PharmaSUG 2017 - Paper AD21

Integration of multiple files into a study report using Word VBA macros

Heather Wood, Jack Shostak, Duke Clinical Research Institute

ABSTRACT

This paper provides an overview of a report integration process which uses Word Visual Basic for Applications (VBA) macros and takes multiple individual output files that have been produced by SAS® programs, or by other means, and compiles them all into an integrated study report.

The process begins with a text file (.TXT) which lists the names and paths of all the outputs that are desired in the study report, in the order in which they need to appear. The report integration VBA macros use this text file to identify and process each output for the study report. The section breaks which SAS® PROC REPORT puts at the end of every page of an output are replaced with page breaks. Then the outputs are all compiled together with a new section break separator between each output. This results in a study report compiled from several outputs with one section break between each output, rather than one section break between each page of the report. After the report is compiled, other report integration macros update the page numbering, create and append a Table of Contents, then create a bookmarked PDF version of the study report.

This paper is for anyone who creates and compiles outputs for study reporting, who uses SAS® version 9 or above. Some familiarity with Visual Basic for Applications may help but is not necessary.

INTRODUCTION

This paper is intended to give some inspiration to programmers and statisticians in clinical research who have to produce a single integrated report by compiling many separate outputs together, then create a dynamic table of contents for that report. There can be anything from a couple dozen to a couple hundred outputs needing compilation into an integrated report, consisting of any combination of tables, listings, figures, divider pages, cover pages or addendum pages.

Manual compilation of an integrated study report can take hours, and perhaps an output would get overlooked or slotted into the wrong place. One goal in the automation of this report integration process was to improve accuracy by having the machine compile the outputs, while working at speed.

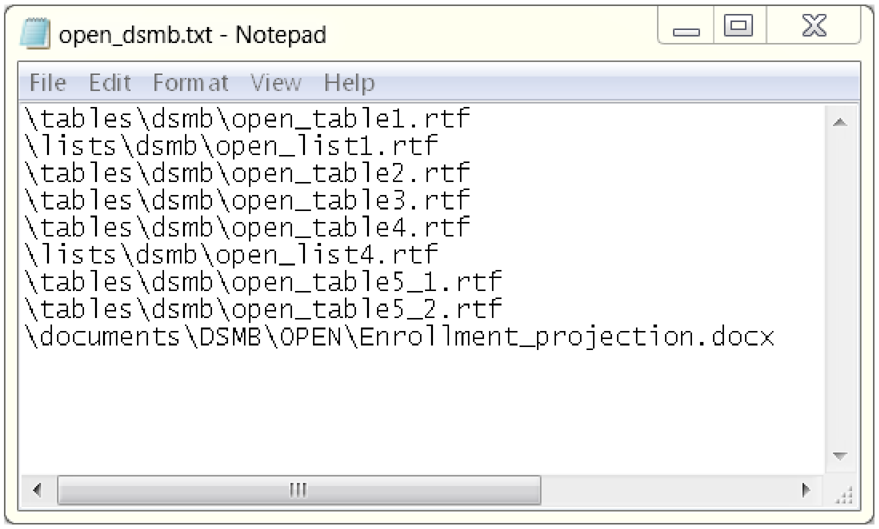
Although there exist some solutions already, such as SAS® PROC DOCUMENT, or various SAS® programming workarounds for Word's limitation of using the {PAGE} function for both overall and within-section page numbering, sometimes these solutions can be more complex for the end user than we would like. Another goal was to have an easy-to-use method where the end user didn't need to be a programmer. They only needed to know how to use Word; thus a statistician who was not a programmer could compile a report for themselves without needing to find a programmer each time. Also, the statistician could create additional documents for the report using MS Office rather than SAS® and compile them into the report easily with no need to adjust any SAS® programming for report creation.

Since this paper is intended to inspire those who would like to try developing VBA tools for working with the MS Office outputs that are our typical end products, to benefit the most from this paper it would be ideal to have some knowledge of VBA coding. However, anyone who knows MS Office and has programming experience can get something from this paper.

INTEGRATION .TXT TEXT FILE – THE BLUEPRINT FOR THE VBA MACROS

The first step in the report integration process is to create a TXT text file that lists the file name and file path of each output to be compiled into a study report. This TXT file can be created using Notepad.

The VBA macros use this TXT file to identify and process the outputs needed in the study report. The outputs are compiled together in the order that they have been listed; thus using a TXT file in this way gives you the flexibility to select any outputs you wish, from any folder within the project drive, and have them compiled into any order you wish.



```
\tables\dsmb\open_table1.rtf
\lists\dsmb\open_list1.rtf
\tables\dsmb\open_table2.rtf
\tables\dsmb\open_table3.rtf
\tables\dsmb\open_table4.rtf
\lists\dsmb\open_list4.rtf
\tables\dsmb\open_table5_1.rtf
\tables\dsmb\open_table5_2.rtf
\documents\DSMB\OPEN\Enrollment_projection.docx
```

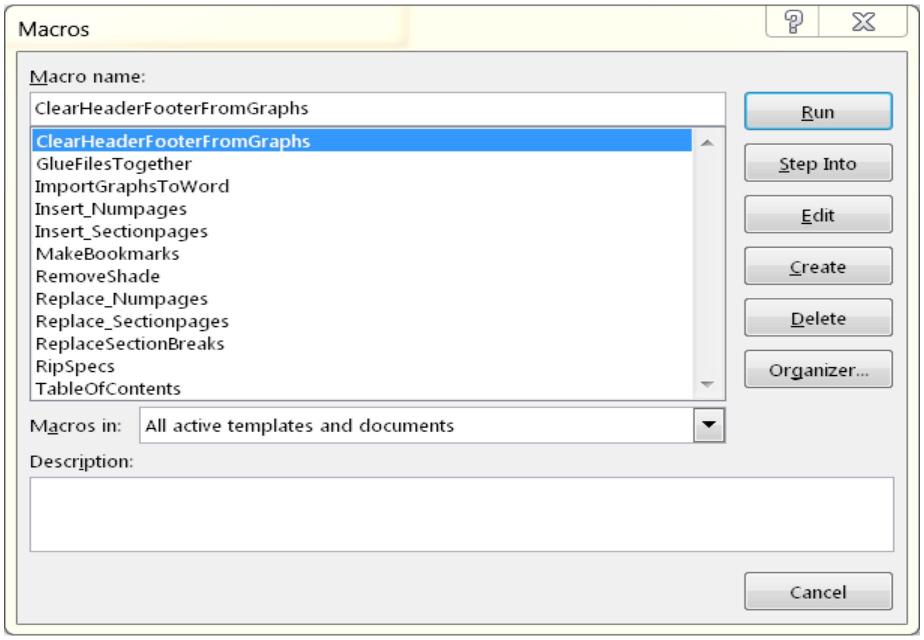
Display 1. TXT File for DSMB Open Report

Often there are several TXT files created in a trial, one for each of the various reports needed. You might have one for a data monitoring committee report, one for an annual safety report, one for an interim report, and so on. Any time you need to create a report, simply select the TXT file designed for that particular type of report when running the VBA macros.

USING ANY ONE OF THE VBA MACROS WITHIN WORD – A BRIEF HOW-TO

The process for accessing and running the VBA macros within Word is quite straightforward. Here are the necessary steps:

1. Open Word. For some macros, you do not need to open any particular document. For other macros, you need to open the document that the macro is to run on.
2. On the **View** tab, click ***Macros***, then click on ***View Macros***. The **Macros** window opens.



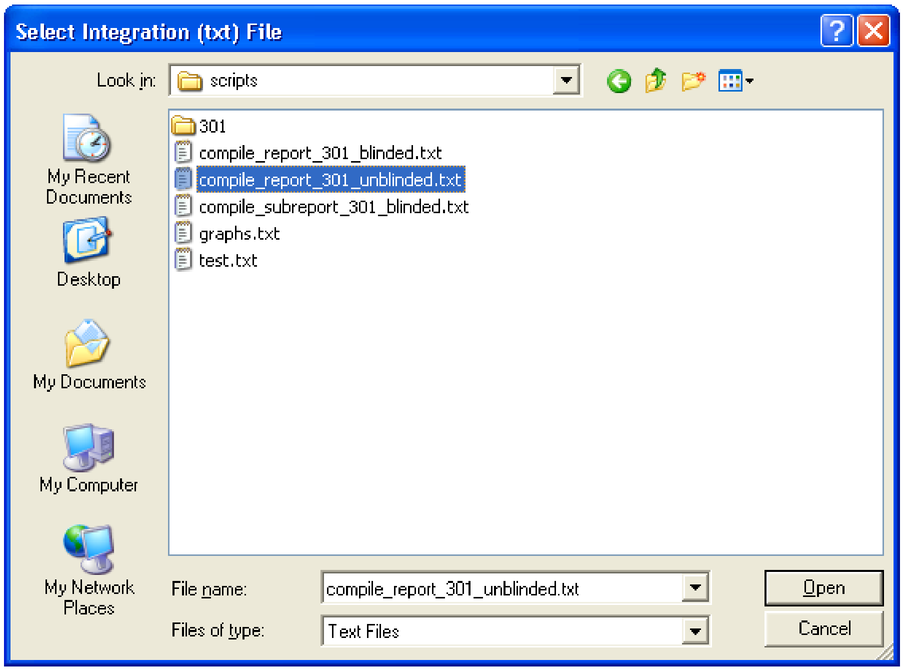
Macro name: ClearHeaderFooterFromGraphs

ClearHeaderFooterFromGraphs
GlueFilesTogether
ImportGraphsToWord
Insert_Numpages
Insert_Sectionpages
MakeBookmarks
RemoveShade
Replace_Numpages
Replace_Sectionpages
ReplaceSectionBreaks
RipSpecs
TableOfContents

Macros in: All active templates and documents

Display 2. Macros Window

3. Click on a macro name to select that macro, then click ***Run***.
4. A popup window asks you to select an integration TXT file. Navigate to your TXT file and then click ***Open***.


Select Integration (txt) File
Look in: scripts
My Recent Documents
Desktop
My Documents
My Computer
My Network Places
301
compile_report_301_blinded.txt
compile_report_301_unblinded.txt
compile_subreport_301_blinded.txt
graphs.txt
test.txt
File name: compile_report_301_unblinded.txt
Files of type: Text Files
Open
Cancel

Display 3. Popup Window for Selecting TXT Files

5. The VBA macro runs, using the outputs listed in the TXT file. When finished, a popup box saying “Run complete” appears. Click OK. Your study report is ready for you to look over.

REPORT INTEGRATION MACRO SUITE

REPLACE SECTION BREAKS WITH PAGE BREAKS MACRO

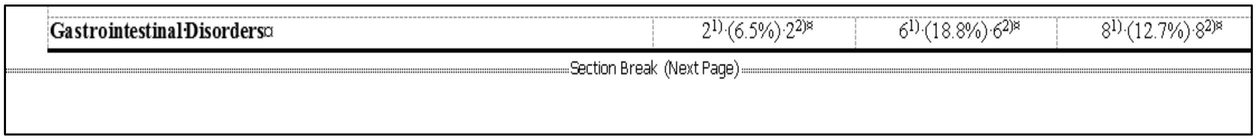
When SAS® produces an output with PROC REPORT, there is a section break on the end of every page of that output. Thus every page of a table, for example, is in its own section. If you need to edit a footnote for that table then you have to edit it in every section, which is to say, on every page. Also, when creating bookmarking or tables of contents, you would get one entry for each page of the table rather than one entry for the entire table. To avoid this, the section breaks need to be replaced with page breaks. The macro ***ReplaceSectionBreaks*** accomplishes this.

When ***ReplaceSectionBreaks*** runs, it uses a TXT file to identify the outputs where section breaks need replacing. If an output in the TXT file has no section breaks, for example it is only one page long, or section breaks were removed previously, then the macro passes over it and continues to the next one. This is the portion of the VBA code which executes every time a section break is found:

```
'Insert pagebreaking in front of each section break
For iSec = wdDocument.Sections.Count To 2 Step -1
   With wdDocument.Sections(iSec).Range
      .Collapse Direction:=wdCollapseStart
      .InsertBreak Type:=wdPageBreak
   End With
Next iSec
'Delete the section break
With wdDocument.Content.Find
   .ClearFormatting
   .Text = "^b"
```

```
      .Forward = True
      Do While .Execute = True
         .Parent.Delete
      Loop
   End With
   wdDocument.Save
```

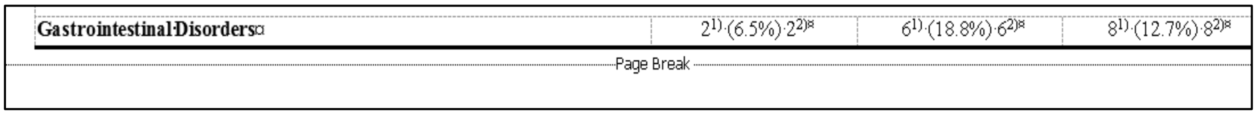
In Display 4 you can see the section break before it is replaced which is shown in Display 5.



Gastrointestinal·Disorders¤	2$^{1)}$·(6.5%)·2$^{2)}$¤	6$^{1)}$·(18.8%)·6$^{2)}$¤	8$^{1)}$·(12.7%)·8$^{2)}$¤

Section Break (Next Page)

Display 4. Original Section Break to be Replaced



Gastrointestinal·Disorders¤	2$^{1)}$·(6.5%)·2$^{2)}$¤	6$^{1)}$·(18.8%)·6$^{2)}$¤	8$^{1)}$·(12.7%)·8$^{2)}$¤

Page Break

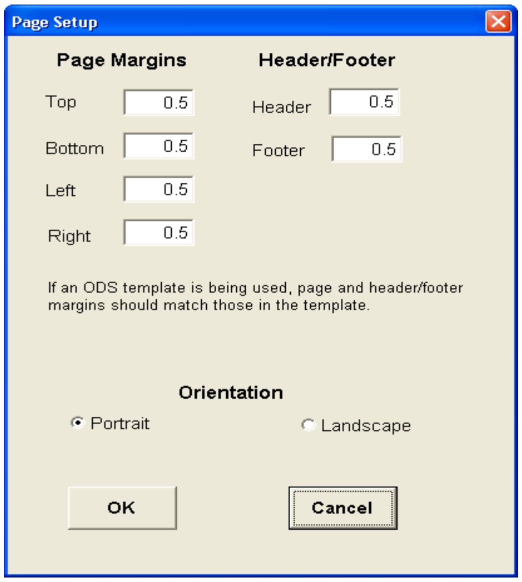
Display 5. The Replacement Page Break

GLUE FILES (OUTPUTS) TOGETHER MACRO

There are many tables, listings, and figures produced for compilation into various study reports during a trial. The ***GlueFilesTogether*** macro uses a TXT file that defines the content of a report, and selects all the outputs listed in that TXT file to ‘glue’ (compile) together.

The end result of running ***GlueFilesTogether*** is an integrated study report consisting of all the required outputs glued together in the order they are listed in the TXT file. Each output is separated from the next by section breaks.

GlueFilesTogether uses the margins and orientation that the user specifies. A popup window collects this information from the user as seen here:



Page Setup

Page Margins

Top 0.5

Bottom 0.5

Left 0.5

Right 0.5

Header/Footer

Header 0.5

Footer 0.5

If an ODS template is being used, page and header/footer margins should match those in the template.

Orientation

[✓ Portrait] Landscape

OK Cancel

Display 6. Popup Window for Margins and Orientation Collection

If divider pages are wanted, or a cover sheet for the report, or any addenda or narratives, then these can be created and saved using MS Office. Their names and paths can be added to the TXT file also. Then you can run ***GlueFilesTogether*** to compile cover sheets, divider pages, or indeed any other document, into the study report alongside the tables, listings, and figures outputs with no need to manually insert them afterwards.

This is the portion of the VBA code that references the TXT file and glues the listed outputs together:

```
'Open the relevant TXT file of documents to load
lngFileNum = FreeFile
Open strPath & "\" & strFileName For Input As #lngFileNum
Do While Not EOF (lngFileNum)
   'This steps through your TXT line by line, storing the output names
   Line Input #lngFileNum, strTextLine
      'Loop through the files in the TXT and glue them with sectionbreaks
      Selection.InsertFile FileName:=strPath & strTextLine, _
      Range:="", ConfirmConversions:=False, _
      Link:=False, Attachment:=False
      Selection.InsertBreak Type:=wdSectionBreakNextPage
Loop
Close #lngFileNum
```

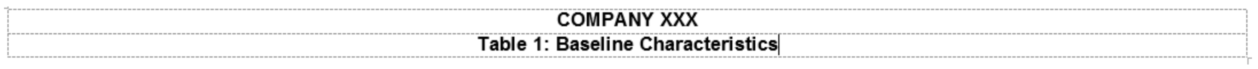
TABLE OF CONTENTS MACRO

After a study report has been created, the next step is usually to give it a table of contents (TOC). The macro ***TableOfContents*** helps you here. After running this macro you will have created a TOC and appended it to the front of your report.

If the report has a cover page then the TOC appears after this cover page, if not then the TOC appears as the top page(s) of the report.

The TOC lists each output in the report with its page number, excluding the cover page. The page number is a real page number; if an output starts on the 8th page of the report then the page number will be ‘8’. The TOC is also interactive, clicking on the entry in the TOC for an output takes you to the first page of that output.

The ***TableOfContents*** macro works by embedding a TOC field code into the first page of each section of the report, which corresponds to the first page of each output in the report. The TOC field code name is the title of the output which is harvested automatically by the macro from the header fields of that output. There is an option to exclude portions of a title if wanted. For example, a company name displayed in the header fields can be excluded from the TOC field code name. For the example output header shown below in Display 7, the TOC field code could be “Table 1: Baseline Characteristics”, since the “COMPANY XXX” part could be excluded if not wanted.


COMPANY XXX
Table 1: Baseline Characteristics

Display 7. Example Header

After the creation and embedding of the TOC field codes into each output, the macro creates the TOC page. Then it uses the text from the TOC field code names to label each TOC entry on the page. There is a permanent link between each TOC field code and its corresponding TOC entry such that any time TOC field codes are updated, then the TOC entry on the TOC will also be updated.

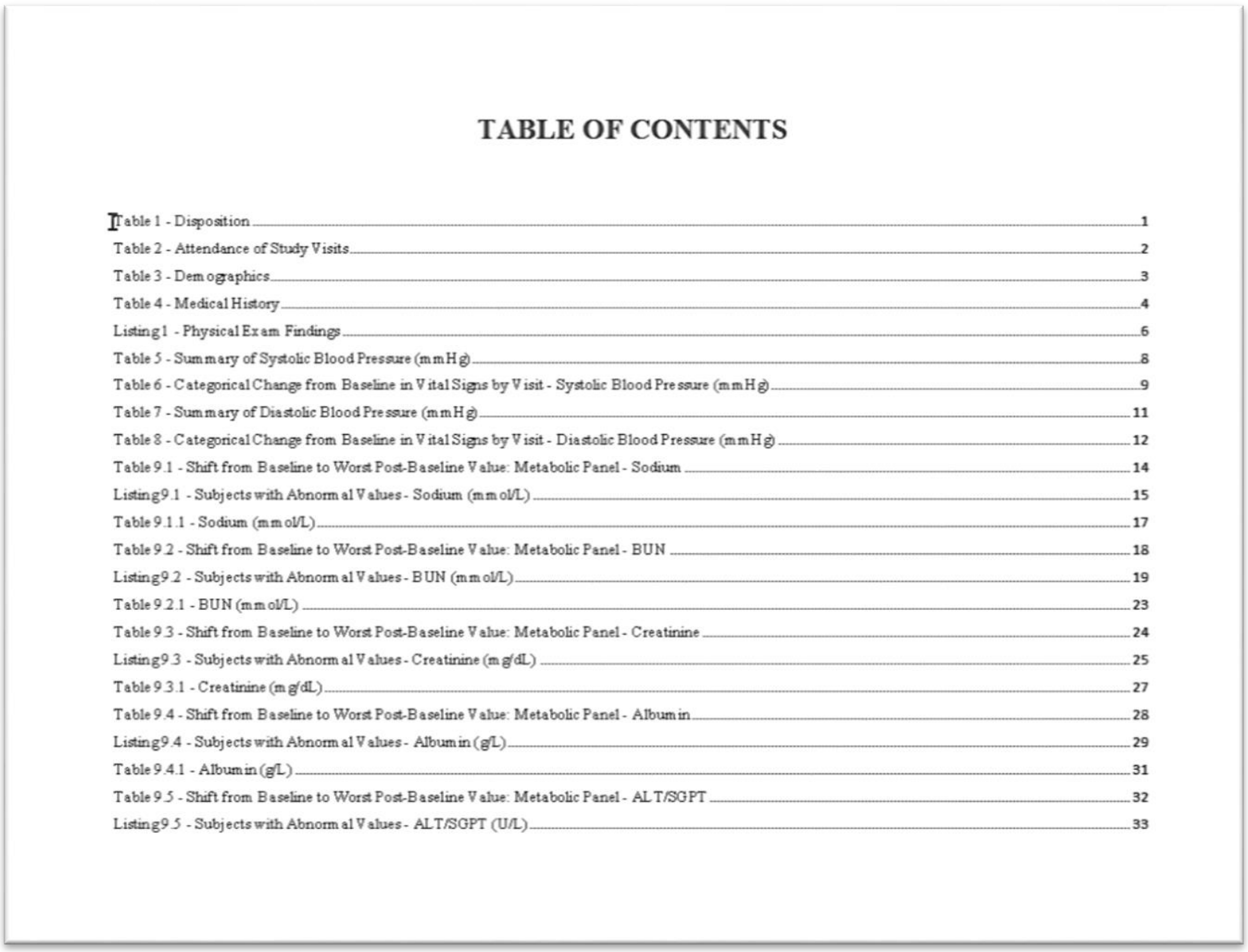


TABLE OF CONTENTS

Display 8. Page from a Table of Contents Produced by the *TableOfContents* Macro

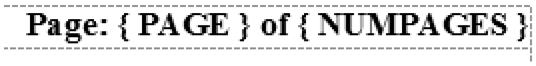
If a report is updated with new outputs added or deleted from it, then you can re-run the ***TableOfContents*** macro to delete the old TOC and create a new one based on the updated report contents. Since it no longer takes a couple of hours (or an afternoon!) to create a TOC, changes to a report are not such an administrative burden as they once were. This macro creates a TOC in less than a minute.

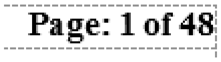
INSERT_NUMPAGES & INSERT_SECTIONPAGES MACROS - RESET PAGE NUMBERING IF NEEDED

Sometimes a newly compiled study report needs to have its page numbering reset after compilation. A cover sheet and a table of contents might need to be excluded from the overall page numbering of the document, or perhaps section page numbering has been used in addition to overall page numbering and so the resulting invalid section numbers need correcting from e.g. "Page 5 of 2".

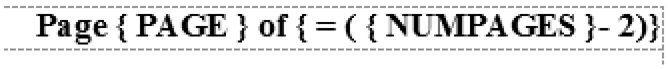
There are two page numbering macros to help here: ***Insert_Numpages*** which will reset the overall page numbering, and ***Insert_Sectionpages*** which will reset section page numbering.

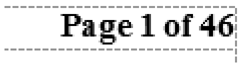
The ***Insert_Numpages*** macro does two jobs. First, it counts the number of cover pages and tables of contents pages, and then subtracts the total from the automatic Word-generated function {NUMPAGES} wherever {NUMPAGES} appears in the report. Second, it resets the {PAGE} function to start from 1 on the first page after the cover page and the tables of contents. Here is a before and after comparison of page numbers on the first page of a study report with one cover page and one table of contents page:


Page: { PAGE } of { NUMPAGES }


Page: 1 of 48

Display 9. Page Numbering Before Running *Insert_Numpages*


Page { PAGE } of { = ({ NUMPAGES }- 2)}

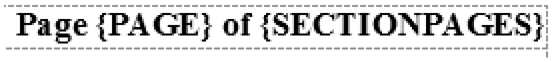

Page 1 of 46

Display 10. Page Numbering After Running *Insert_Numpages*; Cover Sheet and TOC are Excluded

Before running ***insert_numpages*** you have "Page 1 of 48" appearing on the cover sheet. After running ***insert_numpages*** there is no page numbering on the cover sheet since both the cover sheet and the TOC page are now excluded from the numbering system. "Page 1 of 46" displays on the third page of your report, directly after the TOC page.

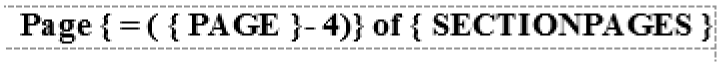
The ***Insert_Sectionpages*** macro resets the automatic word-generated {PAGE} function for each section after the first one. For each succeeding section, it keeps a running total of the pages from all preceding sections, and subtracts that total from {PAGE}.

Here are the {PAGE} and {SECTIONPAGES} functions as they normally appear. This is not changed for the first section of a report:


Page {PAGE} of {SECTIONPAGES}

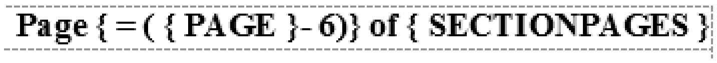
Display 11. Section Page Numbering: 1st Section

Here is the {PAGE} function for the second section of a report, after ***Insert_Sectionpages*** has adjusted it by subtracting "4" to account for the four pages in the first section:


Page { = ({ PAGE }- 4)} of { SECTIONPAGES }

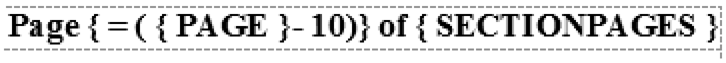
Display 12. Section Page Numbering: 2nd Section, Subtracting 4 Pages of 1st Section

Here is the {PAGE} function for the third section of a report, after ***Insert_Sectionpages*** has adjusted it by subtracting "6" to account for the four pages in the first section and the two pages in the second section:


Page { = ({ PAGE }- 6)} of { SECTIONPAGES }

Display 13. Section Page Numbering: 3rd Section, Subtracting 4 Pages of 1st + 2 Pages of 2nd Section

Here is the {PAGE} function for the fourth section of a report, after ***Insert_Sectionpages*** has adjusted it by subtracting "10" to account for the ten pages from the three earlier sections:


Page { = ({ PAGE }- 10)} of { SECTIONPAGES }

Display 14. Section Page Numbering: 4th Section, Subtracting 4 Pages of 1st + 2 Pages of 2nd + 4 Pages of 3rd Sections

MAKEBOOKMARKS MACRO - BOOKMARK THE STUDY REPORT AND CREATE A BOOKMARKED PDF FILE

The last step in our report integration process is to create a bookmarked PDF file. The macro ***MakeBookmarks*** can help with this. First it constructs a named bookmark for each output in the study report. Then the bookmark is placed at the top of the first page of the output. Finally the PDF file is created and the bookmark and its name is carried through from the Word file to the PDF file.

The name of the bookmark is usually the title of the output. Sometimes the output title is very long, or not optimally-worded for a bookmark; if this happens the bookmark will need editing later after the PDF has been created.

Here is a portion of the VBA code that clears any existing bookmarks, creates a new named bookmark, and then inserts it into the document. The function ***fGetTitle*** that is called here is the piece which constructs the name:

```
'Delete existing bookmarks if any
If oSec.Range.Bookmarks.Count > 0 Then
   For i = oSec.Range.Bookmarks.Count To 1 Step -1
      oSec.Range.Bookmarks(i).Delete
   Next
End If

'Replace invalid characters for Word bookmarks with an underscore
bkmkArray = Array("<", ">", "/", "\", "|", "[", "]","{", "}", ".", "-", "
", ",", ";", ":", "*", "'", "#", "%", "@", "^", "&", "(", ")", "+", "=")

'Get text from header to make bookmark name - call the function for this
sBookmarkName =
fGetTitle(oSec.Headers(wdHeaderFooterPrimary).Range.Tables(1),
sTextExclude)
   For i = 0 To UBound(bkmkArray)
      sBookmarkName = Replace(sBookmarkName, bkmkArray(i), "_")
   Next
'Create the bookmark, attach name to it and insert it
Set rngWhere = oSec.Range
rngWhere.Collapse wdCollapseStart
rngWhere.Bookmarks.Add Name:=sBookmarkName, Range:=rngWhere
```

After the ***MakeBookmarks*** macro runs, the new named bookmarks are displayed in your Word study report as I-beams. See the screenshot below which shows part of a table of contents in Word with the I-beam bookmark visible at the extreme left of the first line.

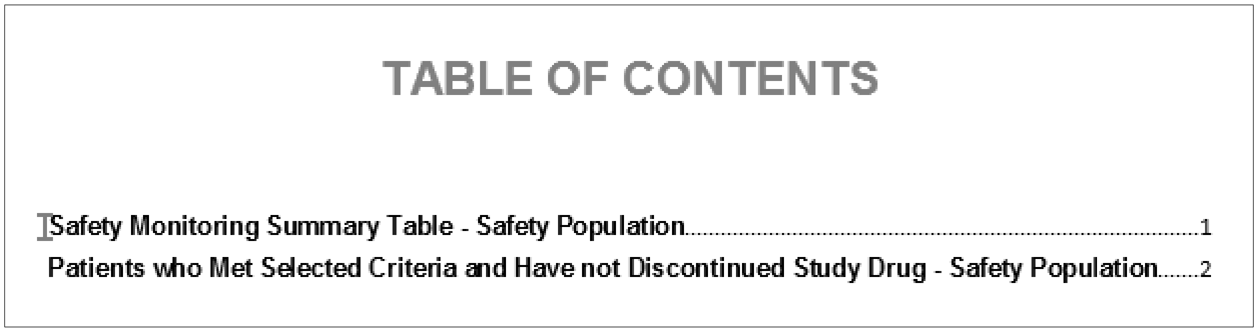


TABLE OF CONTENTS

Display 15. Bookmark Placed Upon Table of Contents Page

The name of the bookmark is not displayed in the Word study report file, however it becomes visible in the bookmark section (on the left) of the PDF study report file.

Once you have your PDF file you may need to manually edit some bookmark names for optimal presentation. This is easy to do by right-clicking on the bookmark name, selecting ***Rename*** from the popup menu, editing as needed and then clicking ***Save***.

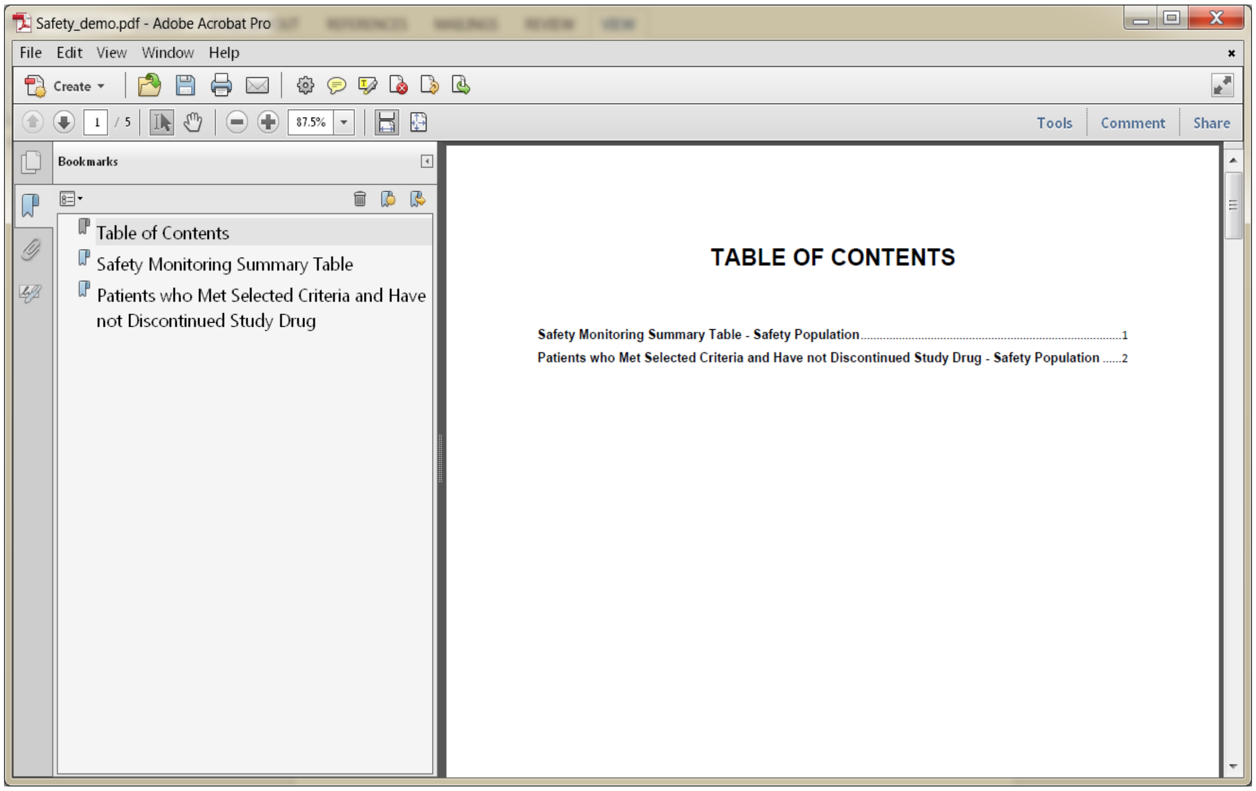

Safety_demo.pdf - Adobe Acrobat Pro
File Edit View Window Help
Create
1 / 5
87.5%
Tools
Comment
Share
Bookmarks
Table of Contents
Safety Monitoring Summary Table
Patients who Met Selected Criteria and Have not Discontinued Study Drug

TABLE OF CONTENTS

Safety Monitoring Summary Table - Safety Population........1
Patients who Met Selected Criteria and Have not Discontinued Study Drug - Safety Population 2

Display 16. A Bookmarked PDF File

CONCLUSION

This report integration process was developed over a period of time in fits and starts, partly to automate often burdensome administrative tasks involved in study report creation, and partly to speed up the reports creation process. As one component of report creation after another was automated, they all slotted together to form a suite of VBA macros that became widely used within our department.

I deliberately kept each VBA macro as a single-use macro covering only one part of the report integration process to give maximum flexibility to use only what is needed. For example, a study report might not need to be converted to a bookmarked PDF, but needs a TOC. Another study report might not need a TOC but needs to have the section page numbering reset after being compiled. Keeping a suite of macros helps with customizing each study report as needed. It also made it easier to keep developing and updating each macro in line with user suggestions without affecting another part of the process.

Other smaller macros were developed as add-ons; such as one that stamps a draft study report on every page with a DRAFT watermark, or one that enables the insertion of page numbering into a report where it had not been originally required. The VBA macros described here are the most common ones used for report integration, though.

Although there are now other options available such as SAS® Proc Document, some of these VBA macros pre-dated these options. When ***GlueFilesTogether*** was first written there was no easy way to compile documents. ***GlueFilesTogether*** has now been used for long enough to have had a lot of kinks worked out, not to mention being refined along the way to support several different formats and types of reports while still being easy to use for the end-user. Thus many users preferred to keep using them, especially those users who are not so adept with SAS® or who are using non-SAS generated documents in addition to SAS outputs.

From a coding point of view, sometimes when working with RTF or DOCX documents it was simply more straightforward to write VBA code to solve a problem I was having than to write SAS code. It was possible to code the solution in either language, however VBA is native to MS Office while SAS® is not; so a complex SAS program could become a shorter and simpler VBA program when my need was to manipulate Word files rather than SAS® files.

ACKNOWLEDGMENTS

I would like to thank all of my colleagues at the Duke Clinical Research Institute who have willingly been guinea-pigs during the continuing evolution of these VBA macros and who have given me many suggestions for improvements. They would not be as well developed as they are without this input.

RECOMMENDED READING

- *Writing Word Macros: An Introduction to Programming Word Using VBA*
- *VB and VBA In a Nutshell*

CONTACT INFORMATION

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

Heather Wood
heather.wood@duke.edu

Jack Shostak
jack.shostak@duke.edu