

AD07

A Tool to Automate TFL Bundling

Mark Crangle

ICON Clinical Research

Introduction

- Typically, requirement for a TFL package is a bookmarked PDF file with a table of contents
 - Often this means combining individual files into one document
 - Can be difficult if individual files are in different formats to meet sponsor requirements
- Existing method at ICON used SAS macros to read in TFL metadata and output VB script which were used in MS Word to pull the TFLs together
 - Worked well with smaller sets of TFLs but could be unreliable and slow to run with large numbers of TFLs
- Existing metadata system was being retired and opportunity to build a new system addressing some of the issues of the old method

Building a New Tool

- The new solution had several key requirements
 1. Able to generate a PDF file with a Table of Contents and bookmarks that both linked through the document
 2. Uses TFL metadata to ensure TOC and bookmarks match the content of TFLs
 3. Increased automation to reduce the amount of user input needed
 4. **Quicker and more reliable than existing solution**
 - Existing solution joined all files together in Word then converted that to PDF
 - With increasing size of TFL files, this was exponentially increasing the computation time for each step
 - Needed to find a way to split the conversion and bundling into smaller parts so that performance was the same regardless of the package size
 - This was the focus of the early development of the new tool

Creating a PDF file from a Web Page

- Adobe Acrobat contains a function to convert an entire web page to PDF
 - Reads in the HTML file and creates output file matching the input HTML in appearance



- Links in the HTML file work in the body of the document and are created as PDF bookmarks
- There is an option to append the link contents into the document
- If we could develop a HTML page to represent the Table of Contents, with links to the individual files, then this could be converted into one document by Adobe Acrobat

Creating a PDF file from a Web Page – HTML syntax

Only relatively simple HTML syntax was needed to create our table of contents:

```
<html>
<head>
<title>Table of Contents</title>
</head>
<body>
<table>
<tr>
<td><a href='file.pdf'>Table 14.1.1</a></td>
<td>Title</td>
<td>Page #</td>
</tr>
</table>
</body>
</html>
```

Other cells can be added to the TOC for the TFL title and page number within the document

to the individual TFL file with the <a> tag

The href option gives the location of the file which can be either relative or absolute location

This is displayed as the TFL number which is a clickable link

Creating a PDF file from a Web Page – Appending TFLs

- Appending TFLs of differing file-types didn't work as expected
 - Adobe used default page sizing, margins and font that often wasn't as required
- Therefore, we had to ensure that we could supply individual PDF files in the hyperlinks
- Ideally, TFLs would be created in PDF format for this but we included a step to convert and non-PDF files to PDF
- Decided to limit conversion to file types that could be opened in MS Word
 - Any figures would need to be provided in an RTF or already converted to PDF
 - “Raw” image would not be accepted

Defining the Process

- 1. Create a PDF file and apply formatting changes to bookmark text and file properties of each one
- 2. Save the PDF file and apply formatting changes to bookmark text and file properties of each one
- 3. Create a PDF file and apply formatting changes to bookmark text and file properties of each one
- 4. Save the PDF file and apply formatting changes to bookmark text and file properties of each one
- 5. Create a PDF file and apply formatting changes to bookmark text and file properties of each one
- 6. Save the PDF file and apply formatting changes to bookmark text and file properties of each one
- 7. Create a PDF file and apply formatting changes to bookmark text and file properties of each one
- 8. Save the PDF file and apply formatting changes to bookmark text and file properties of each one
- 9. Create a PDF file and apply formatting changes to bookmark text and file properties of each one
- 10. Save the PDF file and apply formatting changes to bookmark text and file properties of each one

Building the Tool

- After considering SAS macros, we decided to use an Excel spreadsheet and macros for the tool
 - Existing metadata could be easily read into Excel
 - Control of MS Word for TFL conversions directly rather than going through DDE
 - Make use of Adobe's Inter Application Communication (IAC) library directly from VBA macros
- TFL information would be taken from metadata and imported to Input sheet along with input from the user
- Options for certain aspects of the packaging and conversion entered into Options sheet and saved as macro variables by VBA code

Challenges – Sending Keypress Events

- Some parts of the process can only be done from menus in MS Word or Adobe Acrobat
 - Setting initial print settings required for PDF conversion
 - Reading in the HTML file in Adobe
- To avoid user input, solution was to use KeyPress method in VBA code to send key press events directly to active application
 - Using keyboard shortcut keys, the menus could be navigated this way
 - Sequence of key presses would be dependent on program versions
 - User was trained on when these commands would be run so as not to activate any other applications

Challenges – Waiting for File Conversions

- Conversion to PDF was done by printing from Word to PostScript (PS) files and then using PDF Distiller to convert that to PDF
 - Commands for PS file sent from VBA macro to Word
 - Commands to PDF conversion sent using PDF Distiller library available in VBA
- For both steps, after sending the command to start creating the file, macro tries to move to next step but fails if file creation has not finished
- To prevent this, created a loop that would not exit until file size had stopped increasing

```
lngFSize = FileLen(tempPSFileName)
```

```
flag = 0
```

```
i = 1
```

```
Do While (flag = 0)
```

```
    newsize = FileLen(tempPSFileName)
```

```
    If newsize = lngFSize Then flag = 1
```

```
Else
```

```
    lngFSize = newsize
```

```
    Application.Wait (Now() + TimeValue("00:00:02"))
```

```
    i = i + 1
```

```
End If
```

```
Loop
```

If file size is different then reset the comparison size variable to the current size, increment the loop and wait 2 seconds for the next iteration

Challenges – Updating PDF Formatting

- Default bookmark text in final file just uses the filename so this needed to be updated
 - Can also update document properties and default view so that document always opens with
- Use Inter-Application Communication libraries created by Adobe and available in VBA
 - Library of OLE objects that can be referenced directly in VBA code to control document properties and appearance
 - Nothing exists in IAC to create a PDF file from web page so this part is still done with SendKeys as above
- To open an instance of Adobe Acrobat use the code

```
Set Acroapp = CreateObject("AcroExch.App", "")  
Acroapp.Show
```

Challenges – Updating PDF Formatting

- After opening application we can define objects in the Application (AV) and Portable Document (PD) layers to access document information
 - AV Layer controls the user interface for Adobe
 - PD Layer provides access to the information within the document and can perform basic manipulations

```
Set PDYourDoc = CreateObject("AcroExch.PDDoc", "")  
Set AVYourDoc = CreateObject("AcroExch.AVDoc", "")  
bFileOpen4 = AVYourDoc.Open("C:\pdf.pdf", "Package")  
Set PDYourDoc = AVYourDoc.GetPDDoc
```

Load the PD layer
information of the document
that is open in the AV layer

- Further methods then exist in the PDDoc object to set bookmark text, document properties and default view when the document is opened

Conclusion

- The challenge was to come up with a robust replacement for our existing packaging tool with greater levels of automation
- By using Excel as the input method and running VBA macros we've been able to simplify the input and link to metadata
- Using SendKeys method, removes the need for user interaction in the more complicated parts of the process, reducing the opportunity for error
- Inter-Application Communication is used to directly control document properties so that the final PDF file does not need any further user manipulation

ANY QUESTIONS?