Paper BB-15

Combining External PDF Files by Integrating SAS® and Adobe® Acrobat

Brandon Welch, Rho®, Inc., Chapel Hill, NC
Ryan Burns, Rho, Inc., Chapel Hill, NC

ABSTRACT

As SAS programmers of various disciplines, we often utilize additional software to complete a programming task. One popular approach is integrating SAS and Microsoft® Excel and/or Microsoft Word via Visual Basic for Applications (VBA). Using similar techniques, one may establish a link between SAS and Adobe Acrobat via Inter-application Communication (IAC). This linkage provides programmers the ability to automate processes by embedding Adobe-specific language in their VB scripts. In this article, we illustrate the construction of a SAS program that outputs a Visual Basic Scripting (VBS) file, which is then submitted to combine PDF files. We demonstrate the approach in three steps. We first scan a folder using SAS File I/O functions to identify the PDF files to combine. Then we construct the VB code in a DATA STEP centered on the files residing in the folder. Finally we output the contents of our DATA STEP into a VBS file which is submitted within the same SAS program. The techniques we present offer a good overview of basic data step programming that will educate SAS programmers at all levels.

INTRODUCTION

The SAS system provides power tools for statistical programmers, but sometimes it is necessary to integrate other languages to accomplish certain tasks. A variety of different software packages offer solutions to our clients' needs, and automating tasks in the MS Windows environment are commonplace. Rarely, instances arise that necessitate automating tasks in Adobe Acrobat. Understanding how to communicate with these software packages and how to interact with the appropriate files is vital and is often challenging.

The following are popular examples of how SAS interacts with MS Excel and MS Word. Here we export a SAS data set to an Excel workbook:

```
DATA test;
  do id = 1 to 10;
    treat = ranbin(12345,1,0.5) + 1;
    resp  = ranpoi(12345,50);
    output;
  end;
RUN;

PROC EXPORT
  data = test
  outfile = "C:\exprtex1.xls"
  dbms = excel replace;
  sheet = "sheet1";
RUN;
```

Similarly, as introduced in SAS Version 9:

```
libname forxls 'C:\lbnengx1.xls';
DATA forxls.sheet1;
  set test;
RUN;
libname forxls clear;
```

MS EXCEL WORKBOOK

id	treat	resp
1	1	55
2	2	46
3	1	54
4	1	54
5	2	55

6	1	44
7	1	48
8	1	58
9	1	44
10	1	45

Communicating with MS Word is more complicated. A popular approach uses DDE in a FILENAME statement. Here we write the text “I used to live in SAS, but now I live in a RTF” to a Rich Text File (RTF).

```
options noxwait;
filename sasword dde 'winword|system';
```

We use the NOXWAIT SAS system option to avoid the appearance of the MS DOS shell. Next, we open a blank MS Word document. In its simplest form:

```
DATA _null_;
  rc = system('start winword');
  call sleep(2,1);
RUN;
```

The reader is encouraged to see Vyverman (2003) for a more complete explanation and in-depth examples.

Once the blank file is open, we insert the text using the WordBasic command **Insert()**:

```
DATA _null_;
  file sasword;
  put "[Insert('I used to live in SAS, but now I live in a RTF')]";
RUN;
```

The following saves, closes, and exits MS Word:

```
DATA _null_;
  file sasword;
  put '[FileSaveAs("C:\ex1.rtf"]';
  put '[FileClose 2]';
  put '[FileExit 2]';
RUN;
```

These are only a few ways SAS interacts with a non-SAS file. Slightly more complicated approaches employ VBA and or VB. This leads to the focus of this paper – automating the stacking of PDF files. We accomplish this by constructing a VB script in a data step. The contents are then exported to a VBS file and submitted. Before we show how to build this data step, here is a quick tutorial on the necessary VB functions and an explanation of the importance of IAC.

VISUAL BASIC AND INTER-APPLICATION COMMUNICATION

VB SCRIPT FORM

Our goal is to generate a VB script that has the following form:

To create NewDoc.pdf, for the i^{th} target PDF file and i = 1 to n,

```
Dim doc1
   .
   .
   .
Dim docn

Set Doc1 = CreateObject(“AcroExch.PDDoc”)
   .
   .
   .
Set Docn = CreateObject(“AcroExch.PDDoc”)
```

```
File1 = Doc1.Open("path\target1.pdf")
   .
   .
   .
Filen = Docn.Open("path\targetn.pdf")

Stack = Doc1.InsertPages(Doc1.GetNumPages - 1, Doci+1, 0, Doci+1.GetNumPages, 0)
   .
   .
   .
Stack = Docn.InsertPages(Docn.GetNumPages - 1, Docn-1, 0, Docn-1.GetNumPages, 0)

SaveStack= Doc1.Save(1, "path\NewDoc.pdf")
```

For example, to stack two PDF files (SESUG1.pdf and SESUG2.pdf) and name the stacked file *NewDoc.pdf*, our VB script is:

```
Dim Doc1
Dim Doc2

Set Doc1= CreateObject("AcroExch.PDDoc")
Set Doc2= CreateObject("AcroExch.PDDoc")

file1= Doc1.Open("C:\SAS\SESUG2011\Displays \ SESUG1.pdf")
file2= Doc2.Open("C:\SAS\SESUG2011\Displays \ SESUG2.pdf")

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

SaveStack= Doc1.Save(1,"C:\SAS\SESUG2011\Displays\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 V9.2, MS Windows XP Professional and Adobe Acrobat 8 Standard.

PART1: CREATING OBJECTS

```
Dim doc1
   .
   .
   .
Dim docn

Set Doc1 = CreateObject("AcroExch.PDDoc")
   .
   .
   .
Set Docn = CreateObject("AcroExch.PDDoc")

File1 = Doc1.Open("path\target1.pdf")
   .
   .
   .
Filen = Docn.Open("path\targetn.pdf")
```

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, 2007). 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 (file*x*) using the **Open()** method.

PART2: STACKING THE FILES

```
Stack = Doc1.InsertPages(Doc1.GetNumPages - 1, Doci+1, 0, Doci+1.GetNumPages, 0)
   .
   .
   .
Stack = Docn.InsertPages(Docn.GetNumPages - 1, Docn-1, 0, Docn-1.GetNumPages, 0)

SaveStack= Doc1.Save(1, "path\NewDoc.pdf")
```

We create an instance of the variable Stack for each interaction of the stack. This variable is for debugging purposes since the **InsertPages()** method returns -1 if successful and zero if unsuccessful. **InsertPages()** requires five parameters. The first parameter is the page number after the subsequent document is inserted (Adobe Systems, 2006). For instance, for the first iteration of the stack, we insert after the last page of *Doc1*. We automatically retrieve a document's page number using the **GetNumPages** method. We use this as the first parameter, but subtract unity because the first page in a PDDoc object is always zero (Adobe Systems, 2006). The second parameter is the document to insert into *Doc1*. The third is the starting page of the inserted document, which in this case is always the first page (start page = 0). The fourth parameter is the inserted document's number of pages. We use the **GetNumPages** function again to gather the number of pages. Finally, the last parameter is used for copying book marks. The final step of the program is to save with the **Save()** method.

VB Function	**Meaning**
`Dim`	VB statement used for declaring variables
`Set`	VB statement used to create objects
`CreateObject("AcroExch.PDDoc")`	VB function used to create an object.
`Open()`	Adobe-specific method used to open a PDDoc file
`InsertPages()`	Adobe-specific method used to "inserts the specified pages from the source document after the indicated page within the current document" (Adobe Systems, 2006)
`Save()`	Adobe-specific method used to save a document

Table 1. Map of Key VB/Adobe-specific Statements, Functions, and Methods Used to Stack PDF files

BUILDING THE VB SCRIPT

The primary vehicle we use for stacking PDF files is the VBS file. In the following example, we combine six PDF files from a mock clinical trials study. We stack five tables with a figure.

PART0: IDENTIFYING FILES TO COMBINE

First we create macro variables to denote the path and document names respectively – these will carry throughout the remainder of the program:

```
%let Dir = C:\SAS\SESUG2011\Displays;
%let NewFile = NewDoc;
```

Here we use SAS I/O code to identify all PDF folders in the target folder to stack:

```
DATA prep;

  folder = strip("&Dir");

  rc = filename('files',folder);
  did = dopen('files');
  numfiles = dnum(did);
  iter = 0;

  do i = 1 to numfiles;
    text = dread(did,i);
    if index(upcase(text),".PDF") then do;
      iter + 1;
```

```
        output;
      end;
      call symput("FileNum",put(iter,8.));
    end;

    rc = dclose(did);

  RUN;

  %put NUMBER OF PDF FILES TO STACK: %cmpres(&FileNum);
```

Note that at each loop we capture the number of PDF files in the macro variable &FileNum. At the end of the loop, the macro variable resolves to six files to stack.

OUTPUT TO LOG

```
NUMBER OF PDF FILES: 6
```

OUTPUT DATA SET:

FOLDER	TEXT
C:\SAS\SESUG2011\Displays	SESUG1.pdf
C:\SAS\SESUG2011\Displays	SESUG2.pdf
C:\SAS\SESUG2011\Displays	SESUG3.pdf
C:\SAS\SESUG2011\Displays	SESUG4.pdf
C:\SAS\SESUG2011\Displays	SESUG5.pdf
C:\SAS\SESUG2011\Displays	SESUG6.pdf

Now we use these six records to build the VB Script.

PART1: CREATING OBJECTS

From the list identified above, we translate the file names to properties using the **Set** statement. Within the **Set** method we use the **CreateObject()** function to create objects for each file. Using the **Open()** method, we open each PDF file to act on each object. In addition, we define an ORDER variable. We use this variable to sort the data set so the VB script will run in the appropriate order upon submission.

```
  %*PART 2;
  DATA indat1;
    length code $150;
    set prep;
    code = 'Dim Doc'||compress(put(_n_,8.));
    order = 1;
    output;
    code = 'Set Doc'||compress(put(_n_,8.))||'= CreateObject("AcroExch.PDDoc")';
    order = 2;
    output;
    code = 'file'||compress(put(_n_,8.))||
           ' = Doc'||compress(put(_n_,8.))||
           '.Open("'||strip("&Dir.\")||strip(text)||'")';
    order = 3;
    output;
  RUN;

  PROC SORT data = indat1;
    by order;
  RUN;
```

OUTPUT DATA SET:

CODE	ORDER
Dim Doc1	1
Dim Doc2	1
Dim Doc3	1
Dim Doc4	1
Dim Doc5	1
Dim Doc6	1
Set Doc1= CreateObject("AcroExch.PDDoc")	2
Set Doc2= CreateObject("AcroExch.PDDoc")	2
Set Doc3= CreateObject("AcroExch.PDDoc")	2
Set Doc4= CreateObject("AcroExch.PDDoc")	2
Set Doc5= CreateObject("AcroExch.PDDoc")	2
Set Doc6= CreateObject("AcroExch.PDDoc")	2
file1 = Doc1.Open("C:\SAS\SESUG2011\Displays\SESUG1.pdf")	3
file2 = Doc2.Open("C:\SAS\SESUG2011\Displays\SESUG2.pdf")	3
file3 = Doc3.Open("C:\SAS\SESUG2011\Displays\SESUG3.pdf")	3
file4 = Doc4.Open("C:\SAS\SESUG2011\Displays\SESUG4.pdf")	3
file5 = Doc5.Open("C:\SAS\SESUG2011\Displays\SESUG5.pdf")	3
file6 = Doc6.Open("C:\SAS\SESUG2011\Displays\SESUG6.pdf")	3

PART2: STACKING THE FILES

The final piece involves building the code to stack the files. We accomplish by using the **InsertPages()** method:

```
%*PART 3;
DATA indat2;
  length code $150;
  set prep end = eof;
  code = 'Stack = Doc1.InsertPages(Doc1.GetNumPages - 1, Doc'||
         compress(put((_n_ - 1) + 2,8.))||
         ', 0, Doc'||
         compress(put((_n_ - 1) + 2,8.))||
         '.GetNumPages, 0)';
  if eof then code = 'SaveStack= Doc1.Save(1, "'||
                     strip("&Dir.\")||
                     strip("&NewFile.")||'.pdf"'||')';
RUN;
```

OUTPUT DATA SET:

CODE
Stack = Doc1.InsertPages(Doc1.GetNumPages - 1, Doc2, 0, Doc2.GetNumPages, 0)
Stack = Doc1.InsertPages(Doc1.GetNumPages - 1, Doc3, 0, Doc3.GetNumPages, 0)
Stack = Doc1.InsertPages(Doc1.GetNumPages - 1, Doc4, 0, Doc4.GetNumPages, 0)
Stack = Doc1.InsertPages(Doc1.GetNumPages - 1, Doc5, 0, Doc5.GetNumPages, 0)
Stack = Doc1.InsertPages(Doc1.GetNumPages - 1, Doc6, 0, Doc6.GetNumPages, 0)
SaveStack= Doc1.Save(1, "C:\SAS\SESUG2011\Displays\NewDoc.pdf")

PART4: OUTPUT VBS FILE AND SUBMIT

The final step is to output the code in the above steps to a VBS file and submit the file:

```
*OUTPUT VBS CODE;
filename temp "&Dir.\PDFStack.vbs";
DATA allcode;
  set indat1 indat2;
  file temp;
  put code;
RUN;

*RUN VBS PROGRAM;
systask command "&Dir.\PDFStack.vbs";
```

CONCLUSION

This article offers only a few of the many ways to automate tasks outside of the SAS language. One possible modification to the code we present is to create a sort order for the PDF files used to stack - in our example the order of the displays are inherently sorted by the file name. In many real-world situations, the order of the files is important. For example, if combining across different clinical trials studies, it may be advantageous to put all demographic and disposition displays first, followed by the efficacy displays. This is possible using the code we have provided - with minimal modifications. With the ability of building foreign code (such as VB) within a data step, programmers may adapt their code to fit any PDF stacking need.

REFERENCES

Adobe Systems. Adobe® Acrobat® SDK Version 8.0. “Interapplication Communication API Reference”. http://www.adobe.com/content/dam/Adobe/en/devnet/acrobat/pdfs/iac_api_reference.pdf (2006)

Microsoft Corporation. “Visual Basic for Applications Language Reference”. http://office.microsoft.com/client/helpcategory.aspx?CategoryID=CH806073319990&lcid=1033&NS=WINWORD (2007)

Vyverman, K. (2003). “Fancy MS Word Reports Made Easy: Harnessing the Power of Dynamic Data Exchange (Against All ODS, Part II)”. Proceedings of 28th SAS Users Group International Conference.

Xtreme Visual Basic Talk. “Combine PDF Files”. http://www.xtremevbtalk.com/archive/index.php/t-111697.html (2003)

ACKNOWLEDGMENTS

Steve Noga
Rita Slater
Eva J. Welch

CONTACT INFORMATION

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

Brandon Welch
Rho, Inc
6330 Quadrangle Dr., Ste. 500
Chapel Hill, NC 27517
Phone: 919-595-6339
Fax: 919-408-0999
Email: Brandon_Welch@rhoworld.com
Web: www.rhoworld.com

APPENDIX

```
 %let Dir = ;
 %let NewFile = ;

 %*********************;
 %*GET FOLDER CONTENTS;
 %*********************;

 %*PART 1;
 DATA prep;

   folder = strip("&Dir");

   rc = filename('files',folder);
   did = dopen('files');
   numfiles = dnum(did);
   iter = 0;

   do i = 1 to numfiles;
     text = dread(did,i);
     if index(upcase(text),".PDF") then do;
       iter + 1;
       output;
     end;
     call symput("FileNum",put(iter,8.));
   end;

   rc = dclose(did);

RUN;

%put NUMBER OF PDF FILES TO STACK: %cmpres(&FileNum);

****************;
*BUILD VB SCRIPT;
****************;

 %*PART 2;
 DATA indat1;
   length code $150;
   set prep;
   code = 'Dim Doc'||compress(put(_n_,8.));
   order = 1;
   output;
   code = 'Set Doc'||compress(put(_n_,8.))||'= CreateObject("AcroExch.PDDoc")';
   order = 2;
   output;
   code = 'file'||compress(put(_n_,8.))||
          ' = Doc'||compress(put(_n_,8.))||
          '.Open("'||strip("&Dir.\")||strip(text)||'")';
   order = 3;
   output;
 RUN;

 PROC SORT data = indat1;
   by order;
 RUN;

 %*PART 3;
 DATA indat2;
   length code $150;
   set prep end = eof;
   code = 'Stack = Doc1.InsertPages(Doc1.GetNumPages - 1, Doc'||
          compress(put((_n_ - 1) + 2,8.))||
          ', 0, Doc'||
          compress(put((_n_ - 1) + 2,8.))||
         '.GetNumPages, 0)';
  if eof then code = 'SaveStack= Doc1.Save(1, "'||
                     strip("&Dir.\")||
                     strip("&NewFile.")||'.pdf"'||')';
RUN;

********************;
*END BUILD VB SCRIPT;
********************;

*OUTPUT VB SCRIPT;
filename temp "&Dir.\PDFStack.vbs";
DATA allcode;
  set indat1 indat2;
  file temp;
  put code;
RUN;

*RUN VB SCRIPT;
systask command "&Dir.\PDFStack.vbs";
```