SESUG Paper IACS-75-2025

Behind the Scenes with SAS® Catalogs: Formats, Macros and More

Tejalkumari Tailor, The Emmes Company

ABSTRACT

SAS® catalogs play a crucial role in organizing formats, macros and other components that support efficient and consistent programming - particularly in clinical trials where standardization is essential. However, many programmers aren't fully aware of how catalogs work or how to manage them effectively.

This session will introduce the fundamentals of SAS catalog structure, demonstrate how to create and retrieve catalog entries, and offer practical tips for moving catalogs between projects. You'll learn how catalogs can reduce redundancy, simplify code management and improve collaboration across programming teams.

Whether you're just beginning to explore SAS catalogs or looking to optimize your existing processes, this presentation will provide you with the tools and insights needed to make the most of this powerful.

INTRODUCTION

In clinical trial SAS programming, precision, consistency and regulatory compliance are no-negotiable. While datasets and analysis programs often take the spotlight, an equally important but less visible component underpins much of our work - SAS Catalogs. These specialized files act as containers for a wide variety of SAS objects, including formats, macros, templates and compiled code. While they operate quietly in the background, catalogs can have significant impact on the efficiency, portability and reproducibility of results.

From mapping CRF data into SDTM domains to generating ADaM datasets and creating TFLs, catalogs ensure that formats for variables like race, adverse event severity or visit windows are consistently applied. They also store compiled macros that streamline repetitive tasks and preserve efficiency across multiple studies in a clinical development program. Yet catalogs only gain attention when error occurs or submission packages require precise catalog handling for regulatory reviewers.

This paper takes you behind the scenes to explore the structure and functionality of SAS catalogs in the context of clinical trials. We will cover the structure and type of SAS catalogs, how catalogs store and manage formats, macros and compiled code, discuss strategies for reusing and maintaining them across multiple studies.

By the end, readers will be equipped with practical knowledge to make SAS catalogs an intentional part of their clinical programming workflow, ensuring efficiency, consistency and compliance in trial deliverables.

SAS CATALOGS

SAS catalogs are special SAS files that store many 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 types of catalog entries.

Unlike datasets, which store raw or derived data, catalogs hold compiled code, definitions and templates that are used to process, display or format data. In clinical trials, catalogs often work quietly in the background but directly influence the accuracy, consistency and regulatory compliance of deliverables.

SAS catalog entries are fully identified by a four-level name of the form:

libref.catalog.entry-name.entry-type

- Libref: the logical name of the SAS data library in which the catalog is stored.
- Catalog: a valid SAS name for the file.
- Entry-name: a valid SAS name for the catalog entry.
- Entry-type: is controlled and assigned by SAS when entry is created.

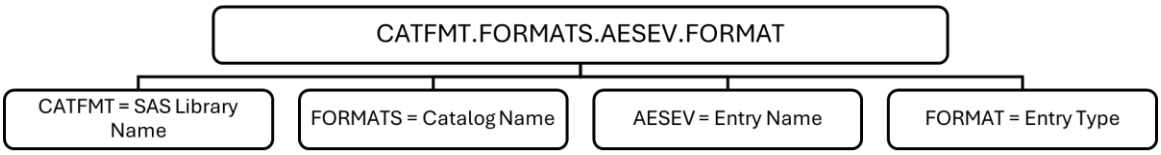

Figure 1. shows component of SAS Catalog name

SAS Catalogs file has a **.SAS7bcat** extension as. The structure of a catalog looks very much like a Windows directory with files of different types coexisting together. Catalogs store information in form of small, distinct units called catalog entries.

CATALOG TYPES

SAS catalogs can contain many different types of entries, each serving a specific purpose in programming workflows. In clinical trials, these catalog entry types are often used to support standardized data handling, reporting and regulatory submissions. The most common types used in catalogs are SOURCE, FORMATS, MACRO and TEMPLATE.

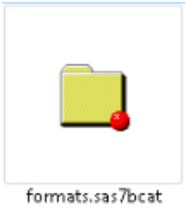

Figure 2. SAS Catalog Icon.

FORMATS

The FORMATS catalog is one of the most widely used catalogs in clinical trials. It stores all used defined formats and informats created using PROC FORMAT. These formats control how data values are displayed in reports and outputs, ensuring consistency across multiple reports.

To save a format or formats in a permanent format catalog, use **PROC FORMAT** to create a format by either defining **value statement** or by using PROC FORMAT's **CNTLIN statement**.

Default catalog name is *formats.sas7bcat.*

Program 1 illustrates how to create a user defined format and store it in catalog. The code creates formats named AESEV and SEX and stores it in catalog CATFMT.FORMATS.

```
%let catfmtdir=%str(Path to store format);
libname catfmt "&catfmtdir.";

proc format library=catfmt.formats;
      value AESEV
            0="None"
            1="Mild"
            2="Moderate"
            3="Severe";
      value $SEX
            "M"="Male"
            "F"="Female";
run;

options fmtsearch=(catfmt.formats work.formats);


40     proc format library=catfmt.formats;
41         value AESEV
42           0='None'
43           1='Mild'
44           2='Moderate'
45           3='Severe' ;
NOTE: Format AESEV has been written to CATFMT.FORMATS.
46         value $SEX
47           "M"="Male"
48           "F"="Female";
NOTE: Format $SEX has been written to CATFMT.FORMATS.
49     run;

NOTE: PROCEDURE FORMAT used (Total process time):
    real time          0.02 seconds
    cpu time           0.00 seconds

50     options fmtsearch=(work.formats catfmt.formats);
```

This format that assigs label to an adverse event (AE) severity value is made available from the catalog file “formats” 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 catalog’s new location.

If a catalog appears in the FMTSEARCH= list, the catalog is searched in the order in which it appears in the list. In example above SAS will first search the catalog CATFMT.FORMATS and then WORK.FORMATS.

MACROS

The MACRO catalog stores compiled versions of SAS macros created in the session. Unlike the macro source code written in programs, the complied version in the MACRO catalog allows faster execution and avoids recompiling each time the macro is called. Default macro catalog is named **SASMARC**.

When a macro is defined using %MACRO and %MEND, SAS compiles it and stores it in the WORK.MACRO catalog by default. This catalog is temporary and deleted at the end of the session. To make macros permanent, programmers can store them in autocall libraries or precompiled catalog

libraries. When compiled macro is called in SAS program, the macro processor skips the compiling step, retrieves the compiled macro code and executes the already compiled code. This time the compiled macro code will not be deleted when the SAS session ends, and it can be reused at later time.

Steps to create compiled Macro

1. Set SYSTEM Options **MSTORED** and **SASMSTORE=**
   - **MSTORED:** enables storage of compiled macros in a permanent SAS library.
   - **SASMSTORE=libref:** Identifies the libref of a SAS library with a catalog that contains, or will contain, stored compiled SAS macros.

2. Use Macro Options to store compiled code, store source code, assign descriptive title for macro entry and secure source code for protect access.
   - **STORE** option: stores the compiled macro in a SAS Catalog in a permanent SAS library.
   - **SOURCE** option: stores the macro source code along with the compiled code. The SOURCE option requires that the STORE option and MSTORED option be set.
   - **DES=””** option: adds descriptions for the macro entry. It must be in quotation marks.
   - **SECURE** option: enables you to write secure macros that prevent access to the macro source code, which helps to protect intellectual property contained in the code.

Program 1. illustrate how to define a macro, store it in catalog and use compiled macro. The code creates macro named LAYOUT and stores it in catalog CATMAC.SASMACR.

Create and Store:

```
%let catmacdir=%str(Path to store macros);
libname catmac "&catmacdir.";

options mstored sasmstore=catmac;
/*Macro to define page orientation and left/right/top/bottom margine
based on orientation*/
%macro layout(orientation= )/ store source
                  des='Setup Orientation of page and margin' ;
      /**************************
      ********* SAS Code *********
      **************************/
%mend layout;

/*Macro to sort dataset*/
%macro sort(DS=, byvar= )/store source
                   Des='Sort any dataset by key variables' ;
      proc sort data=&DS. out=&DS._s ;
            by &byvar.;
      run;
%mend;
```

Use:

```
%let catmacdir=%str(Path to store macros);
libname catmac "&catmacdir.";

options mstored sasmstore=catmac;
%sort(ds=DM, byvar=%str(USUBJID SEX));
```

```
26  options mstored sasmstore=catmac;
27  %macro sort(DS=, byvar= );
28     proc sort data=&DS. out=&DS._s ;
29        by &byvar.;
30     run;
31  %mend;

32  %sort(ds=DM, byvar=%str(USUBJID SEX));

NOTE: There were 19 observations read from the data set WORK.DM.
NOTE: The data set WORK.DM_S has 19 observations and 2 variables.
NOTE: PROCEDURE SORT used (Total process time):
    real time          0.01 seconds
    cpu time           0.01 seconds
```

How to retrieve the source code from compiled macro

The **%COPY** macro statement in SAS retrieves the source code of a stored compiled macro from a SAS macro library and can write the source code to a file, the SAS log or another destination. It is essential that macro has been compiled with **SOURCE** option. Macros are compiled with the SECURE option cannot recover source code by %COPY statement.

Program 3 illustrates how to retrieve a source code of compiled macro and write into log or external .sas file.

Retrieve macro source code:

```
%let catmacdir=%str(Path to store macros);
libname catmac "&catmacdir.";
```

Syntax

```
%copy macro name/<options> ;
%copy macro name/<options> out="External file location";
```

Print code in log window

```
%copy sort/source ;
```

```
86  %copy sort/source ;
%macro sort(DS=, byvar= )/store source;
  proc sort data=&DS. out=&DS._s ;
    by &byvar.;
  run;
%mend;
```

Print code in external file

```
%copy sort/source out="Enter external file path\ retrieved_sort.sas";
```

SOURCE CODE

The SOURCE catalog stores SAS program code as reusable entries. Entries are typically created using %INCLUDE statements with catalog references or through utilities like **PROC CATALOG**. Source elements are the same as text file used to hold SAS code, original macro statements or other general-purpose statements such as options.

Program 4 illustrates how to create a source code catalog and how to use it.

Create SOURCE catalog:

```
%let catsrcdir=%str(Path to store SOURCE);
libname catsourc "&catsrcdir.";

filename procpri catalog 'catsourc.sascode.procpri.source';
/*PROC PRINT*/
data _null_;
      file procpri;
      put 'proc print;
           run;';
run;
```

```
88   %let catsrcdir=%str(Path to store macros);
89   libname catsourc "&catsrcdir.";
NOTE: Libref CATSOURC refers to the same physical library as CATMAC.
NOTE: Libref CATSOURC was successfully assigned as follows:
    Engine:      V9
    Physical Name: File path
90
91   filename procpri catalog 'catsourc.sascode.procpri.source';
92   /*PROC PRINT*/
93   data _null_;
94      file procpri;
95      put 'proc print;
96          run;';
97   run;

NOTE: The file PROCPRI is:
    Catalog Name=CATSOURC.SASCODE.PROCPRI.SOURCE,
    Catalog Page Size=4096,
    Number of Catalog Pages=4,
    Created=Fri, Aug 22, 2025 12:12:01 AM,
    Last Modified=Fri, Aug 22, 2025 12:12:01 AM,
    Filename=File location\sascode.sas7bcat,
    Release Created=9.0401M7,
    Host Created=W32_10PRO,
    Owner Name=XYZ,
    File Size=          9KB,
    File Size (bytes)=9216

NOTE: 1 record was written to the file PROCPRI.
    The minimum record length was 24.
    The maximum record length was 24.
NOTE: DATA statement used (Total process time):
    real time          0.54 seconds
    cpu time           0.06 seconds
```

How to use SOURCE catalog:

```
filename fileref catalog 'libname.catalogname';
%include fileref(entryname);

filename sascode catalog 'catsourc.sascode';
%include sascode(procpri);
```

```
105  filename sascode catalog 'catsourc.sascode';
106  %include sascode(procpri);

NOTE: There were 19 observations read from the data set WORK.DM_S.
NOTE: PROCEDURE PRINT used (Total process time):
    real time          0.02 seconds
    cpu time           0.00 seconds
```

LOG & OUTPUT

The normal part of a SAS program is the production of LOG and OUTPUT files. This information can be stored in the catalog containing the code to document the program's operation and the original code that made up the program. Doing this makes the code almost self-documenting.

A LOG catalog entry stores the SAS execution log inside a catalog entry rather than just in the .log file.

Program 5 illustrates how to generate LOG and OUTPUT catalog files.

```
proc printto new
      log = catsourc.sascode.source_code.log
      print = catsourc.sascode.source_code.output;
run;
/*****************************
* SAS Code to generate log    *
* SAS Code to generate Output *
*****************************/
proc printto ;
run;
```

CATALOGS PROCEDURE

The CATALOG procedure manages entries in SAS catalogs. PROC CATALOG is the primary SAS procedure for viewing, managing and maintaining catalog entries. While PROC FORMAT and PROC TEMPLATE create entries, it is PROC CATALOG that allows programmers to inspect, copy, rename, delete and reorganize catalog contents.

Syntax:

```
proc catalog catalog=<libref.catalog-name> ;
      statement1;
      statement2;
quit;
```

1. **Contents:** List all entries in a catalog. Useful for verifying formats or macros are present.

```
proc catalog catalog=catfmt.formats;
       contents;
quit;

20   proc catalog catalog=catfmt.formats;
21        contents ;
22   quit;

NOTE: PROCEDURE CATALOG used (Total process time):
      real time          0.05 seconds
      cpu time           0.00 seconds
```

Contents of Catalog CATFMT.FORMATS

#	Name	Type	Create Date	Modified Date	Description
1	AESEV	FORMAT	08/22/2025 17:41:19	08/22/2025 17:41:19	
2	SEX	FORMATC	08/22/2025 17:41:19	08/22/2025 17:41:19	

```
proc catalog catalog=catmac.sasmacr;
       contents;
quit;

111  proc catalog catalog=catmac.sasmacr;
112       contents ;
113  quit;

NOTE: PROCEDURE CATALOG used (Total process time):
      real time          0.01 seconds
      cpu time           0.00 seconds
```

Contents of Catalog CATMAC.SASMACR

#	Name	Type	Create Date	Modified Date	Description
1	SORT	MACRO	08/22/2025 21:18:06	08/22/2025 21:18:06	Sort any dataset by key variables

2. **Copy:** Copies some or all the entries in one catalog to another catalog. Essential for promoting objects from a development library to production library. To copy selected entries, use SELECT statement with COPY and to copy all except the entries specified use EXCLUDE statement with COPY.

```
proc catalog catalog=catfmt.formats;
      copy out= catfmt2.formats;
quit;
```

```
39  proc catalog catalog=catfmt.formats;
40     copy out=catfmt2.formats;
41  quit;

NOTE: Copying entry AESEV.FORMAT from catalog CATFMT.FORMATS to catalog CATFMT2.FORMATS.
NOTE: Copying entry SEX.FORMATC from catalog CATFMT.FORMATS to catalog CATFMT2.FORMATS.
```

```
proc catalog catalog=catfmt.formats;
      copy out=catfmt2.formats;
      select AESEV.format ;
quit;
```

```
42  proc catalog catalog=catfmt.formats;
43     copy out=catfmt2.formats;
44     select AESEV.format ;
45  quit;

NOTE: Copying entry AESEV.FORMAT from catalog CATFMT.FORMATS to catalog CATFMT2.FORMATS.
```

```
proc catalog catalog=catfmt.formats;
      copy out=catfmt2.formats;
      exclude AESEV.format ;
quit;
```

```
46  proc catalog catalog=catfmt.formats;
47     copy out=catfmt2.formats;
48     exclude AESEV.format ;
49  quit;

NOTE: Copying entry SEX.FORMATC from catalog CATFMT.FORMATS to catalog CATFMT2.FORMATS.
```

3. **Delete:** Delete entries from SAS catalog. Best practice is to use cautiously in validated environment. To delete *all* entries use PROC CATALOG with KILL option. To delete specified entries, use DELETE statement and to delete all except the entries specified use SAVE statement.

```
proc catalog catalog=catfmt2.formats KILL ;
quit;

57   proc catalog catalog=catfmt2.formats kill ;
58   quit;

NOTE: Deleting entry AESEV.FORMAT in catalog CATFMT2.FORMATS.
NOTE: Deleting entry SEX.FORMATC in catalog CATFMT2.FORMATS.
NOTE: PROCEDURE CATALOG used (Total process time):
    real time          0.00 seconds
    cpu time           0.01 seconds



proc catalog catalog=catfmt2.formats;
       save AESEV.format ;
quit;

62   proc catalog catalog=catfmt2.formats;
63      save AESEV.format ;
64   quit;

NOTE: Saving entry AESEV.FORMAT in catalog CATFMT2.FORMATS.
NOTE: Deleting entry SEX.FORMATC in catalog CATFMT2.FORMATS.

NOTE: PROCEDURE CATALOG used (Total process time):
    real time          0.03 seconds
    cpu time           0.00 seconds



proc catalog catalog=catfmt2.formats;
       delete sex.formatc ;
quit;

77   proc catalog catalog=catfmt2.formats;
78      delete sex.formatc ;
79   quit;

NOTE: Deleting entry SEX.FORMATC in catalog CATFMT2.FORMATS.
NOTE: PROCEDURE CATALOG used (Total process time):
    real time          0.00 seconds
    cpu time           0.00 seconds
```

4. **Alter Catalog entries:** There are multiple statements which can help to alter catalog entry-name. CHANGE can change the name of catalog entries; EXCHANGE can switch the names of two catalog entries and MODIFY can change the description of a catalog entry.

```
proc catalog catalog=catfmt.formats;
        change SEX.formatc=GENDER.formatc ;
quit;

114  proc catalog catalog=catfmt.formats;
115       change SEX.formatC=GENDER.formatc ;
116  quit;

NOTE: Changing entry SEX.FORMATC to GENDER.FORMATC in catalog CATFMT.FORMATS.
NOTE: PROCEDURE CATALOG used (Total process time):
    real time          0.00 seconds
    cpu time           0.00 seconds



proc catalog catalog=catfmt.formats;
        EXCHANGE SEX.formatC=GENDER.formatc  ;
quit;

191  proc catalog catalog=catfmt.formats;
192       EXCHANGE SEX.formatC=GENDER.formatc ;
193  quit;

NOTE: Exchanging entries SEX.FORMATC and GENDER.FORMATC in catalog CATFMT.FORMATS.
NOTE: PROCEDURE CATALOG used (Total process time):
    real time          0.00 seconds
    cpu time           0.00 seconds



proc catalog catalog=catfmt.formats;
        MODIFY AESEV
       (DESCRIPTION="Assign toxicity Grade based on numeric grades")
       /entrytype=format  ;
quit;

200  proc catalog catalog=catfmt.formats;
201       MODIFY AESEV (DESCRIPTION="Assign toxicity Grade based on numeric grades")
          /entrytype=format ;
202  quit;

NOTE: Description changed for entry AESEV.FORMAT in catalog CATFMT.FORMATS.
NOTE: PROCEDURE CATALOG used (Total process time):
    real time          0.00 seconds
    cpu time           0.00 seconds
```

Contents of Catalog CATFMT.FORMATS

#	Name	Type	Create Date	Modified Date	Description
1	AESEV	FORMAT	08/22/2025 21:51:26	08/22/2025 21:55:39	Assign toxicity Grade based on numeric grades

CONCLUSION

SAS Catalogs are often treated as background element in programming, but a closer look reveals that they are central to how SAS organizes, stores and reuses key programming resources. From FORMATS that ensure consistent representation of data to MACROS that automate standard derivations and reporting to SOURCE, LOG and OUTPUT entries that provide traceability and transparency, catalogs are the hidden backbone of a well-governed SAS environment.

Within the realm of clinical trials, catalogs deliver value in two major ways. First, they promote efficiency by supporting reusable, standardized programming tools, reducing duplication across studies. Second, they strengthen compliance by ensuring consistent outputs, preserving metadata and code and providing audit ready records. These capabilities are especially important in a regulatory setting where accuracy, reproducibility and transparency are essential.

Ultimately, working with SAS catalogs is about more than technical know-how- it is about recognizing and leveraging the infrastructure that underpins dependable clinical trial programming. By bringing these "behind the scenes" elements into focus, programmers and statisticians can create workflows that are not only more efficient but also more reliable, reproducible, and regulatory-ready.

REFERENCES

Randall Cates, 2002, "What's In A Name; Describing SAS® File Types", *SUGI 27,* Paper 69-27.

David D. Chapman, 2003, "Using SAS Catalogs to Develop and Manage SAS DATA Step Program." *SUGI 28,* Paper 110-28.

David D. Chapman, 2004, "Using SAS® Catalogs to Develop and Manage SAS® DATA Step Programs." *SUGI 29,* Paper 051-29.

Louise Hadden, 2015, "PROC CATALOG, the Wish Book SAS® Procedure", Paper 3459-2015.

SAS® 9.4 Langauage References: Concepts – Catalogs.

SAS® 9.4 Procedures Guide – The CATALOG procedure.

SAS® 9.4 Macro Language: Reference.

SAS® 9.4 Procedures Guide – The FORMAT procedure.

RECOMMENDED READING

https://documentation.sas.com/doc/en/pgmsascdc/v_066/proc/n11st7possem5en1vo07v2nlxg0c.htm

https://documentation.sas.com/doc/en/lrcon/9.4/n00qg1hma0vur6n1sntt9ju9e78k.htm

https://support.sas.com/documentation/cdl/en/proc/61895/HTML/default/viewer.htm#a000146231.htm

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Tejalkumari Tailor
The Emmes Group
401 N Washington St., # 700
E-mail: ttailor@emmes.com