Paper 051-29

Using SAS® Catalogs to Develop and Manage SAS® DATA Step Programs

David D. Chapman,US Census Bureau, Washington, DC

ABSTRACT

A key concept in program management is to develop the code in one place, make changes at one location, move the code to where it is needed for testing or use, and when making changes make sure they work before changing production code. SAS catalogs are ideal for developing, testing, and managing data step programs. This paper reviews SAS catalogs and the different kinds of elements they contain. It discusses Source, Log, Output, Macro, Format, and CATAMS elements. For each it gives an example of how to put it into a catalog and how to retrieve and use it from the catalog. It also discusses how to manage and sequence code coming out of the catalog to make up your data step and procedure programs. The use of PROC CATALOG to view catalog contents and copy elements to another location, and PROC UPLOAD and PROC DOWNLOAD to move the SAS catalogs to a production location on a different computer are discussed. Managment issues on how to use PROC CONTENTS and PROC UPLOAD and PROC DOWNLOAD to move the SAS catalogs to a production location on either the same or a different computer are shown. The use of the Source Control Manager (SCM) to management data step and procedure code from a SAS catalog is also discussed.

SAS CATALOGS

SAS catalogs are special SAS files that store many different kinds of information in smaller units called catalog entries. Each entry has an entry type that identifies its purpose to the SAS System. A single SAS catalog can contain several different types of catalog entries. Some catalog entries contain system information such as definitions. Other catalog entries contain application information such as window definitions, help windows, formats, informats, macros, or graphics output. You can list the contents of a catalog using various SA S features, such as SAS Explorer and PROC CATALOG.

SAS catalog entries are fully identified by a four-level name of the form: "*libref.catalog.entry-name.entry-type*". An example is "SUGI29.PAPER151.CONTROL.SOURCE". You commonly specify the two level name for an entire catalog, as follows: *libref.catalog (e.g. SUGI29.PAPER110)*. "libref" is the logical name of the SAS data library in which the catalog is stored. "Catalog" is a valid SAS name for the file. The entry name and entry type are required by some SAS procedures. "*Entry-name*" is a valid SAS name for the catalog entry; "*entry-type*" is controlled and assigned by SAS when the entry is created. When using a catalog entry, if the entry type has been specified elsewhere or if it can be determined from context, you can use the entry name alone.

CATALOG ELEMENTS

A SAS catalog is a special type of SAS file that contains elements. Elements can be several different types. All elements in a file can the same type of element or all elements can be a different type. Different types of elements are described below. A catalog can contain all the code and information necessary to execute a data step program. There are many different types of catalog elements. Elements most commonly used with a data step program are listed below. The most common elements used in catalogs are: SOURCE, FORMAT, MACRO(not a special catalog entry), OUTPUT, LOG, and SOURCE. The use of these elements is given in later sections.

SOURCE

This probably the most common element in use. It is the equivalent to a SAS program (e.g. program1.sas) file or ASCCII file. These elements are used to hold basic data step or procedure code. There can be a different element for each datastep and proc or a data step can be divided between several different elements. The source element is similar to a text file used to hold SAS code, original macro statements, or other general purpose statment such as options. Source code is added to a catalog either interactively or in batch. When it is created in batch, a data _null_ statement is often used. The except from the LOG file below illustrates how to add data step program code to catalog and then retrieve and use it.

```
44   libname sugi29 "c:\sugi29";
NOTE: Libref SUGI29 was successfully assigned as follows:
      Engine:        V9
      Physical Name: c:\sugi29
44 !                           run;
45
46   FILENAME  SrcCode CATALOG 'sugi29.paper151.source_code.source';
47   DATA  _NULL_;
48   FILE SrcCode ;
49   PUT 'proc datasets ;';
50   PUT 'RUN ; ' ;
51   RUN;
NOTE: The file SRCCODE is:
      Catalog Name=SUGI29.PAPER151.SOURCE_CODE.SOURCE,
      Catalog Page Size=4096,
      Number of Catalog Pages=21,
      Created=13:30 Sunday, January 11, 2004,
      Last Modified=11:27 Saturday, January 31, 2004,
      File Name=c:\sugi29\paper151.sas7bcat,
      Release Created=9.0000M0,Host Created=XP_PRO
NOTE: 2 records were written to the file SRCCODE.
      The minimum record length was 6.
      The maximum record length was 15.
```

```
NOTE: DATA statement used (Total process time):
      real time           0.02 seconds
      cpu time            0.03 seconds
```

The key elements are to have a filename statement that identifies a source entry in a catalog and that filename be used with the “FILE” statement in the data _NULL_ data step.

FORMAT

User define formats are commonly used with data step programs. User define formats can be created, stored in a catalog, and retrieved from the catalog for later use with data steps or procedures. User defined format are created using PROC FORMAT. These formats are used to recode variables, as the basis for classifications that are part of tabulate totals using PROC TABULATE or PROCREPORT, or to display data in output.

PROC FORMAT is used to create the formats and store them in the catalog. The code to create a user define format and store it in a catalog is illustrated in the LOG file below. The code creates a format name “$REGION” and stores it in the catalog “sugi29.paper151".

```
10   proc format library=sugi29.paper151 ;
22   value $REGION
32    '1' = "NorthEast"
42    '2' = "NorthEast"
52    '3' = "Central"
62    '4' = "Central"
72    '5' = "South"
82    '6' = "South"
92    '7'  = "South"
10    '8'  = "West"
11    '9'  = "West" ;
NOTE: Format $REGION has been written to SUGI29.PAPER151.
12   run;
NOTE: PROCEDURE FORMAT used (Total process time):
      real time           0.01 seconds
      cpu time            0.02 seconds
```

This format that assigns a census region code based on census division code is made available from the catalog file “paper151” by setting a SAS system option. Once a format has been stored in a catalog, it can be accessed by specifying the system option FMTSEARCH (). If you physically move the catalog, you must either change the name of the libname or redefine the libname to reflect the catalogs new location.

```
Options fmtsearch (sugi29.paper151);
```

SAS searches for formats in a specific order. It first seaches the catalog WORK.FORMATS, then the LIBRARY.FORMATS catalog, and finally the “SUGI29.PAPER151" catalog.

LOG AND OUTPUT

A normal part of a SAS data step progam is the production of a LOG and OUTPUT file. This information can be stored in the catalog containing the code. This allows the catalog to contain both the OUTPUT and the LOG file describing the program’s operation and the original code that made up the program. Doing this makes the code self documenting. If there is a lot of output, it could also make the catalog quite large. This is particularly useful when the code may vary each time it is used and a new catalog is created for each use. The code below illustrates how to use the PRINTTO procedure to store results from the OUTPUT and LOG windows in the program catalog.

```
1    PROC PRINTTO  NEW
2                  LOG =SUGI29.PAPER151.PROGRAM.LOG
3                  PRINT=SUGI29.PAPER151.PROGRAM.OUTPUT;
4    RUN;
NOTE: PROCEDURE PRINTTO used (Total process time):
      real time           0.00 seconds
      cpu time            0.01 seconds
```

The code above puts the LOG statements associated with executing the PROC CATALOG statement into the element program of type LOG and puts the output of the CONTENTS procedures into the element program of type OUTPUT. The “NEW” option creates a new element each time it is run. If the “NEW” value is not used, data is appended to whatever is in the element.

CATAMS

A CATAMS element is an element for holding data. This data can be retrieved from the catalog to create a SAS data set or view. In theory, the size of the data set created is limited only by the size of the catalog that can be created. In practice, data sets create from CATAMS elements are often used to hold program parameters. One application used a CATAMS element to hold IP addresses of different computer systems.

Paper 051-29

The LOG file below reads the SAS data step “sashelp.company” and writes the data to a CATAMS catalog element.

```
515   filename data catalog "sugi29.paper151.company.catams";
516
517   data _null_;
518   file data;
519   set sashelp.company;
520   PUT LEVEL1 $
521   LEVEL2 $
522   LEVEL3 $
523   LEVEL4 $
524   LEVEL5 $
525   JOB1 $
526   DEPTHEAD $ ;
527   RUN;
NOTE: The file DATA is:
      Catalog Name=SUGI29.PAPER151.COMPANY.CATAMS,
      Catalog Page Size=4096,
      Number of Catalog Pages=9,
      Created=13:30 Sunday, January 11, 2004,
      Last Modified=15:54 Sunday, January 11, 2004,
      File Name=c:\SUGI29\paper151.sas7bcat,
      Release Created=9.0000M0,Host Created=XP_PRO

NOTE: 48 records were written to the file DATA.
      The minimum record length was 57.
      The maximum record length was 85.
NOTE: There were 48 observations read from the data set SASHELP.COMPANY.
NOTE: DATA statement used (Total process time):
      real time           0.06 seconds
      cpu time            0.06 seconds
```

Once data is entered into a CATAMS element, it can be either by used to create a SAS data set or used directly from the catalog as a SAS Data View. The LOG file containing the code that reads and prints the data contained in the CATAMS element create above is given below.

```
530   DATA COMPANY_VIEW/VIEW=COMPANY_VIEW;
531   INFILE DATA;
532   INPUT LEVEL1 $
533   LEVEL2 $
534   LEVEL3 $
535   LEVEL4 $
536   LEVEL5 $
537   JOB1 $
538   DEPTHEAD $;
539   PUT _INFILE_;
540   RUN;
NOTE: DATA STEP view saved on file WORK.COMPANY_VIEW.
NOTE: A stored DATA STEP view cannot run under a different operating system.
NOTE: DATA statement used (Total process time):
      real time           0.04 seconds
      cpu time            0.05 seconds

541  PROC PRINT DATA=COMPANY_VIEW;RUN;
NOTE: The infile DATA is:
      Catalog Name=SUGI29.PAPER151.COMPANY.CATAMS,
      Catalog Page Size=4096,
      Number of Catalog Pages=8,
      Created=13:30 Sunday, January 11, 2004,
      Last Modified=15:54 Sunday, January 11, 2004,
      File Name=c:\SUGI29\paper151.sas7bcat,
      Release Created=9.0000M0,Host Created=XP_PRO

International Ai TOKYO ADMIN CONTRACTS So Suumi MANAGER 1
International Ai TOKYO ADMIN CONTRACTS Steffen Graff ASSISTANT 2
    {lines deleted to save space}
International Ai NEW YORK TECHN. SERVICES MIS Claire Voyant TRANSLATOR 2
International Ai NEW YORK TECHN. SERVICES MIS Roy Hobbs TECH. CONS. 2
NOTE: 48 records were read from the infile DATA.
      The minimum record length was 57.
      The maximum record length was 85.
NOTE: View WORK.COMPANY_VIEW.VIEW used (Total process time):
      real time           0.17 seconds
      cpu time            0.07 seconds
```

```
NOTE: There were 48 observations read from the data set WORK.COMPANY_VIEW.
NOTE: PROCEDURE PRINT used (Total process time):
      real time           0.20 seconds
      cpu time            0.10 seconds
```

CATAMS elements are excellent tools to save small data sets associated with a SAS application.

MACROS

Macros used as autocall macros can be stored and used from a SAS catalog. To create an autocall macro called "work_ds" that executes the PROC DATASETS command. If a macro is stored in an catalog and accessed as an autocall macro, the name of the catalog element must be the same and as the macro. The LOG file below show how to create the catalog element for the autocall macro.

```
11   LIBNAME  SUGI29 'c:\sugi29';
NOTE: Libref SUGI29 was successfully assigned as follows:
      Engine:        V9
      Physical Name: c:\sugi29
11 !                                RUN;
12   FILENAME   mymac  CATALOG 'sugi29.paper151.work_ds.source';
13   DATA  _NULL_;
14   FILE mymac;
15   PUT '%macro work_ds;' ;
16   PUT 'proc datasets ;';
17   PUT '%mend ;' ;
18   PUT 'RUN ; ' ;
19   RUN;
NOTE: The file MYMAC is:
      Catalog Name=SUGI29.paper151.WORK_DS.SOURCE,
      Catalog Page Size=4096,
      Number of Catalog Pages=5,
      Created=23:39 Monday, January 19, 2004,
      Last Modified=23:39 Monday, January 19, 2004,
      File Name=c:\sugi29\test.sas7bcat,
      Release Created=9.0000M0,Host Created=XP_PRO
NOTE: 4 records were written to the file MYMAC.
      The minimum record length was 6.
      The maximum record length was 15.
NOTE: DATA statement used (Total process time):
      real time           0.03 seconds
      cpu time            0.04 seconds
```

To use the macro "work_ds" to display files in the work directory, the LOG below illustrates the code to do it. It is important that the file name for the catalog with the macro is the same as the filename use with the SASAUTOS option.

```
31   option  MAUTOSOURCE ;
32   FILENAME   mymac  CATALOG 'sugi29.paper151';
33   OPTIONS SASAUTOS=mymac ;
34   %WORK_DS
```

Any number of macros can be stored and accessed from a program code. Macros could also be created by storing just the source code and recalling the code to create the macro. A log filed of an example is given below.

```
82   libname sugi29 "c:\sugi29";
NOTE: Libref SUGI29 was successfully assigned as follows:
      Engine:        V9
      Physical Name: c:\sugi29
82 !                           run;
83
84   FILENAME   SrcCode1  CATALOG 'sugi29.paper151.SrcCode1.source';
85   DATA  _NULL_;
86   FILE SrcCode1 ;
87   PUT 'proc datasets ;';
88   PUT 'quit; ' ;
89   RUN;
NOTE: The file SRCCODE1 is:
      Catalog Name=SUGI29.PAPER151.SRCCODE1.SOURCE,
      Catalog Page Size=4096,
      Number of Catalog Pages=21,
      Created=13:30 Sunday, January 11, 2004,
      Last Modified=12:01 Saturday, January 31, 2004,
      File Name=c:\sugi29\paper151.sas7bcat,
      Release Created=9.0000M0,Host Created=XP_PRO
```

```
NOTE: 2 records were written to the file SRCCODE1.
      The minimum record length was 6.
      The maximum record length was 15.
NOTE: DATA statement used (Total process time):
      real time           0.03 seconds
      cpu time            0.00 seconds

92   %MACRO DB;
93   %include  SrcCode1 /source2;
94   %mend;
95   %db
NOTE: %INCLUDE (level 1) file SRCCODE1 is file SUGI29.PAPER151.SRCCODE1.SOURCE.96  +proc datasets ;
                                        Directory
         Libref         WORK
         Engine         V9
         Physical Name  C:\DOCUME~1\DAVIDC~1\LOCALS~1\Temp\SAS Temporary Files\_TD2612
         File Name      C:\DOCUME~1\DAVIDC~1\LOCALS~1\Temp\SAS Temporary Files\_TD2612
                                Member    File
                        #  Name     Type      Size  Last Modified
                        1  SASMACR  CATALOG   5120   31JAN2004:11:56:10
97  +quit;
NOTE: PROCEDURE DATASETS used (Total process time):
      real time           0.03 seconds
      cpu time            0.03 seconds
NOTE: %INCLUDE (level 1) ending.
98   quit;
```

Catalogs should also be able to be used with compiled macros stored in a catalog.

PROGRM RELATED CATALOG ELEMENTS

Many SAS procedures make use of catalog entries to store code or parameters for later reuse. One of these is the PROC REPORT procedure; another is the SQL QUERRY window. The common element is that the procedure saves output to a catalog element for reuse later. For PROC REPORT it is to produce the same table with a different data set or under different circumstance. For the SQL QUERRYY window it is to use a previous query statement to extract and save data from the same or a different data file.

The PROC REPORT procedure allows a use to store the code that describes an output table as a catalog entry. This catalog entry can be used later with multiple data sets without change. The source code for creating such a report catalog entry and later reusing it is given below. The code below creates a catalog entry to produce PROC REPORT tables.

```
491 proc report data=sashelp.company1999 outrept=sugi29.paper151.report02 nowd;
492 column level1 level2 N;
493 define level1 /group;
494 define level2 / group;
495 define n / analysis sum;
496 run;
NOTE: Definition stored in SUGI29.PAPER151.REPORT02.
NOTE: There were 48 observations read from the data set SASHELP.COMPANY.
NOTE: PROCEDURE REPORT used:
real time 0.19 seconds
cpu time 0.01 seconds
```

The code below retrieves and uses the report format already created above to produce a report using a different file.

```
499 proc report
500 data=sashelp.company2000 report=sugi29.paper151.report02 nowd;run;
NOTE: There were 48 observations read from the data set SASHELP.COMPANY.
NOTE: PROCEDURE REPORT used:
real time 0.15 seconds
```

SAS STATEMENTS USED WITH CATALOGS

The key to using SAS catalogs is using the FILENAME statement and the %INCLUDE statement. The filename statement allows us to reference catalog elements either individually or collectively. The %INCLUDE statement allows us to submit the code contained in a catalog element. These two statements are the basis for much of the work done with catalogs.

FILENAME STATEMENT WITH CATALOG OPTION

The filename statement allows us to associate a single name with either a particular catalog element or an entire catalog. The key is the catalog option to the filename statement. A filename can be assoiciated with a single element of a catalog

```
FILENAME   DATA_READ  CATALOG "SUGI29.PAPER151.READ.SOURCE";
```

A filename can also be associated with the entire catalog. This make referencing easier and more logical with the %INCLUDE statement.

```
FILENAME   CATALOG_READ  CATALOG "SUGI29.PAPER151";
```

The filename statement is usually used in conjunction with the %INCLUDE statement.

%INCLUDE

The %INCLUDE statement inserts code located in a source file or catalog into the program immediately following the statement. The two most common forms of the statement are for files and catalogs. The code below illustrates how individual files or individual source entires in a catalog can be referenced with and without the catalog option.

```
186  libname sugi29 "c:\sugi29";
NOTE: Libref SUGI29 was successfully assigned as follows:
      Engine:        V9
      Physical Name: c:\sugi29
186!                            run;
187
188  FILENAME   SrcCode1  CATALOG 'sugi29.paper151.SrcCode1.source';
189  FILENAME   pgm1                          "c:\sugi29\pgm1.sas" ;
190
191  DATA  _NULL_;
192  FILE SrcCode1 ;
193  PUT 'proc datasets library=sasuser ;';
194  PUT 'quit; ' ;
195  RUN;
NOTE: The file SRCCODE1 is:
      Catalog Name=SUGI29.PAPER151.SRCCODE1.SOURCE,
      Catalog Page Size=4096,
      Number of Catalog Pages=22,
      Created=13:30 Sunday, January 11, 2004,
      Last Modified=12:21 Saturday, January 31, 2004,
      File Name=c:\sugi29\paper151.sas7bcat,
      Release Created=9.0000M0,Host Created=XP_PRO
NOTE: 2 records were written to the file SRCCODE1.
      The minimum record length was 6.
      The maximum record length was 31.
NOTE: DATA statement used (Total process time):
      real time           0.01 seconds
      cpu time            0.02 seconds

197  DATA  _NULL_;
198  FILE pgm1 ;
199  PUT 'proc datasets  library=sasuser ;';
200  PUT 'quit; ' ;
201  RUN;
NOTE: The file PGM1 is:
      File Name=c:\sugi29\pgm1.sas,
      RECFM=V,LRECL=256
NOTE: 2 records were written to the file PGM1.
      The minimum record length was 6.
      The maximum record length was 32.
NOTE: DATA statement used (Total process time):
      real time           0.04 seconds
      cpu time            0.02 seconds

204  %include srccode1/source2;run;
NOTE: %INCLUDE (level 1) file SRCCODE1 is file SUGI29.PAPER151.SRCCODE1.SOURCE.
205 +proc datasets library=sasuser ;
                                      Directory

      Libref         SASUSER
      Engine         V9
      Physical Name  C:\Documents and Settings\David Chapman\My Documents\My SAS Files\9.0
      File Name      C:\Documents and Settings\David Chapman\My Documents\My SAS Files\9.0
                                 Member      File
                       #  Name   Type        Size  Last Modified
                       1  AGENTS    DATA        5120   06DEC2002:15:35:13
                       2  CNTAINER  DATA        5120   06DEC2002:15:31:22
                            {OUTPUT DELETED TO SAVE SPACE}
                      23  _SCMPLAY  DATA       17408   28NOV2003:12:04:02
                      24  _SCMPREF  DATA       17408   28NOV2003:12:03:17
206 +quit;
```

```
NOTE: PROCEDURE DATASETS used (Total process time):
      real time           0.06 seconds
      cpu time            0.06 seconds
NOTE: %INCLUDE (level 1) ending.
208  %include pgm1/source2;run;
NOTE: %INCLUDE (level 1) file PGM1 is file c:\sugi29\pgm1.sas.
209 +proc datasets  library=sasuser ;
                                      Directory
      Libref         SASUSER
      Engine         V9
      Physical Name  C:\Documents and Settings\David Chapman\My Documents\My SAS Files\9.0
      File Name      C:\Documents and Settings\David Chapman\My Documents\My SAS Files\9.0
                                Member      File
                       #  Name  Type        Size  Last Modified
                       1  AGENTS    DATA        5120   06DEC2002:15:35:13
                       2  CNTAINER  DATA        5120   06DEC2002:15:31:22
                              {OUTPUT DELETED TO SAVE SPACE}
                      23  _SCMPLAY  DATA       17408   28NOV2003:12:04:02
                      24  _SCMPREF  DATA       17408   28NOV2003:12:03:17
210 +quit;
NOTE: PROCEDURE DATASETS used (Total process time):
      real time           0.05 seconds
      cpu time            0.06 seconds
NOTE: %INCLUDE (level 1) ending.
```

The statement can be used to reference an entire catalog for ease of use and clarity. The filename statement is associated with than entire catalog file instead of the a single element of the catalog. This filename statement is particularly helpful if you a retrieving many different elements from a catalog.

```
FILENAME PGM_FILE CATALOG "SUGI29.PAPGER151"
%INCLUDE PGM_FILE (PROGRAM1) / SCOURCE2;
```

When a filename refers to the entire catalog, the %include can refer to just the source element within the parentheses. The "SCOURCE2" option specifies that the code will be written to the LOG. This is illustrated in the section below on controling program flow.

CONTROLING PROGRAM FLOW FROM THE CATALOG

To use the catalog in our data step programs, we need to retrieve and submit the program code from the catalog in a specific order. In practice, this can be done by a simple data step that uses only LIBNAME, FILENAME, and &INCLUDE statements. The %include submits the code in the program elements in the desired order. This sequencing code - groups of %include statements – could themselves be a separate source code element of the catalog. An example is given below.

```
Libname sugi29 "c:\sugi29";
filename master catalog "sugi29.paper151";
%include master (StartUp);
%include master (program_a);
%include master (program_b);
%include master (program_merge);
```

This code identifies the catalog and then inserts and submits code from four different catalog elements in the following order: StartUp, program_a, program_b, and program_merge. This code also can be easily modified to work with MP/CONNECT by adding the required MP/CONNECT statements in the sequencing program.

```
options sascmd = "sas82/dmr/device=grlink/noterminal/no$syntaxcheck";
options autosignon=yes ;

RSUBMIT StartUp WAIT=YES;
       LIBNAME SUGI29 'C:\SUGI29";
       FILENAME MASTER CATALOG 'SUGI29.PAPER151' ;
       %INPUT MASTER(STARTUP);
ENDRSUBMIT;
SIGNOFF StartUP;

RSUBMIT task_ONE wait=no;
       LIBNAME SUGI29 'C:\SUGI29";
       FILENAME MASTER CATALOG 'SUGI29.PAPER151' ;
       %INPUT MASTER(PROGRAM_A);
ENDRSUBMIT;

RSUBMIT task_TWO wait=no;
       LIBNAME SUGI29 'C:\SUGI29";
       FILENAME MASTER CATALOG 'SUGI29.PAPER151' ;
```

```
        %INPUT MASTER(PROGRAM_B);
ENDRSUBMIT;

WAITFOR _ALL_ TASK_ONE TASK_TWO ;

RSUBMIT Merge wait=no;
        LIBNAME SUGI29 'C:\SUGI29";
        FILENAME MASTER CATALOG 'SUGI29.PAPER151' ;
        %INPUT MASTER(PROGRAM_Merge);
ENDRSUBMIT;
```

Programs can may be started either interactively or in batch. The key to this is either to use the options on the pull down menus to put code in the program editor or to construct a SAS program file to do it in batch. Code can be submitted interactively by going to the file pull down menu and picking "Open Object" Once the catalog and entry are selected, you can pick "Open in program editor" to put all the code into the program editor where it can be submitted. The code above can be put directly into a "program.sas" file an submitted as a batch file. The code could also be put into what I call a control element, and only the control element submitted.

```
LIBNAME   SUGI29  "c:\sugi29";
FILENAME MASTER CATALOG  "sugi29.paper151";
%include master (control);
```

PROGRAM EDITOR

Code can be submitted interactively through the PROGRAM EDITOR. When the SOURCE2 option is used with the %INCLUDE statement, the code submitted appears in the LOG file. The interactive PROGRAM EDITOR often works best for testing. Programs are easily inserted into the program editor using the "OPEN AS OBJECT" command from the FILE menu. This allows catalog code to either be edited directly or inserted into the PROGRM EDITOR.

BATCH

For standard, repetitive programs submission by batch is often the preferred way to go. They can be scheduled to rule at a specific time or in a particular sequence. The batch code could be just a set of %INCLUDE statements or it could be just a single element that contains all the sequencing. I prefer putting all the code into a single source element that I name "control" or "control_mp". An example is given below.

```
LIBNAME   SUGI29  "c:\sugi29";
FILENAME MASTER CATALOG  "sugi29.paper151";
%include master (control);
```

MANAGING THE CATALOG

Once program code is developed in a catalog, it often needs to be moved from a development directory or computer to a the production environment directory on the production computer. There are several ways to do this. When the catalog is moved to a different directory on the same computer, PROC CATALOG or PROC DATASETS can be used. When the catalog is to be moved onto a different computer SAS/CONNECT with PROC UPLOAD or PROC DOWNLOAD can be used.

PROC CATALOG

PROC CATALOG is a procedure that is used to manage and move a catalog. It can also be used to show the contents of a SAS catalog. Moving elements from a test directory to a production directory is illustrated below. The LOG showing the results of a copy is given below.

```
7 libname sugi29 "c:\";
        NOTE: Libref SUGI29 was successfully assigned as follows:
        Engine: V8
        Physical Name: c:\
7       ! run;
8       proc catalog cat= sugi29.paper151 ;
9       copy out= work.code;
10      run;
NOTE: Copying entry ONE.SOURCE from catalog SUGI29.PAPER151 to catalog WORK.CODE.
NOTE: Copying entry READ.SOURCE from catalog SUGI29.PAPER151 to catalog WORK.CODE.
```

PROC CATALOG can be also be used in to list all the elements in a catalog as the code and output below illustrate.

```
proc catalog cat=work.data;
        contents;
        run;
Contents of Catalog WORK.DATA
# Name     Type          Create Date          Modified Date
-----------------------------------------------------------
1 ONE     SOURCE       06DEC2002:18:24:31   06DEC2002:18:24:31
2 READ    SOURCE       06DEC2002:17:26:07   06DEC2002:17:26:07
3 THREE   SOURCE        06DEC2002:18:24:56   06DEC2002:18:24:56
```

PROC CATALOG is used to identify what elements are in a catalog and when they were created. It can also be used to move an entire catalog or individual elements of a catalog between SAS libraries.

PROC UPLOAD/DOWNLOAD AND SAS/CONNECT

SAS/CONNECT can be used to transfer catalog files between two different computers or to use files on two different computers at the same time. One computer is referred to as the local computer(often a PC); the other the remote computer (often a UNIX or OpenVMS Server). PROC UPLOAD is used to move files from the local to the remote computer. PROC DOWNLOAD is used to move files from the remote computer to the local computer. The code below is remotely submitted after a connection has been made with the remote computer.

```
PROC UPLOAD     INCAT  = DEVELOPKMENT_TEST.PGM_CODE
                OUTCAT = PROJECT.PRODUCTION_CODE;
RUN;
```

The key is to use the "INCAT" and "OUTCAT" options. PROC UPLOAD and PROC DOWNLOAD, part of transfer services under SAS/CONNECT, is another way to move catalogs between different computer systems. PROC UPLOAD and PROC DOWNLOAD work similar to PROC CATALOG. This is also a situation when remote library services under SAS/CONNECT may be appropriate. The use of PROC UPLOAD/DOWNLOAD allows SAS programmer to develop and manage their production code in catalogs on a windows computer and move the code to a UNIX or OpenVMS computer for production work.

TESTING THE CATALOG

During program development or program maintenance, you often need to replace or modify of the code. To avoid problems it is wise to try the new code before actually making the change. SAS makes this easier with CATNAME statement. The CATNAME statement logically concatenates two catalogs together under on logical name called a "catref". In the code below the contents of the two catalogs "*test_code*" and "*production code*" are combined into one logical or virtual catalog named "*program_code*" which is associated with the filename "*code*". In this example the new code to be tested is in the catalog "project.test_code. Catalog elements in the test catalog would replace any catalog elements with the same name in the production catalog.

```
CATNAME program_CODE (project.test_code  project.production_code);
filename code catalog   work.program_code ;
&include code (run ) / source2 ;
run;
```

This technique is used to introduce, change, or test new parameters stored in a CATAMS file, new formats or macors, or new source code without making any change to working code. If problems are encountered, you can revert back to the original code by just changing the CATNAME statement. When the test code is accepted, you just need to copy the catalog element from the test catalog to the production catalog.

The rules for concatenation with the CATNAME statement are:

- The concatenation catalog is searched in the order listed on the statement and the first occurrence of the entry found is used.
- When a catalog entry is opened for writing to, it will be created in the first catalog listed.
- When a catalog entry is opend for reading, the element in the first catalog read will be used.
- When an element is deleted or renamed, only the first occurrence of the entry is affected.
- Any time a list of catalog entries is specified, only the first occurrence is given.

The effect of these rules are that when new test code is place in a catalog ahead of the production catalog. The new test code will be used and the production code left unaffected. The effect is the same as if we had change the production catalog and only used the production catalog.

SOURCE CONTROL MANAGER

Source Control Manager (SCM) is an application that gives the SAS data step programmer a way to manage application development. SAS SCM is included with SAS/AF® software. The main features of SCM are:(1) A interface that handles multiple libraries and catalogs, (2) A way to checkin/checkout of SAS files, (3) Revision control, (4) Version labeling, (5) Easy Testing of changes BEFORE committing to production WITHOUT modifying your code, and (6) Easy distribution of versions of your application to other locations and remote hosts. SCM's primary focus is on catalog entry. This information of the Source Control Manager is based on the SAS 8.2 help file. This information is available by typing "HELP SCM" on the command line. SCM is started by typing "SCM" on the command line. Information on the Source Control Manager for version 6.12 can be downloaded from the SAS web site (www.sas.com/apps/demodownloads/scm_exp_612_153.sysdep.jsp).

Graphical User Interface

The SCM Graphical Interface is an tree view of the SAS files. Libraries, datasets, catalogs, and catalog entries are arranged hierarchically. It allows for: Easy navigation of the SAS files, A Small footprint, Pop-menus to access both SCM and other SAS functions and tools, You can apply an operation to all selected files, and easy editing of files.

Checkin/Checkout

Checkin/Checkout in SCM allows several developers to work on the same set of files without overwriting each other's work. When you check out a catalog entry, for example, no one else can check it out until you check it back in.

Testing Before Committing

The SCM is designed to allow you to test your checked-out changes BEFORE you check them back in. We use the new CATNAME function to concatenate your LOCAL (testing) catalogs in front of the TEAM (production) catalogs. You can test without modifying your SCL code.

Revision Control
After each checkin, SCM creates a backup of the file. These back-ups are kept indefinitely if you so desire.. You can easily recall previous revisions of your work, and track how each file has changed over time. You can quickly revert back to a previous revision of a file.

Version Labeling
Along with Revision Control, you can create Version Labels. A Version Label is basically a snapshot of a set of files at a given point in time. With a version label, you can quickly reproduce an image of an application as it existed in the past. You can create a version label at the end of each week, or at every major milestone in the application's development cycle.

Distribution
Once you have a Version Label of an application, you can easily create a copy in other locations on the network, or on other remote machines using SAS/CONNECT®. This makes it easy to do development work in the SCM environment on one platform, and send the finished product to other places.

CONCLUSION

Catalogs are excellent tools to use in the development of applications and in their management when put into production. All code (source statements, logs, output, formats, macros, and parameter files) are stored in a single file - a SAS catalog. Moving the catalog files moves everything needed to execute the program. The Source Control Manager(SCM) is a more complete tool for managing catalogs for data step and procedure coded. SCM works well with teams and in tracking changes.

REFERENCES

SAS Institute Inc. SAS LANGUAGE: CONCEPTS, Cary, NC, SAS Institute Inc. 1999, 554 pages.
SAS Institute Inc. SAS LANGUAGE: REFERENCE, Cary, NC, SAS Institute Inc. 1999, 554 pages.

ACKNOWLEDGMENTS

DISCLAIMER

This paper reports the results of research and analysis undertaken by Census Bureau staff. It has undergone a more limited review by the Census Bureau than its official publications. This report is released to inform interested parties and to encourage discussion.

CONTACT INFORMATION

Your comments and suggestions are encouraged and welcomed. Contact the author at:

David D. Chapman
Economic Planning and Coordination Division
US Census Bureau
Washington, DC 20233-6100
Work Phone: 301-763-6535
Fax: 301-457-4473
Email: david.d.chapman@census.gov