PharmaSUG 2011 - Paper SAS-AD02

# Beyond the Basics: Advanced REPORT Procedure Tips and Tricks Updated for SAS® 9.2

Allison McMahill Booth, SAS Institute Inc., Cary, NC, USA

## ABSTRACT

This paper is an update to the 2007 SAS Global Forum Paper "Beyond the Basics: Advanced PROC REPORT Tips and Tricks"—a compilation of tips for producing desired PROC REPORT output with SAS 9.1.3. This paper offers tips and tricks for getting desired PROC REPORT output with the new functionality and enhanced features that are specific to using SAS® 9.2, such as:

- exploring the new SPANROWS option for repeating group and order variables on a continuing page
- using the ODS TAGSETS.RTF statement with PROC REPORT
- using the new border control for creating gridlines in all destinations
- changing nodes in an Output Delivery System (ODS) PDF bookmark by using the DOCUMENT procedure

Other new topics include:

- conditionally executing a LINE statement
- changing the style attributes of spanned headers
- adding text at the end of a report, even with a PAGE option
- taking control of across variables

Sample code and output are included with each topic. The examples presented in this paper assume an advanced knowledge of PROC REPORT.

## INTRODUCTION

SAS 9.2 has introduced some exciting changes for PROC REPORT. The paper explores the new items that are specific to PROC REPORT and SAS 9.2, as well as additional topics that span the different versions of SAS and different destinations. Specifically, this paper presents the following topics:

- overview of PROC REPORT for SAS 9.2
- new SAS 9.2 PROC REPORT topics for ODS
- new SAS 9.2 and SAS 9.1.3 topics for
  - ODS only
  - ODS and listing output
- what is new for PROC REPORT and SAS 9.3

So, let us get started….

## OVERVIEW OF PROC REPORT FOR SAS® 9.2

SAS 9.2 introduced several enhancements to the REPORT procedure. Here is a brief summary of these enhancements:

- The ability to repeat the value of a group and order variable across a page break was a top-ten item from the SASware Ballot.® The new SPANROWS option in the PROC REPORT statement provides this functionality in ODS destinations. This new option also places the group and order variables in a box that spans the entire grouping instead of on a per-cell basis. The SPANROWS option does not repeat the values for the group and order variables in traditional ODS RTF output.
- In ODS, the new Table of Contents now supports the CONTENTS= option in the PROC REPORT, BREAK, RBREAK, and DEFINE statements. The new border control style attributes for changing the border (also referred to as gridlines) color, width, and style are also supported. The CALL DEFINE statement has a new attribute option of STYLE/MERGE, which allows the styles to be concatenated rather than replaced—by default, styles were replaced in the previous versions of SAS.
- Some other enhancements include the addition of the MODE statistic and the ability to use the PROBT statistic as the alias for the PRT statistic. Other enhancements, which were suggested by SAS customers, include the ability to use the OUT= option in the PROC REPORT statement in conjunction with the BY

statement, the ability to change the page number between the BY groups with the new BYPAGENO=n option, and support for the ODS DOCUMENT and ODS OUTPUT destinations.

- Finally, a new enhancement for PROC REPORT that was added in the third maintenance release for SAS 9.2 is the ability to take advantage of new in-database technology. The REPORT procedure has been enhanced to run inside the Teradata Enterprise Data Warehouse (EDW), DB2 under UNIX, and Oracle. The result of using the in-database technology is less data movement and significant performance improvements.

Let us look further at some of these new enhancements.

## NEW SAS® 9.2 PROC REPORT TOPICS FOR ODS

### THE NEW CALL DEFINE STATEMENT ATTRIBUTE OF STYLE/MERGE

Before SAS 9.2, if multiple styles were applied to the same element through a CALL DEFINE statement, only the last CALL DEFINE would apply. The styles were always replaced. For the most part, this replacement did not cause a problem. However, when multiple CALL DEFINE statements were applying a style to the same element, then the coding became tricky.

In order to apply all of the styles appropriately, you would use only one CALL DEFINE for a particular element. However, this approach required more compute block code logic and more CALL DEFINE statements. The more CALL DEFINE statements that had style options for the same element, the more complicated the compute block code would become.

To illustrate the problem, look at the compute age block in the following code:

```
ods pdf;
proc report nowd data=sashelp.class;
   where age in(14,15);
   compute age;
      if age.sum=15 then call define(_row_,'style','style=[background=green]');
      if sex='M' then call define(_row_,'style','style=[foreground=blue]');
      if sex='F' then call define(_row_,'style/merge','style=[foreground=red]');
   endcomp;
run;
ods pdf close;
```

You want to change the font color of the table row to red when SEX is **F** and to blue when SEX is **M**. In addition, the table row needs to have a background color of green when AGE is **15**. In the preceding code, when SEX=M, the style attribute replaces the previous style with the style of FOREGROUND=BLUE for the row.

In SAS 9.2, STYLE is the same as STYLE/REPLACE. However, when SEX=F, the STYLE/MERGE option merges the FOREGROUND=RED style attribute with the existing styles. The resulting output is shown in Figure 1.

| Name | Sex | Age | Height | Weight |
|---|---|---|---|---|
| Alfred | M | 14 | 69 | 112.5 |
| Carol | F | 14 | 62.8 | 102.5 |
| Henry | M | 14 | 63.5 | 102.5 |
| Janet | F | 15 | 62.5 | 112.5 |
| Judy | F | 14 | 64.3 | 90 |
| Mary | F | 15 | 66.5 | 112 |
| Ronald | M | 15 | 67 | 133 |
| William | M | 15 | 66.5 | 112 |

**Figure 1. ODS PDF Output Showing the Results of Using and Not Using the STYLE/MERGE Option**

Based on the preceding code, the code that merges all of the styles, without using the STYE/MERGE option, looks like this:

```
compute age;
   if age.sum=15 and sex='M' then
      call define(_row_,'style','style=[background=green foreground=blue]');
   if age.sum=15 and sex='F' then
      call define(_row_,'style','style=[background=green and foreground=red]');
   if age.sum ne 15 and sex='F' then
      call define(_row_,'style','style=[foreground=red]');
   if age.sum ne 15 and sex='M' then
      call define(_row_,'style','style=[foreground=blue]');
endcomp;
```

## THE NEW SPANROWS OPTION

In the listing output, when group and order variables break across a page and continue on the next page, the values for these variables are repeated on the first row of the next page. With ODS output, the values for the group and order variables are not repeated on the next page when the same values continue to the next page. Before the new SPANROWS option became available with SAS 9.2, the only way to ensure that the values for the group and order variables repeat was to create the page break using the PAGE option in a BREAK statement. If the data set did not already have a variable to page by, then the data set would need to be preprocessed to create such a variable. This process can be very time consuming and tedious due to measurement issues.

The following code and output in Figure 2 illustrates a blank cell instead of the desired value of M under the Sex column on the next page:

```
data spanrows_example;
   set sashelp.class
       sashelp.class
       sashelp.class;
run;

ods pdf file='c:\no_spanrows.pdf';
proc report nowd data= spanrows_example;
   col sex age name height weight;
   define sex / order;
run;
ods pdf close;
```

| | | | | |
|---|---|---|---|---|
| | 15 | Mary | 66.5 | 112 |
| M | 14 | Alfred | 69 | 112.5 |
| | 14 | Henry | 63.5 | 102.5 |
| | 12 | James | 57.3 | 83 |
| | 13 | Jeffrey | 62.5 | 84 |
| | 12 | John | 59 | 99.5 |
| | 16 | Philip | 72 | 150 |
| | 12 | Robert | 64.8 | 128 |
| | 15 | Ronald | 67 | 133 |
| | 11 | Thomas | 57.5 | 85 |

| Sex | Age | Name | Height | Weight |
|---|---|---|---|---|
| | 15 | William | 66.5 | 112 |
| | 14 | Alfred | 69 | 112.5 |
| | 14 | Henry | 63.5 | 102.5 |
| | 12 | James | 57.3 | 83 |
| | 13 | Jeffrey | 62.5 | 84 |

**Figure 2. ODS PDF Output Showing that the ORDER Variable Does Not Repeat Across a Page**

The ability to repeat the same values for the group or order variable that continues on the next page was a ballot item on the 2007 SASware Ballot and was voted as a top 10 software enhancement by SAS customers. Subsequently, the new SPANROWS option was added to the PROC REPORT statement in SAS 9.2. Not only does the new SPANROWS option enable you to repeat the values for the group and order variables on the next page, it also creates a single cell for each level of group and order variables that spans all of the rows for that level. Adding this ability was the 24th most requested software enhancement for the 2007 SASware Ballot.

The following code and the output in Figure 3 show the result of using the new SPANROWS option in the PROC REPORT statement:

```
ods pdf file='c:\spanrows.pdf';
proc report nowd data= spanrows_example spanrows;
   col sex age name height weight;
   define sex / order;
run;
ods pdf close;
```

| | | | | |
|---|---|---|---|---|
| | 15 | Mary | 66.5 | 112 |
| M | 14 | Alfred | 69 | 112.5 |
| | 14 | Henry | 63.5 | 102.5 |
| | 12 | James | 57.3 | 83 |
| | 13 | Jeffrey | 62.5 | 84 |
| | 12 | John | 59 | 99.5 |
| | 16 | Philip | 72 | 150 |
| | 12 | Robert | 64.8 | 128 |
| | 15 | Ronald | 67 | 133 |
| | 11 | Thomas | 57.5 | 85 |

| Sex | Age | Name | Height | Weight |
|---|---|---|---|---|
| M | 15 | William | 66.5 | 112 |
| | 14 | Alfred | 69 | 112.5 |
| | 14 | Henry | 63.5 | 102.5 |
| | 12 | James | 57.3 | 83 |
| | 13 | Jeffrey | 62.5 | 84 |

**Figure 3. ODS PDF Output Showing the Repeat of an ORDER Variable Value and a Single Cell for the Value**

The SPANROWS option works fairly well. However, some measurement issues might interfere with the values for the group and order variables repeating properly. In this case, going back to the old method of creating a variable value to page by might be the only option. Also, ODS RTF output does not repeat the values for the group and order variables on the next page.

## THE ODS TAGSETS.RTF STATEMENT

As mentioned previously, the new SPANROWS option repeats the values for the group and order variables when there is a continuation of the same value on the next page for all ODS destinations, except when it is used with ODS RTF (also referred to as traditional RTF). In SAS 9.2, the new RTF TAGSET (referred to as TAGSETS.RTF) was introduced as another way to create RTF output. When the new SPANROWS option is used with TAGSETS.RTF, SPANROWS tries to repeat the values for the group and order variables. Using the UNIFORM option and PAGEPANELS=NONE with the ODS TAGSETS.RTF statement might help with some of the undesirable layout issues.

```
ods tagsets.rtf uniform pagepanels=none;
```

The UNIFORM option keeps the whole table in memory. When it is used, breaks between spanning lines do not lose context and are not measured again when the data does not suit the layout. PAGEPANELS=NONE uses a simple

model that does not try to adjust the number of observations that are formatted to the page. Looking ahead to SAS 9.3, when SPANROWS is requested in the PROC REPORT statement, the UNIFORM and PAGEPANELS=NONE options will be applied automatically.

What are the advantages of using TAGSETS.RTF over traditional ODS RTF with PROC REPORT? TAGSETS.RTF was written with the specification that titles and footnotes should appear in the body and not within the header and footer controls. In traditional RTF, when this is desired, the BODYTITLE option is used but it does not allow for repeats on page breaks with long tables.

Also, TAGSETS.RTF attempts to know when a page break occurs and traditional RTF allows Word to format long sections into a page. The following code illustrates this issue:

```
data example;
   do i= 1 to 32;
      output;
   end;
run;
ods rtf file='c:\myrtf.rtf';
ods tagsets.rtf file='c:\myrtf_tradtional2.rtf';
proc report nowd data=example
   style(report)=[rules=groups frame=below]
   style(header)=[background=_undef_];
   define i / display 'header';
   compute after _page_;
      line 'end of page';
   endcomp;
run;
ods rtf close;
ods tagsets.rtf close;
```

The code specifies that there is text and a frame at the end of the output data on each page. The ODS RTF output, shown in Figure 4, does not show the text from the COMPUTE AFTER _PAGE_ or the FRAME=BELOW style option at the end of the page. However, the output from TAGSETS.RTF that is shown in Figure 5 does show the text from the COMPUTE AFTER _PAGE_ and the FRAME=BELOW style option.

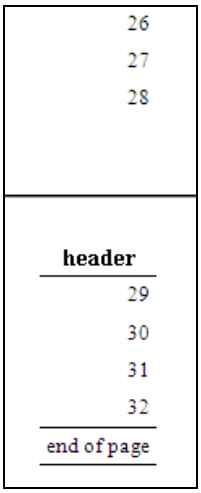


**Figure 4. ODS RTF Ouptut**

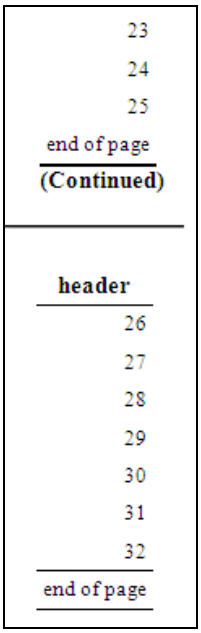


**Figure 5. ODS TAGSETS.RTF Output**

Notice that in Figure 5, there is the new additional text **(Continued)**. This is part of TAGSETS.RTF by default. The text of **(Continued)** can be removed by adding the following PROC TEMPLATE code to the preceding code and adding the style to the ODS TAGSETS.RTF statement. Here is the updated code:

```
proc template;
   define style styles.nocontinued;
      parent=styles.rtf ;
      style Continued from Continued /
            pretext=" "
            font=("Arial", 1pt) width=1%;
      style parskip from parskip /
            font=("Arial", 1pt);
   end;
run;
ods tagsets.rtf file='c:\myrtf_tradtional.rtf' style=styles.nocontinued;
```

The resulting output is shown in Figure 6.

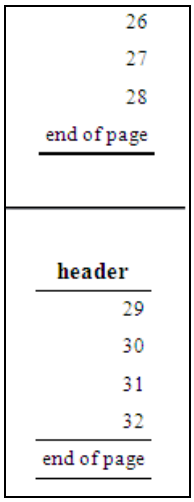


**Figure 6. The Text (Continued) Removed**

ODS TAGSETS.RTF has the ability to include user-written tagset functions. A customized tagset can be written that augments or adds functions and inherits from TAGSETS.RTF. Also, modifications to the TAGSETS.RTF can be delivered outside the scope of normal production releases. This means that some problems can be addressed more quickly instead of waiting for the next release of SAS.

## NEW BORDER CONTROL LANGUAGE FOR DRAWING GRIDLINES

In the 2007 SAS Global Forum paper "Beyond the Basics: Advanced PROC REPORT Tips and Tricks" (Booth 2007), the "Adding Lines" section shows how to add gridlines to different ODS destinations. ODS HTML has the most control of the gridlines, ODS RTF has control of the horizontal gridlines, and ODS PDF has very little control of the gridlines.

SAS 9.1.3 does not provide a lot of overall control for the gridlines. This all changed with SAS 9.2. The new border control functionality enables you to control the gridlines. You can control the border color, width, and style for each side of the border—top, left, bottom, and right.

A common question that SAS customers ask about gridlines in the header is related to how you open the header for the first column. In the following code, there is a spanned header in the COLUMN statement.

```
ods pdf;
proc report data=sashelp.class nowd
   style(header)=[background=white];
   col ( sex)
       ('spanned header' age weight height);
run;
ods pdf close;
```

The resulting output that is shown in Figure 7 shows the spanned header, but it also shows a blank cell over the first column.

| | spanned header | | |
|---|---|---|---|
| Sex | Age | Weight | Height |
| M | 14 | 112.5 | 69 |
| F | 13 | 84 | 56.5 |
| F | 13 | 98 | 65.3 |
| F | 14 | 102.5 | 62.8 |

**Figure 7. ODS PDF Output Showing a Blank Header Cell and a Spanned Header**

With the new border control functionality in SAS 9.2, you can control the header gridlines and open the header cell for the first column. If you look at the header in Figure 7, you can see that there is a border above Sex. You can add a style attribute to the DEFINE statement of the first column to change the header top border to white, which is the background color. This will remove the border between the first and second row. Now you have to add back the frame border at the top for the first row. You can do this by adding the style attribute as a spanned header to change the very top border to black. The following code is modified to show these changes and the resulting output is shown in Figure 8:

```
ods escapechar='^';
ods pdf;
proc report data=sashelp.class(obs=4) nowd
   style(header)=[background=white];
   col ('^S={bordertopcolor=black} ' sex)
       ('spanned header' age weight height);
   define sex / style(header)=[bordertopcolor=white];
run;
ods pdf close;
```

| | spanned header | | |
|---|---|---|---|
| Sex | Age | Weight | Height |
| M | 14 | 112.5 | 69 |
| F | 13 | 84 | 56.5 |
| F | 13 | 98 | 65.3 |
| F | 14 | 102.5 | 62.8 |

**Figure 8. Creating an Open Header Cell by Adding Border Control Functionality**

Another common question that SAS customers ask about gridlines is related to creating underlines in the header and a vertical bar in the detail data to separate the sections of the report. You can do both by using the SAS 9.2 border control functionality. The following code and the resulting output table in Figure 9 show the default output without the underlines and a vertical bar:

```
data class;
   set sashelp.class(obs=3);
   _empty='';
run;

ods escapechar='^';
ods tagsets.rtf;
proc report nowd data=class


   style(report)=[rules=none frame=void];
   col name ('this is spanned'
            ('A' age sex) _empty ('B' weight height));
   define _empty/' ' style(column)={cellwidth=1%};
   compute after _page_ / style={bordertopcolor=black bordertopwidth=2};
      line ' ';
   endcomp;
run;
ods tagsets.rtf close;
```

| | this is spanned | | | |
|---|---|---|---|---|
| | A | | B | |
| Name | Age | Sex | Weight | Height |
| Alfred | 14 | M | 112.5 | 69 |
| Alice | 13 | F | 84 | 56.5 |
| Barbara | 13 | F | 98 | 65.3 |

**Figure 9. ODS TAGSETS.RTF Default Output**

To produce the underlines, add the border control functionality to the COLUMN statement as spanned headers. You can create the vertical bar by adding the border control functionality to the DEFINE statement for the variable that is used to separate the detail section of the report. The following updated code and the resulting output in Figure 10 show you how to do this:

```
ods tagsets.rtf;
proc report nowd data=class
   style(report)=[rules=none frame=void]
   style(header)=[background=white borderbottomcolor=black borderbottomwidth=2
                  background=_undef_];
   col ('^S={borderbottomcolor=black borderbottomwidth=2} '
       ('^S={borderbottomcolor=white borderbottomwidth=2 } '
        '^S={borderbottomcolor=white borderbottomwidth=2 } ' name)
       ('^S={borderbottomcolor=black borderbottomwidth=2}this is spanned'
       ('^S={borderbottomcolor=black borderbottomwidth=2} A' age sex)
       ('^S={borderbottomcolor=white borderbottomwidth=2' _empty)
       ('^S={borderbottomcolor=black borderbottomwidth=2} B' weight height)));
   define _empty/' ' style(column)={cellwidth=1% borderrightcolor=red
                                    borderrightwidth=5};
   compute after _page_ / style={bordertopcolor=black bordertopwidth=2};
      line ' ';
   endcomp;
run;
ods tagsets.rtf close;
```

| | this is spanned | | | |
|---|---|---|---|---|
| | A | | B | |
| Name | Age | Sex | Weight | Height |
| Alfred | 14 | M | 112.5 | 69 |
| Alice | 13 | F | 84 | 56.5 |
| Barbara | 13 | F | 98 | 65.3 |

**Figure 10. ODS TAGSETS.RTF Output Using Border Control Functionality to Create Underlines and a Vertical Separator Bar**

## CHANGING NODES IN AN ODS PDF BOOKMARK USING THE DOCUMENT PROCEDURE

The ODS DOCUMENT statement and PROC DOCUMENT became production in SAS®9. However, it was not until SAS 9.2 that PROC REPORT could be used with ODS DOCUMENT and PROC DOCUMENT. Traditionally, PROC REPORT has the ability to store a report definition in a catalog entry with the OUTREPT= option in the PROC REPORT statement. This catalog entry, with the entry type of **REPT**, could then be recalled in a subsequent PROC REPORT submission using the REPORT= option. Typically, OUTREPT= and REPORT= were used for reports that remained the same in structure but not in the data.

The introduction of ODS provided the ability to route your output to an ODS destination. However, there was no way to avoid running the same PROC REPORT code with the same input data, until SAS 9.2. In SAS 9.2, you can run PROC REPORT with ODS DOCUMENT and PROC DOCUMENT. This new capability enables you to store the REPORT output and then replay it to different destinations as needed. This is especially important if the original program took a long time to run or the input data is difficult or impossible to reproduce.

Additionally, ODS DOCUMENT and PROC DOCUMENT allow you to modify the report structure. For example, running the following code in SAS 9.2 produces a Table of Contents as shown in Figure 11:

```
ods pdf file='mypdf3.pdf';
ods proclabel 'Section 1';
data example;
   set sashelp.class;
   holder=1;
run;

proc report nowd data=example(obs=2) contents='Group 1';
   define holder / noprint order;
   break before holder / page contents='';
run;
ods proclabel=' ';
proc report nowd data=example(obs=4) contents='Group 2';
   define holder / noprint order;
   break before holder / page contents='';
run;
ods proclabel=' ';
proc report nowd data=example(obs=4) contents='Group 3';
   define holder / noprint order;
   break before holder / page contents='';
run;
ods pdf close;
```

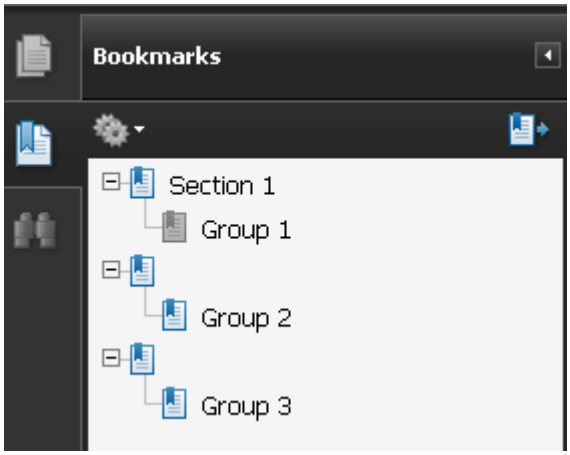


**Figure 11. ODS PDF Bookmarks before Using ODS DOCUMENT and PROC DOCUMENT**

Notice that none of the PROC REPORT bookmarks from the new CONTENTS= option are connected to the top node of Section 1. However, if you run the same code using ODS DOCUMENT, you now have the report structure in a document and can make the necessary changes within PROC DOCUMENT. Within the following PROC DOCUMENT code, the MOVE and DELETE options are used to rearrange the Table of Contents in the ODS PDF output, as shown in Figure 12:

```
ods proclabel 'Section 1';
data example;
   set sashelp.class;
   holder=1;
run;
ods document name=mydocrep(write);
proc report nowd data=example(obs=2) contents='Group 1';
   define holder / noprint order;
   break before holder / page contents='';
run;
ods proclabel=' ';
proc report nowd data=example(obs=4) contents='Group 2';
   define holder / noprint order;
   break before holder / page contents='';
run;
ods proclabel=' ';
proc report nowd data=example(obs=4) contents='Group 3';
   define holder / noprint order;
   break before holder / page contents='';
run;
ods document close;

proc document name=mydocrep ;
   move Report#2\Report#1 to report#1;
   delete Report#2;
   move Report#3\Report#1 to report#1;
   delete Report#3;
run;
ods pdf file='mypdf.pdf';
   replay;
run;
ods pdf close;
quit;
```

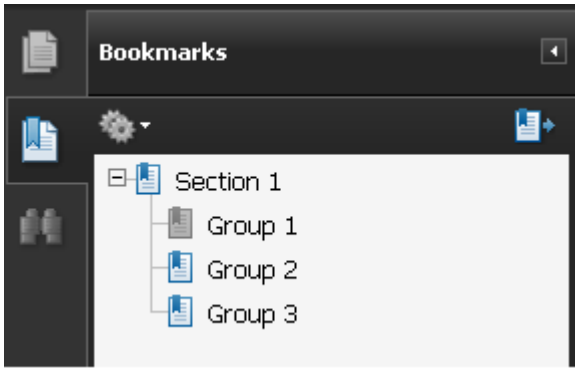


**Figure 12. ODS PDF Bookmarks after Using ODS DOCUMENT and PROC DOCUMENT**

## A NEW SAS® 9.2 AND SAS® 9.1.3 PROC REPORT TOPIC FOR ODS

### CHANGING THE STYLE ATTRIBUTES OF SPANNED HEADERS

Another question that customers frequently ask is related to changing the background attribute of a spanned header. You can change the style of a header globally by using the STYLE(HEADER) in the PROC REPORT statement or through a template code. You can also specify the STYLE(HEADER) in the DEFINE statement to make a change to a column header. Another option is to use in-line formatting, but the background might not apply to the whole cell in some destinations.

The following code produces the table in Figure 13.

```
ods html;
proc report nowd data=sashelp.class
   style(header)=[background=white];
   col age (('gender' sex),(weight height));
   define age / style(header)=[background=lightgreen];
   define sex / across style(header)=[background=yellow] ' ';
   define weight / style(header)=[background=orange];
   define height / style(header)=[background=tan];
run;
ods html close;
```

| | Gender | | | |
|---|---|---|---|---|
| | F | | M | |
| Age | Weight | Height | Weight | Height |
| 253 | 811 | 545.3 | 1089.5 | 639.1 |

**Figure 13. ODS HTML Output Showing a Spanned Header**

Notice that all of the header cells inherit the BACKGROUND=WHITE style from the STYLE(HEADER) option in the PROC REPORT statement. The other cells with values inherit the background from the STYLE(HEADER) option in the DEFINE statement. There is very little control over the header cells.

For illustration purposes, suppose you want the top row of the headers to have the background color of yellow. You want the second row cell above Age to inherit from the STYLE(HEADER) option in the PROC REPORT statement. And for fun, you would like to change the background of F to pink and M to light blue. Is this possible?

Yes, it is possible. You know that you can control a header in a DEFINE statement and you also know that a value for an across variable can span over multiple columns. If you combine these two, you can then make the changes that you need to most of the header cells. You can change the background of the across variable with a format.

In the following code, a DATA step is used to add a new variable flag, which in turn is going to be as an across variable in the PROC REPORT code:

```
data new;
   set sashelp.class;
   flag=' ';
run;
proc format;
   value $gender 'F'='pink' 'M'='lightblue';
ods html;
proc report nowd data=new
   style(header)=[background=white];
   col flag=flag2,age flag,(( sex),(weight height));
   define flag / across missing 'Gender' style(header)=[background=yellow];
   define flag2 / across missing 'x' ' '
          style(header)=[background=yellow foreground=yellow];
   define age / style(header)=[background=lightgreen];
   define sex / across style(header)=[background=$gender.] ' ';
   define weight / style(header)=[background=orange];
   define height / style(header)=[background=tan];
run;
ods html close;
```

In this technique, the spanned header text is moved to the DEFINE statement as a column label. Note that in order to apply a background to a blank cell, you might need to add a column label value. In this case, add a FOREGROUND= using the same color as the BACKGROUND=. See the DEFINE statement for Flag2. Figure 14 shows the resulting output.

| | Gender | | | |
|---|---|---|---|---|
| | F | | M | |
| Age | Weight | Height | Weight | Height |
| 253 | 811 | 545.3 | 1089.5 | 639.1 |

**Figure 14. ODS HTML Output Using an Across Variable and Formats to Change the Header Background Color**

## NEW SAS® 9.2 AND SAS® 9.1.3 PROC REPORT TOPICS FOR ODS AND LISTING OUTPUT

### ADDING TEXT AT THE END OF A REPORT, EVEN WITH A PAGE OPTION

In PROC REPORT, if there is a BREAK statement with the PAGE option and a break at the end of the report through an RBREAK or a COMPUTE AFTER block, the end of report values will print on the next page. No options are available to keep the end of report values on the same page as the last detail values. The next two examples show how to keep the end of report values from going to the next page.

In this example, a macro variable is created before PROC REPORT to hold the last value of the BREAK variable Age. In the PROC REPORT code, a conditional statement is used in a COMPUTE AFTER age block to determine when the value for Age and the value for the macro variable are the same. When the conditional statement is true, then the end of report LINE statement is written after the last value of Age. Here is the code and the resulting output (Figure 15).

```
proc sort data=sashelp.class out=sorted;
   by age;
   data _null_;
      set sorted end=eof;
      by age;
      if eof then do;
         call symput('last',trim(left(age)));
      end;
run;
ods html style=minimal;
proc report nowd data=sashelp.class style(column header)=[background=white
                                                          rules=none];
   define age / order;
   break after age / page;
   compute after age/style=[just=l];
      length text $20;
      num=0;
      LINE age ' years old';
      if age=&last then do;
         text='***this is the end***';
         num=25;
      end;
      line text $varying. num;
   endcomp;
run;
ods html close;
```

| Name | Sex | Age | Height | Weight |
|---|---|---|---|---|
| Janet | F | 15 | 62.5 | 112.5 |
| Mary | F | | 66.5 | 112 |
| Ronald | M | | 67 | 133 |
| William | M | | 66.5 | 112 |
| 15 years old | | | | |

| Name | Sex | Age | Height | Weight |
|---|---|---|---|---|
| Philip | M | 16 | 72 | 150 |
| 16 years old<br>***this is the end** | | | | |

**Figure 15. Using a Macro to Determine the Last BREAK Variable Value to Print an End of Report LINE Statement**

This next example also shows you how to keep the end of report values on the same page as the last detail values. In this example, you use multiple PROC REPORT steps and remove the page break. In listing output, you can remove the page break by adding OPTIONS FORMDLIM=' '. In ODS destinations, you can use STARTPAGE=NO. The following code shows the FORMDLIM=' ' and STARTPAGE=NO options.

```
options formdlim='';
ods pdf file='c:\mypdf.pdf';
proc report nowd data=sashelp.class style=[rules=groups frame=void]
   style(column)=[cellwidth=.75in];
   col name age weight height;
   define age / order;
   break after age / page;
run;
```

```
options formdlim=' ';
ods pdf startpage=no;
proc report nowd data=sashelp.class noheader style=[rules=groups frame=void]
   style(column)=[cellwidth=.75in];
   col name1 blank1 weight height;
   define blank1 / computed format=$9.;
   define name1 / computed format=$8.;
   format weight height comma10.;
   compute name1/char;
      name1='Total';
   endcomp;
   compute blank1/char;
      blank1=' ';
   endcomp;
run;
ods pdf close;
```

Figure 16 shows the resulting ODS PDF output with the STARTPAGE=NO option.

| Name | Age | Weight | Height |
|---|---|---|---|
| Philip | 16 | 150 | 72 |
| | | | |
| Total | | 1,901 | 1,184 |

**Figure 16. ODS PDF Showing the End of a Report Row Using Multiple PROC REPORT Steps and STARTPAGE=NO**

## CONDITIONALLY EXECUTING A LINE STATEMENT

By design, PROC REPORT always prints a LINE statement. However, when the format for the entire LINE statement is 0, the LINE statement is created in memory but does not print in the corresponding output. The SAS format $VARYING*w*. *length-variable* is used in this technique because the value for *length-variable* can be changed each time the COMPUTE block is executed.

In the following example code, an IF statement is used to determine when AGE=15.

```
ods html ;
proc report nowd data=sashelp.class style(column header)=[background=white
                                                          rules=none];
   column  Name Sex Age Height Weight;
   where age in(15,16);
   define  Name / display format= $8.  "Name" ;
   define  Sex / display format= $1.   "Sex" ;
   define  Age / order format= best9.  "Age" width=5;
   define  Height / sum format= best9. "Height" ;
   define  Weight / sum format= best9. "Weight" ;
   compute after age;
      length  text $ 50;
      if age=15 then do;
         text = 'age=15 only';
         num=50;
      end;
      else do;
         text = "";
         num=0;
      end;
      line  text $varying. num ;
   endcomp;
run;
ods html close;
```

When the IF condition is true, a DATA step variable is created to hold the LINE statement text and *length-variable* is set to an appropriate length for the text to print. When the IF statement is false, the DATA step variable is set to blank and *length-variable* is set to **0**. The LINE statement is used to print the value of the DATA step variable using the format $VARYING*w*. *length-variable*. The resulting output is shown in Figure 17.

| Name | Sex | Age | Height | Weight |
|---|---|---|---|---|
| Janet | F | 15 | 62.5 | 112.5 |
| Mary | F | | 66.5 | 112 |
| Ronald | M | | 67 | 133 |
| William | M | | 66.5 | 112 |
| age=15 only | | | | |
| Philip | M | 16 | 72 | 150 |

**Figure 17. ODS HTML Showing a LINE Statement that Appears to Be Conditional**

## TAKING CONTROL OF ACROSS VARIABLES

Across variables can be one of the most confusing concepts of PROC REPORT. This is especially true when you have to reference an across column or a cell in a COMPUTE block. The column number in the form of _C*n*_, where *n* is the actual column number, has to be used for each column under the across variable. The following code shows a basic PROC REPORT step with an across variable. The resulting output is shown in Figure 18:

```
ods html;
proc report nowd data=sashelp.class
   style(column header)=[background=white rules=none];
   col ('Gender' sex) age,weight ('Avg Weight' weight);
   define sex / group ' ';
   define age / across;
   define weight / mean format=8.2 ' ';
run;
```

| | Age | | | | | | |
|---|---|---|---|---|---|---|---|
| Gender | 11 | 12 | 13 | 14 | 15 | 16 | Avg Weight |
| F | 50.50 | 80.75 | 91.00 | 96.25 | 112.25 | . | 90.11 |
| M | 85.00 | 103.50 | 84.00 | 107.50 | 122.50 | 150.00 | 108.95 |

**Figure 18. ODS HTML Showing an Across Variable**

If you wanted to add a format based on the row value for each column under the across variable and make some style attribute changes for your ODS output, you need to have a separate CALL DEFINE for each column and each row under the across variable.

However, you can control how the format and style is applied to the columns under an across variable. Place the COMPUTE block CALL DEFINE within a DO loop and use the index variable as the column ID instead of the _C*n*_. The following code shows how to apply a format and a style using a CALL DEFINE within a DO loop. The resulting output is shown in Figure 19.

```
proc format;
   value ffmt low-90='under 90'
               91-high='over 90';
   value mfmt low-100='under 100'
               101-high='over 100';
proc report nowd data=sashelp.class
   style(column header)=[background=white rules=none];
   col ('Gender' sex) age,weight ('Avg Weight' weight);
   define sex / group ' ';
   define age / across;
   define weight / mean format=8.2 ' ';
   compute age;
      if sex='F' then do;
         do i=2 to 7;
            call define(i,'format','ffmt.');
            call define(i,'style','style=[background=pink]');
         end; end;
      if sex='M' then do;
         do i=2 to 7;
            call define(i,'format','mfmt.');
            call define(i,'style','style=[background=lightblue]');
         end; end;
   endcomp;
run;
```

| | Age | | | | | | |
|---|---|---|---|---|---|---|---|
| Gender | 11 | 12 | 13 | 14 | 15 | 16 | Avg Weight |
| F | under 90 | under 90 | over 90 | over 90 | over 90 | . | 90.11 |
| M | under 100 | over 100 | under 100 | over 100 | over 100 | over 100 | 108.95 |

**Figure 19. Using a DO Loop to Apply a Format and Style to Columns Under an Across Variable**

However, what if you needed more control of the across variables? What happens when the input variable value for the across variable is not a static number of unique values each time the code is submitted? The question now is how to code for a changing number of across columns, especially when calculations are based on the across column values.

Look at the following code:

```
data test;
   input month_incurred paid $ amt rate;
   datalines;
1 tier1 38.28 0.3553
1 tier2 48.81 0.2631
1 tier3 61.18 0.0845
2 tier2 30.01 0.1021
2 tier3 01.65 0.1112
3 tier3 88.92 0.2937
;
run;
ods html;
proc report nowd data=test;
   col ('Month' month_incurred) paid,( amt rate) ('Rate' newrate);
   define month_incurred / group ' ';
   define paid / across 'Amount Paid';
   define newrate / computed ' ';
   define rate / noprint;
   define amt / ' ';
   compute newrate;
      newrate=sum(_c3_,_c5_,_c7_);
   endcomp;
run;
```

The input data shows that there are three unique values of the across variable. The last column of the report uses some of the values of the across columns in the calculation. The calculation is straight forward in this case. The resulting output is shown in Figure 20.

| | Amount Paid | | | |
|---|---|---|---|---|
| Month | Tier1 | Tier2 | Tier3 | Total Rate |
| 1 | $38 | $49 | $61 | 0.7029 |
| 2 | . | $30 | $2 | 0.2133 |
| 3 | . | . | $89 | 0.2937 |

**Figure 20. PROC REPORT with the COMPUTED Column Based on the Across Column Variable Values**

When you change the number of unique values in the across variable, you also need to change the COMPUTED variable calculation. Furthermore, if you want to create an output data set from the preceding code, the output data set will contain the column numbers as the variable names and labels instead of the actual variable values that contributed to the across column value. You need better control of the across variables.

In this situation, the best option is to not use an across variable and to reshape the data set before the PROC REPORT step. In order to reshape the data set, you need to summarize the data. In the following code, a PROC REPORT step is used to summarize the data and create the calculated computed variable. An output data set is created and used as input for the PROC TRANSPOSE step. Now all of the column data is in separate variables.

In the final PROC REPORT step, the previous across variable values are now columns that can be named in the COLUMN statement. To account for the possible across variable values that might show in the input data set, you can add each unique variable value to the COLUMN statement. PROC REPORT sees these variables that are not from the input data set as COMPUTED. The NOZERO added to the DEFINE statement keeps the column from printing in the output report. The resulting output is the same as for the preceding code (Figure 20).

```
proc report nowd data=test out=grouped;
   col month_incurred paid amt rate newrate;
   define month_incurred / group noprint ;
   define paid     / group noprint;
   define amt / noprint;
   define rate / noprint;
   define newrate / noprint;
   compute before month_incurred;
      totrate=rate.sum;
   endcomp;
   compute newrate;
      newrate=totrate;
   endcomp;
run;
proc transpose data=grouped(where=(_break_ eq ' ')) out=trans;
   by month_incurred newrate;
   id paid;
   var amt;
run;
   /* PROC REPORT without using an across. */
proc report data = trans nowd  missing
   style(column header)=[background=white rules=none];
   column month_incurred ('Amount Paid' tier1 tier2 tier3 tier4) newrate;
   define month_incurred / group "Month" missing;
   define tier1 / 'Tier1' format=DOLLAR12.;
   define tier2/ 'Tier2'  format=DOLLAR12. nozero;
   define tier3 / 'Tier3' format=DOLLAR12. nozero;
   define tier4 / 'Tier4' format=DOLLAR12. nozero;
   define newrate /  'Total Rate';
run;
```

## WHAT IS NEW FOR PROC REPORT IN SAS® 9.3

In SAS 9.3, PROC REPORT is planned to support the multilabel (MLF) format. This enhancement was also a previous SASware Ballot item. PROC REPORT will be able to use the format label or labels for a given range or overlapping ranges to create subgroup combinations. The MLF will be supported in the DEFINE statements for both group and across variables. This ability also applies to the REPORT window and output data sets.

Here are some other planned additions for PROC REPORT in SAS 9.3:

- The PROC REPORT SPANROWS option will work better with TAGSETS.RTF and will reduce some of the measurement issues.
- Hyperlinks will no longer be applied to blank cells.
- You will be able to use the column number, variable name, and _COL_ as the CALL DEFINE column-id within a COMPUTE AFTER block.
- You will be able to use the n-literal in the REPORT window.

## CONCLUSION

As you can see, SAS 9.2 includes a lot of changes for PROC REPORT. This paper presented an overview of the changes and further explored them showing code examples with the corresponding output. New tips and tricks were offered that could be used in SAS 9.2 and SAS 9.1.3 for different output destinations. Finally, this paper gave you a preview of what is new for PROC REPORT and SAS 9.3.

Now, you are challenged to use the new SAS 9.2 functionality and enhanced features for PROC REPORT to better control your output.

## RECOMMENDED READING

Booth, Allison McMahill. 2007. "Beyond the Basics: Advanced PROC REPORT Tips and Tricks." *Proceedings of the SAS Global Forum 2007 Conference*. Cary, NC: SAS Institute Inc. Available at support.sas.com/rnd/papers/sgf07/sgf2007-report.pdf.

## CONTACT INFORMATION

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

Allison McMahill Booth
SAS Institute Inc.
SAS Campus Drive
Cary, NC 27513
E-mail: support@sas.com
Web: support.sas.com