PhUSE 2012

Paper PP20

Using the Latest ODS Features in SAS 9.2

Chris Gibby, OCS Consulting, Den Bosch, the Netherlands
Kevin Scase MSc, WIL Research Europe B.V., the Netherlands

ABSTRACT

In this paper we discuss the methods used in adapting a Flexible Reporting System (FRS) utilising new features in SAS 9.2 / ODS. OCS Consulting maintains the FRS for WIL Research Europe B.V., a pre-clinical life science organisation. The FRS enables import of data from lab results and presents this in a PDF document. Originally the FRS used Adobe distiller to convert a postscript file into PDF, as the system was built using an earlier version of SAS, pre ODS. Due to two converging factors there was a requirement to enhance the FRS enabling direct output to PDF. These factors were: 1) updates in SAS 9.2 ODS, 2) the need for reports to adhere to electronic common technical document (eCTD) guidelines. This paper will discuss the technical details in achieving this, using a combination of PROC REPORT, PROC DOCUMENT and ODS to create one PDF file containing many reports / graphs, fully bookmarked and with hyperlinked TOC.

INTRODUCTION

OCS developed a flexible reporting system (FRS) for WIL Research Europe B.V., a pre-clinical life science company which is part of the WIL Research group of companies. The FRS is used to import pre-clinical data and select a variety of reports and graphs to present in one PDF file. The FRS was initially developed in an older version of SAS, pre ODS, and as such to output PDF files it was necessary to first create a postscript file from SAS and then use Acrobat distiller to convert this into a PDF file. While this works very well, it does not offer the use of some smart developments in SAS – in particular the advent of ODS.

Due to new requirements for the attributes of the electronic common technical document (eCTD) from the international conference on harmonisation (ICH) the reports produced by the FRS were no longer produced with the correct specifications. Because of these requirements and due to advances in ODS and new features in SAS 9.2, WIL Research Europe B.V. requested that OCS redevelop the output modules of the FRS to utilise these features while at the same time adhering to the new eCTD standards.

Specifically the ICH required the eCTD page margins, font weight and font size to be standardised for all new submissions. Also the report should fit on both A4 and letter size paper, as well as providing bookmarks and a table of contents with hyperlinks. Finally the initial view of the PDF should be set to bookmarks and page.

These specifications would have been difficult to achieve with the current output module of FRS using postscript to distiller functionality. Fortunately with the use of SAS ODS and some specific features in SAS 9.2 the requirements for the new eCTD guidelines are readily achievable. With these features we can remove the postscript – distiller functionality and instead run our datasets through a PROC REPORT / ODS DOCUMENT combination and output to an ODS PDF destination. For clarity it should be noted that all features described here are also available with SAS 9.3.

NEW FEATURES IN SAS 9.2

The new features in SAS 9.2 that help us with the requirements described above are as follows:

1. The key feature in SAS 9.2 is the compatibility of PROC REPORT with ODS DOCUMENT (and hence PROC DOCUMENT).
2. Use of the CONTENTS= option in the PROC REPORT BREAK statement, so we can control the nodes on PDF bookmarks.
3. Addition of the PDFPAGEVIEW and PDFPAGELAYOUT options.

We can also utilise these existing ODS features:

4. Use of ODS PDF ANCHOR to create hyperlinks from our table of contents to the report.
5. Use of ODS PDF STARTPAGE to control where we break our reports. With careful calculation we can determine where our columns and rows break, creating individual documents so reports break where we want them to.

Let's take a look at points 1-3 in more detail to explain why they are helpful to us in our report generation with ODS:

1. PROC REPORT is an excellent procedure for presenting quality reports. If we are using an application to create multiple reports to be displayed in one PDF, the ODS PDF destination must remain open for every report we generate with PROC REPORT. Because there are a lot of data steps required to shape the data for each report this becomes impractical and not very controllable. However now PROC REPORT can be used with ODS DOCUMENT this problem is solved. Each report can be sent to a document and stored in a permanent library where it can be accessed when required. Successive reports (documents) can be produced like this without having to hold open a PDF file. When our last report (document) is ready we can use PROC DOCUMENT to replay our stored reports one by one to a PDF file. This gives us great flexibility in how we build and process our reports (documents), including how we order them in the output file.
2. To produce a smart PDF document it is also nice to be able to produce tidy bookmarks without excessive levels (or nodes). Although bookmarks are already partly controllable with PROC REPORT they are much easier to control using PROC DOCUMENT. Therefore once we have created our documents we can also easily manipulate the bookmarks. Furthermore another new addition to PROC REPORT in SAS 9.2 is the use of the CONTENTS= option on a BREAK statement. This enables us to remove the 'Table 1' node from PROC REPORT, which cannot be achieved from PROC DOCUMENT itself.
3. One of the guidelines from the eCTD is that the PDF initial opening should be set to bookmarks and page. Fortunately in SAS 9.2, ODS has two new options PDFPAGEVIEW and PDFPAGELAYOUT which allow us to achieve this.

We have described here the background to the changes required to the FRS and how SAS 9.2 can help us implement these requirements. During the course of the paper we will discuss the approach and the technical solutions we implemented.

TECHNICAL SOLUTIONS

In summary the following steps are required to create our reports in one output PDF file:

1. Produce report ready dataset.
2. Determine how many lines and columns will fit on an A4 page.
3. Calculate control macro variables and paging variables.
4. Sequentially execute PROC REPORT statements to an ODS DOCUMENT destination.
5. Sequentially execute PROC DOCUMENT statements to configure report bookmarks.
6. Create a front page table of contents with hyperlinks.
7. Replay all documents to an ODS PDF file.

Let's explain each of these steps to detail the approach and highlight the technical solutions:

1. Produce report ready dataset

We start with a report ready dataset containing our data, statistics, column and group information, titles, sub-titles, etc. In our report ready datasets we always ensure all data is converted to character variables. We assume that the reader is already familiar with producing datasets suitable for PROC REPORT.

2. Determine how many lines and columns will fit on an A4 page

One of the key elements of the project was to be able to accurately control how much information, both number of columns and number of lines, that will fit onto a page without spilling onto another page.

In our report dataset we have all the information required for presenting the report. We know the number of columns and we know the type and length of data to expect in each column. We also know the number of observations that will be presented to the report.

What we don't know yet is how many columns and lines will appear on a page before the report automatically breaks onto a new page. We would like to control this so we always know which columns and which line numbers will fit onto a page so the report always breaks where we want it to.

The precise number of columns and lines that will fit onto a page will depend on the attributes of the PDF file we create, e.g. the margin sizes, font size, style template, etc. Therefore the first thing we must do is set these attributes so we can always be in control of our reports.

First we will set our margins – this will directly control the number of columns that will fit across the top of the page. Next we must create a user template (using PROC TEMPLATE) that removes all cell padding, cell spacing and border-widths, so there is no 'hidden' space that will disrupt our calculations. This is outside the scope of the paper but there are plenty of examples for creating custom style templates. Finally we must choose our font type and size. We must use the same font and font size throughout the report, otherwise we will lose control. Once these settings have been chosen they must remain in effect for all of our reports. The author discovered that font weight was unimportant, i.e. bold, standard or italic can be interchanged throughout the report without loss of control. The author also found that DPI considerably varied the number of lines that would fit on one page. DPI is useful for graph quality however it also affects the exact layout of data in a report.

Now that the report attributes have been set we can use an easy trick to figure out how many lines will fit exactly down one page. We will create a report using PROC REPORT and sequentially list numbers from 1 to 100. The report will be written to a PDF file (remembering to use all the attributes discussed above). Then we simply check at which number the page has broken.

For argument's sake let's say that with this approach we discover our settings allow us to fit 80 lines on one page of A4. So now we know the dimensions of our A4 page, we know our margin sizes and we know how many lines will fit on a page. We can now use this information directly with our dataset to figure out how many pages will be required to display the reports and where the data should be broken, at which column and at which line.

3. Calculate control macro variables and paging variables

In our report dataset we already know the type of data to expect in each column and we can determine the maximum (SAS) length of each column using the length() function. We can loosely correlate our column lengths into a length in mm using a column multiplier. A wider column in the dataset will therefore have a wider column in mm on the page. In this way we can use the exact dimensions of the A4 page (210mm wide - left margin - right margin) and the width of each column in mm, to determine how many columns from the dataset will fit across a page and whether there are more columns than the width of the page.

To do this we can simply set an array for the total number of columns in the dataset and loop through the array, column by column, keeping a cumulative total with respect to the page width. For each new column in the array we check to see whether the width of this column will exceed the remaining space on the page. If so then we create a macro variable containing the column (variable) names that will fit on the first page and then continue with the array processing to determine the columns on the next page. In this way we can create a series of column identifiers (macro variables) that we can add to the column statement in our PROC REPORT. We can think of these as 'blocks' of columns, with each block fitting on one page. We run a new PROC REPORT statement for each block of columns to ensure controllable page breaking. For example:

```
%let col_block1=Var1 Var2;
%let col_block2=Var3 Var4;
proc report data=din;
  column &col_block1;
  ----------------;
run;
proc report data=din;
  column &col_block2;
  ----------------;
run;
```

To determine the line (row) number that we need to break to a new page is slightly easier, because we can easily compare each dataset observation to the total page size (number of lines that we determined earlier). We can create a page number variable to keep track of this – for all observations up to the total lines that fit on one page, set this to page 1. Then increment this to page 2, and so on throughout the dataset.

To demonstrate this we use the sashelp.bweight dataset as an example (below), selecting only the first 50 observations. For simplicity let's assume we can only fit (or only want) 25 rows per page. Therefore we set page=1 for the first 25 observations and page=2 for the last 25 observations.

```
data bweight;
 set sashelp.bweight (obs=50);
 subjid=1000 + _n_;
 if _n_ <= 25 then pagenum=1;
 else pagenum=2;
run;
```

4. Sequentially execute PROC REPORT statements to an ODS DOCUMENT destination

The beauty of ODS DOCUMENT is that when we run PROC REPORT we can selectively write a portion of data to a document, depending on the number of columns / rows that will fit on a page. Therefore we can create a library of documents with a precisely controlled amount of data on each page. When we come to replay the documents we know exactly where to break using the ODS STARTPAGE=NOW option.

When we create our documents we must run a new PROC REPORT for each page that we wish to control the breaking. There are two elements to this, first to control the breaking of the columns and second the breaking of the rows. Again using the sashelp.bweight dataset for simplicity we determine that we can fit five columns across a page and we wish to create a report with nine columns from the dataset. We decided previously that we require 25 rows on each page.

Step 1: Create two documents for two pages with the first block of columns that fit across one page. If we are standardising our program this is where we would utilise a macro variable for the column names.

```
ods document name=work.report1(write);
  proc report data=bweight(where=(pagenum=1));
  column subjid boy visit weight m_wtgain;
  --------------;
ods document close;

ods document name=work.report2(write);
  proc report data=bweight(where=(pagenum=2));
  column subjid boy visit weight m_wtgain;
  --------------;
ods document close;
```

Step 2: Create another two documents with the remaining columns that didn't fit across the initial page:

```
ods document name=work.report3(write);
  proc report data=bweight(where=(pagenum=1));
  column subjid married smoke cigsper;
  --------------;
ods document close;

ods document name=work.report4(write);
  proc report data=bweight(where=(pagenum=2));
  column subjid married smoke cigsper;
  --------------;
ods document close;
```

We have now created four documents each containing one page of our report. Therefore we are now in complete control of presenting and breaking our pages where we want them, not where SAS decides to!

We can be much more creative with our page numbering. This is only a simple example but for a report with many groups and subtitles, etc. we can test to see row by row how many lines we have left on our page. We can therefore calculate whether one row or a group of rows will still fit in the remaining space of the page. Depending on the outcome we either maintain the current page number or increment to the next page, so we ensure PROC REPORT processes this whole group on the same page.

5. Sequentially execute PROC DOCUMENT statements to configure report bookmarks

For the first page of each report we require a bookmark to be associated with it in the PDF. Since we have a separate document for each page of the report, it is only the first page that requires a bookmark. In this example we use PROC DOCUMENT to set the label of our bookmark and to copy the contents of the document (report) to the level required to be associated with this bookmark. Note that here we are creating our final documents in a permanent library, based on the documents we first created in the work library.

```
proc document name=procdoc.rep1(write);
  make     bkmark;
  dir      \bkmark;
  setlabel \bkmark "BODYWEIGHTS";
  copy \work.report1\Report#1\Report#1\Report#1 to ^;
  dir ^^;
quit;
```

For subsequent pages we must execute the following code so no bookmarks are associated with these pages and hence not displayed in the PDF. Effectively we are copying the document to another level without setting a label (bookmark) at this level.

```
proc document name=procdoc.rep2(write);
  copy \work.report2\Report#1\Report#1\Report#1 to ^;
  dir ^^;
quit;
```

PROC DOCUMENT can be used to control the first two nodes (or levels) of bookmarks in the PDF file. These are the 'Report procedure' and 'Summarised report' nodes. These two nodes can also be controlled using, respectively, the ODS PROCLABEL statement and the CONTENTS= option in the PROC REPORT statement. There is a third node however that could not be edited or removed before SAS 9.2. This is the 'Table 1' node. With SAS 9.2 we can remove this node by creating a constant count variable throughout our dataset and set a BREAK BEFORE statement in conjunction with CONTENTS=, as demonstrated below. Note it is the CONTENTS= option in use with BREAK that is new for SAS 9.2.

```
data din;
  ----------;
  count=1;
run;

proc report data=din;
  column   ----------;
  define   ----------;
  break before count / contents='' page;
run;
```

6. Create a front page table of contents with hyperlinks

Using ODS PDF TEXT statements we can output lines of text to a page, format the text to appear as a hyperlink and set a URL which will be associated with an anchor in the main report. The syntax of this is shown below. Based on the total number of reports we can create a simple and effective front page table of contents linked to each report in the file. The creation of anchors in the report is explained in step 7.

```
ods pdf text = "^{style [url='#rpta' -----] Report A}";
```

7. Replay all documents to an ODS PDF file

We have now created all documents (reports) and stored them in a permanent library. We have also defined our bookmarks and created a table of contents. Everything is now ready to process and create our final PDF document. This is accomplished using PROC DOCUMENT and the REPLAY statement. We can simplify this using a macro, since we know: the number of pages in the report, which pages need to be forced to break and where to set the anchor for linking the top page of each report to the URL in the table of contents. The latter is accomplished with ODF PDF ANCHOR and should be execute immediately before the first document (page) of each report. We would also set our PDF layout and view options here. The code to demonstrate this is shown below.

```
ods pdf file='C:\temp\testout.pdf' style=journal startpage=never;
  %do rpt=1 %to 4;
    %if &rpt=1 %then ods pdf anchor='rpta';;
    ods pdf startpage=now;
    proc document name=procdoc.rep&rpt.;
      replay;
    quit;
  %end;
ods pdf close;
```

The full code that was used for these examples can be viewed in Appendix A. The reader can execute this code in SAS 9.2 or higher to see step by step how the process described above is working. In this example we have also created a graph and stored this in a document to demonstrate how straightforward it is to mix graphs and reports in a series of documents, and replay these into a single PDF file.

CONCLUSION

SAS ODS and the compatibility of PROC REPORT with ODS DOCUMENT can be a very creative method of processing a large number of reports and graphs and to smartly control the breaking of columns and pages within these reports. By building up a library of documents the user also has great control over bookmarks and the order of processing these reports to a PDF output file.

REFERENCES

Using PROC DOCUMENT to Modify PDF Bookmarks Generated by PROC FREQ, Suzanne M. Dorinski
http://www.nesug.org/proceedings/nesug08/np/np07.pdf

Use ODS and PROC DOCUMENT to collapse a multiple-level table of contents, SAS Institute
http://support.sas.com/kb/42/764.html

Problem Note 31278: Table 1 node generated by PROC REPORT
http://support.sas.com/kb/31/278.html

Usage Note 45808: Links to ANCHOR= values are not stored by ODS DOCUMENT
http://support.sas.com/kb/45/808.html

A SAS® Output Delivery System Menu for All Appetites and Applications, Chevell Parker, SAS Institute
http://www.wuss.org/proceedings10/Applications/3028_2_APP-Parker1.pdf

Beyond the Basics: Advanced REPORT Procedure Tips and Tricks Updated for SAS® 9.2, Allison McMahill Booth, SAS Institute
http://support.sas.com/resources/papers/proceedings11/246-2011.pdf

Let's Give 'Em Something to TOC about: Transforming the Table of Contents of Your PDF File, Bari Lawhorn, SAS Institute
http://support.sas.com/resources/papers/proceedings11/252-2011.pdf

PROC TEMPLATE: The Basics, Lauren Haworth, Genentech Inc.
http://www2.sas.com/proceedings/sugi31/112-31.pdf

CONTACT INFORMATION

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

Chris Gibby
OCS Consulting BV
PO Box 3434
5203 DK 's-Hertogenbosch
The Netherlands
Work Phone: +31 (0) 73 523 6000
Email: sasquestions@ocs-consulting.com
Web: http://www.ocs-consulting.nl

APPENDIX A: FULL SAS CODE EXAMPLE

```
libname procdoc 'C:\temp\documents';
title;

data bweight;
 set sashelp.bweight (obs=50);
 subjid=1000 + _n_;
 if _n_ <= 25 then pagenum=1;
 else pagenum=2;
 count=1;
run;
ods html close;

ods document name=work.report1(write);
  proc report data=bweight(where=(pagenum=1)) nowd split='^' style(header)={just=l};
    column count subjid boy visit weight m_wtgain;

        define count     / group noprint;
    define subjid    / group 'Subject'              style(column)={cellwidth=20 mm just=l};
    define boy       / group 'Gender'               style(column)={cellwidth=20 mm just=l};
    define visit     / group 'Visit'                style(column)={cellwidth=20 mm just=l};
    define weight    / group 'Weight (g)'           style(column)={cellwidth=20 mm just=l};
    define m_wtgain  / group 'Mean^weight gain (g)' style(column)={cellwidth=20 mm just=l};

    break before count / contents='' page;
  run;
ods document close;

ods document name=work.report2(write);
  proc report data=bweight(where=(pagenum=2)) nowd split='^' style(header)={just=l};
    column count subjid boy visit weight m_wtgain;

        define count     / group noprint;
    define subjid    / group 'Subject'              style(column)={cellwidth=20 mm just=l};
    define boy       / group 'Gender'               style(column)={cellwidth=20 mm just=l};
    define visit     / group 'Visit'                style(column)={cellwidth=20 mm just=l};
    define weight    / group 'Weight (g)'           style(column)={cellwidth=20 mm just=l};
    define m_wtgain  / group 'Mean^weight gain (g)' style(column)={cellwidth=20 mm just=l};

    break before count / contents='' page;
  run;
ods document close;

ods document name=work.report3(write);
  proc report data=bweight(where=(pagenum=1)) nowd split='^' style(header)={just=l};
    column count subjid married smoke cigsper;

        define count     / group noprint;
    define subjid    / group 'Subject'              style(column)={cellwidth=25 mm just=l};
    define married   / group 'Married'              style(column)={cellwidth=20 mm just=l};
    define smoke     / group 'Smoker'               style(column)={cellwidth=20 mm just=l};
    define cigsper   / group 'Cigarettes^per day'   style(column)={cellwidth=20 mm just=l};

    break before count / contents='' page;
  run;
ods document close;

ods document name=work.report4(write);
  proc report data=bweight(where=(pagenum=2)) nowd split='^' style(header)={just=l};
    column count subjid married smoke cigsper;

        define count     / group noprint;
    define subjid    / group 'Subject'              style(column)={cellwidth=25 mm just=l};
    define married   / group 'Married'              style(column)={cellwidth=20 mm just=l};
    define smoke     / group 'Smoker'               style(column)={cellwidth=20 mm just=l};
    define cigsper   / group 'Cigarettes^per day'   style(column)={cellwidth=20 mm just=l};

    break before count / contents='' page;
  run;
ods document close;
```

```
GOPTIONS gsfname=gname gsfmode=replace dev=jpeg reset=global cback=white ftitle="arial unicode
ms"
         ftext="arial unicode ms" gunit=pct htitle=3.1 htext=2.0 NOBORDER DISPLAY TRANSPARENCY;

ods document name=work.plot1(write);
  proc gplot data=sashelp.class gout=procdoc.gseg uniform;
    plot age*height;
  run;
  quit;
ods document close;


/* Create bookmark using report name */
%macro bookmarks;
  %do rpt=1 %to 4;
    %if &rpt=1 %then %do;
      proc document name=procdoc.rep1(write);
        make     bkmark;
        dir      \bkmark;
        setlabel \bkmark "BODYWEIGHTS";
        copy \work.report1\Report#1\Report#1\Report#1 to ^;
        dir ^^;
      quit;
       %end;
       %else %do;
         proc document name=procdoc.rep&rpt.(write);
              copy \work.report&rpt.\Report#1\Report#1\Report#1 to ^;
              dir ^^;
         quit;
    %end;
  %end;
  proc document name=procdoc.plot1(write);  * Note lib must be cleared on successive runs;
       make graph;
       dir \graph;
       setlabel \graph "AGE v HEIGHT";
       copy \work.plot1\Gplot#1\GPLOT#1 to ^;
       setlabel \graph#1\GPLOT#1 '';
       dir ^^;
  quit;
%mend bookmarks;
%bookmarks


options nocenter PDFPAGEVIEW=FITWIDTH PDFPAGELAYOUT=CONTINUOUS orientation=portrait papersize=A4;
option bottommargin="2.5 cm" leftmargin="2.5 cm" rightmargin="2.5 cm" topmargin="2.5 cm" nodate
nonumber;
title;
%macro op_report;
  ods pdf file='C:\temp\testout.pdf' style=journal startpage=never;

    options orientation=portrait;
    ods escapechar='^';
    ods pdf text = ' ';
    ods pdf text = '^{style [color=black just=l font_face=Arial font_size=8pt font_weight=bold ]
CONTENTS} ';
    ods pdf text = ' ';
    ods pdf text = "^{style [url='#rpta' linkcolor=white color=blue just=l font_face=Arial
font_size=8pt]Bodyweights}";
    ods pdf text = ' ';
    ods pdf text = "^{style [url='#rptb' linkcolor=white color=blue just=l font_face=Arial
font_size=8pt]Age versus Height}";

    %do rpt=1 %to 4;
         %if &rpt=1 %then ods pdf anchor='rpta';;
      ods pdf startpage=now;
      proc document name=procdoc.rep&rpt.;
           replay;
      quit;
    %end;
```

```
        options orientation=landscape;
        ods pdf anchor='rptb' startpage=now;
    proc document name=procdoc.plot1;
      replay;
    quit;

  ods pdf close;
%mend op_report;
%op_report
```