

User Defined Format Documentation

Michel Schexnayder, Louisiana State University

Abstract

SAS user defined formats provide a tool that can save programming and execution time as well as improve the appearance of output. User defined formats enhance the appearance of reports by substituting intelligible descriptions in place of obscure codes. The proc format cntlin= option converts a code table into a user defined format. A SAS permanent library is the appropriate place to store user defined formats. With the growth of this format library comes the need to document its contents. This documentation consists of: How can the format be updated? Where did the descriptions come from? When was the format last updated? SAS automatically provides some of the needed documentation. Combining the data SAS automatically stores with a small additional file produces the documentation needed to maintain a large user defined format library.

Introduction

This paper assumes the reader is familiar with SAS format library concepts. Jack Shoemaker's SUGI 27 paper 56-27, "Proc Format in Action", describes the proc format cntlin= option and the variable names fntname, type, start, and label.

Format Name Dataset

The simple concept of this paper is a format name dataset. The format name dataset contains one record for each user-defined format stored in the format library. For each format we associate a format description, the associated system (i.e. student records, payroll, accounting) and the program that loads the format into the format library. The combination of the format name dataset and the data that SAS maintains for each format yields a useful report that documents the formats stored in a format library. The appendix contains the source code for the report program.

Source of User Defined Formats

At LSU, we are fortunate that an extensive code table (DB2 database) is part of each production system (student records, payroll, chart of accounts, etc). Part of the online menu screen for the student records code table is shown below.

STUDENT RECORDS CODE TABLE LIST

05/28/02 17:52

Function => _____

Code Table => _____

Code => _____

01 ORGANIZATION	18 MINOR
02 COLLEGE	19 SPECIAL ENROLLMENT
03 COMMENT	20 CALENDAR
04 CONTROL NUMBER	21 EXTRAMURAL
05 COURSE	22 SPEEDE TRADING PARTNERS
06 DISCIPLINARY FINDING	23 COLLEGE ACADEMIC ACTION
07 DEGREE	24 LAW INSTRUCTOR
08 DEPARTMENT	25 HOLMES GROUP PROGRAM
09 SEMESTER	26 FIELD OF STUDY
10 GRADUATE SCHOOL EXAM	27 CONCENTRATION

The code table database contains three variables: code table number, code value, and code description. For example, one observation in 02 College code table contains the code value of 'EDUC' and the code description of 'Education'. Prior to the Proc format cntlin= step the code value will become the 'start' value and the code description will become the 'label' value.

Based on the code table menu, we built the following portion of the format name dataset. Since one format load program has several associated formats, the format name dataset contains two record types. Record type one observations contain the name of the system (SRR, Student Records and Registration) and the location of the format load program (perm.prod.prgms(srrfmts)). Multiple record type two observations usually follow each record type one observation. Record type two observations contain the format name and a format description.

1	SRR	perm.prod.prgms(srrfmts)
2	\$ORGANF	01 ORGANIZATION
2	\$COLF	02 COLLEGE (char4 to name)
2	\$	03 COMMENT
2	\$	04 CONTROL NUMBER
2	\$COURSEF	05 COURSE
2	\$	06 DISCIPLINARY FINDING
2	\$DEGREEF	07 DEGREE
2	\$DEPTF	08 DEPARTMENT NAME
2	\$SEMCDF	09 SEMESTER CODES
2	\$GSEXAMF	10 GRADUATE SCHOOL EXAM

This dataset associates a user-defined format name with a code table number. The code tables with no associated format name indicate a skipped code table. The following code loads the formats to the permanent format library. The variable tblcode is the common value (two digit number) between the student records code table (codes) and the format name dataset (codetbl).

```

data codetbl;      merge codetbl(in=s) codes(in=c); by tblcode;
if s and c;                * build format dataset;
if ffmtname ne '';
keep tblcode ffmtname type start label;
type= 'C';
start=cdevalue; label=codedesc;

                                * permanent user formats;
libname library 'perm.sas.library' disp=old; run;
proc format cntlin=codetbl library=library;

```

Before adding another series of formats to the format library, we must insure that we do not unintentionally overwrite any existing formats. This can be accomplished by comparing the proposed format names with the format names currently contained in the format name dataset.

A second Example

The example below shows a portion of the format name dataset related to the Chart of Accounts (COA) system.

1	COA	PERM.PROD.PRGM\$(COAFMTS)
2	\$ACCTDST	Account distribution code
2	\$ACCTNAM	Account name
2	\$AGENCYF	Agency code (federal accounts)
2	\$AAF	AA AUXILIARY ACCOUNT
2	\$ACF	AC AGENCY
2	\$ATF	AT ACCOUNT TYPE
2	\$BAF	BA BILLING ADDRESS
2	\$BCF	BC BILLING CYCLE
2	\$BFF	BF BILLING FORM
2	\$BTF	BT BILLING TYPE

In this case the format load program loads formats from both the code table (i.e. AA, AC, etc) and the Chart of Accounts master file (acctnam agencyf).

Format Library Documentation

The code found in the appendix produces the following report using the format name dataset and data stored in the format library. Observation 64 describes the account name format 'acctnam' which converts a 9-character account number into a 25-character account description. The format was last updated on 9/3/01 and there are 9,027 account numbers and descriptions.

----- SOURCE=COA SRCDESC=PERM.PROD.PRGM(S (COAFMTS) -----

OBS	NAME	TYPE	LENGTH	DATE	COUNT	DESC
63	ACCTDST	C	9,9,3	09/03/01	19027	Account distribution code
64	ACCTNAM	C	9,9,25	09/03/01	19027	Account name
65	AGENCYF	C	9,9,2	09/03/01	5933	Agency code (federal accounts)
66	AAF	C	3,3,25	09/03/01	32	AA AUXILIARY ACCOUNT
67	ACF	C	2,2,25	09/03/01	51	AC AGENCY
68	ATF	C	1,1,17	09/03/01	7	AT ACCOUNT TYPE
69	BAF	C	3,3,25	09/03/01	1189	BA BILLING ADDRESS
70	BCF	C	1,1,25	09/03/01	20	BC BILLING CYCLE
71	BFF	C	2,2,25	09/03/01	19	BF BILLING FORM
72	BTF	C	2,2,17	09/03/01	2	BT BILLING TYPE

SAS maintains the data contained in the type and date columns of this report. In SAS version 6.08 the length information was stored in the description field. When SAS Technical Support was asked about the lack of the length information in SAS version 8, SAS Technical Support said it was a bug in version 6.08. In any case, a slight adjustment in the proc summary calculates the same information.

Conclusion

The combination of a format name dataset that contains one record for each format stored in a permanent format library and the data that SAS stores in the format library provide the information necessary to document the contents of a format library. The documentation contains the name of the format, the location of the program which loads the format, the minimum and maximum length of the start value, the length of the label value, the date the format was last loaded, the number of start values, and a format description.

References

Proc Format in Action, Jack Shoemaker, SUGI 27 paper 56-27.

SAS Professional Programming Secrets, Aster and Seidman, Windcrest/ McGraw Hill
Highly recommended.

Contact Information

Michel Schexnayder
Applications Consultant
Budget and Planning
Louisiana State University
Baton Rouge, Louisiana 70803

Voice 225 578-5949
Fax 225 578-5963
Email irmike@lsu.edu

Appendix A

The following code generates a SAS user defined format library report.

```
TITLE1 'SAS Format Library';
data DESC; INFILE DESC;
TITLE2 'Source: PERM.PROD.SOURCE(LIBRARY)';
INPUT @1 RCD $1.
      @9 F1 $8.
      @21 F2 $30.;
RETAIN SOURCE SRCDESC;
length name $8;
if RCD GT '0';
if RCD = '1' THEN DO; SOURCE=F1; SRCDESC=F2; END;
if RCD = '2' THEN DO; NAME =F1; DESC =F2; END;
if RCD = '2' ;
* Keep system, program location, format name, format description;
keep SOURCE SRCDESC NAME DESC;
proc sort; by NAME;

libname LIBRARY 'PERM.SAS.LIBRARY' DISP=SHR; run;
proc CATALOG cat=LIBRARY.FORMATS;
  contents out=cat;          * one record per format;
proc sort data=cat; by NAME;
data cat; set cat;          * keep the description field, if the format
                             * has not been rebuilt by SAS 8;
if desc gt `` then
  length = SUBSTR(DESC,15,8); DROP DESC;

proc format library=library cntlout=format;
data format; set format;
lths = length(start);
lthl = length(label);

PROC summary nway data=format(rename=(fmtname=name LENGTH=LTH));
  class NAME min max lth;
  var lths lthl;
  output out=freq min(lths) =minlths
        max(lths) =maxlths
        max(lthl) =maxlthl;

DATA cat; MERGE cat(IN=F) freq(IN=D); BY NAME;
RENAME _freq_=count;
LENGTH MINC MAXC LTHC $4;
MINC = MINlths;
MAXC = MAXlths;
LTHC = maxlthl;

Length = TRIM(LEFT(MINC)) || ' ' ||
          TRIM(LEFT(MAXC)) || ' ' || TRIM(LEFT(LTHC));

data cat; merge cat(IN=F) freq(IN=D); by NAME;
rename _freq_=count;

data cat; merge cat(IN=F) DESC(IN=D); by NAME;
proc sort; by SOURCE DESC;
proc print uniform ; by SOURCE SRCDESC;
  var NAME TYPE LENGTH DATE COUNT DESC;
proc printto print=disk;
proc print uniform ; by SOURCE SRCDESC;
  var NAME TYPE LENGTH DATE COUNT DESC;
```

The proc printto directs the output of the second proc print to disk. This enables a programmer to access the report in a window on his terminal.