Paper SER10_05

EXTRACTING DATA FROM PDF FILES

Nat Wooding, Dominion Virginia Power, Richmond, Virginia

ABSTRACT

The Adobe Portable Document File (PDF) format has become a popular means of producing documents for use on other computers when the author cannot be certain of the software available on the other machines. However, once a document is stored in this format, it may generally be considered to be "read only" and the Adobe Acrobat Reader software does not offer a text extraction tool. This paper discusses third-party software that can translate a PDF to ASCII and shows how to automate the process using SAS®.

INTRODUCTION

The Adobe Corporation has developed a highly useful file format for storing and shipping documents. Their marketing strategy is that they sell the Acrobat product which converts the document to this Portable Document File (PDF) format and they offer the Acrobat Reader software free or for a nominal price. A large document such as the SESUG proceedings can be stored in this format on a compact disc and the reader software placed on the disc also. The Reader software is offered for many different mini- and microcomputer operating systems which makes this format even more attractive for document distribution to audiences with unknown computing capability.

However, once a document is received in this format, it is in a read-only mode as is a printed book and Acrobat Reader does not offer a text extraction tool. This presents a challenge if one receives data in the PDF format and one needs to be able to use and manipulate these data.

I was recently faced with extracting data from some 2000 individual PDF files and was able to use a third-party software which I will generically call GhostScript to extract these data. Furthermore, I was able to use SAS®' capability of running external programs to automate the extraction process.

EXTRACTION TOOLS

Adobe Acrobat (the file creator product includes a file conversion tool but most users do not have this product; rather One can always open the file and manually transcribe the needed information. Given access to Optical Character Reader (OCR) software and a scanner, one can print a few pages, scan them, and then convert them to text. I have used this when dealing with a small table that I did not want to rekey.

There is also a product called Ghostscript which is described as an interpreter of Postscript and PDF formats. It is available for Unix and DOS/Windows systems and, depending on the release, may be downloaded free-of-charge by individual users. It appears to have been originally developed and maintained by one individual and is now maintained by a small shop called Artofcode, LLC. The Ghostscript product has a fairly primitive interface so a number of individuals have written and distributed GUI front ends. I have only used the one called GSView. Packaged with Ghostscript is an executable module called PS2ASCII which is what I used for the automation process.

One useful item to note is that GhostScript can be used to view graphics that are stored in PostScript format for embedding in documents. In the past, SAS® Technical Support has recommended using GhostScript to check SAS Graph® output as Postscript which would not display after being embedded in a document.

My recommendation is that if you have only a few files from which you need data, use GhostScript via the GSView interface. If you have more than a few files, keep reading.

MY PROBLEM

Following the several-year drought in eastern Virginia in the early 2000s, Dominion has been taking another look at the water supply for the reactors at the North Anna Nuclear Power Station and part of this work has involved modeling the thermal loading of the lake. One of the parameters that needed to be included was lake level. The level of Lake Anna is recorded four times daily by the operator at the dam and these values are entered into a database of various operational measurements that, when printed, runs for about 30 or 40 pages per day. About seven years ago, these records were compiled on compact disks in PDF format with each day comprising at least one separate file. For this study, we needed to extract the levels from six disks which meant that 6*365 files had to be opened, the appropriate lines found, and the data recorded. This presented an interesting challenge.

A SOLUTION

Basically, I needed some means to have each file opened, the data converted to ASCII text, the appropriate data found and extracted, and then stored. I needed a way to automate GhostScript.

SAS® has, for many years, offered the “X” command which allows one to send a command to the operating system. My first memory of it was jokes told of how one could use the MVS TSO “Send” command to send anonymous messages. This command was intended to be a sort of instant messaging system and it automatically attached your user id to the message. For some reason, when one sent a message from within a SAS® session under TSO, the user id was deleted and the sender could attach a fake ID. Unfortunately, the systems programmers at the SAS® Institute found out about the situation and fixed it.

When used properly, the command can be used to submit programs, operating systems commands, or batch type files for execution. You need to include this in a data step such as

```
 DATA _NULL_;
      X “command” ;
RUN;
```

In the windows environment, the command could change to a different directory, run an executable file or run a batch file.

For running an executable file, at the minimum you need to specify the file name and path. You may also have to specify file names and switches.

In my case, my null step was

```
DATA  _NULL_;
       X “CD C:\GS”;
       X “GSSETGS.BAT”;
       X “PS2ASCII INPUTPDF OUTTXT”;
RUN;
```

Here, I point to the GhostScript directory where PS2ASCII is located, I run a batch file that has several GhostScript settings, and I then run the file converter while telling it the file to read and where to store the output.

The batch file was described in the online GhostScript documentation but took some tinkering to get the proper output. It is listed in Appendix 1.

THE ACTUAL SAS® JOB

The SAS® job was organized as follows: first, PS2ASCII read the PDF and stored the output in a file; then, a DATA step read and parsed this file looking for the word “Lake” which started the lake level readings. The lake levels were captured and appended to a file of these values. The parsing step was not particularly complicated but required tinkering since the records were not always consistent. Appendix 2 has an outline of the job.

There were several important details in making the job work. First, the X command opens a DOS window. In order to not have to close this window manually, you will need.

```
OPTIONS NOXWAIT;
```

In my program, I also included the option NOXSYNC but this may not have been necessary.

In order for PS2ASCII to have time to complete it’s operation and close the file, I added the step

```
DATA _NULL_;
     SLEEP(15);
```

```
RUN;
```

which told SAS® to pause for 15 seconds.

Once the code was working properly, I packaged it in a macro. I then wrote a simple loop and put statement routine that created and stored a year's worth of macro calls. In the case of these files, I could not used a macro generator since there were a few days that had two files and I had to manually add these additional macro calls.

Given the 15 second pause for each file and rest of the time for the program execution, each CD required almost two hours to be read. As I mentioned, there were several occasions when the entries did not follow the usual form so after each run, I had to scan the output and sometimes modify the parsing routine and rerun the CD.

OTHER USES OF THE SAS® 'X' COMMAND

I have mentioned a couple of these but here are a few more.

DDE (Direct Data Exchange) can be used to have SAS® communicate with some windows-based products such as Excel. When using DDE, both SAS® and the other program must be running and the X command is one means of launching EXCEL and is the method for passing instructions to Excel.

System utilities can be invoked with X. In addition to changing a directory, one can issue rename and delete commands. I have seen SAS®/L postings where someone wanted to search a directory and used the X command to issue a DOS DIR statement that did the listing and stored the list so that SAS® could parse it.

I should point out that

```
X  “command”;
```

And

```
SYSTEM “command”;
```

Are synonymous.

Additionally, SAS® can run a compiled routine (assuming that it can find it) by simply coding

```
PROC whatever;
```

This assumes that one can pass any necessary parameters to the compiled routine. I have seen postings about running the MVS utility IDCAMS in this way.

CONCLUSION

SAS® can sometimes use other software to accomplish tasks that are impossible or difficult to achieve with standard SAS® tools.

APPENDIX 1 THE GHOSTSCRIPT BATCH FILE

```
@echo off
@rem $Id: gssetgs.bat,v 1.3 2001/06/22 16:09:22 lpd Exp $

rem Set default values for GS (gs with graphics window) and GSC
rem (console mode gs) if the user hasn't set them.
 set GSC=gswin32c
```

APPENDIX 2 The SAS® JOB

This job has very detailed code for parsing the specific file that I was reading. It is intended to only be an example of what one might need to do to read a PDF.

```
LIBNAME NAT 'C:\MYFILE';

OPTIONS MPRINT MACROGEN SYMBOLGEN;


DM 'OUTPUT; CLEAR';
DM 'LOG; CLEAR';
DM 'PGM';
DM 'CAPS ON';
DM 'NUM ON';

DATA  ALL;* THIS IS A DUMMY ENTRY THAT SETS UP THE FILE WHERE THE READINGS WILL BE
STORED;
        DATE=.;
        _1=.;
        _2=.;
        _3=.;
        _4=.;

 RUN;

 OPTIONS NOXWAIT NOXSYNC MPRINT DATE;

 %MACRO DOIT(DATE,FILE);

DATA _NULL_ ;

        X 'CD C:\GS';

        X 'GSSETGS.BAT' ;
        X   "PS2ASCII  E:\BB&FILE..PDF C:\PARK\BB.TXT ";

DATA _NULL_;
        X=SLEEP(15);
        FILENAME PDF 'C:\PARK\BB.TXT'; * WHERE THE TRANSLATED FILE IS STORED;

DATA A; *FILE SPECIFIC CODE FOLLOWS;
        INFILE PDF LRECL=256;
        INFORMAT CK $CHAR256.;
        INPUT   ;
        CK=UPCASE(_INFILE_);
        IF INDEX(CK,'NOTE') THEN DELETE;
        COUNT=0;
        IF INDEX(CK,'LAKE') THEN DO UNTIL(COUNT GT 5);
                INPUT   ;
                CK=_INFILE_;
                CK=COMPBL(CK);
                CK=LEFT(CK);
                OUTPUT;
                COUNT+1;
        END;
        DROP COUNT;
```

```
DATA A;
          SET;
          IF INDEX(CK,':');
          LENGTH STRING $20;
          DO I=1 TO 20;

                     DATE =&DATE;
                      FORMAT DATE MMDDYY10.;
                      STRING=SCAN(CK,I,' ');
                    IF STRING GT ' ' THEN DO ;
                               STRING=COMPRESS(UPCASE(STRING));
                               INITIAL_LENGTH=LENGTH(STRING);
                              STRING=(COMPRESS(STRING,'ABCDEFGHIJKLMNOPQRSTUVWXYZ,-'));
                              LEVEL=1*STRING;
                              IF LEVEL THEN OUTPUT;
                    END;
          DROP CK;
          END;

 RUN;

DATA A;
          SET;COUNT+1;* THIS SETS UP AN ID VARIABLE FOR TRANSPOSE;

PROC TRANSPOSE OUT=A (DROP=_NAME_);* I want each day to be one obs;
          VAR LEVEL;
          ID COUNT;
          BY DATE;

DATA ALL;
          SET ALL A;

PROC PRINT;

RUN;
          LIBNAME PDF CLEAR;
RUN;
%MEND DOIT;


%DOIT(13149,000001)
%DOIT(13150,000002) etc
```

Contact the author at:

Nat Wooding
Dominion Virginia Power
4111 Castlewood Rd.
Richmond, Va 23235

(804) 271- 5313
Nathaniel_wooding@dom.com