Paper 056-2007

Seven Steps to Regulatory Publication Style With Proc Report

Dennis Gianneschi, Amgen Inc., Thousand Oaks, CA

ABSTRACT

This paper outlines SAS® 9.1.3 coding steps to produce output that meets "Regulatory Publication Style". This style is specific to my clinical trial reporting environment and is composed of requirements coming from two separate sources. The first source is an internal company document called the *Style Guide for Regulatory Submission,* which all output submitted to a regulatory agency must follow. The second source is the publishing software itself, which processes all output into the regulatory filing. Examples of style guide requirements are: RTF output, Arial font, and that footnotes must attach to the bottom of the table without a box drawn around them. Publishing system requirements are: a) The user controls the type of line breaks, either soft, {\line}, or hard, {\par}, in the title, which controls how much of the title is picked up by the publishing software and put in the table of contents. b) Each line, such as a spanning underline or the line between the table body and the footnotes, must have a minimum thickness of 19 twips. c) Section breaks are not allowed. d) Table title and footnotes cannot be in the header or footnote section of the RTF document. e) Table title must be "free text" on the page and not in a table structure.

The seven steps to achieve this style are: 1) proc template, 2) statistical procs (optional), 3) reformat data (optional), 4) add pagination and footnotes, 5) proc report, 6) post process RTF file, and 7) View in WORD® (optional).

The steps utilize features of ODS style templates, proc report, specific RTF codes, the macro language, and SAS data step programming. A simple, easy to follow coding style is used and consideration was given to efficient use of computer resources. The steps discussed here are limited to producing a report where text wrapping in the report is predictable and the number of rows per page is constant. For example, the code could not produce a correctly paginated adverse event listing, where wrapping of long text strings makes the number of rows per page nonconstant. Even with the constant rows per page limitation, the steps can produce a wide variety of useful listings and statistical tables for regulatory filings, ad hoc analysis, exploratory analysis, or validation reporting.

INTRODUCTION

Highly stylized RTF output with proportional fonts roughly doubles the amount of code required to produce an output compared to plain text or HTML output. Proportional fonts makes size determination and therefore pagination difficult. WORD has its own rules for processing the RTF which must be understood in order to correctly anticipate content based pagination. Given all these technical challenges, some companies still have a policy of producing plain text output which is converted to a fixed font Courier in a WORD document. My company has pursued RTF output with great passion for the last ten years because we see it as a way to dramatically improve the readability and usability of the output, as well as the quality of a regulatory filing. Clinicians and medical writers prefer the RTF table structure readability features such as table centering, text centering, decimal alignment, and alignment of footnotes with the table. These features provide the clearest possible presentation of the data. RTF usability features include the ability to integrate the outputs with text in WORD for a study report, internal presentation, or publication. The quality and speed of compiling the filing is improved because outputs can be copied among different sections due to the consistent format of statistical tables found in summary documents, individual study reports, and appendices.

With the release of 9.1.3, much less post processing of the RTF output file from proc report is required to meet the "Regulatory Publication Style". The author hopes that these coding steps can translate and be useful in your computing environment. The step by step approach provides the opportunity to add, delete, or modify a step to meet your specific requirements. For example, the post processing translates section breaks to page breaks in the RTF file that proc report writes out. These section breaks interfere with my publishing software's function of inserting a running header and footer. In your computing environment, using a different publishing system, you may not need the post processing step. Steps may be added to handle content based pagination or multi-panel reports. SAS macro language, .NET® , or Java™ with SAS Integration Technologies may be used to automate the steps. Example A below shows a simple listing where steps 2 and 3 were not needed. To produce a statistical table, where SAS procs are called, steps 2 and 3 would be needed. Example A has been reviewed and approved my company's medical writers and processed successfully by our regulatory publishing software.

STEP #1 – PROC TEMPLATE

Step 1 uses proc template to create a style template to implement as many *Regulatory Publication Style* elements as possible. Here is Example A without the style template specified, using the default SAS 9.1.3 RTF style.

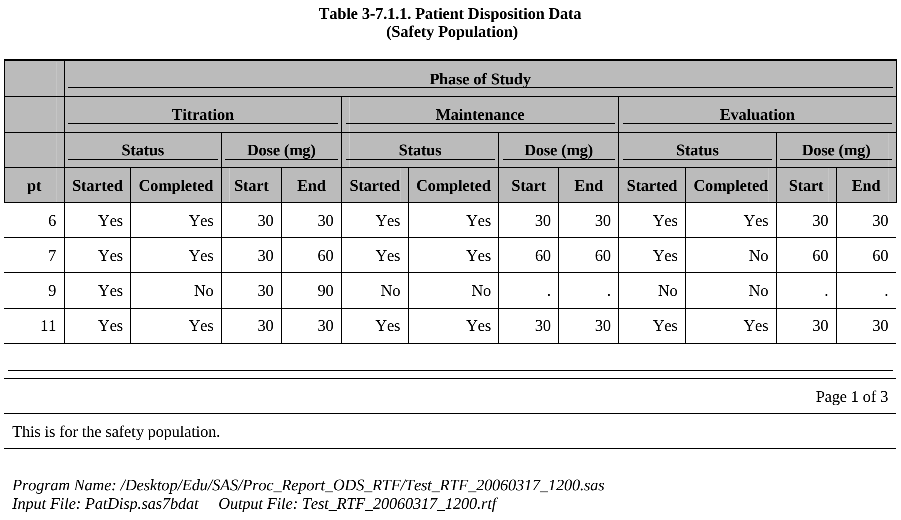

Table 3-7.1.1. Patient Disposition Data
(Safety Population)

	Phase of Study											
	Titration				Maintenance				Evaluation			
	Status		Dose (mg)		Status		Dose (mg)		Status		Dose (mg)	
pt	Started	Completed	Start	End	Started	Completed	Start	End	Started	Completed	Start	End
6	Yes	Yes	30	30	Yes	Yes	30	30	Yes	Yes	30	30
7	Yes	Yes	30	60	Yes	Yes	60	60	Yes	No	60	60
9	Yes	No	30	90	No	No	.	.	No	No	.	.
11	Yes	Yes	30	30	Yes	Yes	30	30	Yes	Yes	30	30

Page 1 of 3

This is for the safety population.

Program Name: /Desktop/Edu/SAS/Proc_Report_ODS_RTF/Test_RTF_20060317_1200.sas
Input File: PatDisp.sas7bdat   Output File: Test_RTF_20060317_1200.rtf

Here is Example A using the `“US_Letter_Landscape_10pt”` style template.

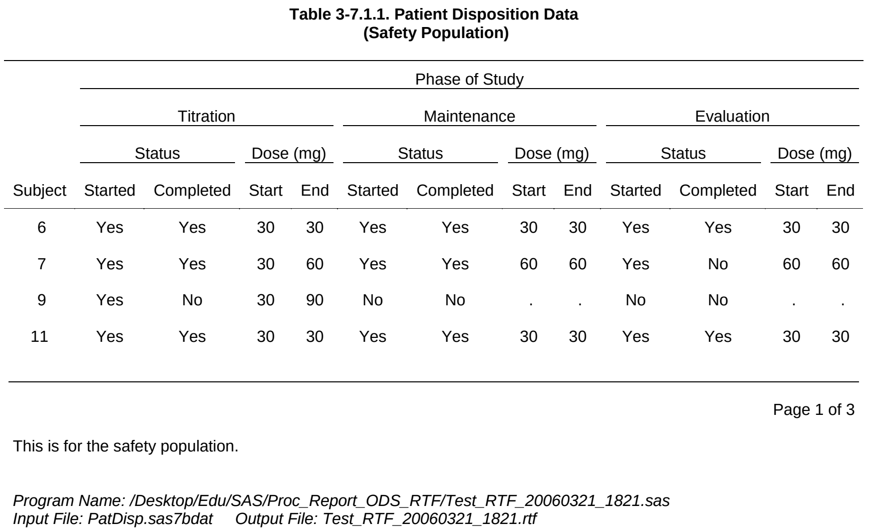

Table 3-7.1.1. Patient Disposition Data
(Safety Population)

	Phase of Study											
	Titration				Maintenance				Evaluation			
	Status		Dose (mg)		Status		Dose (mg)		Status		Dose (mg)	
Subject	Started	Completed	Start	End	Started	Completed	Start	End	Started	Completed	Start	End
6	Yes	Yes	30	30	Yes	Yes	30	30	Yes	Yes	30	30
7	Yes	Yes	30	60	Yes	Yes	60	60	Yes	No	60	60
9	Yes	No	30	90	No	No	.	.	No	No	.	.
11	Yes	Yes	30	30	Yes	Yes	30	30	Yes	Yes	30	30

Page 1 of 3

This is for the safety population.

Program Name: /Desktop/Edu/SAS/Proc_Report_ODS_RTF/Test_RTF_20060321_1821.sas
Input File: PatDisp.sas7bdat   Output File: Test_RTF_20060321_1821.rtf

The style template is one of many generated from a SAS macro program containing proc template code. Other style templates support portrait orientation and European paper sizes. The template sets the following *Regulatory Publication Style* elements: paper size, orientation, font, dividing lines, and margins. These style elements are encoded in the template name. For example the template "EU_A4_Landscape_9pt" would produce 9pt size output in the body of the table, in landscape orientation, on European A4 size paper. Other templates available are US standard letter size paper: US_Letter, _9pt, _10pt, and _11pt. European standard A4 paper size: EU_A4, _9pt, _10pt, and 11pt. The remainder of the *Regulatory Publication Style* elements are achieved in steps 4, 5, and 6 through data step processing, ODS, RTF, and proc report coding. Note that it is not possible to set the running headings and footings with proc template and these are critical for *Regulatory Publication Style*. Running heading and footing are set below with the proc report parameters `headery` and `footery`.

STEP #2 – CALL STATISTICAL PROCS (OPTIONAL)

Step 2 is optional and calls the SAS statistical procedures of interest, using ODS output file destination to capture the statistics in a dataset that proc report can display. Use Irene Zhao's method of normalizing the dataset structure of all statistical procedures provide a consistent input to the downstream reformatting data step. Irene has a consistent structure and naming convention for all key variables, row labels, and statistics. Output from any statistical procedure can be easily stacked with any other proc and input to the next step.

STEP #3 – REFORMAT DATA FOR DISPLAY (OPTIONAL)

Step 3 is optional and uses SAS data step programming or possibly a proc transpose to reformat the data to display format. For listings, no reformatting is required. For statistical tables, reformat to treatment column display format.

STEP #4 – ADD PAGINATION AND FOOTNOTES

Step 4 is required and uses SAS data step programming to add pagination and control variables to the dataset. These variables are specified in a particular order in the BY and COLUMN statements of the proc report code discussed in the next section. Each control block is controlled by one variable and has only one style available. Several rows, with different styles, justification and font, are needed at the bottom of the table. A key finding is that by assigning the value of _Page to other variables and arrange them in the appropriate order in the COLUMN and By statements, the compute blocks can be activated in the correct order. The variable names begin with an underscore to avoid collision with existing dataset variables.

Pagination and Footnote Compute Block Variables

Item	Name	Description
1	`_N_Detail_Lines`	The constant number of rows to display on each page
2	`_Abs_Line`	The absolute, unique row number of each row in the report
3	`_Line`	The relative row number of each row in a page
4	`_Max_Page`	The maximum number of pages. Value is correct on the last obs.
5	`_User_Foot`	Assigned the same value as `_Page`. Must be a different variable to make the compute block print out the user footnotes.
6	`_Bottom_Line`	Assigned the same value as `_Page`. Must be a different variable to make the compute block print out the bottom line.

STEP #5 – PROC REPORT

Step 5 calls proc report. The code is devoid of any SAS statements like `title` and `footnote` which violate *Regulatory Publication Style*. Titling is done in the proc report statement with the following code: `pretext="\fs22 \b &Titles. \b0 {\line}"`. Pagination and footnotes are attached to the bottom of each table on the page by `compute` blocks. The SAS macro language is used to code macro variables which hold long text strings that clean up the downstream proc report code, making the code easier to read.

Macro Variables to Improve Proc Report Code Readability

Item	Name	Description
1	`Titles`	Titles with RTF to control multiple lines.
2	`User_Foots`	User footnotes with RTF to control multiple lines.
3	`Std_Foot1`	First standard footnote.
4	`Std_Foot2`	Second standard footnote.
5	`Span`	"\brdrb\brdrs\brdrw1" is the RTF code to underline. Proc report supports any number of levels of spanning text by using the parenthesis operators ( ) in the `column` statement.

The following are ODS RTF statement parameters are critical for *Regulatory Publication Style*.

ODS RTF Statement Parameters

Item	Name	Description
1	`Headery=1080`	Set running heading to .75" in twips (1440 twips per inch).
2	`Footery=360`	Sets running footing to .25" in twips.
3	`NOTOC_DATA`	Prevents table of contents information to be written into the first column of the table.

The following are ODS style elements that are set in proc report.

ODS Style Elements Settings in Proc Report

Item	Name	Description
`Style(report)`		
1	`asis=on`	Specifies that leading spaces and line breaks will be honored allowing for user controlled indentation.
2	`protectspecialchars=off`	Allows user to pass RTF codes through proc report into the RTF output file to be processed by WORD. Otherwise, these codes would displaying it as text.
3	`outputwidth=9.25 in`	This is a good setting for landscape. Interacts with the cell widths to adjust cell width for the column headings. If omitted, only the cells widths are used and heading width is ignored.
4	`pretext="\fs22 \b &_Title. \b0 {\line} "`	Passing in titling information here instead of titles avoids putting hard returns in your titles, which cause problems for medical writing.
`Style(header column)`		
5	`rules=groups`	Produces two spanning lines across the entire report. One above the column heading and another below.

An important feature of the `column` statement is the nesting and spanning operators `()` , the left and right parenthesis.

Nested Column Labeling Code in Example A and Explanation

```
("&Span. Phase of Study"
        ("&Span. Titration"   ("&Span. Status"    (I_strt_1 I_cmpl_1))
                              ("&Span. Dose (mg)" (DS_str_1 DS_end_1))) … )
```

&Span is a macro variable containing the RTF code that will draw the underline below the "Phase of Study" text which will span across all of display type variables. The nested parenthesis and position of the spanning text and underlining provide a clear message of structure to the end user of the information in the table. For example, "Status" is nested within "Titration", nested within "Phase of Study" providing the reader with the best visual information clues graphically, without the redundant clutter of words that might be necessary in the column headings otherwise.

Lets look at some of the `define` statements used in the proc report.

Example A – `define` Statement Code and Explanation

```
    define _Page     / order   noprint;

    define PT       / order   style=[cellwidth=.6 in just=c];

    define I_strt_1 / display style=[cellwidth=.6 in just=c];
```

_Page is defined as an order variable so that the compute block code can be triggered by a break statement. The PT variable is an order variable providing a row label where the first value is displayed and successive rows with the same value are blanked, similar to a statistical table structure. The `cellwidth=.45 in` is required for tables with more than 9 columns. By default, columns are dimensioned with a with of 1.0 inch. With 9.25 inches of available width on the US_Letter_Landscape_10pt page, proc report, by default, put out 9 columns on the first page and the other columns on the next page expanding the column width to fill the page width. Since this table has 13 columns on one page, we have to specify column width so that all of the columns fit on one page. Use the calculation width=9.25/#cols and round down to the nearest 10th of a decimal. For this table it worked out to .7 in for each column, resulting in a total of 9.1 inches. The "Completed" column label would not fit into a .7 inch column without wrapping so further adjustment had to be made. The result was most of the widths were set to .6 inches. So for a table with more than 9 columns, there is some necessary trial and error to get the heading text to wrap correctly.

Report Compute Blocks
Compute blocks draw the line at the bottom, display page number, user and standard footnotes. All must be controlled by different order variables in the dataset. The by variable and order in of `compute` block variables the `column` statement is critical. `by _Page; column _User_Foot _Page _Bottom_Line _Abs_Line PT …`

Example A – `Compute` Code and Description

<table>
<tr><th>Item</th><th>Code</th><th>Description</th></tr>
<tr><td>1</td><td>compute after _Bottom_Line /
style={protectspecialchars=off};
    line "&Span";
endcomp;</td><td>Draw the line at the bottom spanning the entire report.</td></tr>
<tr><td>2</td><td>compute after _Page /
    style={just=r font_size=10pt};
    Page_Count = "Page "||trim(left(put(_Page,
4.)))||" of &Max_Page.";
        line Page_Count $;
endcomp;</td><td>Display page numbering at the right, below the bottom line.</td></tr>
<tr><td>3</td><td>compute after _User_Foot /
    style={just=l font_size=9pt};
    line "&User_Foot1.";
endcomp;</td><td>Display user footnotes (optional). This compute block can be removed if there are no user defined footnotes.</td></tr>
<tr><td>4</td><td>compute after _page_ /
   style={just=l font_size=9pt
          font_style=italic };
       line " ";
        line "&Std_Foot1.";
        line "&Std_Foot2.";
endcomp;</td><td>Display standard administrative footnotes with _page_ automatic break variable. Writes information immediately before or after the table while still attached to the table.</td></tr>
</table>

STEP #6 – POST PROCESS RTF FILE

Step 6 reads in the RTF file produced by proc report and translates lines with a thickness of 15 to a thickness of 19 twips using `tranwrd(record, "\brdrw15", "\brdrw19");` This width does not vanish when tables are processed by the regulatory software. The only other function is to replaces section breaks with page breaks using `tranwrd(record,'\sect\sectd\','\page\');`

STEP #7 – VIEW IN WORD (OPTIONAL)

View in WORD. Hit the ¶ button form the standard formatting toolbar, to view hidden formatting. You should not see hard returns in the titles, table of content information in the first column of the table, or section breaks. These violate *Regulatory Publication Style*. Verify that the margins are correct, including the running headings and footings. Use the *File, Page Setup,* pull-down menu, and choose the *Margins* tab. Once the code is working, producing *Regulatory Publication Style*, this step is optional.

CONCLUSIONS

This paper outlined SAS 9.1.3 coding steps to produce output that meets my company's specific requirements that have their sources in medical writing style and regulatory publishing software. The steps consist of straightforward coding techniques and efficient use of computing resources. A wide variety of listings and tables is possible where pagination is predictable. The steps provide a framework from which to add, delete, or modify to meet your specific output requirements.

REFERENCES

Zhao, Irene. 1997. "Generic Methodology in Report Building by Using SAS@ Macros ." http://www2.sas.com/proceedings/sugi22/CODERS/PAPER84.PDF

Microsoft's reference material for RTF: http://msdn2.microsoft.com/en-us/library/aa140277(office.10).aspx
Additional reference for RTF codes: **http://www.biblioscape.com/rtf15_spec.htm**

ACKNOWLEDGMENTS

Thanks to James M. Kirkpatrick, David Edwards, Roger DeAngelis, Michael Hagendoorn, and Jim Johnson for their contributions to the coding techniques presented here. Thanks to Jack Schoemperlen, George Li, and Dana Soloff for their IBS-EM 13 support. Thanks to Randy Poindexter of SAS Technical Support for some key RTF codes.

CONTACT INFORMATION

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

Dennis Gianneschi
Amgen, Inc.
Mail Stop 24-2-C
One Amgen Center Drive
Thousand Oaks CA 91320-1799
E-mail: dgiannes@amgen.com