PharmaSUG 2019 - Paper PO-137

A Practical SAS® Macro for Converting RTF to PDF Files

Xiao Xiao, Brian Wu, and Kaijun Zhang, FMD K&L Inc.

ABSTRACT

As part of clinical trial reporting, it has been a raising demand to create a large numbers of RTF outputs when a study reaches its major milestones in a study. CROs often receive requests from clients on converting all reports in a user-friendly file format document, usually PDF format, to ease delivery and facilitate review process.

One solution for RTF to PDF conversion is via some ready tool, another alternative way is through SAS programming. The mechanism under SAS System for OS/2 and Windows enables it to talk to other PC applications by Dynamic Data Exchange (DDE). Specifically, DDE enables the creation of fully customized Microsoft Word documents from a Base SAS program, and is therefore ideal for automating the conversion of RTF files to high quality PDF files. This paper presents an alternative effective method where a practical SAS macro was developed and utilized under 2 scenarios: a.) quickly convert an extremely large size RTF file into a PDF file; b.) convert a queue of RTF files reliably into PDF files. The details of step-by-step macro development were also provided and discussed.

INTRODUCTION

Currently, RTF2PDF software package is a common tool to convert rtf files to PDF files in clinical trials. Some of companies also developed their own tools such as SAS programs. However, for some of extreme large rtf files which may have hundreds of thousands of pages, the conversion may be time-consuming and PC memory insufficiency.

VBScript is a variant of MS Visual Basic language. It is written in text files with a .vbs extension. One of the advantages of VBScript language is understandable for MS Office applications. The use of VBScript can greatly enhance the capabilities of using a SAS program to control Office applications. Dynamic Data Exchange (DDE) can read word (RTF files) table into SAS dataset. In this presentation, VBScript is created to split extreme large rtf files by every X page, for example, every 1000 pages. Microsoft Visual Basic language (VB), DDE and Base SAS are used together to reduce the RTF file to PDF file conversion time. We demonstrated this approach in the following three steps: a) the product splits the file into a series of small sized rtf files, b) the product converts all sub-files automatically into PDF files, c) combine all pdf files into a single file.

DYNAMIC INTERACTION BETWEEN SAS PROGRAM AND VBSCRIPT

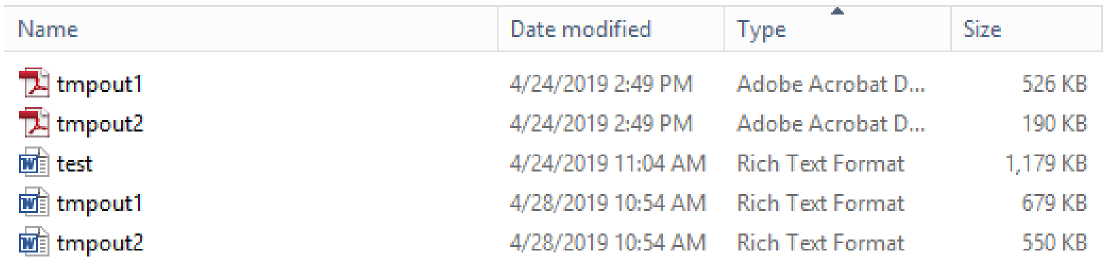
The basic idea in our application is to utilize the interaction between SAS program and VBScript where calling VBSscript opens an executive file. In this executive file, we will call SAS system to run a SAS program. Below are key ingredients to execute a SAS program from running VBScript.

STEP 1 SET UP DIRECTORY:

```
****Directory for VBScript

%let vbssplit=K:\conversion\SplitDoc.vbs;

*****Directory for RTF which undergoes splitting;

%let inrtf=K:\conversion\test.rtf;

*******Directory for saving multiple pdf files converted from small sized splitted rtf files. This serial pdf
files's name start with prefix "tmpout" plus auto-sequenced number;

%let outprefix=K: \conversion\tmpout;
```

STEP 2. SAS PROGRAM TO CALL VBSCRIPT TO SPLIT RTF FILE. THE FILE NAMES ARE TMPOUT1, TMPOUT2, TMPTOUT3, …. N. THEN THE SERIAL RTF FILES WITH SMALL SIZE ARE CONVERTED INTO MULTIPLE PDF FILES.

```
data _null_;
  /* execute vbs */
  cmd = "&vbssplit &inrtf &outprefix 10 ";
  call system(cmd);
run;
```



Name	Date modified	Type	Size
tmpout1	4/24/2019 2:49 PM	Adobe Acrobat D...	526 KB
tmpout2	4/24/2019 2:49 PM	Adobe Acrobat D...	190 KB
test	4/24/2019 11:04 AM	Rich Text Format	1,179 KB
tmpout1	4/28/2019 10:54 AM	Rich Text Format	679 KB
tmpout2	4/28/2019 10:54 AM	Rich Text Format	550 KB

Picture 1. Display of rtf file with name test.rtf which is split into two rtf file: tmpout1 and tmpout2 which are converted into two pdf files.

STEP 3. MULTIPLE PDF FILES COMBINED INTO A SINGLE PDF FILE (WELCH, B AND BURNS, R. 2011, P3).

As example shows as step 2, two PDF files can be combined into a single PDF file.

```
Dim Doc1

Dim Doc2

Set Doc1= CreateObject("AcroExch.PDDoc")

Set Doc2= CreateObject("AcroExch.PDDoc")

file1= Doc1.Open("C:\conversion \ tmpout1.pdf")

file2= Doc2.Open("C:\conversion \ tmpout2.pdf")

Stack = Doc1.InsertPages(Doc1.GetNumPages - 1, Doc2, 0, Doc2.GetNumPages, 0)

SaveStack= Doc1.Save(1,"C:\conversion\NewDoc.pdf")
```

EXPLANATION OF VB SCRIPT FUNCTIONS

The bridge between MS Windows and Adobe Acrobat is provided through Inter-application Communication, which is accomplished via OLE and DDE in Microsoft Windows. All of the objects, statements and functions fall under the umbrella of OLE Automation. For this paper, we use SAS V9.4, MS Windows 10 Professional and Adobe Acrobat X Standard.

Although not required, declaring variables using the Dim statement is customary. In the event of combining several files (declaring more variables), Options Explicit at the beginning of the VB script is useful. If the compiler finds an undefined variable, an error is issued. Options Explicit ensures a null value is assigned to the variable. The Set statement assigns a value to a property (Microsoft Corporation, 2016). We assign PDF-type objects to specific variables (DocX) using the CreateObject() function and use the “AcroExch.PDDoc” class. Adobe Acrobat with the Software Development Kit (SDK) installed is necessary for the VB script to successfully create this object. Finally, we generate a series of file variables (filex) using the Open() method.

The VB script contains a main loop that iterates through the input RTF document from the beginning to the end. In each iteration, it selects a chunk of pages and copies them into the clipboard. Afterwards it creates a small document and pastes the clipboard pages into the small document. Since the small document may have a different page orientation, we explicitly set the orientation of the small document to that of the input document. Another problem is that each small document starts with Page 1 by default, and we have to change the starting page number to be consistent with the original input. In addition, the document header has a field for total number of pages. This value is much smaller in each small document than the original input. We loop through the headers to update the total number to be consistent with the input document. So all information has been reserved in the split files.

VBSCRIPT FOR SPLITTING RTF FILE INTO MULTIPLE RTFS AND THEN CONVERTED INTO MULTIPLE PDF FILES.

```
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
'  This script takes three arguments:
'  inputFilePath:    the full path of the input RTF file
'  outputFilePrefix: the prefix of the output file path, the small files will be
'                    named as prefix1.rtf, prefix2.rtf, etc
'  chunkSize:        the number of pages in a small RTF file
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
Option Explicit
Sub SplitDoc()
  Dim totalPages, fromPage, toPage
  Dim chunkIndex, chunkSize
  Dim WordApp , bigDoc, smallDoc
  Dim hdr, asect, afield, fromPos, toPos

  '''The number of pages in each small output file
  chunkSize = CInt(WScript.Arguments.Item(2))

  '''Some constants from Microsoft Word library
  Const wdActiveEndPageNumber = 3, wdNumberOfPagesInDocument = 4
  Const wdGoToPage = 1, wdGoToNext = 2, wdGoToAbsolute = 1
  Const wdStory = 6, wdCharacter = 1
  Const wdFormatOriginalFormatting = 16
  Const wdOrientLandscape = 1, wdOrientPortrait = 0
  Const wdFormatPDF = 17, wdFormatRTF = 6
  Const wdHeaderFooterPrimary = 1, wdHeaderFooterFirstPage = 2
  Const wdCollapseEnd = 0
  Const wdFieldNumPages = 26, wdFieldPage = 33
  Const wdSaveChanges = -1, wdDoNotSaveChanges = 0

  '''Open the input document
  Set WordApp = CreateObject("Word.Application")
  WordApp.Visible = FALSE
  Set bigDoc = WordApp.Documents.Open(WScript.Arguments.Item(0),,False)
  bigDoc.Activate

  '''Get the total number of pages
  totalPages = bigDoc.Range.Information(wdNumberOfPagesInDocument)

  chunkIndex = 1
  fromPage = (chunkIndex - 1) * chunkSize + 1

  '''Loop until the end of the document
  While fromPage <= totalPages
     '''Get the starting position of a chunk
     WordApp.Selection.GoTo wdGoToPage, wdGoToAbsolute, fromPage
     fromPos = WordApp.Selection.Range.Start

     '''Get the ending position of a chunk
     toPage = chunkIndex * chunkSize
     If toPage+1 > totalPages Then
       'Go to the end of document
            WordApp.Selection.EndKey wdStory
     Else
       'Go to the next page of toPage
       WordApp.Selection.GoTo wdGoToPage, wdGoToAbsolute, toPage+1
       'Move backwards to the end of toPage
       WordApp.Selection.MoveLeft wdCharacter, 1
     end if
     toPos = WordApp.Selection.Range.End

     '''Select a chunk
     WordApp.Selection.Start = fromPos
     WordApp.Selection.End   = toPos
```

```
'''Copy the selection to the clipboard
WordApp.Selection.Copy

'''Create a new small documnet
Set smallDoc = WordApp.Documents.Add
smallDoc.Activate

'''Set the orientation
if bigDoc.PageSetup.Orientation = wdOrientPortrait Then
  smallDoc.PageSetup.Orientation = bigDoc.PageSetup.Orientation
Else
  'Sometimes bigDoc.PageSetup.Orientation has a weird nonzero value
  smallDoc.PageSetup.Orientation = wdOrientLandscape
End If

'''Paste the clipboard to the small document
WordApp.Selection.PasteAndFormat (wdFormatOriginalFormatting)

'''Update the starting page number in the small document
Set hdr = smallDoc.Sections(1).Headers(wdHeaderFooterPrimary)
hdr.PageNumbers.RestartNumberingAtSection = true
hdr.PageNumbers.StartingNumber = fromPage

'''Change the total page number in the small document
For Each asect in smallDoc.Sections
  For Each hdr in asect.Headers
    hdr.Range.Collapse wdCollapseEnd
    For Each afield In hdr.Range.Fields
      If afield.Type = wdFieldNumPages Then
        'Delete the existing total page number
        afield.Select
        WordApp.Selection.TypeBackspace
        'Insert the total number from the big document
        WordApp.Selection.InsertAfter "" & totalPages
      End If
    Next
    Next
   Next

   '''Save the small document
   smallDoc.SaveAs WScript.Arguments.Item(1) & chunkIndex & ".rtf" , wdFormatRTF
   smallDoc.SaveAs WScript.Arguments.Item(1) & chunkIndex & ".pdf" , wdFormatPDF
   smallDoc.Close wdDoNotSaveChanges

   chunkIndex = chunkIndex + 1
   fromPage = (chunkIndex - 1) * chunkSize + 1
   bigDoc.Activate
 Wend

 '''Empty the clipboard
 Dim WshShell
 Set WshShell = WScript.CreateObject("WScript.Shell")
 WshShell.Run "cmd.exe /c echo. >NUL  | clip", 0, True

 '''Close the big document
 bigDoc.Close wdDoNotSaveChanges
 WordApp.Quit
 Set bigDoc = Nothing
 Set WordApp = Nothing
End Sub

''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
'                         Starting the main program
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
If WScript.Arguments.count < 3 then
 wscript.echo "Usage: splitdoc.vbs inputRTF outputPrefix chunkSize"
Else
 call SplitDoc
End If
```

CONCLUSION

VBS working together with SAS helps to automate the process of converting rtf files to PDF files by dealing with MS Office documents. It can be written and launched within SAS and allows to perform commands in MS Office that cannot be directly executed in SAS.

This paper illustrates how to make VB script, DDE and SAS software work together to convert huge size rtf files to PDF files without considering memory enough available and reducing conversion time to meet tight timeline with quality deliverables. Further It is dynamic, secured and error free.

REFERENCES

[1] Welch, B and Burns, R. 2011. "Combining External PDF Files by Integrating SAS® and Adobe® Acrobat." Proceedings of 19th Annual Southeast SAS Users Group. Alexandria, VA. Available at: https://analytics.ncsu.edu/sesug/2011/BB15.Welch.pdf

ACKNOWLEDGMENTS

Sheng Zhang

Chao Wang

CONTACT INFORMATION

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

Kaijun Zhang
FMD K&L Inc
kaijun.zhang@klserv.com

Xiao Xiao
FMD K&L Inc
xiao.xiao@klserv.com