Paper 006-2012

ODS PDF and RTF Application Development:

Steps to Ensure Success and Examples of Useful Coding Techniques

Benno Kurch, Trading and Software Development, Inc., Merrifield, VA

ABSTRACT

For an ODS developer at a beginning or advanced level, creation of tailor-made ODS PDF or RTF reports is not an out-of-the-box, turnkey operation. Creation of such reports may require creative thinking and much experimentation. Implementing a larger font size, for example, when converting a line-printer ("legacy") report using a monospace font to an ODS report may require report re-design and consultation with business review personnel. This paper will present some challenges encountered in a recent upgrade from legacy reports to more refined, formatted PDF and RTF output, and the recommended solutions to those challenges. Taking time up front to think through potential issues can significantly reduce development time and facilitate the production of properly designed ODS reports that will impress users with their professional look.

INTRODUCTION

Our initial project mandate was to convert 17 pharmaceutical reporting applications producing about 100 legacy reports to ODS PDF and RTF reports. The conversion effort required developing SAS® ODS reporting code with UNIX SAS 9.2. The PDF reports represented the initial stage of producing reports for FDA submission. The RTF reports were primarily for our Medical Writing group. The legacy report applications called post-processing programs to manipulate reports that the REPORT procedure or the TABULATE procedure generated. The post-processing programs added page X of Y information, section headers or desired blank rows.

We elected to produce the RTF reports using ODS measured RTF (also known as the RTF tagset). We just used the ODS TAGSETS.RTF statement, the TEMPLATE procedure and PROC REPORT to produce RTF files. We did not make changes to the default tagset since there was a substantial learning curve involved. For the prototype reports we came close to producing the reports as we expected to see them. However, we could not figure out how to prevent footnotes from going to the next page. We were not knowledgeable of the RTF language so we did not attempt to post-process the RTF. SAS Institute, Inc. sent us TAGSETS.RTF code to correct the problem. This code contains many other useful features and provides a great basis for TAGSETS.RTF development. You can access this code by clicking on the Downloads tab from this SAS support link: https://support.sas.com/kb/40/327.html

Our experience with producing a measured RTF report illustrates a classic problem of underestimation. Since we could produce a prototype quickly with the new technology, we did not realize how much time and knowledge it would take to properly implement the whole new methodology. Minimal examples at the SAS web site and on the Web in general compounded the difficulty (TAGSETS.RTF had just been introduced into SAS when we began using that tagset). By using the ODS TAGSETS RTF statement without tagset code edits and with a few changes to your style template using PROC TEMPLATE, you can produce an almost adequate report. However, because of the inherent complexity of RTF reports, use of tagsets code and a thorough understanding of the RTF language is a requirement to produce a finely tuned report that will meet your users' professional standards. TAGSETS.RTF is powerful software that promises to make measured RTF creation even easier since SAS Institute is putting more resources into development of the measured RTF destination. Several papers listed in the References section make it easy to understand how to use and create tagsets code.

Problems that surfaced for us in implementing ODS PDF reporting were these: (1) Right justification of Page X of Y in first title line did not work when report replayed in PROC DOCUMENT (2) Incomplete suppression of all PDF bookmark nodes other than the primary one (3) Determination of the optimal column width settings to produce as much output on the page as possible and produce a professional looking report (4) Center decimal alignment of numeric fields (5) Suppression of grid line between N and percentage columns. We also encountered other minor problems. Based on your client's reporting requirements and also based on your operating system implementation, the spectrum of problems that appear for your ODS PDF reporting may encompass some of these problems or will vary.

THE MOST IMPORTANT CONSIDERATIONS WHEN CONVERTING A LEGACY REPORTING SYSTEM TO AN ODS PDF OR RTF BASED SYSTEM

For many companies the reports being converted may have been created 5 years ago or more. The business group probably will be excited when they see a prototypical ODS PDF or RTF report since the report gives them the slick, professional look they are seeking. The business group informed about ODS capability will realize that maintenance of programs post-processing PROC REPORT output is history since, for one thing, ODS will add Page X of Y information to PDF (and RTF) reports.

Not aware of the potential problems involved in the light year jump technology, the business group may give the green light to convert the reports. This may occur before any consideration is given to major report redesign. Before embarking on the re-programming step, the ODS reports developer needs to stop and alert the business group of the probable need to re-design reports.

Font changes may force the re-design of a whole report. For example, a lab report may have taken up the whole width of the report page when it was printed in 8 point Courier, a small fixed width font. If the report is converted to 10 point Arial, a common proportional font, the report will no longer fit across the page and may overflow to the next page or be truncated. Your users want to display as many columns of information on the page as possible so comparison is optimized. One way to approach the re-design if a row often extends to multiple rows is to combine columns. For example, age, race and sex may occupy 3 columns. You may want to combine age, race and sex into one column (so if age has a value of 46, race white and sex male, you could represent the column label as Age/Race/Sex and the values as 46/White/Male). If enough values for other columns other than Age, Race and Sex extend to multiple rows then you may want to represent the information vertically (For the label, i.e., you might want Age/ on the first line, Race/ on the second line and Sex on the third line).

Legacy reports often have characters all with the same size, a fixed line size and fixed page size. Post-processing programs leverage this functionality to make decisions about keeping a group of information on the same page. ODS PDF reports typically employ proportional fonts. These reports have infinite line size and page size. Breaking to a new page is a logical and no longer physical consideration. To illustrate, let’s say you have a group of System Organ Class records that you want to keep together on a page. For a legacy report you can determine how many lines are left for display on the page and how many records are in your group. If there are more records in the group then can fit on the page, your post-processing program forces a page eject, and prints the “System Organ Class” sectional header line (a literal not in a GROUP variable). With ODS PDF your easiest option is to create a page break variable. However, by doing this you most likely will have report pages with lots of wasted space. ODS PDF doesn’t have the capability to measure line height size. Developing an algorithm to do this would be time consuming.

The developer armed with knowledge should request a meeting with all parties involved in using the reports. Because programming ODS reports can be labor intensive, a meeting is crucial at the start of the process. The developer needs to ask: “Do the current legacy reports optimally advance business needs? If not, how should the reports be changed to do this?”. If reports have not been upgraded for 5 years or more, then most likely the user community will embrace the opportunity to re-vamp the reports. We found in developing a large-scale ODS application that there were many report items users wanted to change. The developer needs to acquire a clear understanding of what the user community desires before thinking about ODS programming details.

When you elicit user feedback about reporting requirements you may find that the report conversion task has become more complicated. Once you start intensively converting and you run into programming issues you may find the task becomes even more challenging. Scope creep, the developer’s nemesis, constantly intrudes. To combat this you need to establish regular communication with your business group and to establish limits. Should a change, for example, go in our first version or should it be delayed to a subsequent production version? We recommend weekly technical meetings with representatives from the business and programming groups and general business group meetings for review of reports. Your company may even want to implement ODS reporting in the background of some structured development methodology such as Rational Unified Process (RUP), for example. When formulating possible solutions before engaging in a business group discussion, the developer needs to become as informed as possible about existing or potential problems. The developer can do this by talking to other developers who have already built an ODS system, reading some of the excellent SAS books written about ODS and by looking at SAS blogs.

REPORTING SOFTWARE TO IMPLEMENT FOR CONVERSION OF MULTIPLE REPORTING TYPES: REPORT, TABULATE AND DATA _NULL_:

If legacy programs are written primarily with DATA _null_ logic, converting to the use of PROC REPORT and/or PROC TABULATE will greatly increase your ability to improve the style of your report output. The learning curve for each of these procedures is large, so training time for each needs to be factored into your decision. Also you need to factor in more time for the ODS portion of the conversion since ODS for PROC REPORT and PROC TABULATE sometimes involves different ODS coding techniques.

At the start of our project we decided to convert our PROC TABULATE code to PROC REPORT code. Doing so took less effort than you might think, just a few hours or a day to convert a report. With business approval we re-designed the reports so they would be easier to produce with REPORT (less TABULATE centric). The REPORT COMPUTE block allowed us to produce reports with more of the tailoring that DATA _NULL_ provides. While some programmers and users preferred PROC TABULATE or DATA _NULL_ to PROC REPORT, we found that long-term maintenance and consistency across reporting platforms were simplified by using only REPORT. We strongly suggest selecting one reporting type for all the reports, to reduce re-programming difficulties.

ESTABLISHING THE TEMPLATE (FOUNDATION) FOR ODS PDF DEVELOPMENT

PROC TEMPLATE can appear intimidating at first. You see that the default template style Styles.rtf doesn't give you the report you're looking for. However, we quickly found template examples at the SAS support site and on the web for ODS PDF. After some experimentation we came up with template code that required no modification as we progressed forward.

```
ods path work.templates(update) sashelp.tmplmst(read) sasuser.templat(update);

proc template;
  define style yourstyl / store=templates;
         parent = Styles.rtf;
  class fonts /
    'TitleFont'            = ("Times New Roman, Times Roman, Times" ,10pt,Bold )
    'TitleFont2'           = ("Times New Roman, Times Roman, Times" ,10pt,Bold )
    'footFont'             = ("Times New Roman, Times Roman, Times" ,10pt,Bold )
    'StrongFont'           = ("Times New Roman, Times Roman, Times" ,10pt,Bold)
    'EmphasisFont'         = ("Times New Roman, Times Roman, Times" ,10pt,Italic)
    'headingEmphasisFont' = ("Times New Roman, Times Roman, Times" ,10pt,Bold)
    'headingFont'          = ("Times New Roman, Times Roman, Times" ,10pt,Bold)
    'docFont'              = ("Times New Roman, Times Roman, Times" ,10pt)
    'FixedEmphasisFont'    = ("Courier" ,9pt,italic)
    'FixedStrongFont'      = ("Courier" ,9pt,Bold)
    'FixedHeadingFont'     = ("Courier" ,9pt,Bold)
    'BatchFixedFont'       = ("Courier" ,9pt)
    'FixedFont'            = ("Courier" ,9pt)
  ;
  class color_list /
    'link'= blue
    'bgH' = white
    'bgT' = white
    'bgD' = white
    'fg'  = black
    'bg'  = white
  ;
  style Table from Output /
      rules       = all
      cellpadding = 4px
      cellspacing = 0
      bordercolor = lightgrey
  ;
  class systemtitle /
    protectspecialchars=OFF
    asis=ON
  ;
  class systemfooter /
    font=Fonts('footFont')
    protectspecialchars=OFF
    asis=ON
  ;
```

```
    class header /
      protectspecialchars=off
    ;
    class data /
      protectspecialchars=off
    ;
    class rowheader /
      protectspecialchars=off
    ;
    class usertext /
      protectspecialchars=off
    ;
    class byline /
      protectspecialchars=off
    ;
    end;
  run;
```

Depending on your business preferences you will need to do some tailoring of the template code. You may want to change the type size and font for the various elements, place more padding around the text of the cell, and place more space between cells. We built the templates dynamically; we added macro code around the template code, for example, to allow users at run-time to produce a template with no grid lines within the table body (rules=groups). Your site may want to create permanent templates if your reports never have grid lines within the table body or you never want to change the bolding attribute.

This information from the SAS on-line documentation for BASE SAS: ODS fonts is very useful:

The fonts associated with the following three strings are the most commonly modified:

The string 'DocFont' controls the fonts for the majority of the output, such as the data values and the items in the table of contents.

The string 'headingFont' controls the fonts for the headers.

The string 'TitleFont' controls the fonts for the titles and footnotes.

Regarding style color_list:

'fg' = black is specified. In the parent style Styles.RTF 'fg' = black is also specified.

If 'fg' = black removed from color_list warning message will appear in log,

i.e., Warning: Could not locate style reference 'std.colors("bylinefg")').

'fg' - color specified affects color of text in cells throughout the table.

'bg' - color specified affects background color of entire table except for title, footer and column header areas.

'bgH' - color specified just affects background color of header area.

You will also need to set your margins. We chose margins suitable for a pharmaceutical application:

```
  options
         topmargin    = 0.75in
         bottommargin = 0.75in
         leftmargin   = 0.75in
         rightmargin  = 0.75in;
```

You can also do this in the template:

```
  Style body from Document /
         topmargin    = 0.75in
         bottommargin = 0.75in
         leftmargin   = 0.75in
         rightmargin  = 0.75in
```

PRODUCING YOUR PDF

Example code:

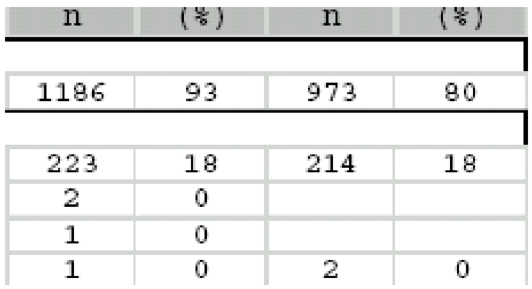
```
ods listing close;
options pdfcopy  pdfaccess  pdfassembly  pdfcomment  nopdfcontent;

ods pdf file  = "./yourfile.pdf"
        compress = 6
        newfile    = NONE
        pdftoc     = 1
        startpage  = YES
        style      = yourstyle
        uniform
        nobookmarkgen
 ;
```

Finally, after adding your PROC REPORT code, issue the command:

```
ods PDF close;
```

LINE SUPPRESSION BETWEEN REPORT CELLS

Your client may request that you suppress the grid line between the count and percent columns (Figure 1):



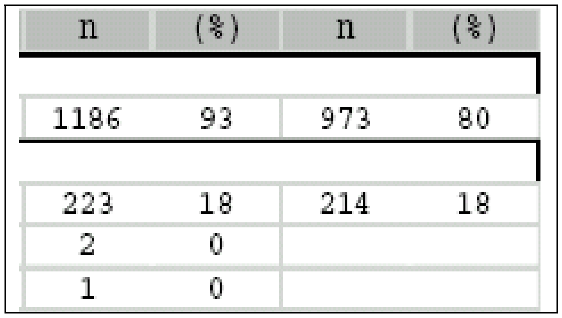
n	(%)	n	(%)
1186	93	973	80
223	18	214	18
2	0		
1	0		
1	0	2	0

Figure 1: PDF output with the gridline between n and %.

You may think when specifying RULES=ALL that you can set the BORDERRIGHTWIDTH=0 for the N column and set BORDERLEFTWIDTH=0 for the percent column. You will find that setting these values for PDF or RTF does not work; the line between N and percent still appears. You can white out the line with BORDERRIGHTCOLOR=WHITE or BORDERLEFTCOLOR=WHITE. That may work depending on your version of Adobe Acrobat or the template you are using. To remove the gap we recommend adding in this PROC REPORT code snippet a BORDERTOPWIDTH= and BORDERBOTTOMWIDTH= specification (output in Figure 2):

```
     define count    / analysis  sum  "Count"

       %if ( &produce_grid_lines= 1 ) %then %do; /* count - grid lines */

          %if ( &produce_percentages = 1 ) %then %do; /* percentages */

                      style(column)={just=center   width=15mm
                                          borderrightcolor=white
                                   bordertopwidth=1px
                                   borderbottomwidth=1px
                                }
                      style(header)={just=center}

          %end; /* percentages */

          %else %do; /* no percentages */
```

```
                          style(column)={just= center  width=15mm}
                          style(header)={just=center}

         %end; /* no percentages */

      %end; /*count - grid lines */

      %else %do; /* count - no grid lines */

                            style(column)={just=center  width=15mm}
                            style(header)={just=center}

      %end; /* count - no grid lines */
    ;

    %if ( &produce_percentages = 1 ) %then %do; /* display percentages */

        define _pct_    /  "Percentages"

           %if ( &produce_grid_lines = 1 ) %then %do; /* percent - grid lines */

                            style(column)={just=center   width=15mm
                                           borderleftcolor=white
                                             bordertopwidth=1px
                                             borderbottomwidth=1px
                                           }
                            style(header)={just=center}

           %end; /* percent - grid lines */

           %else %do; /* percent - no grid lines */

                            style(column)={ just=center   width=15mm }
                            style(header)={just=center}

           %end; /* percent - no grid lines */
        ;

    %end; /* display percentages */
```

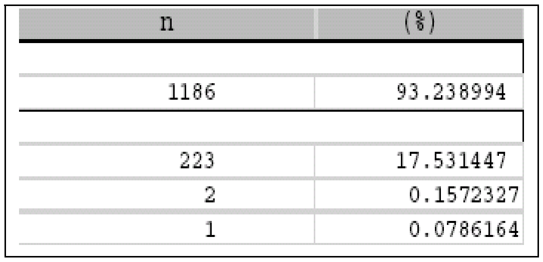


n	(%)	n	(%)
1186	93	973	80
223	18	214	18
2	0		
1	0		

Figure 2: PDF output with no gridline between n and %.

CENTER ALIGNMENT OF DECIMAL VALUES WITHIN A CELL

For legacy reports with fixed character sizes (a non-proportional font) center aligning decimal values is easy. For SAS 9.2 producing ODS PDF files with a proportional font this is not easy. When producing a PDF file SAS software as of 9.2 does not allow you to center a value in a cell and align on the decimal. You can accomplish this to some extent by specifying STYLE(COLUMN)={JUST=CENTER ASIS=ON} and assigning a PICTURE format to the value. This technique is not perfect, however, and you will find with different combinations of column numbers that sometimes the alignment will noticeably not appear centered. We don't recommend assigning a PICTURE format. We recommend using JUST=D (ASIS=ON not necessary) to right align on the decimal (output in Figure 3).



n	(%)
1186	93.238994
223	17.531447
2	0.1572327
1	0.0786164

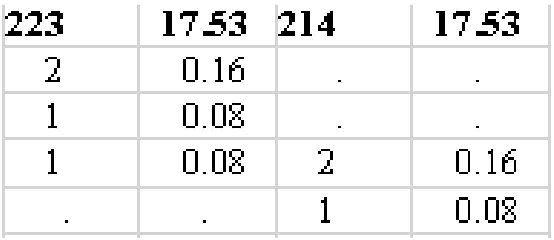
Figure 3: PDF output center aligned

If producing an ODS RTF file you can easily center align on the decimal using in-line RTF commands with the 'R' destination code. An example of this code for your PROC REPORT DEFINE statement:

```
     define _count_ /  "Count"
                       Style(column) = [pretext='^R"\ql\tqdec\tx250 "' ];
     define _pct_   /  "Percentages"
                       Style(column) = [pretext='^R"\ql\tqdec\tx250 "' ];
```

The 'ql' means to left justify, '\tqdec' set the decimal tab and '\tx250' set the space for the decimal tab at 250 twips. A twip is 1/1440 of an inch.

Part of the output file is shown below (Figure 4):



223	17.53	214	17.53
2	0.16	.	.
1	0.08	.	.
1	0.08	2	0.16
.	.	1	0.08

Figure 4: RTF output center aligned

PRODUCING ONE PDF BOOKMARK NODE WITH NO SUB-NODES

To suppress sub-nodes there are some tacks you can take (reference SAS support problem note 31278 (http://support.sas.com/kb/31/278.html). 3 nodes appear by default. The ODS PROCLABEL statement controls the production of the first (primary) node. By default the value for this node is "The Report Procedure". Via the PROCLABEL statement you replace this value with your own such as "Table 1". To suppress the production of the second node (a sub-node), code the CONTENTS= option with no space between the quotes on your PROC REPORT statement:

```
      Proc report data=x nowd list contents=''
```

To suppress the third node, also a sub-node, you have to counteract a SAS change to PROC REPORT's table of contents entry in SAS 9.2: You have to create a dummy variable, reference it within PROC REPORT and specify the CONTENTS= option for the variable within the BREAK statement:

```
      Data x;
        Set x;
        … ;
        dummy_var = 1;
      run;

      proc report  data=x nowd list contents='';
        columns dummy_var  … ;
        define dummy_var  / group noprint;
        break before dummy_var / contents='' page;
```

Even after you implement this code, sub-nodes will appear in your Table of Contents (bookmarks) if your PROC REPORT code contains a BY statement.

You may realize that the DOCUMENT procedure can be used to remove/manipulate sub-nodes as well. This solution will work but becomes a bit complicated when a page by report is involved or you are stacking output from multiple procedures (REPORT, FREQ, for example) in one PDF. A solution that avoids the use of DOCUMENT and is quite simple is the following code. The solution involves creating a dummy report page with no visible character (we tested this code extensively and found no problem):

```
ods pdf file=”./example.pdf”
        startpage=YES
        nobookmarkgen;

proc report  data=x nowd;
   column a;
   define a …
run;
ods pdf bookmarkgen  startpage=no;

data test;
    x="00"x;
    output;
run;
options label;
ods proclabel = "Table X";

title;
footnote;

proc report data=test nowd contents=''
            style={font_size=.1pt   foreground=white   cellpadding=0
            rules=none  frame=void);
    column x ;

    define x / group noprint
               style={font_size=.1pt   foreground=white   cellpadding=0
               rules=none frame=void};
    break before x / contents='' page;
 run;
 ods pdf close;
```

Thank you to Bari Lawhorn of SAS Institute, Inc. for providing this very creative solution!

USING ‘@’ AS THE ESCAPECHAR WITH LASTPAGE WILL NOT WORK IN PDF OUTPUT

For our legacy reports, post-processing programs added Page X of Y information. To create ODS PDF files, in our reporting programs we produced this information using the THISPAGE and LASTPAGE functions. However, while experimenting with various escape characters, we realized that the LASTPAGE function does not work when ‘@’ is used as the ODS escape character. Please refer to below link for usage note and the possible workaround: http://support.sas.com/kb/38/614.html

INSERTING BREAKS IN TITLES AND FOOTNOTES

Sometimes we needed to add more titles or footnotes than the limit of 10 SAS allows. With ODS, by using line breaks it is easy to have more than 10 titles or footnotes. For more information on how to achieve this, please refer to the following paper from Cynthia Zender:

http://www2.sas.com/proceedings/forum2007/099-2007.pdf

ESTABLISHING REPORT COLUMN WIDTH SETTINGS

For columns defined as numeric types, the SAS ODS PDF algorithm often gives you what you want regarding default column width settings for numeric variables. However, with character columns the algorithm may give you too much space for a column, or too little space so wrapping occurs or 'panels' the report on the page. Since data to display, font size, cell spacing and cell padding primarily are the determinants of column width, you cannot assign fixed widths to the columns and always obtain professional looking reports. We approached the problem by setting up macro variables for each column width and assigning estimated widths. After producing a number of reports we knew that counts and percentages, for example, would require between 10 and 15 millimeters (mm). The leftmost field on the report (ex. system organ class) might take 120mm. A p-value would take 15mm. We ran reports iteratively until we obtained the relatively optimal width settings. It is best to do this with a report whose data distribution is typical for the report. We confirmed with SAS Technical Support that SAS does not provide any way of knowing what the widths for columns are, since styles and data vary drastically. Therefore, when absolute control over column and table width is desired, you are forced to take the iterative route.

To allow your users maximum flexibility with assigning widths you might opt for this strategy: (1) allow SAS or your reporting program to determine default widths and (2) allow the user to override any of the SAS or program defaults. PROC REPORT DEFINE statements for which SAS assigns widths will not have explicit width specifications. An example of the basic macro coding is as follows:

```
%global
          category_dflt_width_ods_pdf
          category_width_ods_pdf
          pgm_sets_dflt_widths
          width_units_ods
;

%let width_units_ods = mm;

%if &pgm_sets_dflt_widths = 1 %then %do;

   %if ( %bquote(&category_width_ods_pdf) = %str() ) %then
      %let category_width _ods_pdf = %str(&category_dflt_width_ods_pdf);

%end;

%if ( %bquote(&category_width_ods_pdf) ne %str() ) %then
   %let category_width_ods_pdf
      = %str(width=&category_width_ods_pdf.&width_units_ods);

proc report …
   define category_col / group order=data
                         style(column)={&category_width_ods_pdf }
   …
run;
```

The programmer starts off development with this code. He or she iteratively runs the program, determines the best default width and assigns the value to CATEGORY_DFLT_WIDTH_ODS_PDF. A metadata table could be used to populate this macro variable.

If the user wants to let SAS define all the widths, the user would just add PGM_SETS_DFLT_WIDTHS (if necessary) to their interface and assign a value of 0. The user would then run the reporting program. The user could do this the first time running the report since it is possible that SAS may produce a report that is acceptable to the user. However, I have found that most report width settings will have to be tweaked.

If the user wants to let the program select the width for a column, the user would add PGM_SETS_DFLT_WIDTHS to their interface, set the value = 1 and not provide an override width.

If the user wants to override the SAS selected width value for column CATEGORY_COL, the user would set PGM_SETS_DFLT_WIDTHS = 0, and add CATEGORY_WIDTH_ODS_PDF to their interface with a non-null value.

If he or she wants to override the program selected width value for the column, he or she would set PGM_SETS_DFLT_WIDTHS = 1 and add CATEGORY_WIDTH_ODS_PDF to their interface with a non-null value.

You can also set up code to assign defaults for labels, formats and justification as follows:

```
%if ( %bquote(&count_lbl_ods_pdf) = %str() ) %then
    %let count_lbl_ods_pdf = n;

%if ( %bquote(&count_fmt_ods_pdf) = %str() ) %then
    %let count_fmt_ods_pdf = cntfmt.;

%if ( %bquote(&count_just_ods_pdf) = %str() ) %then
    %let count_just_ods_pdf = %str(just=center asis=on);

%let count_fmt_ods_pdf = %str(format=&count_fmt_ods_pdf);
```

Finally, to make your application user-friendly, you may want to add the metadata items to some kind of user documentation.

Worth investigating rather than going with this approach to determine relatively optimal widths is 'scaling' your report according to the techniques detailed in the SAS Proceedings paper written by Daniel O'Connor of SAS Institute, Inc., "Zoom, Zoom: Get Your Document to Scale on All Paper Sizes". I did not try the scaling technique mentioned in his article so I don't know how well the technique would always work on producing columns with acceptable widths. The article does mention that the technique combats the problem of 'paneling'.
The focus of O'Connor's scaling techniques is oriented toward fitting tables and graphs on a page from left to right whereas the width specification technique I specified above is oriented to the granular column level. So both techniques might be complementary when used together.

PACKAGING REPORT METADATA FOR YOUR USERS

Since a number of things such as data values, font sizes and the SAS reporting algorithm determine report layout, you cannot assign static values to report column metadata. Assigning widths via macro variables provides your users with capability to modify widths when you are ready to deploy your ODS PDF reporting tool. We used macro variables, too, for other report elements such as titles and footnotes. For a busy report there will be a myriad of report metadata items. For user-friendly packaging, employ a sample page of your PDF report to annotate the metadata items. These items can be grouped in these categories: column widths, column formats, column justification, column labels, and cell value split character. We found you can add all of these items with their values to an annotated report and produce a professional-looking example report. To produce an annotated report, use the "call-out" or annotate tools of Adobe Acrobat. For example, your annotation box pointing to a report column might contain these entries: count_lbl_ods_pdf^n, count_fmt_ods_pdf^cntfmt., count_width_ods_pdf^12 and count_just_ods_pdf^just=d. Values would represent defaults. In an annotation box mentioning the split character we assigned the default value as a vertical bar, '|' (Note: Within values a split character such as '@' does not work for ODS. We converted that character to a hexadecimal '036e'x and thus were able to split the value onto multiple lines.)

You may also want to provide your user, too, with documentation in a Word document or Excel spreadsheet that contains the metadata elements, descriptions and default values.

PREPARING REPORT CODING TO ALLOW PROCESSING OF OTHER ODS DESTINATIONS

At the start of your project, your client may agree to produce just an ODS PDF report and not an ODS RTF report. Don't be surprised if when you are about to go live your client changes their mind and wants to produce an RTF report, too (or even an HTML report). Since syntax and capabilities vary between the types, as you are incorporating your ODS reporting code, put in macro coding for conditionally performing code based on the destination. For example: %if &ods_dest = PDF %then %do; . Also to prevent coding problems and to save lots of debugging time use the BQUOTE and NRBQUOTE functions.

You can add this type of wrapper code around your PROC REPORT code:

```
%let ods_dest_list = %str(PDF RTF);
%do _i = 1 %to &ods_dest_count; /* process ODS destinations */

    %let ods_dest = %upcase(%scan(&ods_dest_list,&_i,%str( )));

   [ Assignment statements from above for setting metadata values for report columns for PDF and RTF]

    PROC REPORT …

%end; /* process ODS destinations */
```

Quite possibly your client may want to be able to conditionally execute the legacy code and so you may have to keep the legacy PROC REPORT code within the same program as your ODS PROC REPORT code.

RECOMMENDED SAS SOFTWARE UPGRADE

More than likely you may find that the solution to a problem may involve applying a hot fix or a maintenance pack. You may find as we often did SAS support recommending this. You may also come across a new problem that you bring to SAS support's attention. Before undertaking the ODS project, if feasible, upgrade your software to the latest maintenance pack and apply all relevant hot fixes. By doing so you will greatly facilitate your conversion effort.

CONCLUSION

Foremost you want to remember that converting your legacy reports to PDF and/or RTF reports presents a marvelous opportunity to advance the goals of the business. Enlist the support of your user community and put in a lot of effort in creating reports that package data presentation most efficiently. Establish report consistency and minimize your programming effort by building reports based on report templates whenever possible. To further reduce such effort, settle on one reporting type of software such as PROC REPORT. Give your development group an advantage and access to more capability by upgrading to the latest SAS release. Spend a lot of time visiting the SAS support and blog sites, and researching on the Web so you can keep abreast of recommended solutions. There is a vast amount of information on the Web about the powerful ODS software and more appears every day. Remember countless others have gone down the same road you are on. Remember if you are stymied and the formulation of your solution is taking too long, contact a fellow developer or give SAS Support a call.

ACKNOWLEDGMENTS

Many thanks to Bari Lawhorn at SAS Institute, Inc. for the many discussions we had with her, all the numerous solutions she provided us, and her editing of this paper. Many thanks to Al Pasquino at Pfizer, Inc. and Shirish Nalavade of eClinical Solutions, Inc. for all their recommendations in producing optimal re-designed reports, and for their assistance with analysis, design and programming.

RECOMMENDED READING

Carpenter, Arthur L., 2007, Carpenter's Complete Guide to the SAS REPORT Procedure, Cary, NC: SAS Institute Inc. Available at:
https://support.sas.com/pubscat/bookdetails.jsp?pc=60966

Gebhart, Eric. 2007. "Paper 225-007: ODS Markup, Tagsets, and Styles! Taming ODS Styles and Tagsets" *Proceedings of the SAS Global Forum 2007 Conference*. Available at:
http://support.sas.com/rnd/base/ods/odsmarkup/Paper_225-2007_tagsets_styles.pdf

Haworth, Lauren. 2005 "132-30: SAS with Style: Creating Your Own ODS Style Template for ODS PDF Output" *Proceedings of the SAS Global Forum 2005 Conference*. Available at: http://www2.sas.com/proceedings/sugi30/132-30.pdf

Lawhorn, Bari. 2011. "Let's Give 'Em Something to TOC about: Transforming the Table of Contents of Your PDF File", *Proceedings of the SAS Global Forum 2011 Conference*. Available at:
http://support.sas.com/resources/papers/proceedings11/252-2011.pdf

Lawhorn, Bari and Huntley, Scott. 2010. “035-2010: Getting the Right Report (Again): Your Compatibility Guide for ODS PDF 9.2”, *Proceedings of the SAS Global Forum 2010 Conference*. Available at: http://support.sas.com/resources/papers/proceedings10/035-2010.pdf

O’Connor, Daniel. 2010. “011-2010: Zoom, Zoom: Get Your Document to Scale on All Paper Sizes”, *Proceedings of the SAS Global Forum 2010 Conference*. Available at: http://support.sas.com/resources/papers/proceedings10/011-2010.pdf

Zender, Cynthia. 2007. "Funny ^Stuff~ in My Code: Using ODS ESCAPECHAR". Technical Papers and Presentations. Cary, NC:SAS Institute Inc. Available at: http://www2.sas.com/proceedings/forum2007/099-2007.pdf

Zender, Cynthia. 2007. "Paper 2603-29 Markup 101: Markup Basics” , SAS Institute, Inc., Cary, NC Proceedings of the SAS Global Forum 2010 Conference. Available at: http://support.sas.com/rnd/papers/sugi29/markup-basics.pdf

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

SAS Discussion Forums:
http://communities.sas.com/index/jspa

CONTACT INFORMATION

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

Benno Kurch
Trading and Software Development, Inc.
P.O. Box 868
Merrifield, Va. 22116-0868
Email: bennfran17@yahoo.com