

Skinning the Cat This Way and That: Using ODS to Create Word Documents That Work for You

Elizabeth Axelrod, Abt Associates Inc., Cambridge, MA

David Shamlin, SAS Institute Inc., Cary, NC

ABSTRACT

By supporting formats like Rich Text Format (RTF) and HyperText Markup Language (HTML), the SAS Output Delivery System (ODS) makes turning SAS output into Word documents easy. Tabular output from SAS procedures can be sent directly to a file that Word can read. However, the resulting Word document might fall short of your expected results.

For example, we needed to create a Word document that would serve as a user guide (or codebook), describing over 400 variables in a survey study. The body of the document needed to include detailed information for each variable. The Table of Contents needed to list the page numbers for the sections including the detailed information for each variable.

As a result, we experienced several formatting challenges. Long values in section titles did not wrap correctly, and there was too much white space between sections, to name a few.

In the course of creating a solution, we discovered a number of useful ODS tricks when creating a Word document using SAS output. We researched ways to automate the creation of the document and to minimize the amount of manual formatting in Word.

INTRODUCTION

With SAS today, you are no longer limited to using the standard output tables generated by PROCs, or to customizing text reports by using PUT statements. Using SAS ODS, you have many options to create appealing tables and reports. By using the right ODS destinations, you can directly view SAS output using third-party tools such as Word. However, there are differences in the structure of SAS output and the creation of Word documents that you might want to control. Formatting information on a page is an example. SAS has primitive concepts of pagination, while Word has sophisticated paging capabilities. Such issues can be addressed by intentionally directing a SAS program to construct output that can be maximally leveraged by the target viewing application - in this case, by Word.

This paper describes a project that takes advantage of the text-processing strengths of SAS ODS, Visual Basic for Applications (VBA) scripts, and Word to produce a highly structured Word document.

DESCRIPTION OF THE PROJECT

Every quarter, from a large ongoing survey, survey data from clients is collected in a SAS data set. Each variable in the survey data is analyzed with a simple frequency, and a report (codebook) is generated. The codebook is comprised of two sections: the body and a Table of Contents. The body includes brief, detailed information for each of the approximately 400 variables (for example, variable name, variable label, and frequency table); this variable information is internally referred to as a "variable block." The Table of Contents provides page numbers for the variables (a sample from each of these sections follows).

Before SAS ODS, the body of the codebook was easily generated by a large SAS program. The output was an unformatted text report that could be read by Word. The results were not elegant and required extensive editing. Worse than that, the Table of Contents was created manually.

Therefore, we initialized a project to use the flexibility of SAS ODS to produce a better-formatted codebook whose body included the markup language necessary to enable Word to automatically generate the Table of Contents.

Table of Contents

Name	First Name.....	1
Sex	Gender.....	1
Age	Age in Years.....	1
Height	Height in Inches.....	1
Weight	Weight in Pounds.....	2
Alphabetic Listing by Variable Label.....		3

Sex

Gender

Type: Character

<u>Frequency</u>	<u>Percent</u>	<u>Response</u>
9	47.37	Female
10	52.63	Male

Age

Age in Years

Type: Numeric

<u>Frequency</u>	<u>Percent</u>	<u>Response</u>
19	100.00	11-18

APPROACH

During our research, we found techniques for customizing SAS ODS to generate output that met most of our requirements. We explored methods for integrating the SAS language with Word's automation features. The end result that we were researching was to make SAS ODS do most of the heavy lifting to generate the body of the codebook¹, but to rely on Word to produce the Table of Contents.

To turn the input data set into a document with the appropriate contents and format that we wanted, the original SAS code was abandoned and rewritten in a simpler form, using ODS. Certain facets of generation and formatting were delegated to Word. The most significant use of Word was to generate a Table of Contents. We used Word for this step for several reasons. The ODS Table of Contents feature uses SAS procedure names to title the sections of the document. However, in our codebook, variable block titles are based on variable names and labels and Word can accommodate this. In addition, certain formatting issues can be handled better in Word than in SAS, which is biased toward generating table-based reports.

GETTING STARTED

SAS ODS supports destinations that Word can open directly such as RTF and HTML. These destinations produce RTF and HTML markup.

Markup refers to control codes embedded in a file that dictate how the file should be formatted when opened by an application that understands markup control codes. For example, Web browsers such as Internet Explorer and Netscape are applications that are commonly used to read HTML files, while many Microsoft applications use RTF as their standard markup language. Microsoft Office applications such as Word and Excel can read both RTF and HTML.

Because Microsoft Office applications can read both RTF and HTML, SAS programmers have two good choices when using ODS to turn SAS output into Word documents.

The ODS RTF destination supports horizontal measurement. This means that, when ODS sends output to an RTF file, it can intelligently format columns of information based on the width of the physical page. Options related to the horizontal placement and spacing of cells (for example, the CELLWIDTH= option) are supported by the ODS RTF destination, giving users a significant amount of control over the format of

¹ A similar approach is described by Hadden (2003).

columns. (Support for vertical measurement is underway.)

The ODS HTML destination does not understand horizontal or vertical measurement. Cell placement and spacing options are ignored. However, the ODS HTML destination is based on tagsets, making it extremely configurable. ODS provides tools for extending and modifying tagsets that ship with SAS software or for creating new tagsets as needed. With ODS HTML, it is possible to produce extensively customized reports.

The SAS code for creating output that Word can read is simple. The following code fragment can be used as a pattern:

```
ods DESTINATION file='PATH\FILE.doc';  
  
/* INCLUDE SAS REPORT CODE HERE */  
  
ods DESTINATION close;
```

where *DESTINATION* can be replaced with *RTF* if you want an RTF file or with *HTML* if you want an HTML file. You must specify an appropriate pathname and filename for your output file. By specifying a file extension of .doc, the Windows operating environment associates the output file with Word, and the file generated by SAS is treated as a Word document by Windows.

In our project, this ODS technique is the basis for creating Word documents from SAS code. The remainder of the project work involves creating a customized Table of Contents.

CREATING THE BODY OF THE CODEBOOK

No single text-processing technique can do everything. However, the following tools, when applied in order, put us on a direct path to all of our desired results:

- The TEMPLATE procedure
- Word styles
- Word template
- Word macros

Consequently, these are the steps we took.

1. Run a batch SAS job to generate the body of the codebook (all variable blocks) and to create an RTF output file.
2. Insert the RTF file into a Word document using a user-defined Word template (.dot) file.
3. Run a series of user-defined Word macros to apply additional formatting.
4. Perform a quick, manual review of the body and insert any needed page breaks.

Each of these steps is described in detail.

1. Run a batch SAS job to generate the body of the codebook (all variable blocks) and to create an RTF output file.

In the first section of the batch SAS job, PROC TEMPLATE is used to tailor a SAS style to be more consistent with styles required by our codebook. This code begins with the SAS default style called RTF; modifies spacing between lines of tables, margins, and fonts; and removes table borders. The RTF output is now ready to support many of the formatting requirements of our codebook.

```

*****;
* CREATE A STYLE TEMPLATE (rtfSQUISH), based on the parent style RTF.
* Set margins, remove table borders, set background color. This template
* will be placed in the default template store.
*****;
proc template;
  define style rtfSQUISH;
    parent=styles.rtf;
    style Table from Output /
      leftmargin    = .5in
      frame         = void
      rules         = none
      cellpadding   = 2pt
      cellspacing   = 0
      borderwidth   = 0
    ;
    style Header from HeadersAndFooters /
      background=#FFFFFF
    ;
  replace fonts /
    'Titlefont'    = ("Arial",12pt,Bold)/* system titles & footers */
    'Titlefont2'   = ("Arial",12pt)    /* system titles & footers */
    'docfont'      = ("Arial",9.5pt)   /* data values in tables */
    'StrongFont'   = ("Arial",11pt)    /* row headers */
    'FixedStrongFont' = ("Arial",11pt)
    'EmphasisFont'  = ("Arial",12pt)
    'FixedEmphasisFont' = ("Arial",13pt)
    'HeadingFont'   = ("Arial",11pt)
    'HeadingEmphasisFont' = ("Arial",15pt)
    'FixedHeadingFont' = ("Arial",16pt)
    'BatchFixedFont' = ("Arial",17pt)
    'FixedFont'     = ("Arial",18pt)
  ;
end;
run;

```

The following code generates the body of the codebook (all variable blocks). We want each variable block heading to contain the variable name and the variable label. Using SQL and dictionary tables, we are able to capture these elements and save them into macro variables for subsequent use.

```

proc sql noprint;
  select  name,
         type
  into    :names separated by '#',
         :types separated by '#'
  from    DICTIONARY.COLUMNS
  where   libname = upcase("&libname")
  and     memname = upcase("&FNAME")
  and     memtype = "DATA"
  ;
quit;

```

To capture the number of observations and variables, a similar procedure is run using the Dictionary Table instead of the Dictionary Columns. Although it is possible to create a macro variable that holds all the variable labels, the labels in our data set were long and, when combined, exceeded the allowable length of a macro variable. Therefore, we used the CALL LABEL statement to capture each variable label. In the following code, we cycle through each variable in the file, run a frequency, and print the results. The title for each PROC PRINT statement becomes the variable block heading in the codebook.

```

ods listing close;
options nodate nonumber;
ods noproctitle;
ods rtf file=".\&rtffile"

```

```

startpage=no
style=rtfsquish bodytitle ;

%macro freqs;
  %do i=1 %to &nvar;
    %let name = %scan(&names, &i, '#');
    %let type = %scan(&types, &i, '#');
    %if &type = num %then %let type=Numeric;
    %else %let type=Character;

    data _null_;
      length labit $233;
      set &FNAME (keep=&name obs=1);
      call label(&name, labit);
      call symput('label', labit);
    run;
    title1 j=left "&name" '09'x "&label";
    title2 j=left '09'x "Type: &type";

    proc freq data=&fname (keep=&name) noprint;
      tables &name/missing out=freq;
    run;
    proc print data=freq noobs label;
      label count = 'Frequency'
            percent = 'Percent'
            &name = 'Response';
      format percent 6.2;
      var count /style=[just=r leftmargin=1.425in];
      var percent /style=[just=r];
      var &name /style=[just=l leftmargin=.3in];
    run;
  %end;
%mend freqs;
%freqs

ods rtf close;

```

'09'x is the ASCII code for a tab.

Align column text exactly where you want it.

Frequency results are displayed using PROC PRINT, letting us apply specific formatting required by the client. Even using PROC PRINT, the results are displayed in a table and additional formatting is minimal. If a table splits over multiple pages, column headings are automatically repeated on the next page. If you add or remove variable blocks, change the font, or alter the document such that it re-paginates, column headings will display correctly at the top of each page.

2. Insert the RTF file into a Word document using a user-defined Word template (.dot) file.

If we open the RTF file with Word, Word uses its default template file (normal.dot), a file that contains Word styles and default formatting (tabs, margins, etc.). The resulting document will not contain styles and formatting required in our codebook. By creating a custom, user-defined Word template and storing it as a .dot document, we were able to get better results. This enabled us to establish heading and body styles, pre-set margins and tabstops, and document headers and footers. The tabs ('09'x) that we inserted in the TITLE statements in the SAS code expand to our defined tabstops, based on the custom Word template.

So far, we have used SAS, PROC TEMPLATE, the ODS RTF destination, and a user-defined Word template and are close to what we want in the codebook. We still need to:

- Properly indent long variable names.
- Apply Word styles to headings.
- Insert horizontal lines between variable blocks.
- Control vertical white space (blank lines) between variable blocks.

These and other challenges can be handled by using Word macros.

3. Run a series of user-defined Word macros to apply additional formatting.

Word macros are useful and easy to create when you use the macro recorder. To make macros more powerful, you can edit the VBA code that is generated in the background. Multiple commands can be embedded in a VBA loop.

We created Word macros to address the outstanding formatting challenges listed above. Further discussion of VBA and Word macros will be presented later in this paper.

4. Perform a quick manual review of the body and insert any needed page breaks.

Some final editing and a brief review are needed. User-inserted page breaks can improve readability and the overall layout of the codebook. After this is done, we can work on the Table of Contents.

GENERATING A TABLE OF CONTENTS

Even though the Table of Contents in our codebook has requirements that your document might not have, the techniques applied and lessons learned in refining a Table of Contents are useful. Note that SAS ODS can automatically generate a Table of Contents. This feature is implemented for ODS destinations that are based on HTML or MARKUP tagsets. When the CONTENTS= option is specified with an HTML- or MARKUP-based ODS destination, SAS generates Table-of-Contents information for the report. The sections included in the Table of Contents are based on SAS procedures generating output. Sections are labeled with SAS procedure names and tables that those procedures generate. An example using PROC FREQ is shown in the next figure:

The screenshot shows the SAS Output Frame window. On the left is a Table of Contents with the following structure:

- 1. The Freq Procedure
 - Table Eyes
 - One-Way Frequencies
 - Binomial Proportion
 - Binomial Proportion Test
 - Table Hair
 - One-Way Frequencies
 - Binomial Proportion
 - Binomial Proportion Test

On the right is the main report content, titled "Hair and Eye Color of European Children". Below the title is "The FREQ Procedure". The main table is titled "Eye Color" and has the following data:

Eyes	Frequency	Percent	Cumulative Frequency	Cumulative Percent
brown	341	44.75	341	44.75
blue	222	29.13	563	73.88
green	199	26.12	762	100.00

Below the table is a section titled "Binomial Proportion for Eyes = brown" with a value of 0.4475.

When using an ODS-generated Table of Contents, remember that ODS generates three separate output files. One file is for the report content (seen on the right of the preceding figure); one file is for the Table of Contents (seen on the left); and one file merges the two files into a single view. This might be acceptable if the output files are available from a common storage location like that which network files share, but if they are distributed electronically, routing multiple files can be cumbersome.

Also remember that entries in the Table of Contents are linked to places in the body with hyperlinks. This is helpful for documents that will be viewed electronically, but not helpful for hard-copy documents.

Finally, subject lines in the Table of Contents are based on SAS procedure names and tables that those procedures generate. This approach might not be suitable for your document. Instead, it might be better to trade the SAS specific titles and labels for titles and labels that are more closely related to the business meaning of the report content (which is what our project required).

In SAS 9.1, the ODS RTF destination includes two Tables of Contents experimental features. The first feature, similar to the CONTENTS= option, generates sections based on SAS procedure names and tables that those procedures generate. The syntax for this feature is:

```
ods rtf file='PATH-NAME\FILE-NAME.rtf' contents;

/* INCLUDE SAS REPORT CODE HERE */

ods rtf close;
```

Important: The Table of Contents is a Word object with integrated pagination information. This means that the page number on which the section appears is included in the Table of Contents. This experimental feature requires post-processing in Word. After the SAS code has executed, the Table of Contents object must be populated by opening the Word document and executing the update field command.

The second feature (experimental in SAS 9.1) enables you to create a custom Table of Contents in your output. This feature uses the ODS WORDSTYLE=, PREPAGE=, and ESCAPECHAR= options. The following code demonstrates how these options are used:

```
ods rtf file='PATH-NAME\FILE-NAME.rtf' wordstyle='(\s1 Heading 1;);'
ods escapechar='\';

ods rtf prepage='\R/RTF"\s1 " SECTION 1 TITLE';
/* INCLUDE SAS REPORT CODE HERE */

ods rtf prepage='\R/RTF"\s1 " SECTION 2 TITLE';
/* INCLUDE SAS REPORT CODE HERE */

ods rtf close;
```

The WORDSTYLE= option associates the RTF markup code \s1 with Word's Heading 1 style. The ESCAPECHAR= option tells ODS to interpret the backslash character as an escape character when generating the RTF file. The PREPAGE= option works like a TITLE statement, causing ODS to write the text value as a page heading in the output file.

The code string begins with control codes that cause RTF to interpret the text value as Word's Heading 1 style.

The text following these control codes, SECTION *n* TITLE, is used as title text. Remember, after this SAS code is executed, the resulting RTF output file must be opened with Word to insert and populate a Table of Contents object. This is done manually by positioning the cursor where the Table of Contents should appear, and selecting **Insert**, then **Index and Tables**, and then **Table of Contents**. The Table of Contents object generated by this technique differs from the previous RTF technique that is described in that the Table of Contents entries are based on Word's heading styles. In the other RTF technique, Table of Contents entries are based on RTF field codes. This is important to note because the two techniques cannot be used together to create a single Table of Contents.

A similar and syntactically simpler technique is possible with the ODS HTML destination. This technique is possible because of the way Word interprets HTML tags. When reading HTML files, Word translates:

- text surrounded by HTML <h1> and </h1> tags as Heading 1 style text
- text surrounded by HTML <h2> and </h2> tags as Heading 2 style text, etc.

Because Word uses text that is set with heading styles to build the Tables of Contents, we can imbed HTML heading tags in character values that SAS outputs using the ODS HTML destination.

For example, assume that you want to label report sections with text that Word formats with the Heading 1 style. TITLE statements in SAS are easily used to label report sections. By bracketing the text of each TITLE statement with <h1> and </h1>, Word will apply the Heading 1 style to the TITLE statement text when it reads it. Following is an example:

```
ods html file="h1_example.doc";

title "<h1>Contents of SASHELP.CLASS</h1>";
proc contents data=sashelp.class;
run;

title "<h1>Contents of SASHELP.AIR</h1>";
proc contents data=sashelp.air;
run;

ods html close;
```

Opening the h1_example.doc file with Word and inserting a Table of Contents by selecting **Insert**, then **Index and Tables**, and then **Table of Contents** results in a Table of Contents with TITLE statements values as entries.

AUTOMATING WORD TASKS

Any task performed by using Word's user interface can be automated by using Word macros. Automating tasks is more than a convenience; it is a quality assurance measure, which ensures that the task is performed consistently each time a report is generated.

Word macros are implemented with VBA. VBA is tailored for automating Microsoft Office applications. While VBA programming specifics are outside the scope of this paper, it is easy for the novice to begin using VBA to automate Word tasks. By selecting **Tools**, then **Macro**, then **Record New Macro** in Word, you can create a macro by recording actions that are performed dynamically through Word's user interface. The sequence of actions is played back automatically by invoking the macro. In addition, Word macros can contain conditional logic (for example, if-then/else and loop constructs). If you are interested in creating macros, there are many publications about Word macro programming and VBA.

Stetz (2000) describes a procedure for invoking VBA macros from a SAS job. In short, Stetz shows how to invoke Word and call a generic VBA macro from a SAS program. The generic VBA macro then calls a task-specific macro that is generated dynamically by the controlling SAS code. This technique can be used to tightly couple SAS and Word functionality into a streamlined, automated process that requires little to no user intervention. We will provide a working example of this methodology in supplemental material that will be downloadable from the online SAS Customer Support Center (http://support.sas.com/rnd/base/index_papers.html) by the opening session of SUGI 29.

CONCLUSION

SAS is an exceptional tool for analyzing data and producing reports. The ODS facility extends report writing capabilities by creating documents that can be viewed by other applications such as Word. The ODS RTF and HTML destinations are well suited for use with Word. ODS RTF understands horizontal measurement, which allows for intelligent placement and spacing of columns. ODS HTML is highly customizable, which gives the report writer greater flexibility in the overall structure and content of the output.

It should be noted that the different versions of SAS handle various RTF and HTML statements differently. For example, the code presented in this paper to produce the body of the codebook is run on SAS Version 9.0, but behaves differently under version 9.1.

The default capabilities of ODS form a suitable foundation for any report, but sometimes the default output must be altered or extended. ODS destinations accommodate varying degrees of customization; any further customizations can be handled by using Word's formatting features. For convenience or quality assurance, document modifications or enhancements through Word's interface can be automated by using VBA macros. VBA macros can be directly invoked by using SAS code, thereby furthering the amount of automation possible.

In general, data analysis that is performed by SAS can be turned into a conventional Word document with Word's standard attributes and characteristics. This is desirable for the information consumers who have Microsoft Office deployed on their desktop system. As shown in this paper, there is an abundance of report creation power shared between SAS software and Microsoft Office, and this power can be used collaboratively to leverage the best of both environments in the production of documents.

REFERENCES

Hadden, Louise. "PROC DOC and Beyond: Is PROC CONTENTS Enough?" Proceedings of NESUG 2003, September 2003.

SAS Institute Inc. 2002. *SAS 9 Output Delivery System: User's Guide*. Cary, NC: SAS Institute Inc.

SAS Institute Inc. 2003. "Experimental RTF Features in SAS 9.1", <http://support.sas.com/rnd/base/topics/odsrtf/rtf901.html> (January 30, 2004).

Stetz, Mark. 2000. "Using SAS® Software and Visual Basic for Applications to Automate Tasks in Microsoft Word: An Alternative to Dynamic Data Exchange" *Proceedings of the Twenty-Fifth Annual SAS Users Group International Conference*. Indianapolis, IN.
<http://www2.sas.com/proceedings/sugi25/25/ad/25p021.pdf> (January 20, 2004).

ACKNOWLEDGMENTS

Both authors would like to acknowledge the expert support and advice received from the SAS ODS R&D team.

David would like to thank Elizabeth for her willingness to enter into this adventure. Elizabeth would like to thank David for the invitation to collaborate; Paul Grant for the SQL code to read the dictionary tables; and Rick Sheppe for editing help.

CONTACT INFORMATION

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

Elizabeth Axelrod
Abt Associates Inc.
55 Wheeler Street
Cambridge, MA 02138
Work Phone: 671-349-2458
Fax: 671-349-2675
Email: elizabeth_axelrod@abtassoc.com
Web: <http://www.abtassoc.com/>

David Shamlin
SAS Institute Inc.
SAS Campus Drive
Cary, NC 27513
Work Phone: 919-531-7755
Fax: 919-677-4444
Email: david.shamlin@sas.com
Web: <http://www.sas.com/>

TRADEMARK CITATIONS

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. © indicates USA registration.

Other brand and product names are registered trademarks or trademarks of their respective companies.