, but it is shown in Display 11 to illustrate the changes that PROC DOCUMENT makes.


The Gchart Procedure
Bar chart of Age
The Gplot Procedure
Plot of Age by Name

Display 11. Default First-Level and Second-Level Nodes Created by the GCHART and GPLOT Procedures

In This Example:

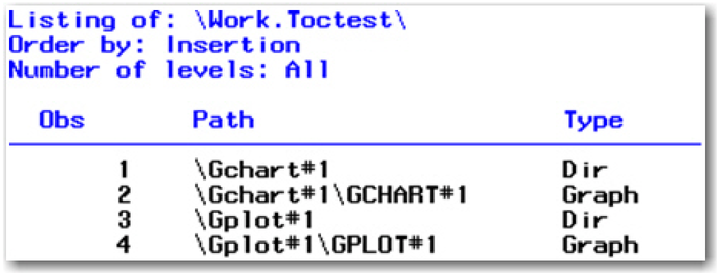
- Display 11 shows the default TOC that is created by two simple PROC GCHART and PROC GPLOT steps.
- The TOC contains the first-level and second-level nodes for each graph in the code sample.
- PROC DOCUMENT also stores the results of these graphs into a document item store called TOCtest.

Part 2: Using PROC DOCUMENT to Replay Only Single-Level Nodes

The following sample code uses PROC DOCUMENT to manipulate the TOCtest item store, producing a final PDF with only two nodes:

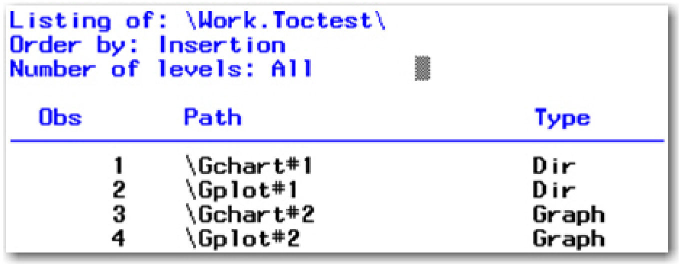
```
ods listing;
proc document name=TOCtest;
   list / levels=all; run;

      /* Move output from under the folders to the highest level. */
   move \Gchart#1\Gchart#1 to ^; run;
   move \Gplot#1\Gplot#1 to ^; run;
      /* Verify that the first-level nodes move out from */
      /* under the folders.                               */
   list / levels=all; run;

   setlabel \Work.test\Gchart#2  "First Node"; run;
   setlabel \Work.test\Gplot#2 "Second Node"; run;

       /* Close the listing output. */
   ods listing close;

      /* Replay the output (PROC GCHART and PROC GPLOT) to a PDF file. */
   ods pdf file="after.pdf" ;
      replay;
run;
ods pdf close;
quit;
```

PROC DOCUMENT generates the following listings:



```
Listing of: \Work.Toctest\
Order by: Insertion
Number of levels: All

   Obs          Path                          Type

     1          \Gchart#1                     Dir
     2          \Gchart#1\GCHART#1            Graph
     3          \Gplot#1                      Dir
     4          \Gplot#1\GPLOT#1              Graph
```

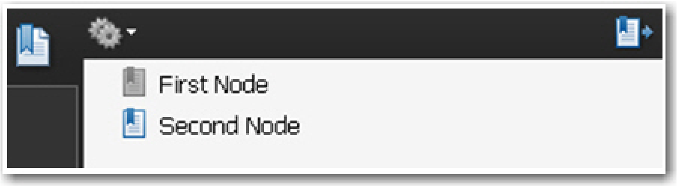
Output 1. PROC DOCUMENT Information about the Item Store TOCtest before Using the MOVE Statements



```
Listing of: \Work.Toctest\
Order by: Insertion
Number of levels: All

   Obs          Path                     Type

     1          \Gchart#1                Dir
     2          \Gplot#1                 Dir
     3          \Gchart#2                Graph
     4          \Gplot#2                 Graph
```

Output 2. PROC DOCUMENT Information about the Item Store TOCtest after Using the MOVE Statements

Then PROC DOCUMENT replays the output into a PDF file with a TOC that contains one node per graph:



Display 12. Replayed Output with Only One Node per Graph

In This Example:

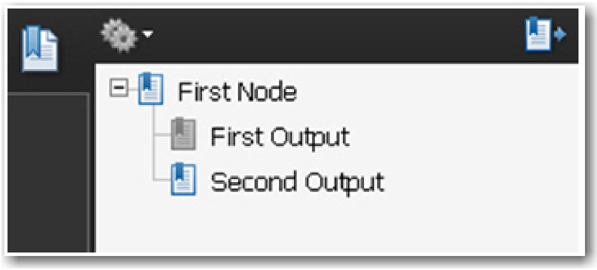
- The results of the LIST statements, shown previously in Output 1 and Output 2, appear in the SAS output window.
- The MOVE statements move the Gchart#1 and Gplot#1 nodes up in the TOC tree, so the respective entries of type GRAPH are on the first-level (or parent-level) node.
- The second LIST statement shows that the GRAPH entries are named Gchart#2 and Gplot#2, so these names are used in the SETLABEL statements in order to rename the nodes.
- The REPLAY statement replays the graph images from the item store to a PDF file (Display 12) called AFTER.PDF.

EXAMPLE 2: MAINTAINING MULTIPLE NODE LEVELS WITH THE COPY, DIR, AND MAKE STATEMENTS IN PROC DOCUMENT

Suppose that you have several tables or graphs on a page and you want to maintain a first-level node along with multiple sublevel nodes. You can do that by using the COPY statement (rather than the MOVE statement from the previous example), the MAKE statement, and the DIR statement. The following sample code uses the TOCtest item store that is created in "Part 1: Creating an Item Store with the ODS DOCUMENT Destination." However, in this code, PROC DOCUMENT creates output with a parent node and two child nodes (Display 13).

```
ods listing;

proc document name=TOCtest;
   list / levels=all;
   run;

   make \test;
   dir \test#1;
```

```
      setlabel \test#1 "First Node";
      setlabel \Gchart#1\Gchart#1 "First Output";
      setlabel \Gplot#1\Gplot#1 "Second Output";

      copy \Gchart#1\Gchart#1 to ^;
      copy \Gplot#1\Gplot#1  to ^;

      ods listing close;
      ods pdf file="after.pdf";
         replay \test#1;
         run;
      ods pdf close;
   quit;
```



Display 13. Replayed Output Showing One Node per Graph in Multiple Node Levels

In This Example:

- The MAKE statement generates a parent node.
- The DIR statement sets the current working directory to be the specified directory.
- The COPY statement copies the graph entries into the parent directory.

EXAMPLE 3: SIMPLIFYING MULTIPLE LEVELS OF NODES

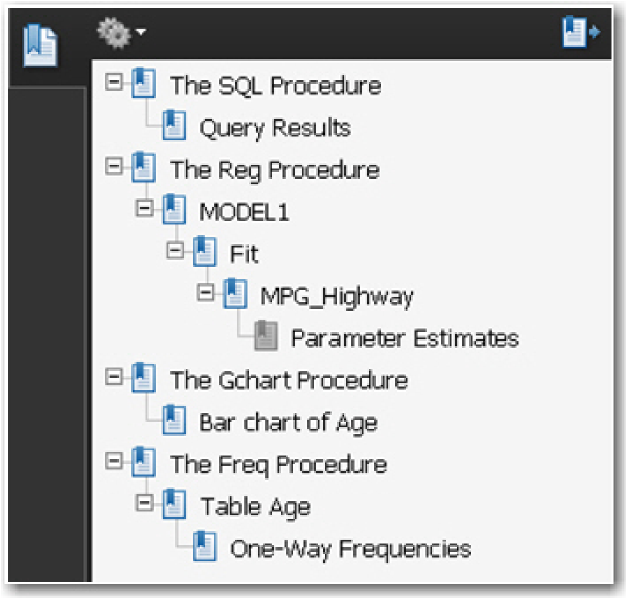
You can use PROC DOCUMENT to move the location of tables and graphs in the TOC tree. The following example, which is a two-part process, uses PROC DOCUMENT to manipulate the item store NESTED in order to simplify a multiple-level TOC.

Part 1: Creating a Default TOC with the SQL, REG, GCHART, and FREQ Procedures

The following code generates a TOC that has multiple levels of nodes (Display 14):

```
proc greplay nofs igout=work.gseg;
   delete _all_;
run;
quit;

ods listing close;

ods document name=nested;
ods pdf file="before.pdf";

proc sql;
   select * from sashelp.cars(obs=3);
run;
quit;

ods select parameterestimates;

proc reg data=sashelp.cars ;
   model mpg_highway=mpg_city;
run;
quit;
```

```
proc gchart data=sashelp.class;
   vbar age / sumvar=height;
run;
quit;

proc freq data=sashelp.class;
   tables age;
run;

ods pdf close;
ods document close;
```



```
The SQL Procedure
  Query Results
The Reg Procedure
  MODEL1
    Fit
      MPG_Highway
        Parameter Estimates
The Gchart Procedure
  Bar chart of Age
The Freq Procedure
  Table Age
    One-Way Frequencies
```

Display 14. Default TOC That Is Created by the SQL, REG, GCHART, and FREQ Procedures

In The First Part of This Example:

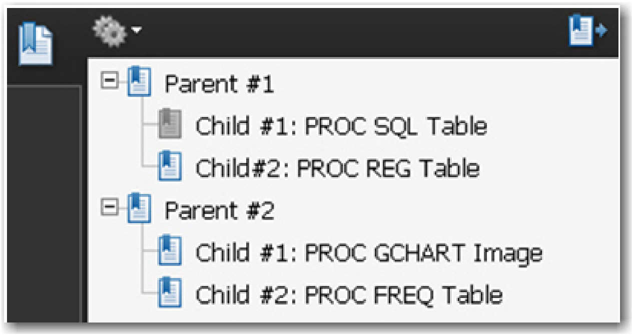
The initial GREPLAY step (in the first part of the code sample) is used for housekeeping purposes. That is, it deletes the previous GRSEG entries in the WORK directory so that graph entries start at NAME#1.

Part 2: Using PROC DOCUMENT to Collapse a Multiple-Level TOC

The second part of this example uses the MAKE, DIR, SETLABEL, COPY, and REPLAY statements. These statements maintain three tables and a graph in the body of the PDF file. In addition, they also simplify the TOC into one that contains only two parent nodes, each of which has two child nodes. This code generates a more readable TOC, as shown in Display 15.

```
proc document name=nested;
   make first;
   dir  \first#1;
      /* Label the highest (parent) node and the two subnodes (child)  */
      /* that are to be copied later by the COPY statements.            */
   setlabel \first#1 "Parent #1";
   setlabel \SQL#1\SQL_Results#1 "Child #1: PROC SQL Table";
   setlabel \Reg#1\MODEL1#1\Fit#1\MPG_Highway#1\ParameterEstimates#1 "Child#2:
   PROC REG Table";

      /* Copy two tables and their subnodes: the PROC REG ParameterEstimates */
      /* table and the PROC SQL table.                                       */
   copy \SQL#1\SQL_Results#1 to ^; run;
   copy \Reg#1\MODEL1#1\Fit#1\MPG_Highway#1\ParameterEstimates#1 to ^;

      /* Move up two levels. */
   dir ^^; run;
```

```
      /* Start a new directory */
   make second; run;
   dir \second#1; run;
   setlabel \second#1 'Parent #2'; run;
   setlabel \Gchart#1\Gchart#1 "Child #1: PROC GCHART Image";
   setlabel \Freq#1\Table1#1\OneWayFreqs#1 "Child #2: PROC FREQ Table";
      /* Copy a graph, a table, and their subnodes (created with  */
      /* PROC GCHART and PROC SQL).                               */
   copy \Gchart#1\Gchart#1 to ^;
   copy \Freq#1\Table1#1\OneWayFreqs#1 to ^; run;
   dir ^^;

      /* Replay the two directories first#1 and second#1. */
   ods pdf file="after.pdf";
      replay \first#1, \second#1;
   run;
   ods pdf close;
quit;
```



- Parent #1
  - Child #1: PROC SQL Table
  - Child#2: PROC REG Table
- Parent #2
  - Child #1: PROC GCHART Image
  - Child #2: PROC FREQ Table

Display 15. Using PROC DOCUMENT to Eliminate and Relabel the TOC Nodes

This new TOC is more readable and easier to navigate than the initial TOC.

In The Second Part of This Example:

- The MAKE statement creates a directory called **`first`**.
- The DIR statement is used three times. It sets paths to both **`first#1`** and **`second#2`**. The second DIR statement is used to move back up the TOC tree, so that subsequent COPY statements are child nodes under **`second#2`** instead of under **`first#1`**.
- The SETLABEL and REPLAY statements work the same way, as demonstrated in previous examples.

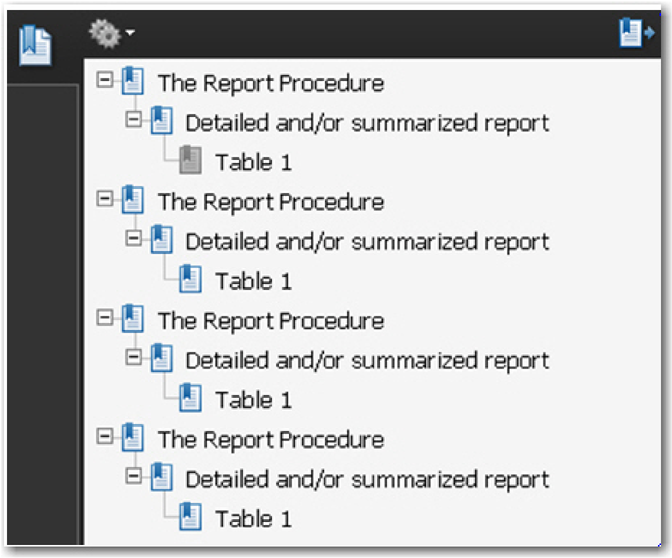
PROC REPORT: TOC'ING A DIFFERENT DIALECT

Sometimes you have to make a special effort in order for your PROC REPORT tables to turn out the way you want. The same is true for the TOC that is created by PROC REPORT. The TOC has all of the correct information in it, but sometimes that information is not what you want nor is it structured the way you want it to be.

The following example illustrates how you can customize a TOC that is created by multiple PROC REPORT steps. This example, which is broken into two parts, uses four PROC REPORT steps to create a final TOC that contains two parent nodes that both have two child nodes. The first part of the example uses the CONTENTS= option in a BREAK statement to replace the text **`Table 1`** in the TOC. The second part uses PROC DOCUMENT to collapse redundant parent nodes so that the final TOC has two parent nodes that both have two child nodes.

The following PROC REPORT steps generate a default TOC. However, the default TOC (Display 16) has many extraneous, redundant nodes.

```
ods pdf file="display16.pdf";

proc report nowd data=sashelp.class(obs=2); run;
proc report nowd data=sashelp.class(obs=4); run;
proc report nowd data=sashelp.class(obs=2); run;
proc report nowd data=sashelp.class(obs=4); run;

ods pdf close;
```



The Report Procedure
Detailed and/or summarized report
Table 1
The Report Procedure
Detailed and/or summarized report
Table 1
The Report Procedure
Detailed and/or summarized report
Table 1
The Report Procedure
Detailed and/or summarized report
Table 1

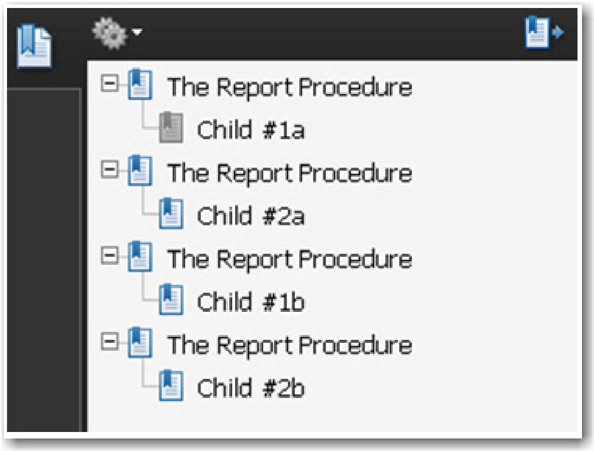
Display 16. The Default TOC That Is Created with Four PROC REPORT Statements

PART 1: MANIPULATING THE TOC BY USING THE CONTENTS= OPTION IN THE BREAK STATEMENT

This next example uses PROC REPORT and the CONTENTS= option to change the node text in the TOC. As discussed in "Changing the Second-Level through *N*-Level Nodes," the CONTENTS= option is available for use in numerous parts of the PROC REPORT syntax. In the following code sample, the option is used in the BREAK statement. This sample generates the TOC shown in Display 17.

```
data new;
   set sashelp.class;
   flag=1;
run;

ods pdf file="display17.pdf";
ods document name=REPORTTOC(write);

proc report nowd data=new(obs=2) contents='Child #1a';
   define flag / noprint order;
   break before flag / page contents='';
run;

proc report nowd data=new(obs=4) contents='Child #2a';
   define flag / noprint order;
   break before flag / page contents='';
run;

proc report nowd data=new(obs=2) contents='Child #1b';
   define flag / noprint order;
   break before flag / page contents='';
run;
```

```
proc report nowd data=new(obs=4) contents='Child #2b';
   define flag / noprint order;
   break before flag / page contents='';
run;

ods document close;
ods pdf close;
```



The Report Procedure
Child #1a
The Report Procedure
Child #2a
The Report Procedure
Child #1b
The Report Procedure
Child #2b

Display 17. Using PROC REPORT and the CONTENTS= Option to Adjust TOC Nodes

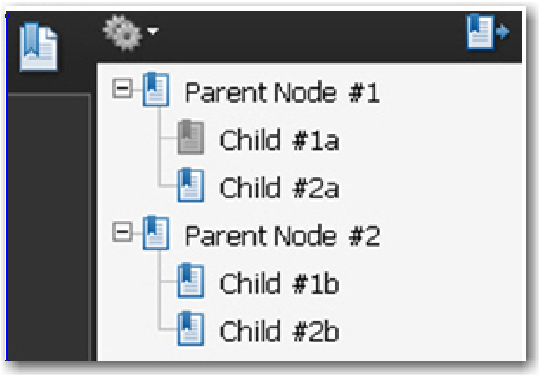
In This Example:

- In order for SAS to eliminate the **`Table 1`** nodes, a variable must exist that is the same for every observation. Such a variable does not exist in SASHELP.CLASS, so the previous sample code creates a variable called FLAG. SAS only uses this variable to manipulate the TOC; therefore, it should not appear in the body of the PDF file. To use the variable FLAG in a subsequent BREAK BEFORE statement, you must define the variable as GROUP or ORDER in the DEFINE statement. This code sample uses the ORDER option. The NOPRINT option is also included in the same DEFINE statement to suppress the printing of the variable in the table.
- The FLAG variable is placed in a BREAK BEFORE / PAGE statement so that the CONTENTS= option can be set to null, which removes the text **`Table 1`** from the body of the PDF file. This technique is described in the SAS Note 31278, "Table 1 node generated by PROC REPORT" (SAS Institute Inc. 2005).

PART 2: COLLAPSING REDUNDANT PARENT NODES BY USING PROC DOCUMENT

This second part of the example collapses the redundant parent nodes into two nodes and renames them. Each new parent node also receives two of the four child nodes. This code sample generates a TOC that eliminates the redundant parent nodes (Display 18).

```
proc document name=REPORTTOC(update);
   ods listing;
     list / levels=all; run;
   ods listing close;
   move Report#2\Report#1 to report#1;
   setlabel report#1 "Parent Node #1";
   delete Report#2;
   move Report#4\Report#1 to report#3;
   setlabel report#3 "Parent Node #2";
   delete Report#4;
run;
ods pdf file='display18.pdf';
   replay; run;
ods pdf close;
quit;
```



Display 18. Replayed TOC with Two Parent Nodes Displaying Two Child Nodes

In This Example:

- A LIST statement is used in PROC DOCUMENT so you can see the contents of the REPORTTOC item store.
- The MOVE statements move up two of the four tables in the TOC tree. These two tables, named REPORT#2 and REPORT#4, are subsequently deleted.
- The two remaining parent nodes (REPORT#1 and REPORT#3) are relabeled as **`Parent Node #1`** and **`Parent Node #2`** by the SETLABEL statements.

THE DOCUMENT FACILITY: KEEP TOC'ING!

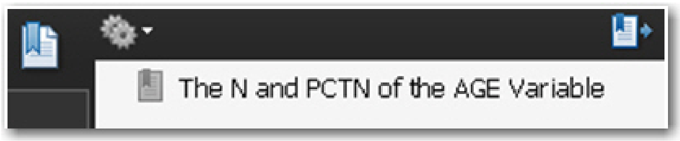
This paper has covered only a few of the capabilities inherent in the ODS DOCUMENT facility. A fuller exploration of the DOCUMENT facility is a topic for another paper (or two). For example, you can automate ODS and PROC DOCUMENT by using SAS macro logic, as described in "A SAS® Output Delivery System Menu for All Appetites and Applications" (Parker 2009).

If you would prefer a point-and-click approach to using the DOCUMENT facility, see "Have It Your Way: Rearrange and Replay Your Output with ODS DOCUMENT" (Zender 2009) for a discussion of the ODS DOCUMENT window, a GUI that is available to SAS programmers who use the SAS windowing environment.

TOGGLING THE GENERATION OF TOC NODES

Suppose you want to keep only certain DATA step or procedure nodes in your TOC. You can prevent generation of these nodes by using the NOBOOKMARKGEN option in the ODS PDF statement. This option is handy when you create your PDF file with any of the previously discussed techniques. The following code sample toggles off the PROC GCHART nodes and the PROC TABULATE nodes, as shown in Display 19.

```
title;
options orientation=portrait;
ods pdf file="display19.pdf" columns=2;

ods proclabel="The N and PCTN of the AGE Variable";
proc tabulate data=sashelp.class contents="";
   class age;
   table age, n pctn / contents='';
run;

ods pdf nobookmarkgen;

proc gchart data=sashelp.class;
   title;
   vbar age ;
run;
quit;

ods pdf close;
```


The N and PCTN of the AGE Variable

Display 19. Using the NOBOOKMARKGEN Option to Eliminate the PROC GCHART Nodes and the CONTENTS= Option to Eliminate the PROC TABULATE Nodes

In This Example:

- The NOBOOKMARKGEN option is used to toggle off the PROC GCHART nodes.
- The CONTENTS="" is used to remove the PROC TABULATE nodes.

Note: Although the nodes are removed**,** the body of the PDF file remains the same. That is, the tables and graphs remain in the body of the file.

If you want to display the nodes of subsequent steps in the same PDF file, you can use the BOOKMARKGEN option rather than NOBOOKMARKGEN, as shown here:

```
ods pdf bookmarkgen;
```

DON'T TOC UGLY

In addition to all of the tools and techniques that are discussed in this paper, there are some pitfalls that you should be aware of as you make changes to your TOC.

USING THE NOLABEL OPTION AFFECTS THE TOC

If you use the global option NOLABEL, it negates the following SAS elements:

- ODS PROCLABEL statement
- SETLABEL statement
- CONTENTS= option
- DESCRIPTION= option

If you find that your TOC does not contain any of the changes that you make, see if the NOLABEL option is in effect. To check, submit the following OPTIONS procedure:

```
proc options option=label;
run;
```

If the NOLABEL option is in effect, reactivate your code by submitting the following statement before the ODS PDF statement that contains the FILE= option:

```
options label;
```

USING A BLANK VALUE FOR THE CONTENTS= OPTION IS NOT THE SAME AS A NULL VALUE FOR THAT OPTION

If you use the CONTENTS= option in the REPORT, TABULATE, and FREQ procedures, you can remove the option only by using a **null value** for the option: CONTENTS="" (no space between the quotation marks). If you use a blank value, CONTENTS=" " (with a space between the quotation marks), then the node, or leaf, appears in the TOC, but it does not have a label.

USING BY STATEMENTS

If you use a BY statement in procedures or DATA steps, the TOC displays just what you see in your BY line in any output destination if the BYLINE option is in effect. That value appears in the output in the following form:

BY-variable-name=BY-variable-value

While you can use the LABEL statement to label the BY-variable name, there is no direct way to control the entire string currently. As described earlier in this paper, you can use the ODS DOCUMENT facility to control the TOC.

WHERE ELSE CAN WE TOC?

This paper focuses on and demonstrates how to change the TOC using the SAS 9.2 ODS PDF destination. The ODS HTML and RTF destinations also have the ability to generate a TOC.

The HTML destination creates a separate contents file when you include the CONTENTS= option in the ODS HTML statement, as shown in this example:

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

The RTF destination creates a TOC when you include the CONTENTS and TOC_DATA options in the ODS RTF statement that contains the FILE= option, as shown here:

```
ods rtf file="file.rtf" contents toc_data;
```

You can use most of the techniques that are discussed in this paper to adjust the TOCs in your RTF and HTML files. The PDDFTOC=, NOTOC, and BOOKMARKGEN options are exceptions. These options are exclusive to the ODS PRINTER or the ODS PDF statements.

CONCLUSION

The PDF destination’s TOC is just like your code: flexible and evolving. This paper describes ways to improve your TOC’ing skills by using the following tools:

- the ODS PROCLABEL statement
- the CONTENTS= and the DESCRIPTION= options
- the DOCUMENT destination and procedure
- the TEMPLATE procedure

These tools provide you with the flexibility and the power to customize your table of contents so that you really leave your audience with something to TOC about!

ACKNOWLEDGMENTS

The author is immensely grateful to the following coworkers who contributed to and edited this paper: Susan Berry, Allison Booth, Jane Eslinger, Tom Hahl, Scott Huntley, David Kelley, Kathryn McLawhorn, and Martin Mincey.

REFERENCES

Parker, Chevell. 2009. "A SAS® Output Delivery System Menu for All Appetites and Applications." *Proceedings of the SAS Global Forum 2009 Conference*. Cary, NC: SAS Institute Inc. Available at support.sas.com/resources/papers/proceedings09/016-2009.pdf.

Zender, Cynthia L. 2009. "Have It Your Way: Rearrange and Replay Your Output with ODS DOCUMENT." *Proceedings of the SAS Global Forum 2009 Conference.* Cary, NC: SAS Institute Inc. Available at support.sas.com/resources/papers/proceedings09/318-2009.pdf.

RECOMMENDED READING

SAS Institute Inc. 2009. *SAS® 9.2 Output Delivery System: User's Guide*. Cary, NC: SAS Institute Inc. Available at support.sas.com/documentation/cdl/en/odsug/61723/PDF/default/odsug.pdf.

CONTACT INFORMATION

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

Bari Lawhorn
SAS Institute Inc.
SAS Campus Drive:
Cary, NC 27513
E-mail: support.sas.com
Web: support.sas.com