PharmaSUG 2023 - Paper MM-315

Macro to Automate Creation and Sync of Shell Document and TOC

Karen Walker, Walker Consulting LLC;
Jeff Cao, RealtimeCRO Inc.

ABSTRACT

In clinical trials, biostatisticians spend time in the development of Statistical Analysis Plan (SAP). The SAP contains mock-up shells for tables, figures and listings (TLF). The metadata provided by the mock shells in SAP/Shell documents includes titles, subtitles, footnotes, comments and formatting details. Statistical programmers are tasked with transferring this information from the mock-ups to an Excel file called table of content (TOC), which SAS® programs use to generate the final TFLs. Manual transfer of this metadata can be labor-intensive, and prone to errors. Both Shell document and TOC are living documents, subject to changes over time. Several SAS users have developed Excel macro and others SAS macro to facilitate the generation of TOC. This paper introduces a novel approach that does much more beyond these. An add-in for Word was developed to facilitate the creation of mock-up shell documents utilizing shell template library (STL). The relevant program index information can be exported from mock-up shells to the TOC programmatically. In addition, the add-in will identify the discrepancies between the shell document and TOC document, synchronizing them while keeping a complete change history and document version history.

INTRODUCTION

Biostatisticians play a critical role in the design, conduct, and analysis of clinical trials. Mockup Shells is one of the documents that biostatisticians develop that defines what tables, figures, and listings (TFL) are to be created by the programmers for a study milestone. The metadata provided by the mock shells include titles, subtitles, footnotes, comments and formatting details. Statistical programmers are tasked with transferring this metadata to an Excel file called TOC, which SAS programs read to generate the final TFLs. In this article, TOC and program index are interchangeable. It can be an Excel document or a database table.

Both Shell document (standalone or part of SAP), owned by biostatisticians, and TOC, owned by statistical programmers, are living documents, meaning they often get updated as needed over the time as study progresses. It is not unusual that after a while, discrepancy start to build up between the two documents. Consequently, the TFLs generated by the programmer may be deviating from what biostatisticians have wanted, which often frustrates both biostatisticians and programmers.

Ideally if there is tool that can quickly identify the discrepancies between the Shell document and TOC document and bring them back in sync, that'd be great. This is exactly one of the main drives that we developed this macro.

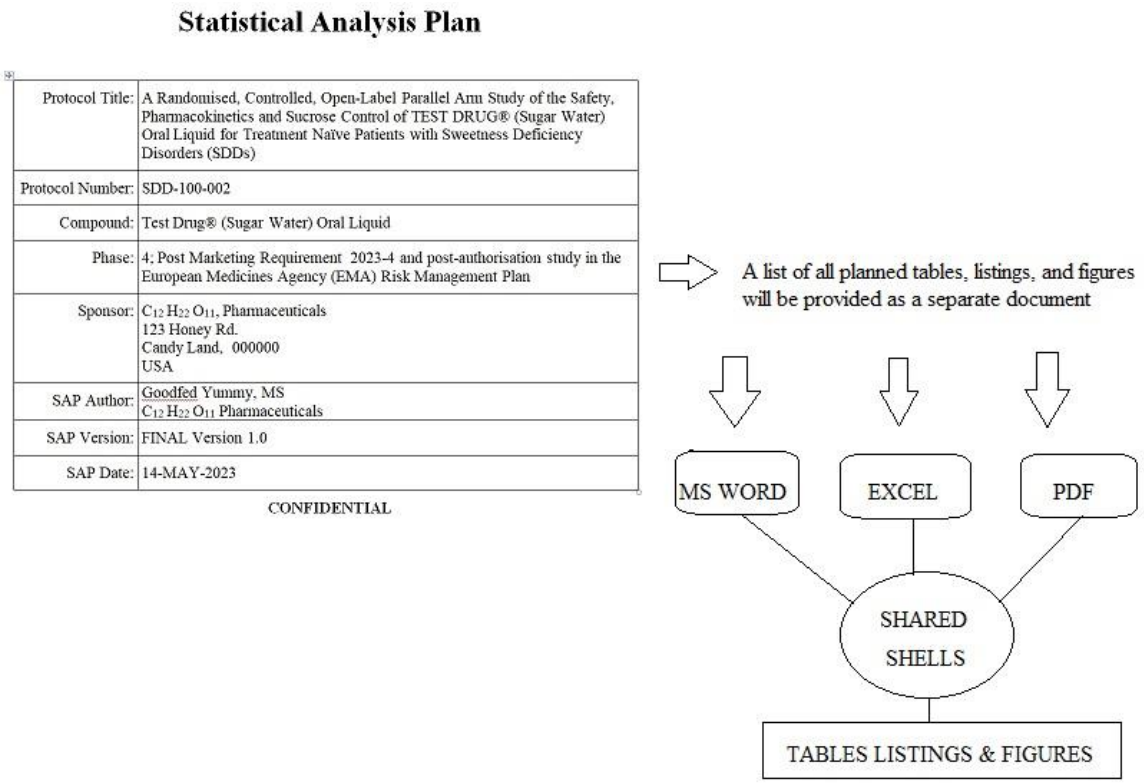
Here are the major objectives before we started the development:

1. Quick and easy to create a new Shell document from template library
2. Quick and easy to create TOC from Shell document
3. Quick and easy to create a Shell document from TOC
4. Quick and easy to keep Shell document and the corresponding TOC in sync
5. Have a complete change history

It took us over 30 months to develop the macro, with an intuitive name ShellTOC, and make it work as expected or exceeding expectation. From the efficiency it brings to us, we feel it worth the efforts.

BEFORE SHELLTOC MACRO

In the past, a Statistical Analysis Plan was written then parsed. The parsing algorithm is triggered by bookmarks for the Title, Column Headers, Body, Footnote, Programmers Notes, and Biostatistician Notes. The parsed data was used to make a program index text file to be used as input parameters to execute the plan. As the program index text file grew larger, so did the need to save the index in a separate document. This document called the shared Shell is the focus for our paper. The file type of the Shell document depends on the operating system and platform. The programming can be "one off" and very specific. The SAP file needs to be converted to Rich Text Format before it could be parsed. Unicode translation constraints were prone to errors, and program testing required much iteration to resolve the issues. See example code in the APPENDIX section.



Statistical Analysis Plan

Protocol Title:	A Randomised, Controlled, Open-Label Parallel Arm Study of the Safety, Pharmacokinetics and Sucrose Control of TEST DRUG® (Sugar Water) Oral Liquid for Treatment Naïve Patients with Sweetness Deficiency Disorders (SDDs)
Protocol Number:	SDD-100-002
Compound:	Test Drug® (Sugar Water) Oral Liquid
Phase:	4; Post Marketing Requirement 2023-4 and post-authorisation study in the European Medicines Agency (EMA) Risk Management Plan
Sponsor:	$C_{12}H_{22}O_{11}$, Pharmaceuticals 123 Honey Rd. Candy Land, 000000 USA
SAP Author:	Goodfed Yummy, MS $C_{12}H_{22}O_{11}$ Pharmaceuticals
SAP Version:	FINAL Version 1.0
SAP Date:	14-MAY-2023

CONFIDENTIAL

A list of all planned tables, listings, and figures will be provided as a separate document

Figure 1 Before ShellTOC macro

Creating a shared shell document from SAP on multiple platforms like Windows, UNIX, JULIA and others presented different challenges. After coding and testing this process over and over again, the code was committed to machine code and now is presented for use in this paper.

ShellTOC took a brand new approach to solve the above problems. It is more portable, more reliable, and can be used in a network/cloud drive or a database.

OVERVIEW – HOW MACRO WORKS

Shell document is a Word document. TOC is either Excel document or database table. We decided to develop a Word macro since it is a constant regardless of the choice of TOC. Many times, both Statistical Programmers and Biostatisticians must update the shared Shell document concurrently. The functionality is inherited from One Drive when the Word Macro is launched as shown by the code snippet below.

```
ods word file="c:\users\sasdck\onedrive - sas\shellTOC.docx";
title "Mockup shells for a study";
proc import out=work.TOC file='TOC_EXCEL.xlsx' dbms=BPS replace'
      sheet = ProgamIndex
run;
quit
```

This is the technical diagram of the macro (Figure 2). Notice that both creation and synchronization actions are bidirectional.

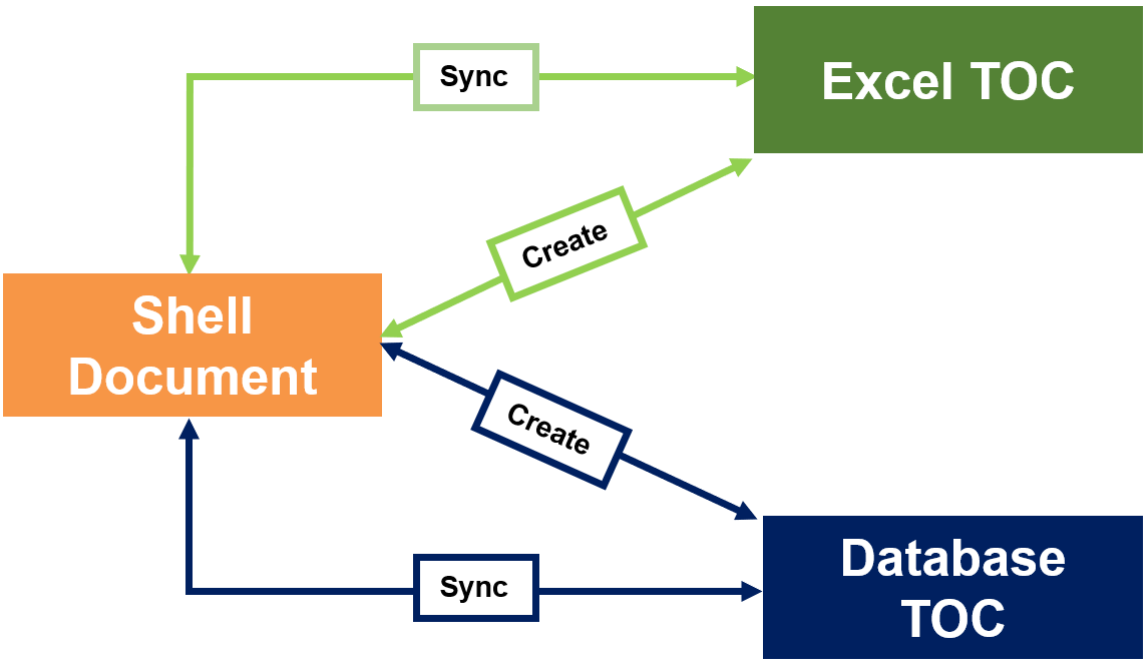


Figure 2 Overview of the macro

QUICK PREPARATION OF SHELL DOCUMENT

Standardization is key to consistent and high quality of work. But maintaining a Shell Template Library in an organization could significantly improve its standardization and operational efficiency.

Shell Template Library

It is important the requirement for a standard shell template be as less restrictive as possible.

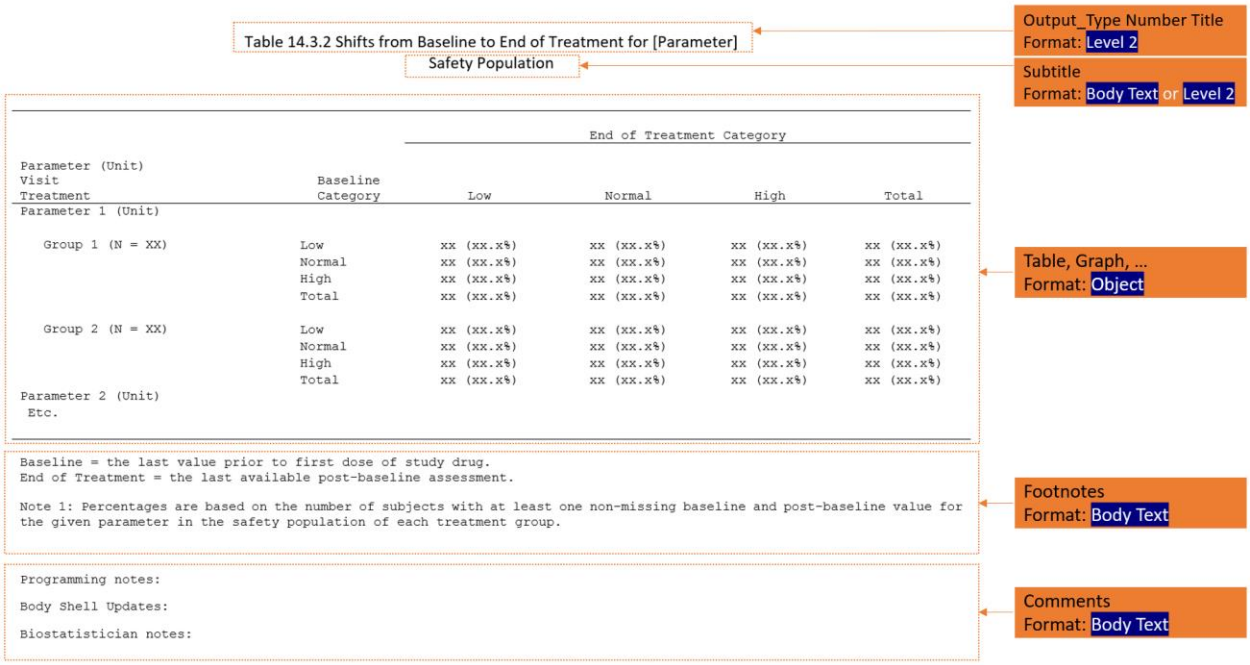


Figure 3 Standard shell template

To establish a shell template library was a lot of easier than we had expected. We collected and classified the existing tables, figures, and listings and formatted them according to the standard (Figure 3). Together they form the library that can be the asset of the company and be leveraged repeatedly.

Shell Document Creation

With the shell template library, you can start a Word document and add selected shells from the library (Figure 4). These shells can be customized once inserted to the document. It also allows adding additional shells via the traditional ways: a) Create from scratch or b) copy/paste from other sources, as long as the format is congruent to the above standard, which is not strict or complicated in anyway.

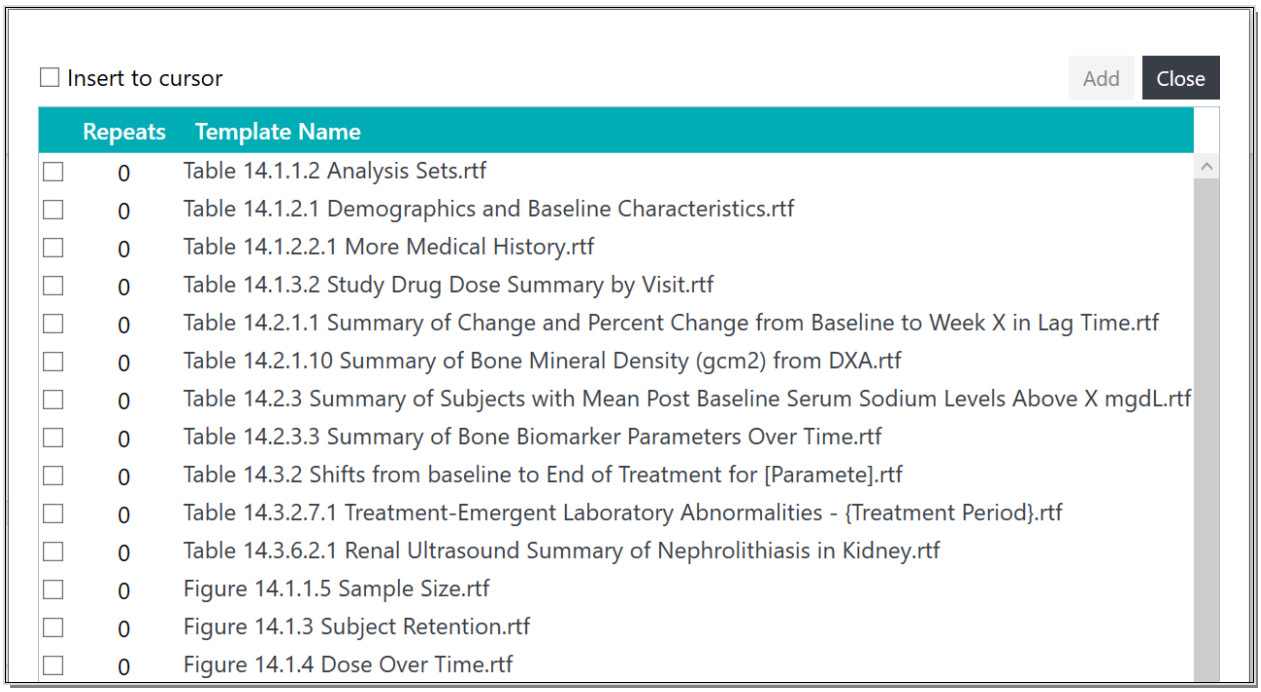


Figure 4 Design of the user interface to add shells from STL

GENERATION OF TOC

Once the shell document is ready, biostatisticians may want to create a TOC for statistical programmers. By clicking a button in the Word macro, a TOC that contains program index information is generated in seconds or minutes, depending on the number of shell instances in the Shell document. The TOC could be in the form of Excel or database table (Figure 5).

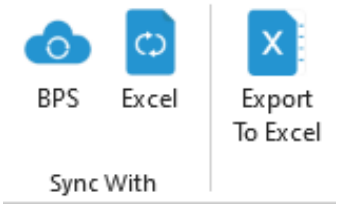


Figure 5 Generation of TOC

GENERATION OF EXCEL TOC

When the Shell document arrives at a good state, a biostatistician clicks the Export to Excel button (Figure 5) to generate an Excel TOC. The Excel TOC accurately captures all the program index information.

Alternatively, the Word macro can be launched as shown by the code snippet below.

```
ods Excel file="c:\users\sasdck\onedrive - sas\ShareTOCExcel.xlsx";   options(....)'
   proc print data= ShareTOCEXCELl
run;
```

GENERATION OF DATABASE TOC

When the Shell document arrives at a good state, a biostatistician clicks the Sync with BPS button (Figure 5), the corresponding program index information will be extracted from Shell document and populated in the database table. The advantages of using database to manage program index includes allowing multi-user concurrent editing and ability to maintain additional audit history.

SYNC BETWEEN SHELL AND TOC

Despite being a macro, the algorithm for the synchronization is way more sophisticated than what we anticipated once getting into the details. With so many scenarios to consider, the description of the algorithm itself could be of length and it might be put it in a follow up pager from this.

The Sync is so powerful that it could successfully identify all changes made in both Shell document and TOC and display them in a summary. The biostatistician or programmer can view the details of each change, add a comment, and click a button to bring both documents back in harmony.

CHANGE HISTORY

A complete and objective audit trail can transform any argument between biostatisticians and statistical programmers into harmony. The change history that ShellTOC produced has all the changes and the details of each change, including what was changed (highlighted), when it was synced and the reason behind the change if applicable.

If ShellTOC is integrated with Biostatistical Programming Studio (BPS) database, all the changes are also captured in the audit trail as well and multiple users can edit the TOC at the same time.

CONCLUSION

In summary, ShellTOC is a breakthrough macro that brought us significant productivity boost, standardization, and improved team collaboration. It

- Allows biostatisticians to quickly create a shell document from shell template library or other resources;
- Allows biostatisticians or statistical programmers to programmatically create TOC from Shell document quickly;
- Allows biostatisticians or statistical programmers to programmatically create a Shell document directly from TOC;
- Allows biostatisticians or statistical programmers to identify discrepancies between Shell and TOC, and bring the content of the two documents in congruent in minutes;
- Automatically logs complete change/version history;

REFERENCES

Goldfarb, I., and Zelichonok, E., (2020), Macro To Produce SAS®-Readable Table of Content From TLF Shells, Proceedings of the PharmaSUG 2020, Paper AD-106

Burmenko, P., and Cardozo, T., (2014) “Come Out of Your Shell: A Dynamic Approach to Shell Implementation in Table and Listing Programs”, Proceedings of PharmaSUG 2014, Paper BB17

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Please feel free to contact anyone of the authors at:

Jeff Cao
RealtimeCRO
jcao@realtimecro.com

Karen Walker
Walker Consulting LLC
kwalker.walker44@gmail.com

APPENDIX

```
options noxwait noxsync; x call "C:\Program Files\Microsoft
Office\Office11\winword.exe";
data _null_;
       wait=sleep(3);
run;
fileName wordLink dde 'WinWord|System';
data _null_;
file wordLink;
       put '[FileOpen.Name = " c:\users\sasdck\onedrive - sas\SAP.docx '"]';
       put '[FileSaveAs " c:\users\sasdck\onedrive - sas\SAP.rtf'",6]';
       put '[FileClose]';
run;
data _null_;
file wordLink;
put '[FileExit]';
run; fileName wordLink clear;
%macro rtf2sas(rtfFile,fileName,surveyNum);
fileName inRTF "&rtfFile";
data shell_toc;
       retain Table;
       infile inRTF lrecl=32632 truncover;
       input rawTxt $ 1-500;
       if index(rawTxt, " Table ") then do;
              Table='Table';
       end;
       if index(rawTxt, " Summary ") then Table='TITLE';
       if index(rawTxt, " ColumnHeader ") then Table='ColumnHeader';
       if index(rawTxt, " Body") then Table='Body'
       if index(rawTxt, " Footnote ") then Table='Footnote'
       if index(rawTxt, " Note ") then Table='ProgrammerNote';
run;

proc transpose data=selected out=record(drop=_name_) prefix=Table;
       var Table;
run;
```

Full code for call execute macro, for syncing. From Excel File and from BPS Database using BPS database reference.

```
%let rc=%sysfunc(system(start winword));
filename word DDE 'Winword|System';

data _null_;
file word;
put '[FileOpen .Name = "' c:\users\sasdck\onedrive - sas\shellTOC.docx" '"]';
put "[EditSelectAll]";
put "[EditCopy]";
put '[FileClose]';
run;

%let rc=%sysfunc(system(start excel));
data _null_;
file excel;
put '[Error(false)]';
put '[Paste]';
run;
data _null_;
put '[Error(false)]';
put '[SAVE.AS("c:\users\sasdck\onedrive - sas\ShareTOCExcel.xlsx")]';
put '[Select.Last.Cell()]';
put '[Copy]';
put '[Quit]';
run;

%macro word2sas(in=, out=temp);
options noxwait noxsync;
%let rc=%sysfunc(system(start winword));
filename word DDE 'Winword|System';
data _null_;
file word;
put '[FileOpen.Name = "' "&in" '"]';
put "[EditSelectAll]";
put "[EditCopy]";
put '[FileClose]';
put '[AppMinimize]';
run;
```

```
%let rc=%sysfunc(system(start excel));
filename excel dde 'excel|system'; data _null_;
file excel;
put '[error(false)]';
put '[paste]';
put '[SAVE.AS("c:\users\sasdck\onedrive - sas\ShareTOCExcel.xlsx ")]';
put '[Select.Last.Cell()]';
put '[Copy]';
put '[quit]';
run;
proc import out=&out datafile= c:\users\sasdck\onedrive - sas\ShareTOCExcel.xlsx ""
replace;
      getnames=no;
      mixed=yes;
run;
filename fd "c:\users\sasdck\onedrive - sas\ShareTOCExcel.xlsx ";
%let rc=%sysfunc(fdelete(fd));
filename _all_ clear;
%mend word2sas;
```