PharmaSUG 2014 – Paper CC08

Additional Metadata for Common Catalog Entry Types

Kenneth W. Borowiak, PPD

ABSTRACT

Anyone who has worked with SAS® has probably added descriptive attributes to entities such as variables (labels and formats), data sets (labels), reports (titles and footnotes), and programs (comments). Though not as well known, one can add descriptive labels to commonly used catalog entry types, namely formats and macros This paper will demonstrate how to add, modify and retrieve this metadata.

Keywords: format, macros, PROC CATALOG, metadata

INTRODUCTION

Anyone who has worked with SAS has probably added descriptive attributes to entities such as variables (labels and formats), data sets (labels), reports (titles and footnotes), and programs (comments). The goal of augmenting these entities with more information[1] includes reducing ambiguity, increasing usability, and creating an audit trail. Information about information is referred to as *metadata*. Creating user defined metadata comes at the expense of time, which itself may be scarce, and requires diligence of the programmer.

This paper explores adding metadata to and retrieving it from three commonly used catalog entry types, namely formats, macros and user-defined functions. While in many cases one can look at the source code for these entities, it is possible that you may need to use a compiled version unknown heritage. Or you might be a developer of a macro who wishes to provide some information on the entry to the end user while purposely shielding them from the source code.

MACROS

A macro is compiled code which when invoked generates code to be executed. Consider the very simple macro below in Figure 1 that writes to the log the date of the previous day.

```
option sasmstore=maclib mstored ;

%macro yesterday( datefmt=date9.) / store
                   secure
                   des="Display yesterday's date in log"
                   ;
 %let today=%sysfunc( today() ) ;
 %let yesterday=%sysevalf( &today.-1 ) ;
 %put WARNING-Yesterday was %sysfunc( putn(&yesterday.,&datefmt.) ) ;
%mend ;
```

Figure 1 - Adding a Descriptive Label to a Macro with the DES Option

[1] The assumption is that the information provided is accurate.

The macro %yesterday is compiled in a macro catalog in the library associated with the libref MACLIB. Notice the use of the DES option on the macro statement to add a descriptive label to %yesterday. Any user pointing to the library associated with the libref MACLIB with the appropriate system options set can call %yesterday. However, they can't see the code that generated the output because of the SECURE option on the macro statement, unless they have access to the program that created %yesterday. By adding a description to the macro, anyone with access to the macro can retrieve any user-defined information about it. Such information may include the author, purpose, version, caveats, where to find detailed documentation about it or location of the source code. The DES option allows up to 40 characters for the description.

In addition to adding a label to a macro at the time it is compiled with the DES option, one can also augment the catalog entry with a description by using PROC CATALOG, as shown below in Figure 2.

```
proc catalog catalog=xxx.sasmacr et=macro;
  modify yesterday(desc='See macro gatekeeper for the source code') ;
  run;
  quit ;

SAS log:
NOTE: Description changed for entry YESTERDAY.MACRO in catalog NESUG.SASMACR.
```

Figure 2 - Adding a Descriptive Label to a Macro with PROC CATALOG

The syntax for PROC CATALOG is similar to that of PROC DATASETS, but operates on catalogs rather than data sets. On the PROC CATALOG statement, you specify the catalog you want to operate on and the entry type (i.e. et=), which is *macro* in this case. To add or modify a description for a macro entry in the catalog, use the MODIFY statement and the DESCRIPTION option after the entry. An alias for the DESCRIPTION option is DESC and the label may contain up to 256 characters.

There are a few ways to retrieve the description associated with a macro. One is to use PROC CATALOG, which is analogous to using PROC CONTENTS to retrieve information on data sets.

```
proc catalog catalog=xxx.sasmacr ;
  contents out=mac_contents(where=(name='YESTERDAY'));
  run ;
  quit ;

SAS log:
NOTE: The data set WORK.MAC_CONTENTS has 1 observations and 8 variables.
```

Figure 3 - Retrieving the Label of a Macro with PROC CATALOG

The CONTENTS keyword will display various attributes about all the entries in the specified catalog to whatever ODS destination is open. You can also send the results to a data set with the OUT= option, where data set options can be used to subset records and variables.

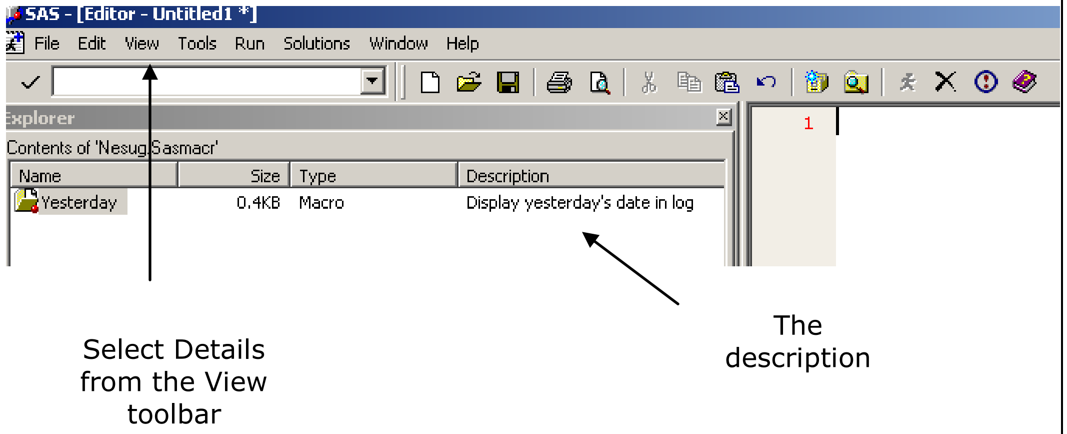
Metadata on macros can also be retrieved by querying the DICTIONARY table CATALOGS or the SASHELP view VCATALG, as demonstrated below in Figure 4.

```
proc sql ;
   select   memname
          , objname
          , objdesc
   from     Dictionary.Catalogs
   where    libname='NESUG'
          & objtype='MACRO'
          & objname='YESTERDAY'
   ;
   quit ;
```

Member Name	*Object Name*	*Object Description*
SASMACR	YESTERDAY	See macro gatekeeper for the source code

Figure 4 - Retrieving the Label of a Macro from Dictionary.Catalogs

If you are working in interactive mode, you can also view the description of catalog entries through the Explorer window with the Details option from the View menu. Figure 5 below is a depiction of SAS in the Windows environment, but can also be an analogous view is available in a Unix environment.


SAS - [Editor - Untitled1 *]
File
Edit
View
Tools
Run
Solutions
Window
Help
Explorer
Contents of 'Nesug.Sasmacr'
Name
Size
Type
Description
Select Details from the View toolbar
The description

Name	Size	Type	Description
Yesterday	0.4KB	Macro	Display yesterday's date in log

Figure 5 - Viewing the Label of a Macro from the Explorer Window

FORMATS/ INFORMATS

Formats are applied to variables, which affect how they are displayed. User-defined formats are stored in catalogs and are available as long as the appropriate librefs are specified in the FMTSEARCH system option. Unlike macros, the values and ranges and their associated labels for formats can always be retrieved by using the CNTLOUT or FMTLIB options on the PROC FORMAT statement. However, you may not be privy to the actual program that generated the format, which may contain useful information such as the author, source of the codes, or why it is needed. This useful metadata can be added as part of the description for format catalog entries.

Unlike macros, descriptions cannot be added to user defined formats at their time of creation with PROC FORMAT. They can be added or modified with PROC CATALOG by specifying the appropriate entry type, as show below in Figure 6.

```
proc catalog catalog=flib.formats et=format;
  modify FConf(desc='Site of future PharmaSUGs: Source: B. Orr') ;
  run ;
  quit ;

SAS log:
NOTE: Description changed for entry FCONF.FORMAT in catalog XXX.FORMATS.
```

Figure 6 - Adding a Descriptive Label to a Format with PROC CATALOG

Retrieving the descriptions for formats can be accomplished in similar ways already discussed for macros. Figure 7 below shows a query against the VCATALG view to extract the description for the format catalog entry FCONF.

```
proc sql ;
  select   memname
         , objname
         , objdesc
  from     Sashelp.VCatalg
  where    libname='WORK'
         & objtype='FORMAT'
         & objname like 'FCONF%'
  ;
  quit ;
```

Member Name	*Object Name*	*Object Description*
FORMATS	FCONF	Site of future PharmaSUGs: Source: B. Orr

Figure 7 - Retrieving a Label of Format from Sashelp.Vcatalg

The discussion in this section has focused on formats, but also applies to its cousin the informat, which defines how a variable is read.

CONCLUSION

Having descriptive information on entities in the SAS system is useful and is generally considered a good idea. While many users are familiar with adding descriptive information to variables, data sets, reports and programs, this paper addresses how formats, informats, macros and user-defined functions can be augmented with user provided metadata. This can be particularly helpful for those who need to use a compiled version of the catalog entries when they are not privy to the code that produced them.

REFERENCES

Crawford, Peter (2006), "The Big Introduction to the Smallest Macro", *Proceedings of the 31st Annual SAS Users Group International*

SAS OnLineDoc®

ACKNOWLEDGEMENTS

The author would like to thank Toby Dunn and Mike Molter for their insightful comments in reviewing an earlier publication of this paper.

DISCLAIMER

CONTACT INFORMATION

Your comments and questions are valued and encouraged.

Ken Borowiak
PPD
3900 Paramount Parkway
Morrisville NC 27560

Ken.Borowiak@ppdi.com
Ken.Borowiak@gmail.com