

Paper CC-015

Using External Files to Maintain an Application

Author, Edgar Adams

ABSTRACT

Tired of re-writing your SAS® code every time the data changes. Then keep your changing data in an external file for the application to read and process. The moment the application is executed is when the “most current” data is used by the application. Maintain the external file or files and the application is always up-to-date. The trick is to store them in a way that uses the SAS format power of processing; storing the data Values and the Meaning of those Values.

INTRODUCTION

The process uses an easy to edit spreadsheet and a few built in functions found in Base SAS.

PROC IMPORT

PROC FORMAT

PROC IMPORT, using a database management system (DBMS) of Excel, allows SAS to read a Microsoft Excel® spreadsheet into a SAS dataset. An additional simple dataset can then be used to create the needed variables for Proc FORMAT to take the spreadsheet information and build usable internal formats. The code necessary for these three steps is shown on the next page and is in the attached documents labeled SAS Code ~~Filename=~~ District-Format.txt.

BODY

For this exercise the scenario will center on a group of Florida Counties or Districts that submit data to a central location. We will act like the central location.

A Microsoft Excel spreadsheet is used to house the names of the Counties or Districts that are expected to provide information to the central location. These are the CHANGING data elements that are read by the SAS program. The Counties or Districts may only change on a yearly basis, but if you need to track down each and every application that your organization has that deals with the Counties or Districts it will take time and every application would need to be changed with the same up-to-date information. Making the change in one location is far better than making the change in two, or three, or more locations.

Shown here as a sample spreadsheet that represents the County or District Numbers and Names:

DistrictNO	DistrictNA
57	Hillsborough
81	Manatee
101	Pasco
103	Pinellas
105	Polk

Reading the above spreadsheet information – District number 57 represents Hillsborough County, District number 81 represents Manatee County, etc. As other Counties or Districts are added or names changed or are re-numbered;

the spreadsheet can grow or be revised without need of changing the SAS program code. Shown below is the external file called DistrictFormat.txt. It contains the three parts mentioned previously

PROC IMPORT step Dataset step PROC FORMAT step

If you look below in the first line of the DistrictFormat.txt file you will see a string "SESUG&YR4." without the quotes. In keeping with maintaining the code by using external files, the "Date" or "Vintage" of the data must be in an external file too. SAS provides a great way to make global variables, using SAS Macro Variables. The %LET statement [For Example: %LET = 2008;] is filed away in a file called YR4.txt. Used in the SAS code as &YR4. the substitution will become 2008 [Where the & starts a macro variable and the . ends a macro variable.] A %INCLUDE statement is used to bring the file YR4.txt into the SAS code.

The Key to using the spreadsheet is to get the information into a usable form and the PROC FORMAT processing in SAS is very convenient for that process.

DistrictFormat.txt file

```
-----
filename OfflList "C:\SESUG&YR4.\DATA\District_names_numbers.xls";
PPROC IMPORT datafile=OfflList
    out=DistsFILE
    dbms=excel REPLACE;
run; ----- Step 1
DATA DistFORMAT; SET DistFILE;
    fMtname="Di_TBL";
    LABEL=DistrictNA;
    START=DistrictNO;
    END= DistrictNO;
    KEEP FMTNAME LABEL START END;
run; ----- Step 2
PROC FORMAT CNTLIN=DistFORMAT;
run; ----- Step 3
-----
```

The above block of code reads in the spreadsheet using a dbms EXCEL engine, creates the statements needed for PROC FORMAT to read and create a format of District Numbers and associated District Names to be referenced by the Format variable name of DiTBL. Once in the program, the uses of this information are to make report titles and meaningful file names.

This code is also saved external to the SAS program code and is brought in using a %INCLUDE statement. Again, for flexibility and maintenance issues, once this code is written and executing properly, it can be inserted into other applications for ease of programming.

Attached is a copy of the entire program SAS code, the DistrictFormat text file, the YEAR text file and portions of the SAS Log.

CONCLUSION

Once you try a simple example like the above and see the power of PROC FORMAT, you may never go back to having to change your SAS code again. Once it works, it works every time. When Districts change names or more are added or even removed, the SAS code runs without any additional changes. Pretty neat, I think. Have a Great Day!

ACKNOWLEDGMENTS

This is a subset of a paper that was awarded the "Best Presentation" in the section "Information Systems & Web Applications" at the 16th MidWest SAS Users Group Conference held in Cincinnati, October 9-11, 2005. That recognition gave me the courage to try another presentation today. And to John Wickizer, my client manager at the Commonwealth of Kentucky, Department of Education, who lead me to this way of thinking.

CONTACT INFORMATION

Edgar Adams
Keane, Inc.
9 Fountain Place
Frankfort, KY 40601
(502) 564-9853
Edgar.Adams@Education.ky.gov

SAS and all other SAS Institute Inc. product or service names are registered trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are trademarks of their respective companies.

SAS Application Code

```

/*****
/* Purpose of this routine...
/*
/* Create a teaching sample of how to maintain an application using
/* external files.
/*
/* Written by: Edgar Adams, Keane Inc.
/* SESUG 2008
*****/

/* This application creates simulated reports for all of the "Districts"
/* by using the district names and numbers contained in the external file.

/* Using the External file, where the changes occur, permits the application
/* to run year after year without modifications. Once it runs error free,
/* it will continue to run, because all of the changes are external to the
/* SAS code.

*The OPTIONS statement is used to establish the layout desired for this process;
OPTIONS Linesize = 80
        Center
        Date
        Nonumber;
*The PDF ODS statement closes the default. Later another ODS statement will all the;
* use of another output presentation
ODS Listing Close;

*Using an INCLUDE statement allows a "YEAR" vintage to be used in data processing;
*Again, it is external to the application so that changes to code are not needed ;
%Include "C:\Processing\YEAR\YR4.txt";
        *Creates a global variable &YR4. Which in this teaching case contains 2008;

*Using an INCLUDE statement allows the read and conversion to FORMAT to occur;
*external to the body of the code. This one TXT file may be referenced by ;
*many SAS applications and if a District change is required, it only requires;
*ONE place to make the change.
%Include "C:\SESUG&YR4.\data\DistrictFormat.txt";

/*****
/* The learning is over, the Application now has the District information in
/* a form that is useable.
/*
/* What will happen next is to cycle through the folder of data sent in by
/* the Districts. Each District that provides a data file will receive a
/* District specific report of their information.
*****/

*Provide the location of the District Files;
Filename DIR "C:\SESUG&YR4.\DATA\DistrictFiles\";

*Date the reports with current date, set into a global variable;
%LET RptDate = %SYSFUNC(date(), mmddyy6.);

*Data from the Districts come in as numbers, need to report both
* Number of District and
* Name of District;
%Macro CreateRpts(FileToRead);
DATA NULL;
    District_Number = &Dist.;
    tempDistrictName = TRIM(put(District_Number,Di_Tbl.));
    *make the District Name into a global variable;
    CALL SYMPUT('DName',TRIM(tempDistrictName));
    *make sure that all district number are three digits long, by

```

```

        making smaller numbers Left Zero Filled;
        tempDistrictNumber= TRIM(Put(District_Number,z3.));
        CALL SYMPUT('DiNum',TRIM(tempDistrictNumber));
run;

*Create individual District Reports in the PDF format;
ODS PDF style=Printer
file = "C:\SESUG&YR4.\REPORTs\Rpt_&DiNum._%TRIM(&DName.)_&RptDate..PDF";

*Begin report process by reading in the data provided by the District;
DATA WorkFile;
    %LET _EFIERR_ = 0;
    INFILE "C:\SESUG&YR4.\Data\DistrictFiles\&FileToRead."
        MISSOVER DSD lrecl=32767 FirstObs=1;
    INPUT Count;
    If _ERROR_ then CALL SYMPUT('_EFIERR_',1);
run;

*Now report the data;
PROC PRINT data= WorkFile Label;
    Title1 "Report for &YR4.";
    Title2 "=====";
    Title3 "District: &DName.";
    Label Count="Number of Items";
    ID Count;
run;
ODS PDF Close;
%Mend; *CreateRpts(FileToRead);

*options symbolgen mprint mlogic;
OPTIONS nosymbolgen nomprint nomlogic;

* Need to loop through folder of district data, determine how many have
* submitted their data and what was the name of the file they submitted;
%Macro CkDirectory(DirRef);
    *Open directory of district data files;
    %LET Drc = %SYSFUNC(DOpen(&DirRef.));
    *Find the number of files;
    %LET NumFiles = %SYSFUNC(DNum(&Drc.));
    *Put a statement into the log on the Count of data files;
    %If (&NumFiles. = .) %Then %PUT The data directory is empty;
    %else %Put The folder contains &NumFiles. files;;

    *Loop through data folder to process reports;
    %Do I = 1 %to &NumFiles.;
        *Locate district number in file name;
        *Required layout is XXX_nnn_XXXXXXXXXXXX.csv;
        *Where the 'nnn' is the position of the district number;

        * Strip out District Number and Read filename;
        %LET Dist = %SUBSTR(%SYSFUNC(dread(&drc,&I)),6,3);
        %LET TheFileName = %SYSFUNC(dread(&drc,&I));

        *Add a LOG note to track progress;
        %PUT District number &Dist. FileName: &TheFileName.;

        * Call the report writer;
        %CreateRpts(&TheFileName);
    %end;

    %LET rc=%SYSFUNC(dclose(&drc));
%mend; * CkDirectory(DirRef);

* Begin the process by providing the location of the District Files;
%CkDirectory(DIR);

QUIT;

```

SAS Code

File name = YR4.txt

```
%Let YR4 = 2008;
```

File name = DistrictFormat.txt

```
filename OfflList "C:\SESUG&YR4.\DATA\District_names_numbers.xls";

PROC IMPORT DATAFILE=OfflList
  OUT=DistFile
  DBMS=excel REPLACE;
Run;

DATA DistFORMAT; SET DistFILE;
  fMtname="Di_TBL";
  Label=DistrictNA;
  Start=DistrictNO;
  End= DistrictNO;
  KEEP FmtName Label Start End;
Run;

PROC FORMAT CNTLIN=DistFORMAT;
Run;
```

SAS LOG

```
30  %Include "C:\ProcessingYEAR\YR4.txt";
32      *Creates a global variable &YR4. Which in this teaching case contains
32 ! 2008;
33
34      *Using an INCLUDE statement allows the read and conversion to FORMAT to
34 ! occur;
35      *external to the body of the code. This one TXT file may be referenced by
35 ! ;
36      *many SAS applications and if a District change is required, it only
36 ! requires;
37      *ONE place to make the change.
37 ! ;
38  %Include "C:\SESUG&YR4.\data\DistrictFormat.txt";
```

NOTE: .DISTFILE was successfully created.

NOTE: PROCEDURE IMPORT used (Total process time):

real time	4.35 seconds
cpu time	0.68 seconds

NOTE: There were 5 observations read from the data set WORK.DISTFILE.

NOTE: The data set WORK.DISTFORMAT has 5 observations and 4 variables.

NOTE: DATA statement used (Total process time):

real time	0.23 seconds
cpu time	0.15 seconds

NOTE: Format DI_TBL has been output.

NOTE: PROCEDURE FORMAT used (Total process time):

real time	0.23 seconds
cpu time	0.06 seconds

NOTE: There were 5 observations read from the data set WORK.DISTFORMAT.

Comments about above log entries.

- 1) Line 30 is the INCLUDE that brings in the "&LET YR4 =2008;" statement. This does not produce any log entries. It just happens and you know it worked when line 38 is executed without an error.
- 2) Line 38 is the INCLUDE that brings in and processes the format District data. This does produce log entries when files are imported and created and when PROC FORMAT is run.

SAS LOG

The folder contains 5 files
District number 057 FileName: DR08_057_Fall data count.csv

NOTE: The data set WORK.NULL has 1 observations and 3 variables.
NOTE: DATA statement used (Total process time):
 real time 0.04 seconds
 cpu time 0.00 seconds

NOTE: Writing ODS PDF output to DISK destination
 "C:\SESUG2008\REPORTS\Rpt_057_Hillsborough_063008.PDF", printer "PDF".

NOTE: Numeric values have been converted to character
 values at the places given by: (Line):(Column).
 6:190

NOTE: The infile "C:\SESUG2008\Data\DistrictFiles\DR08_057_Fall data count.csv"
 is:
 File Name=C:\SESUG2008\Data\DistrictFiles\DR08_057_Fall data count.csv,
 RECFM=V,LRECL=32767

NOTE: 1 record was read from the infile
 "C:\SESUG2008\Data\DistrictFiles\DR08_057_Fall data count.csv".
 The minimum record length was 2.
 The maximum record length was 2.

NOTE: The data set WORK.WORKFILE has 1 observations and 1 variables.
NOTE: DATA statement used (Total process time):
 real time 0.12 seconds
 cpu time 0.00 seconds

NOTE: There were 1 observations read from the data set WORK.WORKFILE.
NOTE: PROCEDURE PRINT used (Total process time):
 real time 0.53 seconds
 cpu time 0.01 seconds

NOTE: ODS PDF printed 1 page to
 C:\SESUG2008\REPORTS\Rpt_057_Hillsborough_063008.PDF.

Comments about above log entries.

- 1) Top line is simply a PUT statement that lets me know how many districts have data to process.
- 2) The remainder of this page is the result of processing one district. It shows the input data file on the second line, the created report on the last line, as well as some information about the amount of data being processed. I use this log information as a double check against the expectations. Each district would have a similar amount of information to review.