PharmaSUG 2019 - Paper AD-104

A Simple SAS Utility to Combine Existing RTF Tables/Figures and Create a Multi-Level Bookmark Hierarchy and a Hyperlinked TOC

Lugang Xie (Larry), Janssen R&D, Raritan, NJ

ABSTRACT

Rich Text Format (RTF) is a popular format that most sponsors adopt to create tables, listings and figures (TLF) in the pharmaceutical industry. However, there is an unmet need for a method to concatenate the outputs into a presentable single RTF file independent of the computing system. This paper presents a simple SAS approach to concatenate any SAS generated RTF files, including both tables/listings and figures, and create a multi-level bookmark hierarchy and a hyperlinked TOC, without having to modify the macros/programs that create the individual outputs. It can be executed in any environment with SAS installed. In addition, a simple approach to convert the combined RTF file into a PDF or DOCX format is also introduced, and it can be used for OSI BIMO listings submission.

INTRODUCTION

BACKGROUND AND APPROACH

While PDF is the FDA supported format, RTF remains the most popular format used for production of individual reports in the pharmaceutical industry. Standard programs/macros to generate clinical reports in RTF format have been developed for years and as such manipulating RTF files has since been increasingly important. For example, concatenated RTF files with a navigation system (e.g. hyperlinked TOC or bookmarks) added enables switching between tables/graphs easier thereby reducing the time required for internal review. In addition, PDF format with a multi-level bookmark hierarchy is often required for agency submission, e.g. OSI BIMO listings. Some companies re-create these outputs in other formats, e.g. PDF or plain text, and created a bookmarked PDF file based on these files. It will be both time and cost efficient if sponsors take advantage of existing programs/macros and convert these individual RTF files into a deliverable in the required format.

Several methods have been developed to handle this task over the past decade, but each of them has its own drawback. For example:

1. Visual Basic approach: it must be executed in windows environment with MS Word installed; it is resource demanding and takes a considerably long time to handle a large number of RTF files.
2. ODS Document approach: it must have the code in the individual TLF programs to create ODS Document objects. It is unable to handle any type of RTF files directly.
3. Other approaches: they either require inserting RTF code in the individual TLF programs or are unable to create a multi-level bookmark hierarchy.

This paper introduces a highly efficient, environment independent approach through RTF code manipulation to concatenate any type of RTF files generated by SAS and create a multi-level bookmark hierarchy and hyperlinked Table of Contents (TOC).

METHODOLOGY AND DISCUSSION

REQUIREMENTS FOR INDIVIDUAL RTF FILES

Each sponsor has its own specific requirements for RTF reports generated by SAS, but fortunately the basic requirements are actually very similar. The approach this paper provides accommodates the most popular requirements among sponsors:

1. Individual reports in RTF format are generated by SAS.

2. Titles are printed in the body rather than in the header section of an RTF file.

CONCATENATE INDIVIDUAL RTF FILES

RTF files are largely plain text ASCII files, which enables us to read the files in to a SAS dataset and manipulate the RTF code for various purposes.

Each SAS generated RTF file includes three parts:
Part 1: the opening section, which includes the control words for file attributes (e.g. "{\rtf1"), file attributes, font table, color table, style sheet, etc.
Part 2: the content section, which includes RTF code for header, footer and body. This part contains the content to be displayed by a word processer. The starting RTF code for this section varies, depending on the way the RTF file is output, RTF template, etc. It usually starts immediately after {\stylesheet}. In the example displayed in Display 1, it starts with either "\paperw15840\paperh12240\landscape" if the page orientation is landscape or "\margl720\margr720" if it is portrait.
Part 3: the closing section, which includes a closing curly brace "}", which may be preceded by one or more control words, e.g. "\par}".

Here is an example of RTF code of a typical SAS generated RTF table output.

Part 1	{\rtf1\ansi\deff0\deflang1033 {\fonttbl{\f0\froman Times New Roman;} {\f1\fswiss\fcharset0\fprq2 Arial;}} {\stylesheet {\s0 Normal;} {\s1 \ql\widctlpar heading 1;} {\s2 \ql\widctlpar heading 2;} {\s3 \ql\widctlpar heading 3;}}
Part 2	\paperw15840\paperh12240\landscape \margl720\margr720\margt2880\margb1800\headery1800\footery1800 {\header\pard\par} {\footer\pard \qr Page \chpgn of {\field{*\fldinst NUMPAGES }} \par} \s15\qc\li0\ri0\sb0\sa0\widctlpar\rin0\lin0\itap0\qc\f0\fs24 \qc\f0\fs24\cgrid Table 1: Summary of Vital Signs \par \trowd\trleft0\trqc … …
Part 3	\pard\par}

Display 1: An example of RTF code of a typical SAS generated RTF table output

If we remove the part 3 of the first RTF file, part 1 of the last RTF file and both part 1 and part 3 of the rest of the RTF files in the middle, and set them all together with "\sect\sectd" inserted between files in a SAS dataset, and write the SAS dataset to a new RTF file, it will result in a single RTF file with all individual files concatenated. Note that "\sect\sectd" starts a new page in an RTF file.

Display 2 illustrates how RTF files are concatenated.

RTF File	RTF Code
First	Part 1 Part 2
Second	\sect\sectd Part 2
Third	\sect\sectd Part 2
more ……	\sect\sectd Part 2 \sect\sectd Part 2 ……
Last	\sect\sectd Part 2 Part 3

Display 2: RTF code for combined RTF file

MULTI-LEVEL BOOKMARKS

A multi-level bookmark hierarchy can be created by adding RTF code around the titles. For example, as per FDA requirements for BIMO listings, three levels of bookmarks need to be created - the first level is study ID, the second level is site number, and the third level is each individual listing titles. This three-level bookmark hierarchy can be created by following two steps:

1. *Identify the title of each table.* The titles are typically surrounded with a consistent pattern of RTF control words, e.g. the title is always preceded by "\qc\f0\fs24\cgrid" and followed by "\par" in the example in Display 1. Again, the surrounding control words vary from RTF file to RTF file, depending on the RTF template used to create these file. As long as they are created by SAS, they will be consistent across different tables.

2. *Insert style control words for the titles*. For example, add \s2 before the site title and \s0 after it, and the site title will appear at the second level in the bookmark hierarchy; add \s2 before the listing title and \s0 after it, and the listing title will appear at the their level in the bookmark hierarchy. This insertion should ONLY be done for the first page of each individual RTF file.

   A cover page can be created containing study ID "ABC-P123" by adding the following code before Part 2 of the first file:

```
\sect\sectd
\pard\par\pard\plain\qc\f0\fs24\b \s1 ABC-P123 \s0 \b0\par
```

   Please note that \s1 and \s0 surround the ABC-P123, enabling the study ID to appear at the first level in the bookmark hierarchy.

   Important: the control words below define the headings and must be included in {\stylesheet} as illustrated in part 1 in Display 1. If the default stylesheet does not include this code, it needs to be added.

```
{\s0 Normal;}
{\s1 \ql\widctlpar heading 1;}
{\s2 \ql\widctlpar heading 2;}
{\s3 \ql\widctlpar heading 3;}
```

If the section titles are not available, there is a workaround that can be considered to adopt – insert into the combined RTF file a site number title before the table titles with font size being 0 and color being white.

HYPER-LINKED TABLE OF CONTENTS

Table of Contents can also be created by adding RTF control words. Reviewers may find it useful in different situations. This paper does not go into details on this topic, but the basic idea of how it can be created is briefly explained as follows:

1. Insert the code below between \sect\sectd and part 2 for each file as illustrated in Display 2. Note that the *n* as in IDX*n* is the order number of individual RTF file. It should be incremental.

```
{\bkmkstart IDXn}{\bkmkend IDXn}
```

For example:

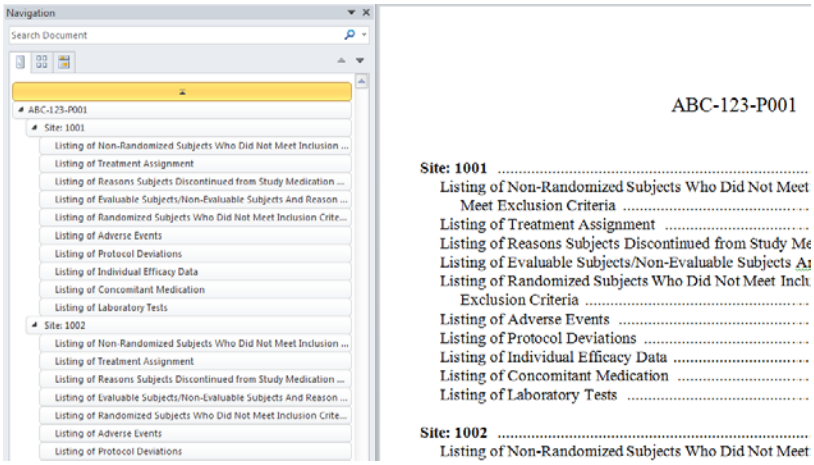
```
{\bkmkstart IDX1}{\bkmkend IDX1}
{\bkmkstart IDX2}{\bkmkend IDX2}
… … etc.
```

2. On the first page created earlier, which only contains the study ID, insert a Word field for each title to populate the hyperlinks. The titles can be extracted from individual RTF files, given the fact that the surrounding RTF code of the titles should be consistent across all the files. For example:

```
\fi-200\li400{\cf0{\field{\*\fldinst { HYPERLINK \\l "IDX1"}}
{\fldrslt {\cf0 Table 1: Summary of Vital Signs }}}
\tab{\field{\*\fldinst { PAGEREF IDX1 \\h}}{\fldrslt{NUMPAGES}}}}\par
\pard\plain\ql\li0\ri0\nowidctlpar\faauto\rin0\lin0\itap0\f1\fs20

\fi-200\li400{\cf0{\field{\*\fldinst { HYPERLINK \\l "IDX2"}}
{\fldrslt {\cf0 Table 2: Summary of Adverse Events }}}
\tab{\field{\*\fldinst { PAGEREF IDX2 \\h}}{\fldrslt{NUMPAGES}}}}\par
\pard\plain\ql\li0\ri0\nowidctlpar\faauto\rin0\lin0\itap0\f1\fs20

… … etc.
```



Display 3: An example of combined RTF file with a three-level bookmark hierarchy and a TOC

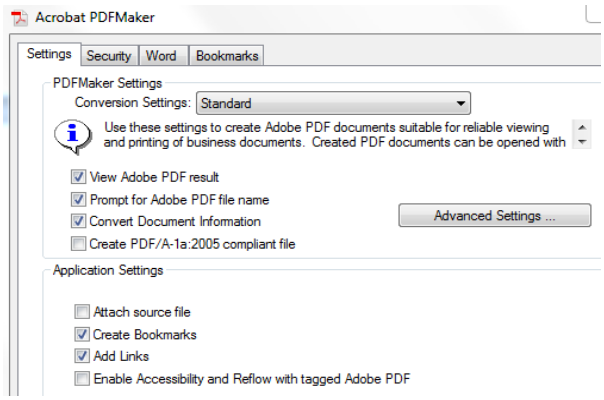
CONVERT COMBINED RTF FILE TO OTHER FORMATS

Formats other than RTF may be preferred depending on the user's need. PDF, for example, is often a required format for submission to regulatory agencies; DOCX is smaller in size and it is convenient to add comments to the file. Three methods are recommended for format conversion – Acrobat PDFMaker (for PDF only), PowerShell scripts and Visual Basic scripts. Bookmark hierarchy in the RTF file is preserved using any of these three methods. This paper only provides details of converting to PDF using PDFMaker and PowerShell scripts, as examples to convert to DOCX can be found on the web.

Method 1: Acrobat PDFMaker

This is a very straight forward approach and the resulting PDF file size is relatively small. Open the combined RTF file with MS Word and in the Acrobat ribbon click Create PDF, and job is done. To considerably reduce conversion time, it is recommended that the following two settings be disabled in Preferences (as illustrated in Display 4):

- Create PDF/A-a-2005 compliant file
- Enable Accessibility and Reflow with tagged Adobe PDF



Display 4: Recommended settings for Acrobat PDFMaker

Method 2: PowerShell Scripts

This approach involves minimal manual process and can be integrated with SAS programs. The resulting PDF file will have all bookmarks collapsed. File can be converted by following a few simple steps:

1. Create a PowerShell script file containing the scripts provided in Appendix and name it as RTF2PDF.ps1
2. Create a folder and save both the script file and the RTF file to convert in the folder, e.g. C:\test
3. Open PowerShell console and run the following commands:

```
cd C:\test
. .\RTF2PDF.ps1
RTF2PDF -path "C:\test"
```

For details of the scripts, please refer to Script to convert Word document to PDF file (PowerShell).

CONCLUSION

This approach is especially useful if sponsors want to take advantage of the existing SAS macros/programs already available to create RTF outputs and want to avoid duplicate efforts or extra software tools.

While other options are also available, there are distinct advantages for this SAS approach to combine RTF files:

1. Efficiency and environment independent. It is more efficient than the VBA approach. It takes considerably less time to create a combined RTF file using SAS than using VBA scripts.

2. Low maintenance. As it is implemented using SAS, every SAS programmer with basic RTF knowledge should be able to update the macro. This is especially important when customization is required.
3. Fully automated process to create combined RTF files. No manual process is involved. It can be added to the end of the batch job for report generation. Combined files can be automatically generated after all the individual outputs are created.

It is worth noting that there is a file size limit of 500 MB for any MS Word documents. In other words, if an RTF file is larger than 500 MB in size, it cannot be opened by MS Word. However, if you created a few smaller combined RTF files and convert them into multiple PDF files, it would circumvent the issue. It should not be a concern for internal review and FDA accepted this approach for recent submissions.

REFERENCES

1. Meeker-O'Connell, Ann and Kassim, Sean. "Overview of Information Requested by CDER's Office of Scientific Investigation (OSI) for NDA and BLA Submissions. October 2012". Available at https://collaboration.fda.gov/p44198603/
2. "Portable Document Format (Pdf) Specifications". September, 2014. Available at http://www.fda.gov/downloads/Drugs/DevelopmentApprovalProcess/FormsSubmissionRequirements/ElectronicSubmissions/UCM163565.pdf
3. "Field codes in Word". Available at http://www.fda.gov/downloads/Drugs/DevelopmentApprovalProcess/FormsSubmissionRequirements/ElectronicSubmissions/UCM163565.pdf
4. "Script to convert Word document to PDF file (PowerShell)". Available at https://gallery.technet.microsoft.com/office/Script-to-convert-Word-f702844d
5. "Rich Text Format (RTF) Specification Version 1.9.1.". Available at https://www.microsoft.com/en-us/download/confirmation.aspx?id=10725
6. Xie, Lugang (Larry) and Yan, Zhiping. "A Fully Automated Approach to Concatenate RTF outputs and Create TOC". PharmaSUG China 2014. Available at http://www.lexjansen.com/pharmasug-cn/2015/DV/PharmaSUG-China-2015-DV28.pdf

CONTACT INFORMATION

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

Name: Lugang Xie (Larry)
Enterprise: Janssen R&D
Address: 920 US Highway 202
City, State ZIP: Raritan, NJ 08869
Work Phone: (908) 927-7415
E-mail: lxie9@its.jnj.com

APPENDIX

```
Function RTF2PDF
{
  [CmdletBinding(SupportsShouldProcess=$true,DefaultParameterSetName="Default")]
  Param
  (
    [Parameter(Mandatory=$true,Position=0)]
    [Alias('p')][String]$Path,
    [Parameter(Mandatory=$true,Position=1,ParameterSetName="Customize")]
    [Alias('sp')][Int32]$StartPage,
    [Parameter(Mandatory=$true,Position=2,ParameterSetName="Customize")]
    [Alias('ep')][Int32]$EndPage
  )

  If($PSCmdlet.ShouldProcess("Convert Word document to PDF file."))
  {
    If(Test-Path -Path $Path)
    {
      #get all related to wd document files
      $wdDocumentFiles = Get-ChildItem -Path $Path -Recurse -Include *.rtf,*.docx
      If ($wdDocumentFiles)
      {
        Add-type -AssemblyName Microsoft.Office.Interop.Word
        $wdApplication = New-Object -ComObject "Word.Application"

        ######define the parameters######
        $wdExportFormat = [Microsoft.Office.Interop.Word.WdExportFormat]::wdExportFormatPDF
        $OpenAfterExport = $false
        $wdExportOptimizeFor = [Microsoft.Office.Interop.Word.WdExportOptimizeFor]::wdExportOptimizeForOnScreen
        $wdExportItem = [Microsoft.Office.Interop.Word.WdExportItem]::wdExportDocumentContent
        $IncludeDocProps = $true
        $KeepIRM = $true
        $wdExportCreateBookmarks = [Microsoft.Office.Interop.Word.WdExportCreateBookmarks]::wdExportCreateHeadingBookmarks
        $DocStructureTags = $true
        $BitmapMissingFonts = $true
        $UseISO19005_1 = $fals

        If($StartPage -and $EndPage)
        {
          $wdExportRange = [Microsoft.Office.Interop.Word.WdExportRange]::wdExportFromTo
          $wdStartPage = $StartPage
          $wdEndPage = $EndPage
        }
```

```
Else
{
  $wdExportRange = [Microsoft.Office.Interop.Word.WdExportRange]::wdExportAllDocument
  $wdStartPage = 0
  $wdEndPage = 0
}

#Convert wd document to PDF file
Foreach($wdDocumentFile in $wdDocumentFiles)
{
  $wdDocumentCounts = $wdDocumentFiles.Count
  If(!$wdDocumentCounts)
  {
    $wdDocumentCounts = 1
  }

  $intNumberwd++

  #get the file name
  $wdDocumentName = $wdDocumentFile.Name
  #get the directory of file
  $wdDocumentPathDirectory = $wdDocumentFile.DirectoryName
  $wdDocumentBaseName = $wdDocumentFile.BaseName
  $wdDocumentPath = $wdDocumentFile.FullName
  #define an output file path
  $OutputFilePath = "$wdDocumentPathDirectory\$wdDocumentBaseName.pdf"

  Try
  {
    #Displays a progress bar within a Windows PowerShell command window.
    Write-Progress -Activity "Converting Word document [$wdDocumentName] to PDF file" `
    -Status "$intNumberwd of $wdDocumentCounts Word document(s)" -PercentComplete $($intNumberwd/$wdDocumentCounts*100)

    #Open the Word document
    $wdDocument = $wdApplication.Documents.Open($wdDocumentPath)
    #Sets the format paramters
    $wdDocument.ExportAsFixedFormat($OutputFilePath,$wdExportFormat,$OpenAfterExport,`
    $wdExportOptimizeFor,$wdExportRange,$wdStartPage,$wdEndPage,$wdExportItem,$IncludeDocProps,`
    $KeepIRM,$wdExportCreateBookmarks,$DocStructureTags,$BitmapMissingFonts,$UseISO19005_1)

    #release the object
    $wdDocument.Close()
    [void][System.Runtime.InteropServices.Marshal]::ReleaseComObject($wdDocument)

    $Properties = @{'File Name' = $wdDocumentName
```

```
            'Action(Convert to PDF)' = If(Test-Path -Path $OutputFilePath)
                                        {"Finished"}
                                       Else
                                        {"Unfinished"}
            }

            $objWord = New-Object -TypeName PSObject -Property $Properties
            $objWord
          }
          Catch
          {
            Write-Warning "A few Word documents have been lost in this converting. $wdDocumentName cannot convert to PDF file."
          }
        }

        ######release the object######
        $wdApplication.Quit()
        [void][System.Runtime.InteropServices.Marshal]::ReleaseComObject($wdApplication)
        [GC]::Collect()
        [GC]::WaitForPendingFinalizers()
      }
      Else
      {
        Write-Warning "Please make sure that at least one Word document file in the ""$Path""."
      }
    }
    Else
    {
      Write-Warning "The path does not exist, plese input the correct path."
    }
  }
}
```