PharmaSUG 2016 - Paper HT06

Combining TLFs into a Single File Deliverable

William Coar, Axio Research, Seattle, WA

ABSTRACT

In day-to-day operations of a Biostatistics and Statistical Programming department, we are often tasked with generating reports in the form of tables, listings, and figures (TLFs). A common setting in the pharmaceutical industry is to develop SAS® code in which individual programs generate one or more TLFs in some standard formatted output such as RTF or PDF. As trends move towards electronic review and distribution, there is an increasing demand for producing a single file as the deliverable rather than sending each output individually.

Various techniques have been presented over the years, but they typically require post-processing individual RTF or PDF files, require a knowledge base beyond SAS, and may require additional software licenses. The use of item stores as an alternative has been presented more recently for TLFs [6][7][8]. Using item stores for Proc Report or the SG procedures, SAS stores the data and instructions used for the creation of each report. Individual item stores are restructured and replayed at a later time within an ODS sandwich to obtain a single file deliverable. This single file is well structured with either a hyperlinked Table of Contents in RTF or properly bookmarked PDF. All hyperlinks and bookmarks are defined in a meaningful way enabling the end user to easily navigate through the document.

This Hands-on-Workshop will introduce the user to creating, replaying, and restructuring item stores to obtain a single file containing a set of TLFs. The use of ODS is required in this application using SAS 9.4 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 an Output Delivery System (ODS) destination such as RTF or PDF. These programs are developed over time, and submitted in batch mode for the 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.

In this setting, all outputs tend to be in individual files. As trends move towards electronic review and distribution, a single file structured with hyperlinks and/or bookmarks is desirable. Many options have been proposed that post-process individual RTF or PDF files ([1], [2], [3], [4],[5],[6]). The focus of this Hands-On-Training is the use of ODS Document and item stores ([6],[7]). While an automated approach using item stores was presented in [8], the goal of this training is simply to develop a basic understanding and learn how to implement these techniques. Automation is outside of the scope of these exercises.

A brief review will cover the replay process presented at PharmaSUG 2013 [7]. A series of examples that create, replay, and modify item stores using Proc Report and Proc SGrender will follow. The final examples will demonstrate the read/restructure/replay of multiple item stores into a single document in a manner that results in an attractive and user-friendly single document with hyperlinks and bookmarks. In addition to the programmatic approach using SAS code, a point-and-click approach will also be presented. The paper will conclude with a brief summary.

REVIEW OF ITEM STORES & THE REPLAY APPROACH

This two-step process in Figure 1 was proposed in [7] for the creation of a single document that contains a set of TLFs with a hyperlinked table of contents and/or bookmarks. The first step of the process uses ODS Document to create item stores for each individual TLF. 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***. Just as one obtains a PDF or RTF, the program also contains an additional file: the item store.

The second step uses Proc Document to combine, restructure, and replay these item stores into a single document within an ODS destination. 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.

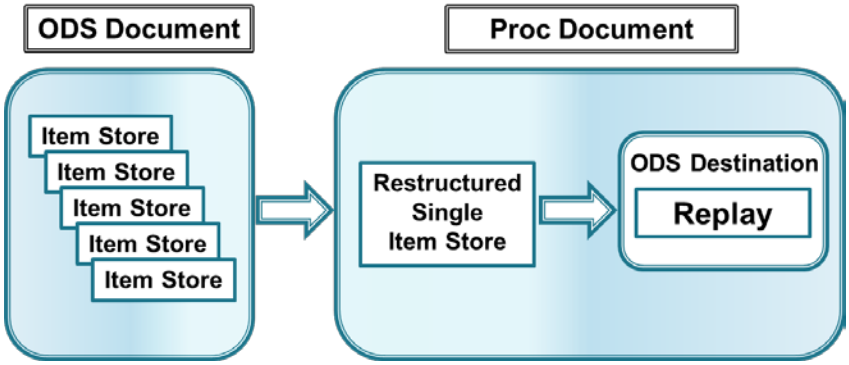


Figure 1: Replay Approach

The discussion presented here focuses on item stores created from Proc Report and SG procedures. For a more general discussion of item stores, see [10].

REVIEW OF CREATING ITEM STORES

While it is common (and reasonable to assume) for Proc Report to be used for tables and listings, it may not be common for figures to be created using Proc SGRender. These techniques apply to all SG procedures, but the author has found there are known issues with (not) displaying titles/footnotes with some SG procedures when the item store is replayed to the PDF destination. While this may seem like a rigid constraint, it is manageable since some SG procedures have an option to output the Graph Template Language (GTL).

Tables and Listings

As stated above, it is easy to create an item store that houses the data and instructions from Proc Report (or Proc SGRender) that can be replayed at a later time. The initial step is to create an ODS Document sandwich that includes all of the desired procedure. **Sample Code 1** below that would create a demographics summary table highlights the statements required around a typical Proc Report.

```
ODS Document name=istore.rdemog (write);
ODS rtf file ="rdemog.rtf" style=AxioConslt;
proc report data=tns_rpt center headline headskip nowindows missing ;
        column sv1 sv2 row1 _1 _2 _3;
Other define statements
run;
ODS RTF close;
ODS Document close;
```

Sample Code 1: Creating an item store

The result of the ODS Document sandwich is an item store named ***rdemog*** that is stored in the location defined by the libname ***istore***. Notice the use of the option (write). This instructs SAS to recreate the item store it already exists. Without the use of (write), SAS will automatically add to the existing item store, which may happen if there are multiple executions of the SAS code.

The item store is a SAS file with its own internal structure only readable by SAS. Thus, to see what is inside of the item store, we must rely on a SAS procedure: Proc Document. **Sample Code 2** below provides the SAS code necessary to see what is inside of the newly created item store.

```
proc document name=istore.rdemog (read);
        list / levels=all;
run;
quit;
```

Sample Code 2: 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. While this may not tell us much about what is ***really*** inside of the item store (you need to replay it for that), it provides us with key information about its structure: it consists of directories and a table. In fact, this structure from Proc Report is very predictable, which will become important in the later examples of this workshop when we seek to combine individual item stores.

Listing of: \Istore.Rdemog\		
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

To regenerate the report, a programmer needs to actually replay the item store. To do so requires the use of Proc Document with a REPLAY statement, as seen in **Sample Code 3**.

```
ODS RTF file ="rdemog_replay.rtf" style=AxioConslt;

proc document name=istore.rdemog (read);
        Replay Report#1l;
run;
quit;

ODS RTF close;
```

Sample Code 3: Replay a report from an item store

Since the Proc Document and REPLAY statement are within an ODS RTF sandwich using the same ODS style, the resulting RTF (named rdemog_replay for demonstration) will be identical to the original RTF.

Tip #1: Item stores are not specific to ODS destination or ODS style. Item stores can be replayed into different ODS destinations using different styles. These files may look different, but the contents displayed will be the same.

Tip #2: The (read) option is used in the Proc Document so the item store itself remains unchanged by Proc Document. Other options such as (write) and (update) are used to modify an existing item store using Proc Document.

While a Table of Contents (ToC) and/or bookmarks may not be useful for a single table, it is reasonable to introduce them at this time. Sample ODS RTF and ODS PDF statements below in **Sample Code 4** show the use of options that will create the desired hyperlinked ToC in RTF and bookmarks in PDF.

```
ODS RTF file ="rdemog_replay.rtf" style=AxioConslt contents=on toc_data;
ODS pdf file =”rdemog_replay.pdf" style=AxioConslt pdftoc=2 contents=on;
```

Sample Code 4: ODS statements with options for ToC

When initially opened in Word, the resulting RTF will contain what appears to be an empty ToC on the first page, with the table starting on page 2. The ToC exists, but it needs to be populated. There are a number of ways to do so. One method is to select the ToC by using Control->A, then hit the F9 key. Other methods exist, such as right clicking on a ToC and selecting “update field”.

When opening the PDF, the user will find there are 2 levels in the bookmarks. The second level actually points to the table. Since the file opens with the bookmarks pane open, it may not be necessary to have bookmarks and a ToC. A ToC within PDF does not work well when figures are in the document, thus in general, should be avoided.

Tip #3: Due to inherent differences between RTF and PDF, you may notice differences in spacing and line breaks when an item store is replayed to both destinations. While the outputs may look slightly different, the contents displayed are the same.

As will be seen in the exercises, the resulting ToC and/or bookmarks will be uninformative. They will simply say “The Report Procedure” and “Table 1”. With straightforward modifications to the SAS code, the user can obtain more informative hyperlinks and bookmarks. More details can be found in [10]. The first step is to add the CONTENTS= option within Proc Report to specify the desired text for the hyperlink or bookmark. The second step is slightly more tedious. The input dataset to Proc Report needs a flag variable that is constant for every record in the data. This flag variable is defined as an ORDER variable with a NOPRINT option. Finally, a BREAK BEFORE statement is added with a necessary contents=’’ option. Assuming the input dataset has a variable named flag which is set to 1 for every record, **Sample Code 5** shows the necessary updates to Proc Report to obtain more meaningful bookmarks and hyperlinks.

```
ODS Document name=istore.rdemog (write);
ODS rtf file ="rdemog.rtf" style=AxioConslt;
proc report data=tns_rpt center headline headskip nowindows missing contents=”Table 1: Demographics”;
        column flag sv1 sv2 row1 _1 _2 _3;
        define flag /order order=data noprint ;
Other define statements
break before flag / page contents='';
run;
ODS RTF close;
```

Sample Code 5: Updates to Proc Report to obtain informative hyperlinks/bookmarks

When creating a ToC after these modifications, the resulting output will have hyperlinks/bookmarks that say “Table 1: Demographics” rather than a generic “Table 1”. The ToC will continue to have a level label “The Report Procedure”. However, this will be ignored later when the Read and Restructure phase of the Replay Approach is reviewed.

The exercises of this Hands-on-Training are meant to allow the user to gain experience in these areas for outputs generated from Proc Report. To this point, creation of item stores for Proc Report (tables and listings) will be reviewed in Exercises 1 and 2 of the workshop. Modifications required for obtaining a hyperlinked ToC and informative hyperlinks/bookmarks will be reviewed in Exercise 3 of the workshop.

Figures

As noted above, the application presented relies on Proc SGRender to create figures. Once the GTL code is obtained, it is easily modified to allow the program to first create a template from GTL, and then create the figure by using Proc SGRender. This also allows for opportunities to provide custom figures without extensive knowledge of GTL.

Tip #4: The SG Procedures and replaying item stores from an SG procedure don't always play well with PDF. When using PDF, there are known issues where titles and footnotes are not displayed. Furthermore, use of the ODS escape ~LASTPAGE will result in a PDF file without a graph.

Tip #5: The size of a figure in PDF may be reduced depending on the number of titles and footnotes when using SAS Version 9.4. The author suggests using the DESIGN size options in GTL to maximize the size of the graph for RTF destination. Once replayed in PDF, the size of the image is reasonable.

As with tables and listings, modifications to SAS code are required, but manageable. The first is to obtain GTL for the figure in preparation to use Proc SGRender. *In doing so, the GTL must be stored in a permanent template store so it is available at a later point in time when the item store is recalled.* **Sample Code 6** shows the necessary update to the GTL code so that the GTL is stored in a permanent template.

```
ods path (prepend) istore.templat (update);
proc template;
  define statgraph fcumenrl;
  begingraph / border=false designwidth=9in designheight=4.5in;
```

Sample Code 6: Storing GTL in a permanent template store

There are a few noteworthy additions to the SAS code in **Sample Code 6.** The first is the use of (prepend) in the ODS path statement, which allows the user to simply add a new location to the beginning of the existing ODS search path. The second is the use of (update). This instructs SAS to save the subsequent graph template in a permanent catalog. If the catalog exists, SAS will simply update it. If the catalog does not exist, then SAS will create it. The third is the use of the SAS program name for the name of the graph template. This allows us to have unique graph templates. All will be stored in the same permanent catalog in a folder defined by the libname istore.

Tip #6: While it may be acceptable to use dashes (-) in file names for SAS programs, dashes are not acceptable for naming item stores or templates.

Remember, the primary reason for storing the GTL in a permanent catalog is that it needs to exist later when the item store is replayed.

As with Proc Report, a minor update is needed to obtain informative hyperlinks and bookmarks. **Sample Code 7** provides example code of the updates required to create an item store for figures created using Proc SGRender.

```
ODS PDF file="fcumenrl.pdf"  style=AxioConslt nogfootnote nogtitle notoc ;
ODS Document name=istore.fcumenrl (write);
proc sgrender data=rpt template=fcumenrl description="Figure 1: Cumulative Enrollment";
run;
ODS Document close;
ODS PDF close;
```

Sample Code 7: Updates to Proc SGRender

Note that with Proc SGRender, the DISCRIPTION= option is used rather than CONTENTS= as in Proc Report. Also see that an item store is created simply by making an ODS Document sandwich.

The same code provided in **Sample Code 2** can be used to see what is inside of the item store resulting from Proc SGRender, with a minor update to change the name of the item store. Once this code is executed, you will see the structure of an item store created using Proc SGRender consisting of one graph. The output is shown below in **Output 2**.

Listing of: \Istore.Fcumenr\		
Order by: Insertion		
Number of levels: All		
Obs	**Path**	**Type**
1	**\SGRender#1**	Dir
2	**\SGRender#1\SGRender#1**	Graph

Output 2: Item Store from Proc SGRender

Note here that this item store consists of *directories* and *graphs* whereas those from Proc Report consisted of *directories* and *tables*. In both cases, the structure of the item store is predictable. This becomes important during the read and re-structure process of creating the single document.

To regenerate the report, a programmer once again needs to replay the item store. As seen with minor updates to **Sample Code 3**, the figure can be replayed using **Sample Code 8**.

```
ODS RTF file ="fcumenrl_replay.rtf" style=AxioConslt;

proc document name=istore.fcumenrl (read);
        Replay SGRender#1l;
run;
quit;

ODS RTF close;
```

Sample Code 8: Replaying a figure from Proc SGRender

Tip #7: It is not necessary to state Report#1 or SGRender#1 in Sample Code 3 and 8. Simply stating “Replay” without any specific table or graph will replay everything inside the item store.

To this point, creation of item stores for Proc SGRender (figures) has been reviewed. Modifications required for obtaining a hyperlinked ToC and informative hyperlinks/bookmarks have been presented. Exercises 4 and 5 in the Hands-on-Training are meant to allow the user to gain experience in these areas.

More on Figures

In the pharmaceutical industry, it is common to provide figures of laboratory parameters or vital sign parameters over time. The resulting figure is typically in a single RTF (or PDF) file with one page per parameter. Each page may have customization with respect to the parameter, such as labeling of the y-axis. A “one-at-a-time” approach can be used to obtain such a figure where a macro is called once for each parameter. Since the macro calls are all within an ODS RTF (or PDF) sandwich, they all end up in a single file.

This concept extends to ODS Documents and item stores as well. To this point, there has only been a single table or graph in each item store. However, this is not a limitation. Item stores can contain multiple tables and/or graphs. The following example considers a lab figure over time where there are multiple lab parameters of interest (4 for demonstration). Consider a single item store created in manner where a graph is created for each parameter within a single ODS sandwich. This results in an item store with multiple graphs. Using techniques in **Sample Code 2** pointing to **name=istore.flabchem (read)**, we see the item store looks like:

Listing of: \Istore.Flabchem\		
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 3: Item store from Proc SGRender with 4 Images

The item store in **Output 3** 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 21 images, one for each of 21 laboratory parameters of interest. For illustration, only 4 were included in **Output 3**. 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. For graphs that might have one image per page, we find a predictable structure in the item store: a directory and graph for each image labeled sequentially.

Based on the above examples, it is clear that the proposed process deals with *directories*, *reports*, and *graphs*. Although not immediately obvious in **Output 3,** 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.***

Exercise 6 in the Hands-on-Training is meant to allow the user to gain experience in viewing item stores that have multiple graphs within them as well as replaying only select components of the item store.

READ, RESTRUCTURE, AND REPLAY ITEM STORES

Now that the user has experience with creating, reading, and replaying item stores, the next step in the application is referred to as **read and restructure**. Extrapolating from the discussion in Exercise 6 of the workshop, we find that a single item store can have multiple tables and graphs. While simply reading in multiple tables and graphs into a single item store may give you a single file deliverable, but it will not yield the desired structure in a Table of Contents or Bookmarks. The structure we wish to obtain allows us to have the TLFs in the single file grouped in a meaningful way. For example, we may wish to have all of the Adverse Events displayed together, followed by the display of all Laboratory reports.

Based on the preceding examples, replaying the contents of the item store should be straightforward using the REPLAY statement. The following discussion focuses on reading and restructuring to obtain a *single item store with the contents of the entire deliverable*. When this is replayed within an ODS sandwich to either RTF or PDF destination, the result is a single document that has all the TLFs with a hyperlinked Table of Contents and bookmarks of the desired structure.

The details of reading in and restructuring item stores in general can be found in Lawhorn [5]. A more applicable discussion can be found in [7] and [8], the predecessors of this Hand-On-Training.

To begin the exercise of read and restructure, two adverse event summary tables are read into a single item store as shown in Sample Code 9 using the COPY command.

```
proc document name=work.app71 (write);
        copy \istore2.raeov\Report#1 to ^;
        copy \istore2.rae\Report#1 to ^;
        list        / levels=all;
run;
quit;
```

Sample Code 9 Reading in Two Tables

The name=work.app71 (write) statement indicates a new item store named app71 will be created in the work directory. The COPY command is used to read in two item stores into a single item store. Recall that an item store consists of directories, tables, and graphs. Using the TO ^ indicates we copy both of the item stores to the same parent directory within app71. The structure of the resulting item store work.app71 can be seen in **Output 4**:

Listing of: \Work.App71\		
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
4	\Report#2	Dir
5	\Report#2\Report#1	Dir
6	\Report#2\Report#1\Report#1	Table

Output 4

Of note in **Output 4** is the sequencing SAS applies when item store are read in. For a more custom Table of Contents and bookmarking, the MOVE, DELETE, and SETLABEL commands are introduced as seen in **Sample Code 10**. It may be helpful to think of \Report#1 and \Report#2 (observations 1 and 4) simply as labels, or names of directories. As will be seen later, these can be renamed to some other label. For our purposes (and for automation), we keep with the SAS naming convention of \Report.

```
proc document name=work.app72(write);
        copy \istore2.raeov\Report#1 to ^;
        copy \istore2.rae\Report#1 to ^;
        move Report#2\Report#1 to Report#1;
        delete Report#2;
        setlabel Report#1 "Adverse Events";
        list      / levels=all;
run;
quit;
```

Sample Code 10 Restructuring the item store

Our first step in restructuring is to MOVE the directory \Report#2\Report#1 and table \Report#2\Report#1\Report#1 (observations 5 and 6) to be in the same directory as \Report#1. The reason is purely for customization of the ToC. Since these two tables are related (both are adverse events), it is desirable to have them in the same section of the ToC.

Once these have been moved to \Report#1, the only thing that remains is an empty directory \Report#2. The DELETE command removes this empty directory. This step of restructuring is to delete observation 4, the directory \Report#2.

The last step is to add a label “Adverse Events” to the section defined by directory \Report#1. The resulting restructured item store is found in Output 5.

Listing of: \Work.App72\		
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
4	\Report#1\Report#2	Dir
5	\Report#1\Report#2\Report#1	Table

Output 5

If we were to replay this restructured item store, the ToC would have one section labeled Adverse Events that consists of two tables. The hyperlinks are defined by the CONTENTS= option in the initial creation of each table.

Exercise 7 in the Hands-on-Training is meant to allow the user to gain experience in reading multiple tables and restructuring into a single item store.

Suppose we wanted to add another section for laboratory data. Or, suppose we wanted to add a summary table of shifts from baseline to the worst post-baseline CTC grade as well as a listing of laboratory abnormalities. Using similar code as **Sample 10** we can add a second section labeled “Laboratory Data”.

```
proc document name=work.app81(write);
        copy \istore2.raeov\Report#1 to ^;
        copy \istore2.rae\Report#1 to ^;
        move Report#2\Report#1 to Report#1;
        delete Report#2;
        setlabel Report#1 "Adverse Events";

        copy \istore2.rlabcwrst\Report#1 to ^;
        copy \istore2.lchemab\Report#1 to ^;
        move Report#4\Report#1 to Report#3;
        delete Report#4;
        setlabel Report#3 “Laboratory Data";

        list        / levels=all;
run;
quit;
```

Sample Code 11: Addition of a lab table and listing

The structure of the resulting item store is seen below in Output 6.

Listing of: \Work.App81\		
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
4	\Report#1\Report#2	Dir
5	\Report#1\Report#2\Report#1	Table
6	\Report#3	Dir
7	\Report#3\Report#1	Dir
8	\Report#3\Report#1\Report#1	Table
9	\Report#3\Report#2	Dir
10	\Report#3\Report#2\Report#1	Table

Output 6 Item store with 2 sections, each containing 2 tables

There are a few things to note with **Output 6**. Recall that when SAS reads in another item store with a directory name \Report, it automatically increments. The laboratory summary table was the 3rd item store read in. There are no MOVE or DELETE statements associated with this table. This is because we are creating a new section in the ToC, and sections are defined by the higher level directories. The first section is defined by \Report#1 whereas this section section is defined by \Report#3. When replayed, the resulting ToC will have two sections labeled according to what was specified in the SETLABEL commands.

Exercise 8 in the Hands-on-Training is meant to allow the user to gain experience in creating a single item store that contains two sections, each with 2 tables.

We are not restricted to sections containing just summary tables or listings (both of which are identified as Tables within the Proc Report item stores). Suppose we wish to add figures of laboratory parameters over time in the section on Laboratory Data using the item store in Output 3 that has multiple graphs (one per parameter). For demonstration purposes, suppose we wish to add the image defined by \SGRender#1\SGrender#1. This is initially copied into the newly structured item store using the COPY command as was done previously. However, we note that the higher level directory is named \SGRender#1. In order for this graph to be in the same section as the other laboratory table(s) and listing(s), the higher level directory needs to also be labeled \Report#3. To do so will require the introduction of the RENAME command.

```
proc document name=work.app91(write);
        copy \istore2.raeov\Report#1 to ^;
        copy \istore2.rae\Report#1 to ^;
        move Report#2\Report#1 to Report#1;
        delete Report#2;
        setlabel Report#1 "Adverse Events";

        copy \istore2.rlabcwrst\Report#1 to ^;
        copy \istore2.lchemab\Report#1 to ^;
        move Report#4\Report#1 to Report#3;
        delete Report#4;

        copy \istore2.flabchem\SGRender#1 to ^;
        rename \SGRender to Report;
        move Report#5\SGrender#1 to Report#3;
        delete Report#5;
        setlabel Report#3 “Laboratory Data";

        list    / levels=all;
run;
quit;
```

Sample Code 12 Adding a figure

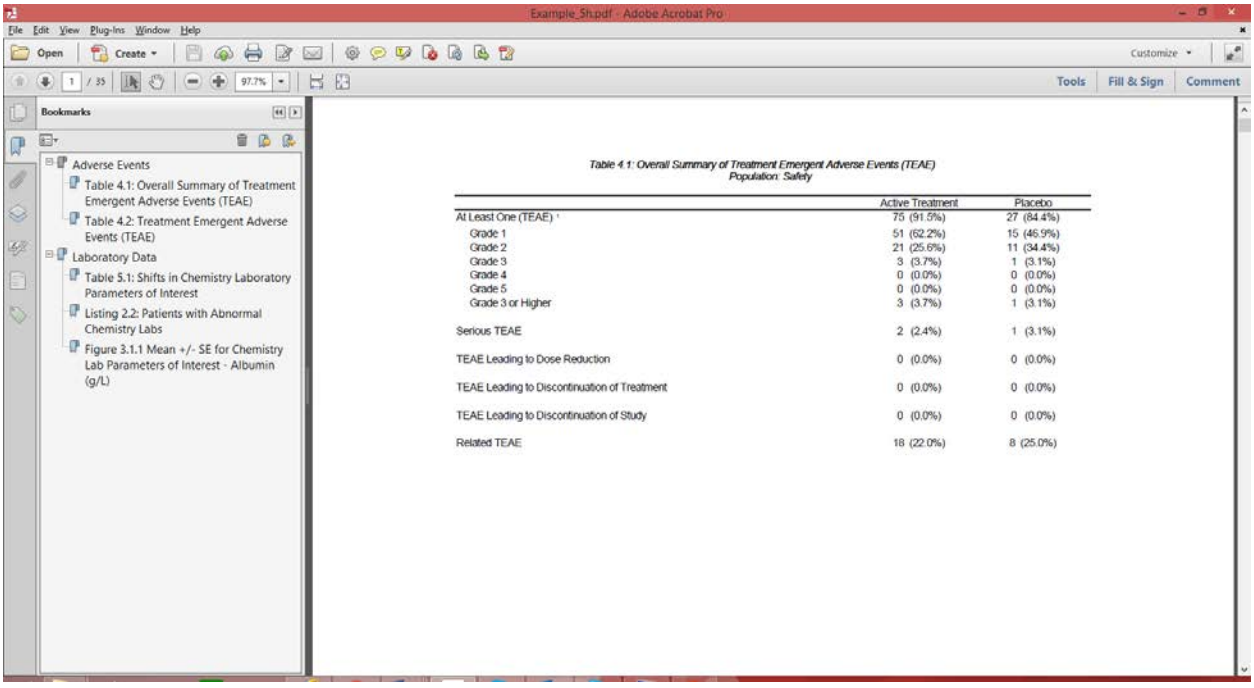
As seen in **Sample Code 12**, the graph is first read in and renamed from SGRender to Report. SAS continues to increment to obtain this newly created directory \Report#5. Again using the MOVE and DELETE commands we find that summary tables, listings, and graphs can all be within the same section of a document. The result of this series of RENAME, MOVE, and DELETE commands is an item store with structure seen in Output #7.

Listing of: \Work.App91\		
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
4	\Report#1\Report#2	Dir
5	\Report#1\Report#2\Report#1	Table
6	\Report#3	Dir
7	\Report#3\Report#1	Dir
8	\Report#3\Report#1\Report#1	Table
9	\Report#3\Report#2	Dir
10	\Report#3\Report#2\Report#1	Table
11	\Report#3\SGRender#1	Graph

Output 7

When replayed, this restructured item store results in two sections defined by types of safety summary data presented: adverse events and labs. There are many other natural ways to combine TLFs into meaningful sections. The techniques presented so far enable such customization, and can be automated so that updates can be made with little modification (see [8]). While automation is beyond the scope of this hands-on-training, a seasoned programmer might recognize the sequencing and series of COPY, MOVE, DELETE commands make macro programming a reasonable choice.

A screenshot of the resulting PDF is seen below in Output #8.



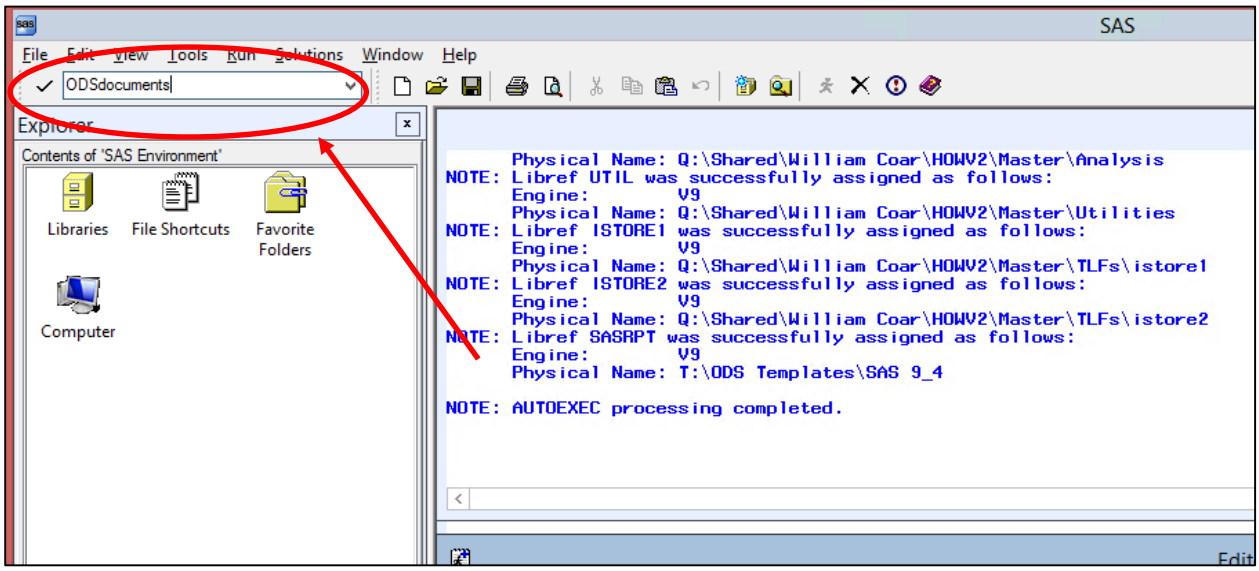
Output 8 Single file deliverable with custom bookmarks

We recognize that Output #8 is the first to demonstrate the REPLAY of a restructured item store. This is a function of producing this written paper to accompany the training. The users will see the resulting output throughout the training. When viewing the sectioning of bookmarks in Output #8, we hope the reader will recognize the similarities between the structure of the item store in Output #7 and the final single file deliverable in Output #8.

A POINT & CLICK APPROACH

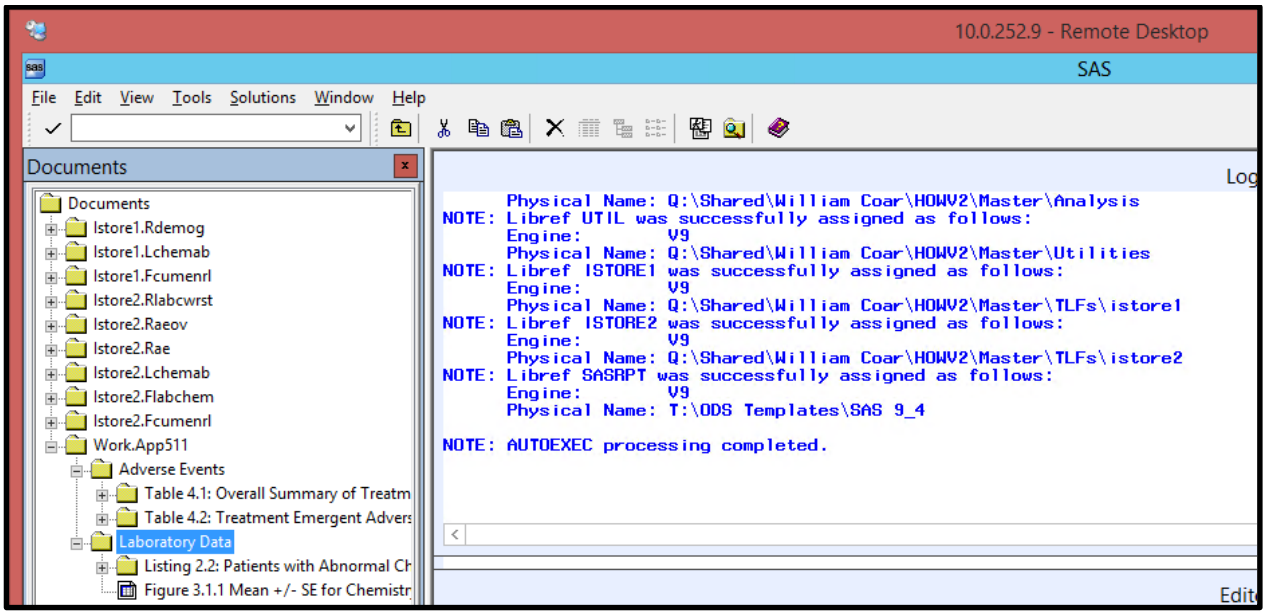
The techniques presented thus far demonstrate a programming approach to creating a restructured item store that produce a highly structured single file deliverable when replayed. Programming approaches are more desirable unless the deliverable is a one-off, and the programmer is not likely to need to re-produce the single file deliverable.

The point and click approach utilizes the ODS Document Window:



The user can then find the item stores in existing libraries. Using right-click commands to create a new directories and move tables or graphs to these new directories, the user can create a custom restructured item store that can be replayed to obtain the desired single file deliverable.

Using the ODS Document window, we can easily create the item store similar to Output #7 that when replayed re-produces the same results in Output #8.



Even though a point and click approach was used to obtain the restructured item store, the programmer still needs to replay the item store to obtain the final single file deliverable. A more step-by-step approach will be presented in the hands-on-training. While the approach is quite simple, such steps with screen shots would be too much for a written paper.

CONCLUDING REMARKS

The techniques presented allow a user to create, read, restructure, and replay item stores. They apply to both RTF and PDF, and allow for a very custom single file that includes a hyperlinked Table of Contents and/or bookmarks. With exception to updating a ToC in an RTF file, all steps to create the single file deliverable are done within SAS. All programs can continue to be run in batch mode, and automation of creating the single file deliverable is possible (see [9]).

These techniques have successfully been implemented on many projects at a CRO for the past 3 years. They continue to be used and have proven to be efficient in creating the single file deliverables. Implementing of such techniques is possible in a production setting, though the author admits that it does require some updates to standard libraries of code, primarily the code for the reporting and graphical procedures. Furthermore, transitioning to Proc SGrender and GTL may be intimidating. While this may seem overwhelming here, once the techniques are implemented in one project, they are far easier to implement in others.

We hope that attendees of this Hands-on-Training leave with a better understating of creating, restructuring, and replaying item stores as they apply to TLFs. While this single training may not be sufficient to fully implement these techniques each attendee's unique user environment, we hope attendees leave with enough confidence to consider the use of item stores as a viable option for single file deliverables.

REFERENCES

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

[2] Osowski, S., Fritchey, T. "Hyperlinks and Bookmarks with ODS RTF", Pharmasug 2006, Paper TT21

[3] Gilmore, J. "Using Dynamic Data Exchange with Microsoft Word", SUGI 22, Paper 308

[4] Gupta, A, "Watermarking and Combining Multiple RTF Outputs in SAS", PharmaSUG 2012, Paper CC06

[5] 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

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

[7] Coar, W. "TLFs: Replaying Rather Than Appending" PharmaSUG 2013, Paper 2343-2014

[8] Coar, W. "Automation of Appending TLFs" PharmaSUG 2014, Paper AD22

[9] 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

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

[11] Coar, W. "Generation of Subset Listing" WUSS 2013, Paper FP-93

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 Statistical Consulting

Axio Research, LLC
Seattle, WA 98121
Email: williamc@axioresearch.com