Paper 76

Managing SAS® Programs

David L. Ward, Innovex DAS, Inc., Piscataway, NJ

ABSTRACT

This paper presents a solution to the ever-present problem of SAS® program documentation. As companies that use SAS® grow, so does the number of SAS® programs that they must manage. This can quickly become a frightening task since SAS® programs are so easy to set up and run. In the course of a few weeks, hundreds of SAS® programs can emerge, and with no good method of documenting them, finding the right program at the right time can be impossible.

INTRODUCTION

To address this problem, our team decided to create a standard program header that contained program-specific information such as author, client, project, etc. The header was included in all new programs and added to some existing ones. Having such a standard has turned out to be extremely useful in more ways than we had imagined. Four practical applications of using a standard header are outlined in this paper.

THE HEADER

We use the following lines of code as our standard program header. (It has been squeezed to fit into one column)

```
/***********************************************
* Program:   xxxxxx.SAS                       *
* Location:  x:\xxxx\xxxxxx\xxxxx             *
* Client:    xxxxxx                           *
* Protocol:  xx-xxx  (xxx xx xxx)             *
* Author:    xxxxxxx                          *
* Created:   xx/xx/xx                         *
* Summary:   xxxxxx xxx xxxxx xxxxxxxxxxx xxx  *
* Revisions: xxxxx xxx xxxx xxxxxxxxxxxx      *
************************** Innovex DAS, Inc. **/
```

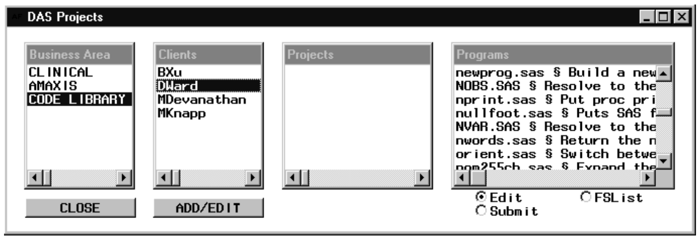
APPLICATION 1: PRINTING A LIST OF PROGRAMS

Programmers often need to produce listings of all programs written for a particular project or directory. When such a list is required (either for a boss or a client), the programmer usually ends up having to maintain a separate file containing this information. This can be a tedious task in which poor maintenance can result in misleading or confusing information. We have developed a macro that prints a standard listing of programs, containing the header information entered by the programmer, to the SAS® output window and creates a SAS® data set containing the same information. The general idea is to read in a list of *.sas files from a piped operating system command, then read the first 10 lines from each file, scanning the text for the header information. The filename “SASPROGS” is a Windows NT command.

```
%macro sasprogs (path);
  filename sasprogs pipe "dir ""&path\*.sas""";
  data programs;
    length line $75 program $20 date time size 8
      client $20 protocol author $25 created $12
      summary $65;
    infile sasprogs pad truncover;
    input line $char75.;
    if index(upcase(line),'.SAS')>0 then do;
      program = scan(upcase(line),4,' ');
      date=input(scan(line,1,' '),mmddyy8.);
      time=input(scan(line,2,' '),time5.);
      size=input(scan(line,3,' '),comma8.);
    end;
    format date mmddyy8. time time5.
      size comma8.;
   if program^=' ' then do;
    file="&path\"||program;
    infile temp filevar=file eof=bot pad
      truncover;
    i=1;
    do while (i<=10);
      input line $char75.;
      pos1=index(line,':');
      pos2=index(line,'''');
      pos3=index(line,'"');
      pos=max(pos1,pos2,pos3);
      line=translate(line,'','''');
      line=translate(line,'','"');
      line=tranwrd(line,'*/','');
      line=translate(line,'','*');
     if pos>0 then do;
      if index(upcase(line),'CLIENT')>0 then
        client=left(substr(line,pos+1,20));
      if index(upcase(line),'PROTOCOL')>0
        then protocol=
        left(substr(line,pos+1,25));
      if index(upcase(line),'AUTHOR')>0 then
        author=left(substr(line,pos+1,25));
      if index(upcase(line),'CREATED')>0 then
        created=left(substr(line,pos+1,12));
      if index(upcase(line),'SUMMARY')>0 then
        summary=left(substr(line,pos+1,65));
     end;
      i=i+1;
    end;
   end;
    keep program date time size client
      protocol author created summary;
    if program^=' ';
   return;
   bot:
  run;
  options pageno=1 nodate;
  title "Listing of SAS Programs Contained in
    &path";
  title2 "Run date: &sysdate";
  footnote "PROGRAM: U:\XXXX\SASPROGS.SAS";
  proc print data=programs width=min
    split='*' noobs;
    id program;
    var date time size client protocol author
      created summary;
    label date='DATE LAST*MODIFIED'
      time='TIME LAST*MODIFIED';
  run;
  title;
  footnote;
  filename sasprogs;
%mend sasprogs;
```

APPLICATION 2: INTEGRATION WITH SAS/AF®

To provide quick access to project specific SAS programs we have created a simple SAS/AF® frame entry that provides many useful options for viewing, editing, or submitting SAS® programs. Populating the list boxes is a data set that contains information about all of our projects. In this data set we store, among other things, the directory where the project’s SAS® programs exist.


DAS Projects

Business Area	Clients	Projects	Programs
CLINICAL	BXu		newprog.sas § Build a new
AMAXIS	DWard		NOBS.SAS § Resolve to the
CODE LIBRARY	MDevanathan		nprint.sas § Put proc pri
	MKnapp		nullfoot.sas § Puts SAS f
			NVAR.SAS § Resolve to the
			nwords.sas § Return the n
			orient.sas § Switch betwe
			nom255ch.sas § Expand the

[✓ Edit] FSList Submit

The “Projects Window” allows our programmers to select a client and project. The SCL® program then loops through all of the SAS® programs in the correct directory and reads in the summary field from the header. This information is displayed alongside the program name and makes it very easy to find the desired program. The screen shot here shows only our in-house code library (macros and %include files) since other information may be proprietary.

The frame entry allows you to quickly navigate to a desired project, view a list of programs with a description of what the program does, and either bring the program into the program editor, submit it, or open it in an FSList® window (to compare multiple programs). The “Programs” list box also allows you to navigate into sub-directories by clicking on the directory name, or open a catalog window for SAS® catalog entries by clicking on the catalog name. The following SCL® code populates the “Programs” list box:

```
projects:
  call notify ('PROJECTS','_GET_DATA_',prott);
  programs=makelist();

  * FIND PROGRAMS PATH CODE FROM PROJECTS DATASET *;
  dsid=open('sasres.projects (where= (busarea="'||
    barea||'" and client="'||client||'" and
    prott="'||prott||'"))','I');
  rc=fetch(dsid);
  progs=getvarc(dsid,varnum(dsid,'progs'));
  path=getvarc(dsid,varnum(dsid,'path'));
  rc=close(dsid);

 * FILL PROGRAMS BOX WITH LIST OF PROGRAMS AND
   DIRECTORIES *;
 if progs^='' then do;
  rc=filename('PROGSL',progs);
  rc=libname('_PROGS_',progs);
  did=dopen('PROGSL');
  do i = 1 to dnum(did);
    file=dread(did,i);
    ** CHECK IF FILE IS A DIRECTORY **;
    fid=mopen(did,file,'I');
    if fid>0 then do;
      do j = 1 to 7;
        rc=fread(fid);
      end;
      rc=fread(fid);
      rc=fget(fid,summary,78);
      rc=fclose(fid);
      if index(summary,'Summary')>0 then file=
        trim(file)||' § '||
        left(scan(summary,2,":*'"));
    end;

    if index(upcase(file),'.SC2')>0 then
      file='¤'||trim(file)||'¤';
    if fid<1 then file='«'||trim(file)||'»';
    if file not in ('«.»','«..»') and
      (index(upcase(file),'.SAS')>0 or
       index(upcase(file),'.SCL')>0 or
       index(file,'«')>0 or index(file,'¤')>0)
      then rc=insertc(programs,file);
  end;
  rc=dclose(did);

  rc=revlist(programs);
  call notify ('PROGRAMS','_REPOPULATE_');
 end;
return;
```

APPLICATION 3: CREATING HEADERS

Since creating a program header is an extra step, it may be difficult to get programmers to use the header at all times. Call it laziness or creative productivity, but we have developed a SAS® macro to automatically generate a filled-in header when creating new programs. The macro is issued at the command line and is simple in concept. First, an open dialog box is displayed prompting the user to create a new SAS® program. The macro then grabs the full file path from the file dialog box and parses the path to extract header information. Our Local Area Network (LAN) is set up so that a typical program may look like: “q:\clinical\client name\project name\sas\programs\myprog.sas” The client, protocol, and program name can be extracted from this information, the username can be obtained by %sysget(username), and the date by %sysfunc(today()). The macro then writes a filled-in header, using data step code (via the global command “gsubmit”), to the new program and includes it into the program editor with the command “include”. The macro function “%nwords”, used on line 18, is a user-written macro that returns that number of words of a string separated by a given delimiter.

```
%macro newprog;

 /*SHOW SAVEAS DIALOG BOX AND GET SELECTION*/
 %let opendlg=;
 gsubmit 'proc display
   cat=sasres.sasres2.opendlg.scl; run;';

 /*IF SELECTION MADE, PROCEED*/
 %if %quote(&opendlg)^= %then %do;

  pgm; clear;

 /*WRITE HEADER INFORMATION TO NEW PROGRAM*/
  %let file=tmp001;
  %let rc=%sysfunc(filename(file,&opendlg));
  %let path=&opendlg;
  %let path=%upcase(&path);
  %let n=%nwords(&path,\);
  %let prog=%scan(&path,&n,\);
  %let client=%scan(&path,2,\);
  %if %quote(&client)=CLINICAL or
    %quote(&client)=AMAXIS %then %let
    client=%scan(&path,3,\);
  %else %let client=;
  %if %quote(&client)^= %then %let proto
    =%scan(&path,4,\);
  %else %let proto=;
  /* GET AUTHORS REAL NAME FROM DOS */
  gsubmit "filename dos pipe 'net user
    /domain %sysget(username)';";
  gsubmit "data _null_;";
  gsubmit "length last first $50;";
  gsubmit "infile dos truncover dlm=',';";
  gsubmit "input @'Full Name' Last First;";
  gsubmit "if last^='' then call symput
    ('author',trim(first)||' '||
    trim(last));";
  gsubmit "run;";
  gsubmit "filename dos;";
  %*let author=%upcase(%sysget(username));
  %let path=%sysfunc(tranwrd(&path,\&prog,
    %str()));
  %let date=%sysfunc(today());
  %let date=%sysfunc(putn(&date,mmddyy8.));
  %if %length(&path)>70 %then %let pos=%eval
    (%length(&path)+15);
  %else %let pos=85;

  /* WRITE CALCULATED HEADER FIELDS INTO THE
PROGRAM */
  gsubmit 'data _null_;file tmp001;';
  gsubmit 'put "/" %eval(&pos-1)*"*";';
  gsubmit 'put "* Program:   &prog"   @&pos
    "*";';
  gsubmit 'put "* Location:  &path"   @&pos
    "*";';
  gsubmit 'put "* Client:    &client" @&pos "*";';
  gsubmit 'put "* Protocol:  &proto"  @&pos "*";';
  gsubmit 'put "* Author:    &author" @&pos "*";';
  gsubmit 'put "* Created:   &date"   @&pos "*";';
  gsubmit 'put "* Summary:   Program Summary Goes
    Here" @&pos "*";';
  gsubmit 'put "* Revisions:"         @&pos "*";';
  gsubmit 'put %eval(&pos-22)*"*" " Innovex DAS,
    Inc. **/";';
  gsubmit 'run;';
  %let rc=%sysfunc(filename(file));

  include "&opendlg";

 %end;
%mend;
```

APPLICATION 4: DOCUMENTING SOURCE ENTRIES

The program header can also be used to create fully documented source entries in SAS® catalogs. We often need to place many SAS® programs saved in ASCII files into one SAS® catalog, saving each program as a source entry. The summary information from the header can be used to automatically assign the description to each source entry. The following macro takes a specified list of programs that exist in the directory “p:\sas resources\code library\dward” and saves each one as a source entry in a SAS® catalog. The macro then creates PROC CATALOG code that updates each source entry with the correct description.

```
%macro sasautos (macros,cat);
  /* PARSE PARAMETER MACROS */
  %let i = 1;
  %do %while (%scan(&macros,&i)^=);
    %let macro&i=%scan(&macros,&i);
    %let i = %eval(&i+1);
  %end;
  %let nummacs=%eval(&i-1);
  %do i = 1 %to &nummacs;
    filename cat catalog "&cat..&&macro&i...source";
    data _null_;
      infile "P:\SAS Resources\Code
        Library\DWard\&&macro&i...sas" truncover;
      input line $char200.;
      if index(line,'* Summary:')>0 then call
        sympUt("sum&i",left(trim
        (scan(line,2,':*'))));
      file cat lrecl=500;
      put _infile_;
    run;
  %end;
  proc catalog cat=&cat;
    %do i = 1 %to &nummacs;
      modify &&macro&i...source (desc="&&sum&i");
    %end;
  quit;
%mend;
```

The macro call %sasautos(nobs nvar, work.sasautos) would copy the programs nobs.sas and nvar.sas into source entries in the catalog work.sasautos and create the following proc catalog code:

```
PROC CATALOG CAT=WORK.SASAUTOS;
MODIFY NOBS.SOURCE (DESC= "Resolve to the number of
observations of a SAS dataset   ");
MODIFY NVAR.SOURCE (DESC= "Resolve to the number of
variables of a SAS dataset   ");
QUIT;
```

CONCLUSION

The method presented here for managing SAS® programs allows programmers to store and retrieve them efficiently. It also makes program documentation more structured, easier to maintain and more accessible. The concept and its applications are simple, very customizable, and require no tricky SAS® code. We have found that storing SAS® programs in such a way greatly reduces time spent looking for programs and understanding what they do.

ACKNOWLEDGMENTS

My special thanks go to David Stokar, senior director of statistical programming at Innovex DAS, for providing important suggestions and ideas about managing our SAS® programs.

CONTACT INFORMATION

David Ward
Innovex DAS, Inc.
501 Hoes Ln., Suite 300
Piscataway, NJ 08854
Phone: (732) 699-9199 Fax: (732) 699-1199
DWard@DAS-International.net