Combine PDFs in Submission-ready Format Quick and Easy

Robin Wu, Lili Li, and Steven Huang, PTC Therapeutics

ABSTRACT

Portable Document Format (PDF) is an essential component in the regulatory submission package (U. S. Food and Drug Administration, 2013; International Council for Harmonisation of Technical Requirements for Pharmaceuticals for Human Use, 2017). The conventional approaches of combining PDFs include using a SAS macro or utilizing a third-party stand-alone tool. Both approaches are usually time-consuming, unstable, and technically challenging in real practices. In the meantime, however, very limited research and utilities have been shared in the SAS community on how to combine PDFs properly and efficiently in a submission-ready format. This paper will address the gap here by introducing a novel approach that implements both a SAS macro and two third-party utilities (a Visual Basic application and a Java application) to make the PDF combination quick, easy, and submission-ready.

KEYWORDS

Combine PDFs, Submission-Ready

INTRODUCTION

U. S. Food and Drug Administration (FDA) and International Council for Harmonisation of Technical Requirements for Pharmaceuticals for Human Use (ICH) have very specific requirements for the combined PDF files in submission packages (U. S. Food and Drug Administration, 2005; U. S. Food and Drug Administration, 2013; International Council for Harmonisation of Technical Requirements for Pharmaceuticals for Human Use, 2015; International Council for Harmonisation of Technical Requirements for Pharmaceuticals for Human Use, 2017). For instance, both FDA and ICH request that the combined PDF file needs to include a Table of Contents (TOCs) page where there are proper hyperlinks for documents of more than five pages. In the combined PDF, FDA and ICH also require bookmarks for each table, figure, or listing (TFL) listed in the TOCs. Even for the TFL outputs in submission package that are produced in non-PDF file formats, FDA instructs that those files "*should also be provided in PDF format*" (U. S. Food and Drug Administration, 2013, p. 2).

To convert and combine TFL outputs into one PDF file, conventional methods generally come in two approaches. The first common approach is to directly combine in SAS. One frequently used method is to produce SAS Item Store catalogs using the ODS Document statement while producing the TFLs. Then a stand-alone SAS macro would be called to read in the SAS Item Store catalogs to be combined into one PDF file at the end. The other common approach is to utilize third-party applications. For instance, Adobe Acrobat is one of the most popular applications that allow the users to manually select, convert, and combine individual files from WORD documents format into combined files in PDF format.

However, both conventional approaches pose significant challenges in practical uses. One most outstanding challenge is that both approaches are time-consuming and resource-consuming, especially when combining a large number of TFL outputs. It is not unusual for the SAS Item Store catalogs method to take hours in converting and combining a couple of hundreds of TFL outputs. Similarly in combining files using Adobe Acrobat, it is not only time-consuming but also very often unstable in processing due to exhaustion of computing resources. The second challenge is in compliance with the regulatory guidelines. For instance, when using third-party applications such as Adobe Acrobat, the final combined PDFs are

not submission-ready as it will not automatically generate the proper TOCs page required by the regulatory agencies mentioned above.

Although it has been generally established that PDF is an integral part of the regulatory submissions while there are many challenges in conventional approaches as discussed above, not much attention has been shed on PDF creation for submission packages, especially on the combined PDF files (Poppe, 2005; Shao & Zhang, 2023; Xin & Wang, 2016). This paper introduces a SAS macro that utilizes a Visual Basic application and a Java application to efficiently create the combined PDF document that complies with regulatory requirements for submission purpose.

Compared to conventional methods, the new SAS macro is 1) highly efficient by significantly reducing the processing time by at least 60% to 90%. The larger of combined file's size, the more in reduction of the processing time; 2) very easy to use as it removes all complex macro set-ups; 3) is resource-effective where the processing does not require much computing resource, thus make it smooth and stable to execute even with large number of TFL outputs to be converted and combined.

PROCESS FLOWCHART

The graph below demonstrates the process flow in and behind the SAS macro itself. The whole utility here can be divided into two components: the SAS macro running on the front-end, and the Visual Basic application and Java application running on the back-end. More details about each of the components will be discussed below.

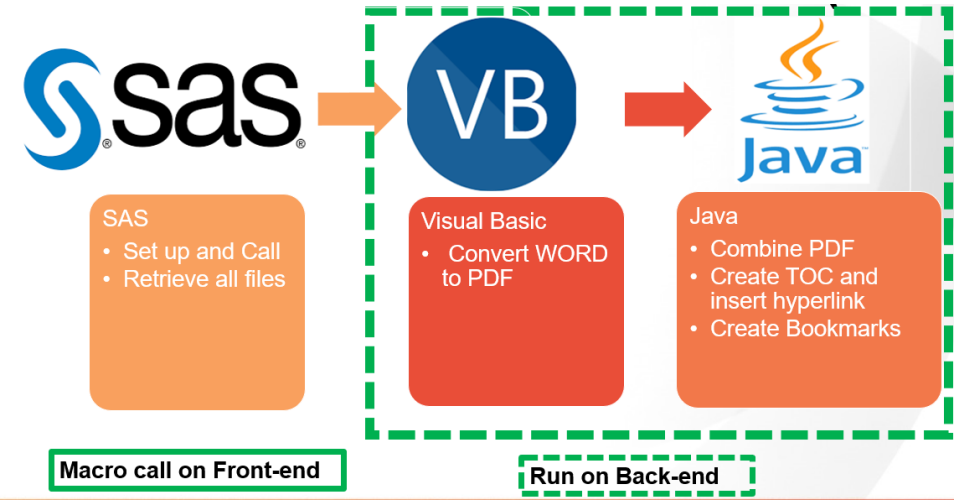

Figure 1. The Process of Combined PDF Production

FRONT-END COMPONENT: SAS MACRO

The SAS macro is the front-end component in this application. As the first step of the process, it retrieves all the TFL output files based in the designated folder path to determine if the TFL output files in WORD document format have corresponding PDF files. If no corresponding PDF file to a particular WORD output (in rft format) is found, the macro will report such case into a temporary data set (*topdf*) where all the TLF

reported will be converted into PDF format at a later step of the application. Details of the codes to realize this function are shared below:

```
filename rtf pipe %sysfunc(quote(dir "%sysfunc(pathname(trtf))\*.rtf" /b));
filename pdf pipe %sysfunc(quote(dir "%sysfunc(pathname(trtf))\*.pdf" /b));

data dirrtf;
    infile rtf pad ;
    length fname $ 100  filex $100;
    input fname $ char100.;
    file=input(scan(fname, 1, ' '), ?? $char100.);
    filex=tranwrd(file,  '.rtf', '');
    /*if ^missing(file);*/
    if substr(fname, 1, 1)^='' ;
    keep file filex ;
run;
proc sort; by filex; run;

data dirpdf;
    infile dirpdf pad;
    length fname $ 100 filex $100;
    input fname $ char100.;
    file=input(scan(fname, 1, ' '), ?? $char100.);
    filex=tranwrd(file,  '.pdf', '');
    if substr(fname, 1, 1)^='' ;
    keep file filex;
run;
proc sort; by filex; run;

data topdf;
   merge dirrtf(in=a keep=filex file) dirpdf(in=b keep=filex);
   by filex;
   if a=1 and b=0;
run;
```

Note, `trtf` is path of the output's location defined by the `libname` statement; Using `pathname` function can convert it back into a macro variable to be directly used in the filename statement as illustrated above.

Using a simple `if a=1 and b=0` statement in the last data step above, the `topdf` data set will yield a list of outputs where any WORD output is still missing corresponding PDF format files and need to be converted into PDFs before combining all the PDF files. Details of rtf to PDF conversion will be further discussed in the back-end components section below.

If WORD output needs to be converted into PDF, the next step of the application is to determine the scope of PDF to be combined. This is achieved by a simple SAS macro to utilize an external excel spreadsheet (TOC file) and generate a text file to be fed into the back-end Java API tool which will be presented with more details below.

As shown in the codes below, only two macro parameters are needed. The `tocfile` denotes the file name of the external excel spreadsheet where it contains all the outputs' proper table number, table title, and output names. The `subset` macro variable is optional only when a separate PDF file for a subset of the outputs is needed.

The page header, font size, font color, and page number location are defined in the filename statement after the back-end Java API tool (*MergePdf.exe*) is called in this macro. Details of this SAS macro codes can be found below.

```
%macro combinePDF(tocfile=, subset= );
  x %quote("P:");
  x %quote(cd "P:\Macro_Library_Development\Combine_PDF\");

  data temp;
    set alltocx;
    &subset;
  run;

  data _null_;
    set temp;
    file "%sysfunc(pathname(trtf))\&tocfile..txt" lrecl=3000;

    if _n_=1 then do;
      put "File Name, TFL No., TFL Title";
    end;

    varlen=length(tocx);
    put tocx $varying3000. varlen;
  run;

  filename cmppdf pipe
  %sysfunc(quote(P:\Macro_Library_Development\Combine_PDF\MergePdf.exe -
  s="%sysfunc(pathname(trtf))\&tocfile..txt" -
  d="%sysfunc(pathname(trtf))\&tocfile._%sysfunc(today(), yymmdd10.).pdf"
  -toc-title="%str(PTC Therapeutics,  Inc. - &protocol)" -toc-fontsize=8 -
  toc-fontfamily=%str(cour.ttf) %str(-toc -b) -add-toc-page=0));

  data _null_;
     infile cmppdf pad;
     length fname $130;
     input fname $char130.;
     putlog fname;
  run;
%mend;
```

BACK-END COMPONENTS: VB APPLICATION AND JAVA API TOOL

As illustrated in the process flowchart, there are two applications running on the back-end: the Visual Basic application and the Java application utilizing iText API.

The function of the VB application component is straightforward. When the SAS macro detects that there are still WORD document files missing corresponding PDF format files, this application will be accessed and called to convert those files into individual PDF format files.

Compared to other programming languages, VB application has its unique advantages as its high compatibility and accessibility into the Window Office System. In this case, the VB application can best reserve the original format from the WORD documents. When we were researching for possible solutions and alternative programming languages, for instance, the blank space on the individual PDF page could be compressed especially with graphic outputs. In other words, the individual PDF in the final combined version would look more crammed than the original WORD output file. While on the other hand, the VB application can preserve the exact original output format from the individual WORD document file when

combining into the final combined PDF file. In other words, what you see in WORD is what you get in the final combined PDF. Sample VB codes to realize this function are included below for further references:

```
Sub SaveWordAsPDF(p_strFilePath, p_strPdfFolderPath)
    'Save Word file as a PDF
    'Initialise
    Dim objWord, objDocument
    Set objWord = CreateObject("Word.Application")

    'Open the file
    Set objDocument = objWord.Documents.Open(p_strFilePath)

    'Save the PDF
    objDocument.SaveAs PathOfPDF(p_strFilePath, p_strPdfFolderPath),
WORD_PDF

    'Close the file and exit the application
    objDocument.Close FALSE
    objWord.Quit
End Sub
```

The Java application component performs a more complex function. As introduced in the SAS macro component section above, when the scope of combination is defined as the TOC file being processed by the SAS macro, the Java application generates a table of content page based on the TOC file and then retrieves the individual PDF files in scope into combining them into the final PDF file. In the meantime, it automatically inserts proper hyperlinks between the table of content page and the corresponding output fil, as well as adds bookmarks for each TLF output for the convenience of reviewer per recommendation by the FDA guideline.

The Java iText API is publicly available. Therefore, this paper will avoid the redundancy here by providing references links to where interested users can find more resources for further references and research purpose:

- Sample codes for creating table of content page(s) when merging into PDFs: https://kb.itextpdf.com/itext/merging-documents-and-create-a-table-of-contents
- Sample codes for adding bookmarks: https://kb.itextpdf.com/itext/bookmark-examples

CONCLUSION

This paper shares a novel approach in combining individual output files into a submission-ready PDF file. Compared to conventional methods, this new tool significantly reduces the processing time and computing resources needed for converting and combining files into a submission-ready PDF file. It is user-friendly to all levels of programmers or even non-programmers.

REFERENCES:

International Council for Harmonisation of Technical Requirements for Pharmaceuticals for Human Use. (2015). *Specification for Submission Formats for eCTD*. Retrieved from https://admin.ich.org/sites/default/files/inline-files/Specification_for_Submission_Formats_for_eCTD_v1_2.pdf

International Council for Harmonisation of Technical Requirements for Pharmaceuticals for Human Use. (2017). *Specification for PDF Formatted Documents in Regulatory Submissions.* Retrieved from https://admin.ich.org/sites/default/files/inline-files/Specification_for_PDF_Format_v1_0.pdf

IText. iText Knowledge Base. (2024). https://kb.itextpdf.com/itext/

Poppe, F. (2005). *Store Your Output and Digest It Later.* SAS Users Group International 30, Philadelphia, PA, United States.

Shao, W. & Zhang, L. (2023). *A utility to combine study level outputs to one PDF file for development safety update report (DSUR) regional appendices.* PharmaSUG 2023, Beijing, China.

Xin, W. & Wang, J. (2016). *Utilizing SAS® and Groovy to combine multiple RTF/PDF reports to one bookmarked PDF document.* PharmaSUG China 2016, Philadelphia, PA, United States.

U. S. Food and Drug Administration. (2013). *Specifications for File Format Types Using eCTD Specifications.* Retrieved from https://www.fda.gov/media/85816/download

U. S. Food and Drug Administration. (2005). *Portable Document Format (PDF) Specifications: Technical Specifications Document.* Retrieved from https://www.fda.gov/media/76797/download

CONTACT INFORMATION:

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

Robin WU

PTC Therapeutics

rwu@ptcbio.com

Lili LI

PTC Therapeutics

lli@ptcbio.com

Steven HUANG

PTC Therapeutics

steven.huang@ptcbio.com