PharmaSUG 2014 - Paper AD22

Automation of Appending TLFs

William Coar, Axio Research, Seattle, WA

ABSTRACT

SAS® reports presenting data from clinical trials are typically in the form of summary tables, listings, and figures (TLFs). Whether a small or large set of reports, a single document that appends output files containing bookmarks and a hyperlinked table of contents may better facilitate review and simplify document management. Although numerous techniques exist to append reports, there is much room for improvement when creating a single document composed of multiple TLFs as well as a table of contents directly from SAS. The proposed process takes advantage of external data used to track SAS programming, ODS Document, and Proc Document.

Generation of the concatenated report requires two primary functions: identification of TLFs to be appended, and then the actual appending. Many companies already have infrastructure for tracking generation and testing individual TLFs produced using SAS. Provided SAS can communicate with the tracking component of the existing system, a list of outputs with the desired sectioning and sorting can be obtained.

The proposed concatenation process is an extension of an application previously presented at PharmSUG 2013 [2]. It requires the use of ODS Document to create individual item stores during the initial creation of each TLF, and the use of Proc Document to manipulate and replay such item stores into a single, well-structured document with a hyperlinked table of contents and bookmarks.

An example showing how to prepare such a report will be presented using SAS 9.3 in a Windows environment.

INTRODUCTION

Although the setting for the application presented may be quite specific, it is not uncommon in the pharmaceutical industry. Individual programs generate one or more TLFs through the Output Delivery System (ODS) destination such as RTF or PDF. These programs are developed over time, and submitted in batch mode for production version of the reports. The final procedure for all summary tables and listings is Proc Report. Figures result from one of the SG procedures.

During the programming process, a tracking document is generally in place to easily find the stage of programming for each of the reports. While some outputs are being tested, others any only be in initial development, or not even started. This tracking document also allows a project manager to easily identify which reports are final. There is often other utility for such a file as well, such as this application provided the necessary columns exist.

The lengthy discussion below is as follows. An example of an existing system to automate appending reports is shown as motivation for obtaining alternative approaches. Second, a review of the use of ODS Document and Proc Document is provided. Although many of the actual programming details are omitted, the key concepts of item stores as they relate to this application are presented. These are essential to understanding the incrementing steps of the proposed automated application. The actual automated process of reading in a program tracker to identify a list of reports, performing diagnostics, restructuring, and finally replaying precede concluding remarks.

EXISTING METHODS

Historically, tables and listings were simply text files (LST) that included printer control language that allowed for nicely formatted printed output. With the introduction of ODS, tables and listings in RFT format are more common. SG procedures and Graph Template Language now allow for programmers to obtain highly custom graphs directly from SAS rather than other software such as R, SPlus®, or Excel®.

A number of methods to create a single document from a set of individual TLF files exist ([1],[6],[7],[8],[9],[10]) when individual files exist in LST, RTF, or PDF format. The more common setting is for reports to exist in RTF format, though at least one method exists for individual PDF files. These techniques were reviewed in [1]. In the settings presented in [1], a typical process for automation might look like something in Figure 1.

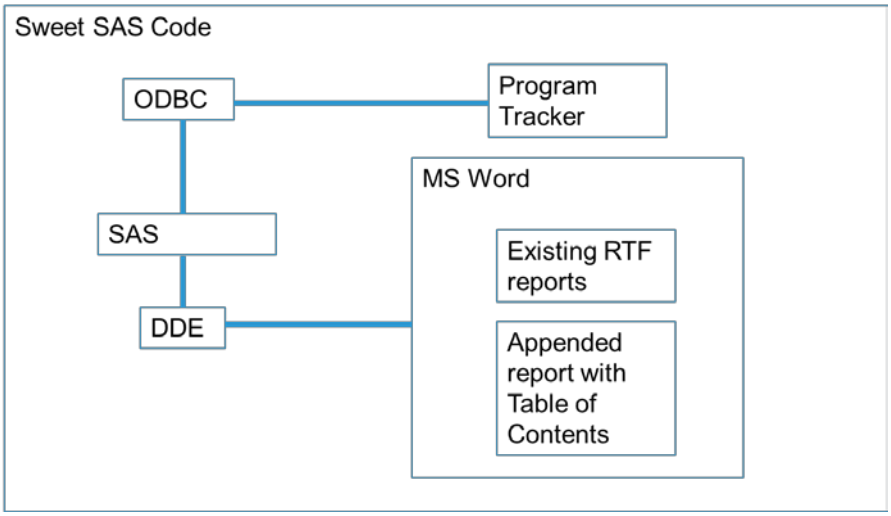


Figure 1: Process flow for post-processing RTF files

Wrapped in some sweet SAS code, this existing method reads in a program tracker from Excel to identify a list of RTF files. The process relies of DDE to open Microsoft Word® and send Word Basic commands. This requires SAS license to communicate with Excel, and it also relies heavily on MS Word. A single document is obtained, though the table of contents is not actually an auto-generated Table of Contents from within Word. It is actually a simple table in Word with cross references to bookmarks throughout the document. The major drawback with this approach is the reliance on DDE. Although DDE can be a powerful tool, it has proven to be unstable in some server environments and the final file format can use improvement. Other methods of creating an appended report rely on Visual Basic, or modification of RTF tags and appending the text files. An alternative approach using current software capabilities is desirable, such as the REPLAY approach discussed in [2].

REVIEW OF THE REPLAY APPROACH

This two-step process in Figure 2 was proposed in [2]. The first step of the process uses ODS Document to create item stores for each TLF. This is done in the initial creation of each TLF. These item stores house the instructions and data used by the procedures called within a block of ODS Document statements. The second step uses Proc Document to combine, restructure, and replay these item stores into a single document within an ODS destination.

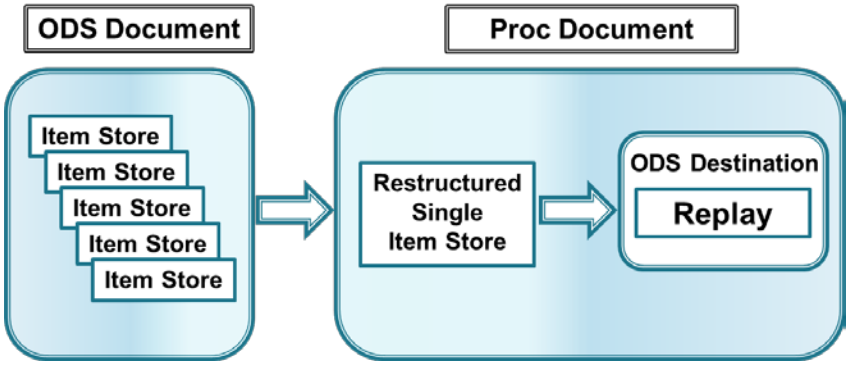


Figure 2: Replay Approach

Since all the item stores are replayed within a single ODS destination, the result is a single file that contains all the reports that encompass the TLFs.

A REVIEW OF FIGURE 2 COMPONENTS

An overview of item stores in the context of Figure 2 will allow the reader to better understand the automated process which is proposed in the next section, particularly the section on diagnostics and the section on restructuring into a single item store.

Recall that for each individual TLF, there is an existing block of ODS statements to create an RTF or PDF. If the additional ODS Document block is added, accompanied by (minor) program updates for style points, an item store will also be created when each program is executed. Item stores hold the data and instructions from the procedure used to create the report. They have procedure specific internal structures that may not make much intuitive sense individually, but structure becomes more apparent when put together into a single restructured item store. The discussion presented here focuses on item stores created from Proc Report and SG procedures. For a more general discussion of item stores, see [4] and [5].

To see what is inside an item store created, the following code can be submitted:

```
proc document name=istore.lrand_closed;
        list / levels=all;
run;
quit;
```

Sample Code 1: Item store from Proc Report

The results are shown in **Output 1**, where we see the item store created when using Proc Report consists of directories and a table. To regenerate the report, a programmer needs to actually replay the item store.

Listing of: \Istore.Lrand_closed
Order by: Insertion
Number of levels: All

Obs	Path	Type
1	\Report#1	Dir
2	\Report#1\Report#1	Dir
3	\Report#1\Report#1\Report#1	Table

Output 1: Item store from Proc Report

Regarding graphics, the use of GTL adds often needed flexibility for producing custom graphics. Although the example below results from using GTL and Proc SGRender, they easily extend to other SG procedures as they all have similar structure. Suppose the code in **Sample Code 1** were resubmitted to list what is inside of an item store created from Proc SGRender, the output might look like **Output 2**.

Listing of: \Istore.Fhemabpbs_closed
Order by: Insertion
Number of levels: All

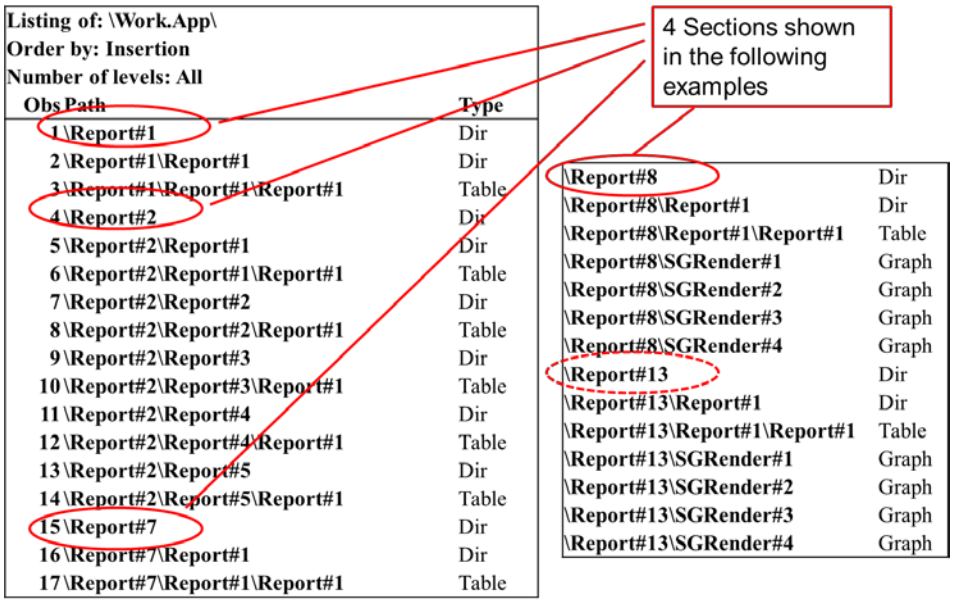
Obs	Path	Type
1	\Sgrender#1	Dir
2	\Sgrender#1\SGRender#1	Graph
3	\Sgrender#2	Dir
4	\Sgrender#2\SGRender#1	Graph
5	\Sgrender#3	Dir
6	\Sgrender#3\SGRender#1	Graph
7	\Sgrender#4	Dir
8	\Sgrender#4\SGRender#1	Graph

Output 2: Item store from Proc SGRender with 4 Images

The item store in **Output 2** was generated for a custom lab figure to assess patient values over time. Although the figure was created in a single RTF file, there were actually 4 images, one for each of 4 laboratory parameters of

interest. Note here that the item store consists of *directories* and *Graphs*. For simpler figures that fit on a single page, the item store would only have a single directory and graph, both labeled Sgrender#1\SGRender#1.

Based on the above examples, it is clear that the proposed process deals with *directories*, *reports*, and *graphs*. Although not immediately obvious in **Output 2,** it is seen that SAS (predictably) increments as additional directories and reports(tables/figures) are added. ***This incrementing is essential when automating the creation of a single restructured report.*** An example of a single restructure item store is shown in **Output 3**.



Output 3: Example of a single Restructured Item Store

The details of actually reading and restructuring into a single item store are discussed in [2], thus omitted here. Briefly, COPY, MOVE, DELETE, and RENAME statements with Proc Document allow a programmer to create a single item store that contains all of the desired TLFs in user-defined sections.

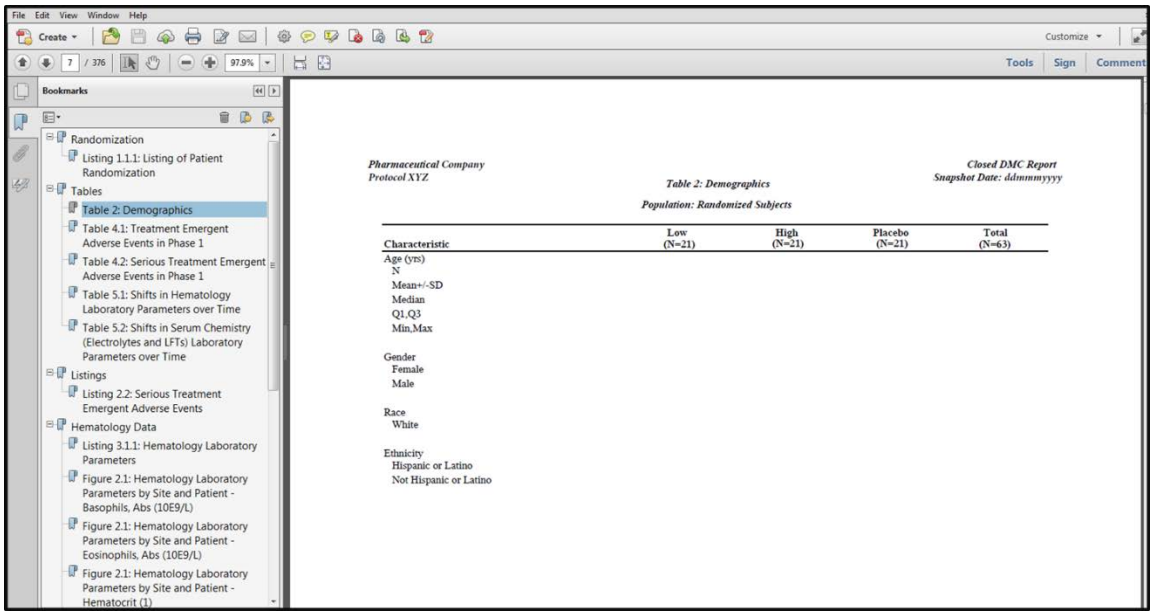
Take note of the directory numbers circled in RED. First, the left-most directory level is the same within each section. Second, the number of this directory is a function of the number of "Reports" (either tables or graphs) previously read into the structured item store. For example, the second section is labeled "Report#2" simply because the first section only had 1 report (ie, one table) in it. The third section is labeled Report#7. Even though this is the third section, the report label is Report#7 because the first report in the third section is actually the 7th report (table or graph) read into the restructured item store.

As seen in the fourth (and fifth) sections, a section can have tables, listings, and/or figures. In the above example, the fourth section (Report#8) has one listing and four figures. Recall both tables and listings come from Proc Report, so the contents of the item store are labeled "Table", regardless if it is a summary table or listing. **Output 3** alone does not suggest Report#8 contains a listing and 4 graphs, but this becomes clear when this item store is replayed.

Even though only the first element in section 4 actually comes from Proc Report, incrementing continues to occur with respect to the label of the next section (Report#13). This is by design of the application. Recognizing that the left-most directory level is really just a label that gets incremented, the application renames the directories of figures to "Report". Thus, we can see there are two types of incrementing: one with respect to the number of item store elements previously read in (to uniquely identify a section) while the other simply increments an element within each section.

SAMPLE OUTPUT

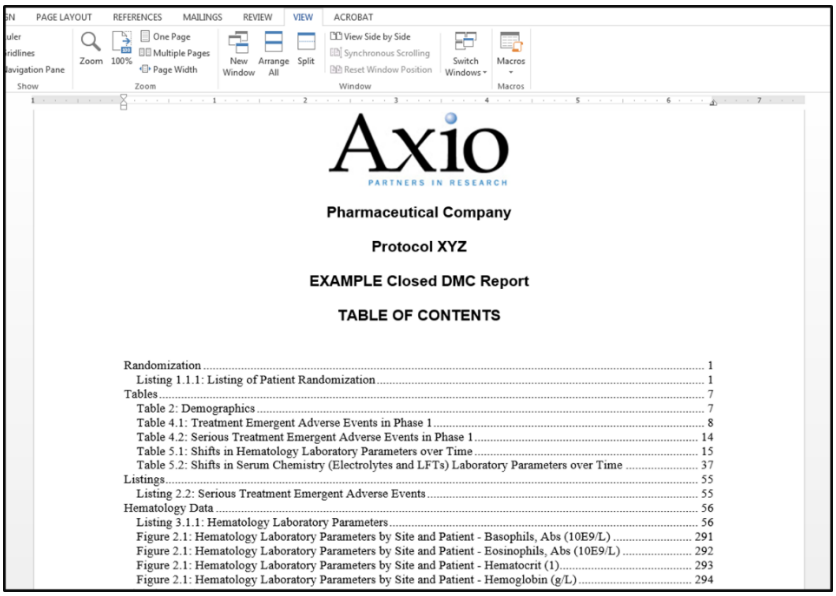
To clear the mind from the above discussion on incrementing, it is useful to view the output from replaying the first 4 sections of the item store in **Output 3** into a (single) RTF file and a (single) PDF.



Example 1: PDF from replaying four sections in the Output 3 item store

When using ODS to output directly to PDF, a highly structured (and searchable) PDF file is created with all of the individual TLFs in the desired sectioning. Although there is no Table of Contents, the bookmarks on the left-hand side provide the similar information and since the bookmarks are hyperlinks to each report, navigation throughout the document is quite easy. The absence of summary statistics in the demographics table in **Example 1** is by design for this paper since there data are proprietary.

The corresponding output in RTF format is seen in **Example 2**.



Example 2: RFT from replaying four sections in the Output 3 item store

When using RTF, a programmer can obtain a highly customizable (hyperlinked) table of contents. It is recommended that these RTF files actually be re-saved as Word documents. Although both **Example 1** and **Example 2** have desirable formats, creation of the restructured single item store in **Output 3** may seem intimidating, especially since the sectioning is likely to change from among projects, or even reports within a project. Automation of these steps would provide tremendous efficiencies.

ODSPD APPROACH

The proposed macro based approach, ODS/Proc Document (ODSPD), automates the replay approach presented in [2]. As seen in **Figure 3**, the process of generating the single document can be simplified.

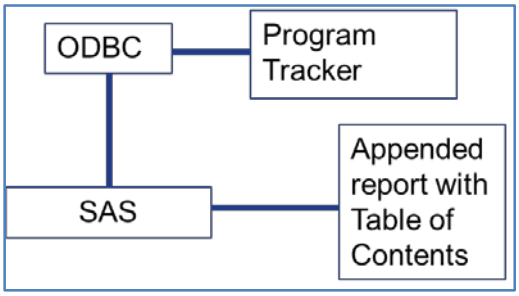


Figure 3: A simplified approach using ODS Document and Proc Document

The proposed approach still requires the ability to communicate with a program tracker, but it eliminates the need for DDE. It allows a programmer to create both RTF and PDF without the need for additional software. Note that SAS will create an RTF file with the necessary components for a hyperlinked Table of Contents, but the user must actually open the file and update the field to populate it. A PDF with bookmarks can be created directly from SAS without the need to purchase a license to PDF writer software.

The program flow with the ODSPD approach is straightforward: read in a list of TLFs from a program tracker, perform some diagnostics (checkis.sas), build the restructured (single) item store (makeapprtp.sas) with desired sectioning, and then replay the desired sections (rplapprpt.sas). Although the actual code is not provided in this paper, it can be made available by request.

READING IN THE PROGRAM TRACKER

Most programming departments seem to have external files used to track program development and testing. Such a "tracker" can be used for our purposes as well with the a few additional columns. The basic requirements are:

- One record per report element defined as a single table, listing, or graph
- (Numeric) Column for sorting sections
- (Character) Column for section names
- Item store name
- (Numeric) Column for sorting within section
- SAS procedure used
- Internal item store sequence number (should it contain multiple tables or graphs)

An example of the tracker used in the previous example is shown below.

Section Number	Section Name	TLF Type	TLF Number	ItemStore Name	Item ID	Proc Type
0	Randomization	Listing	1	lrand_closed	1	Report
1	Tables	Table	1	rdemog_closed	1	Report
1	Tables	Table	2	raep1_closed	1	Report
1	Tables	Table	3	rsaep1_closed	1	Report
1	Tables	Table	4	rhema_closed	1	Report
1	Tables	Table	5	rchem_closed	1	Report
2	Listings	Listing	1	lsae_closed	1	Report
3	Hematology Data	Listing	1	lhemalab_closed	1	Report
3	Hematology Data	Figure	2	fhemabpbs_close	1	Sgrender
3	Hematology Data	Figure	3	fhemabpbs_close	2	Sgrender
3	Hematology Data	Figure	4	fhemabpbs_close	3	Sgrender

Example 3: Tracker in Excel

In order to know verify what is inside the item store for diagnostics and restructuring, the application needs to know

what procedure created the item store, and what report (or graph) number it will be. This has been more useful for graphs, and in particular, graphs of laboratory data done by parameter. The graphing is *not* actually performed using BY VARIABLE processing. Such processes produces items stores with a different structure. For this application, one-at-a-time graphics were utilized which results in a straight forward item store structure containing multiple graphs.

As seen in the red box in **Example 3**, we can expect to find the first listing in section 3 to be associated with Report#1, the first figure to be associated with Sgrender#1, the second figure is associated with Sgrender#2, and so forth because it is known that the item store from creation of the hematology figure had multiple graphs inside. With an understanding of the individual item store structures, a programmer can then predict what *should* be in the actual item store in the diagnostic step of the application.

DIAGNOSTIC: CHECKIS.SAS

When Proc Document experiences problems, uninformative errors occur leaving the programmer frustrated and almost certainly willing to give up. Throughout the development of ODS Document and Proc Dcoument techniques, it has become clear that most of these problems are due to missing or incomplete item stores. Such is the case when programmers overlook warnings or errors in the log files when an individual item store was created. Thus, a diagnostic step has proven to be quite useful.

The macro to perform diagnostics has the following parameters used to predict if an item store specified in the tracker actually exists:

- INDATA: Input dataset (presumably from the tracker)
- INAME: Library name that contains the item stores
- ISNAME: Character variable containing the name of each item store
- PROCTYPE: Character variable containing the name of the procedure that created the item store
- IDNUM: ID number internal to each item store. If multiple reports or graphs are within a single item store, use this IDNUM to identify which internal element is to be used

The general flow of the diagnostic is to first obtain a list of item stores that exist in the specified library as shown in Example 4(a). This can be used to verify that the requested item store files in the tracker actually exist. The macro then looks to see if the requested elements (ie, Report#1, SGrender#1, etc.) exist inside using item store properties.

```
ods table documents=zz_isdocs;
proc document;
      doc lib=&INAME;
run;
quit;                          (a)
```

```
ods table properties=zz_isprop;
proc document name=&THISI;
list / levels=all details;
run;
quit;                          (b)
```

Example 4: Code snippets from checkis.sas

The key concept is the use of **Documents** and **Properties** ODS tables to obtain the information to perform the diagnostic. This information allows a programmer to see if the item stores and elements needed for the report actually exist. Printed output can be used for immediate reference should errors occur in the next step, the restructuring process.

RESTRUCTURE: MAKEAPPRPT.SAS

Of all of the discussion in the document, this macro is actually the backbone of the application. Since its implementation, creation of restructured item stores has become seamless. The macro has the following parameters in addition to those used in CHECKIS.SAS:

- SECNUM: (Numeric) variable to identify and sort different sections
- SECNAME: Character variable that has the name of each section. Section name assumed to be the same for all records within a given section number.
- SRTNUM: Numeric variable for sorting within each section (such as the table number)

It does not take long for a programmer to appreciate the value of automating the creation of the restructured item store. The specific details of restructuring are discussed in [2]. The primary key for automation is recognizing SAS's automatic incrementing of directory and element numbers as they are read into the restructured item store. This was why incrementing was emphasized earlier. Incrementing macro variables within loops handles this quite naturally.

The flow of the restructuring is to first see if IDNUM is specified. This is not required if all item stores are restricted to a single report or graph, as is the case with some programming styles. Should this be the case, the IDNUM macro parameter is not necessary. Thus far, the utility of IDNUM has come from figures that require multiple pages of images.

Creation of macro variables and looping are key concepts of the restructuring macro. Macro variables are generated for lists and numbers of elements associated with each list. Lists are determined for different section names, lists of item stores within sections, and a list of elements (Report#1, SGrender#k, etc.) within each section. Lists in the form of macro variables are a natural fit for looping through each section with the necessary incrementing.

Example code snippets from MAKEAPPRPT.SAS macro are shown below.

```
%do b=1 %to &NSEC;
      %do c=1 %to &&NINSEC&B;

            %let thisst=%scan(&&SECITM&B,&C,' ');
            %let thisrpt=%scan(&&SECRPT&B,&C,' ');
            %let thisid=%scan(&&SECID&B,&C,' ');

            (code for read/restructure)

      %end;
%end;
```

Example 5: Snippet of general looping in makeapprpt.sas

As seen in example 5, looping occurs over the number of sections and the number of elements (reports or graphs) within each section. Information from each list is identified for a given element within a section.

```
copy \istore.&THISST.\&THISRPT.#&THISID. to ^;

%if &THISRPT ne Report %then %do;
            rename \&THISRPT. to Report;
%end;

%LET THISI=%EVAL(&THISI+1);
move Report#&THISI.\&THISRPT.#1 to Report#&THISII.;
delete Report#&THISI.;
```

Example 6: Snippet of restructuring in makeapprpt.sas

As previously noted, it is important to understand two types of incrementing done in this process highlighted by the blue circles in Example 6. Recall each section can be identified by the left-most directly label. This directory label, denoted by the macro variable **THISII**, is the same for all TLFs within a section. This only gets incremented with the introduction of a new section. There is also incrementing with respect to each element read into the restructured item store, denoted by **THISI**. The incrementing of these two macro variables is consistent with that described in **Output 3**.

The above snippet of code in **Example 6** includes a conditional statement required for figures. Since tables and listings tend to be more prevalent that graphs, and both are labeled (by default) by “Report”, any element that is read in that is not labeled as “Report” is renamed accordingly. It is this step that allows tables, listings, and graphs within the same section.

Lastly, a few additional programming statements are needed for the first element within a new section. The first is to increment **THISII**, and is trivial. The second is with respect to the RENAME and MOVE statements. If the first element in a new section is from Proc Report, one only needs to COPY the (single) item store in the restructured item

store. However, if the first element is a figure, both RENAME and MOVE statements are required. This is due to the difference between item store structures from Proc Report and the SG Procedues. The above sample code is a snippet from makeapprpt.sas that does not take into consideration if the element is the first in a (new) section.

The last essential component of makeapprpt.sas is the creation of a global macro variable containing a list to identify all of the sections within the restructured item store. What may not be obvious to this point is that the section label numbers (such as Report#1, Report#2, Report#7, and Report#13 in **Output 3**) will change as item stores are added or removed to each section, or if entire sections are added or removed. To accommodate this, an additional global macro variable containing a list of sections is defined in makeapprpt.sas.

With the restructured item store easily obtained, the programmer simply needs to replay all or some of the desired sections.

REPLAY: RPLAPPRPT.SAS

The last step in the automated process is to replay either the entire report, or individual sections of the report. When this is done inside of the infamous ODS sandwich, the result is a single well-structured file with all of the individual TLFs in the user-specified sections. Compared to the other macros in the application this is trivial, with only two macro parameters:

- INAME: Name of the item store to be replayed
- RLIST: Character string of sections of an item store to be replayed

The general flow is a simple loop through the various sections of the report, replaying each section as seen in **Example 7**. Although the development of this macro was initially for replaying in this application, it is in fact a stand-alone macro, and is now used in other applications.

```
proc document name=&INAME;

%do a=1 %to &numsec;
      replay %scan(&RLIST,&A,' ');
%end;

run;
quit;
```

Example 7: Snippet from rplapprpt.sas to replay sections of a report

As long as the macro call is within an ODS block to the desired (RTF or PDF) destination, a single, well-structured document is obtained.

CONCLUDING REMARKS

The reader is congratulated for making it this far into the discussion. There is a lot to discuss to even begin to understand the totality of the process. The application has enabled automation that allows for great flexibility with sectioning, and even allowing tables, listings, and figures within the same section. The user can easily add or remove report elements, or create additional sections. The application is independent of the user, allows for both RTF, PDF, or both, and does not require knowledge much beyond SAS.

As with many applications, this continues to be a work in progress. Although automation has only recently been achieved, the techniques using ODS Document and Proc Document have been successfully utilized at Axio for over two years. The techniques allow us to provide a single well-structured, hyperlinked, searchable document for distribution and electronic review with minimal to no reliance on other software.

The application continues to provide significant efficiencies, though it does require the use of ODS Document during initial program development. Additional needs will surely be identified with continued use, but the foundation is in place. Natural extensions are currently being developed to expand its utility to patient profiles and subset listings in [3].

REFERENCES

[1] Coar, W. “Appending Reports: A Review and Fresh Look”, WUSS 2012, Paper FP-70

[2] Coar, W. “TLFs: Replaying Rather Than Appending” PharmaSUG 2013, Paper 2343-2014

[3] Coar, W. “Generation of Subset Listing” WUSS 2013, Paper FP-93

[4] SAS 9.2 Output Delivery System: User’s Guide, Understanding Item Stores, Template Stores, and Directories, Available at: http://support.sas.com/documentation/cdl/en/odsug/61723/HTML/default/viewer.htm#a003112776.htm

[5] Lawhorn , B. 2011, Let’s Give ’Em Something to TOC about: Transforming the Table of Contents of Your PDF File, SAS® Global Forum

[6] Shannon, D. “To ODS RTF and Beyond”, SUGI 27, Paper 1-27

[7] Osowski, S., Fritchey, T. “Hyperlinks and Bookmarks with ODS RTF”, Pharmasug 2006, Paper TT21

[8] Gilmore, J. “Using Dynamic Data Exchange with Microsoft Word”, SUGI 22, Paper 308

[9] Gupta, A, “Watermarking and Combining Multiple RTF Outputs in SAS”, PharmaSUG 2012, Paper CC06

[10] Anbazhagan, S, and Patel, S., “A SAS Macro Utility to Append SAS-Generated RTF Outputs and Create the Table of Contents”, PharmaSUG 2012, Paper AD12

ACKNOWLEDGMENTS

The author would like to thank the programmers and statisticians at Axio for their discussions in developing the techniques contributing to this paper. The author would also like to acknowledge the many programmers publishing information on the internet, and apologizes to those not properly referenced above, citing the inherent nature of internet searching which makes it difficult to track so many individual ideas.

CONTACT INFORMATION

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

William Coar, PhD
Biostatistician/Director of Central Region Operations
Axio Research, LLC
Seattle, WA 98121
Email: williamc@axioresearch.com