Paper 116-28

Keep Those Formats Rolling: A Macro to Manage the FMTSEARCH= Option

Pete Lund, Looking Glass Analytics, Olympia, WA

Introduction

User-defined formats in the SAS System® are a wonderful thing. They allow you to label data, classify data and aggregate data. Storing your formats in a permanent catalog is a handy and efficient way to ensure that your formats are always available to your programs.

The FMTSEARCH= option allows you to tell SAS which catalogs to search when a user-written format is referenced. However, this option can be unforgiving and provides little feedback.

This paper presents a macro which allows for more control over the FMTSEARCH= option. Catalogs can be added to the list without deleting those that already exist and can be easily moved to the beginning or end of the list. Also, the current value of the format search list can be displayed as well.

Searching for User-Defined Formats

By default SAS is only going to look in two places when it sees a reference to a user-defined format. In the WORK.FORMATS catalog (the default location) and the LIBRARY.FORMATS catalog.

LIBRARY is a “special” libref that SAS will also search for formats if a catalog called FORMATS is located there. This is the only aspect of LIBRARY that is different than any other libref. It’s not really a reserved name; the default format-searching behavior is its only special feature. You can store datasets and other catalogs in the LIBRARY library.

LIBRARY is a handy place to permanently store formats so that they are available from session to session. As long as the libref is active SAS will find formats stored there.

What about formats stored in other libraries or catalogs? That’s another story!

You’ll find quickly that the LIBRARY.FORMATS catalog can be a bit restricting for storing your formats. Very often you will have multiple projects with multiple formats and it would be nice to store the formats in the same library as the rest of the project data.

However, SAS only searches for formats in the WORK.FORMATS and LIBRARY.FORMATS catalogs - how do you reference all your project formats? The FMTSEARCH= option allows you to specify which format catalogs to look in when a user-defined format is referenced.

Using the FMTSEARCH= Option

The FMTSEARCH= option is a system option, not a PROC FORMAT option, and is used on an OPTIONS statement. The value is a list of catalogs to search separated by spaces and enclosed in parentheses. The catalog names are not quoted and can contain either one-level or two-level names. Two-level names are in the format `<LIBREF>.<CATALOG>` and one-level names imply FORMATS as the catalog name.

The WORK.FORMATS and LIBRARY.FORMATS catalogs are always searched first, unless they are included in the FMTSEARCH list.

```
options fmtsearch=(ProjLib
                   TempLib);
```

In the above example, four catalogs will be searched in the following order (notice that WORK and LIBRARY are included in the list and that the FORMATS catalog is implied for all the one-level names):

- WORK.FORMATS
- LIBRARY.FORMATS
- PROJLIB.FORMATS
- TEMPLIB.FORMATS

```
options fmtsearch=(ProjLib
             Library Work);
```

In this example, three libraries will be searched in the order shown:

- PROJLIB.FORMATS
- LIBRARY.FORMATS
- WORK.FORMATS

As we'll see, the order of the list is important and is different depending on whether WORK and LIBRARY are explicitly referenced.

A few facts about the FMTSEARCH= option will help you to resolve issues of formats that aren't found or are different than you expect.

The FMTSEARCH= List is Ordered

The catalogs are searched in the order listed. When a format is found it is used and the search stops. This is only an issue if you have formats of the same name in multiple catalogs. If so, the first one found will be used.

The FMTSEARCH= List is All Inclusive

With the exception of the WORK and LIBRARY libraries, discussed earlier, the only catalogs searched are those that are in the FMTSEARCH= list. If you have formats stored in other catalogs they will not be found.

This can really start to be an annoyance if you store formats in a number of catalogs for different projects. Every time you change the FMTSEARCH= list the prior value is overwritten. If you just want to add another catalog to the list you need to include what was there before as well. It's not an uncommon experience to spend many minutes (hours!) trying to figure out why the formats that you can see in SAS Explorer or in your SAS code cannot be found only to realize that the FMTSEARCH= list doesn't include the catalog any more.

The FMTSEARCH= List is Not Validated

No checking on the validity or existence of a libname or catalog name is done here. This can cause a lot of gray hairs is you have a slight typo in a libref or catalog name.

A "side effect" of this is that the sequence of the LIBNAME statement and FMTSEARCH= option are unimportant. As long as both are set before the format is referenced, formats in the included catalogs will be found.

Looking at the FMTSEARCH= Value

Just looking at the FMTSEARCH option value can be confusing. Let's follow the value along as we make a few changes.

When SAS is first started the default value for FMTSEARCH is *(WORK LIBRARY)*. We do a little work on a format, $Race, in the work library and then need to work on another project. We want to point to a permanent format catalog so we issue the following command:

```
  options fmtsearch=(ProdLib);
```

If we look at the FMTSEARCH option now it is *(PRODLIB)*. All is right with the world – until you run a procedure and notice that your race values are in the temporary format you were working on earlier!

The problem is that the WORK and LIBRARY format catalogs are always searched. But, once a value has been given to the FMTSEARCH option WORK and LIBRARY no longer are displayed as part of the value unless they are explicitly coded. This means that if you look at the value of FMTSEARCH and do not see WORK and LIBRARY they are at the beginning of the search list.

In order to get our permanent format to take precedence over the temporary format of the same name, our option value would need to be:

```
  options fmtsearch=(ProdLib work);
```

All of these can, and will, bite you as you use more and more formats. So, is there anything you can do??

A FMTSEARCH= Management Macro

One solution to some of these issues was to create a macro to manage the FMTSEARCH= value. The macro has only two important parameters: CAT, which specifies the catalog and ACTION, which specifies what you want to do.

The FMTSEARCH macro does two things: first, it changes the FMTSEARCH option as specified and second, it writes to the log the current FMTSEARCH value and the status of all the catalogs referenced.

The Parameters – CAT

The value of the CAT parameter is simply the format catalog name. It can be either a one-level or two-level name. If a one-level name is listed, the catalog FORMATS is assumed.

The CAT parameter is not required. If it is omitted a description of the macro syntax is written to the log.

The Parameters – ACTION

The ACTION parameter tells the macro what you want done with the catalog. We'll look at the values of the ACTION parameter and then at some examples.

The ACTION parameter is not required, but has a default value (M). See descriptions below.

An example of the results of each ACTION value will given below.

Using the Macro – Values of ACTION

Note: for the examples below, assume that the current value of FMTSEARCH is TestLib and that WORK and LIBRARY maintain their default location at the beginning of the format search list. Also, assume that there is a WORK.FORMATS catalog and that the libref LIBRARY has not been assigned.

- ***D*** – deletes the catalog from the FMTSEARCH value.

```
%fmtsearch(cat=ProdLib,action=D)
```

The ProdLib.formats catalog will be removed from the search list. The value of FMTSEARCH will be *WORK LIBRARY*.

The following log note will be written:

```
===========================================
Current FmtSearch Option value:

     WORK* LIBRARY* TESTLIB

   *implicitly included by default.

===========================================
Status of current catalogs:

NOTE:    WORK.FORMATS EXISTS
WARNING: LIBRARY.FORMATS DOES NOT EXIST
NOTE:    TESTLIB.FORMATS EXISTS
===========================================
```

Note that if you looked at the value of FMTSEARCH that WORK and LIBRARY would not be there. They are there by default.

- ***B*** – adds (or moves) the catalog to the beginning of the format search list –before WORK and LIBRARY.

```
%fmtsearch(cat=ProdLib,action=B)
```

This will move WORK and LIBRARY to the second and third positions in the search list:

```
===========================================
Current FmtSearch Option value:

      PRODLIB WORK LIBRARY TESTLIB

===========================================
Status of current catalogs:

NOTE:    PRODLIB.FORMATS EXISTS
NOTE:    WORK.FORMATS EXISTS
WARNING: LIBRARY.FORMATS DOES NOT EXIST
NOTE:    TESTLIB.FORMATS EXISTS
===========================================
```

Note that WORK and LIBRARY are now explicitly referenced.

- ***E*** – adds (or moves) the catalog to the end of the list.

```
%fmtsearch(cat=ProdLib,action=E)
```

```
===========================================
Current FmtSearch Option value:

     WORK* LIBRARY* TESTLIB PRODLIB

   *implicitly included by default.

===========================================
```

The status note will be the same as B above, with the exception of the order of the catalog list.

- ***M*** – adds (or moves) the catalog to the "middle" of the list, but leaves WORK and LIBRARY in their default location at the beginning. Note: if WORK and LIBRARY have already been explicitly specified in the FMTSEARCH list they will remain in their locations and this ACTION value will place the current catalog before them. This is the default action of the macro.

```
%fmtsearch(cat=ProdLib,action=M)
```

```
=========================================
Current FmtSearch Option value:

     WORK* LIBRARY* PRODLIB TESTLIB

   *implicitly included by default.

=========================================
```

PRODLIB is moved to the middle of the list, after the default WORK and LIBRARY and before TESTLIB. The status note will be the same as B above, with the exception of the order of the catalog list.

Note that this is the default action and `%fmtsearch(cat=ProdLib)` would have produced the same result.

- ***X*** – resets the FMTSEARCH option to its default value including only WORK and LIBRARY.

  ```
  %fmtsearch(action=X)
  ```

- ***L*** – displays the status notes for the current value of the FMTSEARCH option in the SAS log.

  ```
  %fmtsearch(action=L)
  ```

Note that for both actions X and L no CAT value is required and it is ignored if included.

How Does it Work?

An enumerated copy of the macro is given in Appendix 1. We'll step through it piece by piece to see how it works. There are some pieces that have been eliminated to save a little space.

1 – The default catalog is FORMATS and is not required when setting the FMTSEARCH option. In order to facilitate subsequent processing this statement strips off .FORMATS if it is present in the CAT parameter and upper-cases the value.

2 – The macro variable &_FMS is set equal to the current value of the FMTSEARCH option. The GETOPTION function is used to get the value. Note that the value returned by GETOPTION contains parenthesis surrounding the catalog values. These are stripped off here.

As in step 1 above, .FORMATS is removed from any catalog references.

3 – The X action simply resets the FMTSEARCH option to its default value by passing a null catalog list.

4 – The D action deletes the current value by using the TRANWRD function. It translates the current CAT value to a blank value. Note that the value passed to the function contains leading and trailing spaces. This will prevent catalog names that may be embedded in other names from being affected.

The new catalog list, without the current CAT, is stored in a macro variable, &_NewFMS, and passed to the FMTSEARCH option.

Note that this section is called for many of the action values. See further discussion for details.

5 – The M action will place the current CAT value after default WORK and LIBRARY and before other catalogs.

We'll use a little trick to get this to work easily. First, remember that the delete action was already run for the current CAT. This does two things: it eliminates the current value from the FMTSEARCH list and it creates the macro variable &_NewFMS which contains any other catalogs in the list.

Now we can create a new FMTSEARCH value made up of the current catalog followed by any others that were already assigned. Remember that WORK and LIBRARY do not show up in the FMTSEARCH list unless they have been explicitly coded. This behavior works just fine with the macro. If they were explicitly coded they will be in &_NewFMS and remain in their place. If not, they will continue to occupy their default location at the beginning of the list.

6 – This step places the current catalog at the beginning of the list. There is a bit more going on here because we need to explicitly code WORK and LIBRARY so that they come after the current catalog.

Again, remember that the delete action has already been run on the current catalog. We need to check the value of &_NewFMS to see if WORK and/or LIBRARY are already in the list. If they are we'll leave them – it not, we'll add them after the current catalog.

7 – Again, the current catalog has already been deleted and now it is just added after the remaining catalog entries, placing it at the end of the list.

8 – This step displays the FMTSEARCH option value after the actions above have been processed.

First, we use the GETOPTION function again to get the current value of FMTSEARCH as we did above. We then need to search that value to see if WORK and LIBRARY are there. If there are not, that means they will be in their default places at the beginning of the list and we need to add them to the list. Add an asterisk to WORK and/or LIBRARY so we can footnote that they are there by default.

9 – Now comes one of the most important features of the macro. We'll take out &_FMS list and strip out any asterisks we may have put there in step 8.

Now, loop through all the individual catalogs in the list, using the %SCAN function and breaking &_FMS on spaces. We then pass each catalog entry to the CEXIST function. If returns a 1 if the catalog exists and 0 if not. We can use the CEXIST value to write the appropriate note to the log about the existence of each catalog.

Conclusion

I hope that this little macro has inspired you to not only be more brave in your development of your own permanent formats, but also to see how you can use the macro language to take control of circumstances that can seem a bit unruly at first.

Please let me know if you see any areas for improvement or errors.

Acknowledgements

SAS® is a registered trademark of SAS Institute, Inc. in the USA and other countries.

Author Contact Information

Pete Lund
215 Legion Way SW
Olympia, WA 98501
(360) 528-8970
pete.lund@lgan.com
www.lgan.com

Documented electronic copies of the macro are available on request. Drop me an e-mail and I'll send it along. You should also be able to cut and paste the code from the electronic copy of the proceedings.

Appendix 1
Enumerated Code

(Comments and some error checking have been removed)

```
 %macro FmtSearch(Action=M,Cat=,Status=Y);
   %local _FMS _Msg i pos;
   %global _NewFMS;

   %let Action = %upcase(&Action);

   %* <code to check for valid actions and to display macro syntax
       on empty call were removed for the sake of space> ;

❶  %let Cat = %sysfunc(tranwrd(%upcase(&Cat),%str(.FORMATS),%str()));

❷  %let _FMS = %upcase(%sysfunc(compress(%sysfunc(getoption(fmtsearch)),%str(%(%)))));
   %let _FMS = %sysfunc(tranwrd(&_FMS,%str(.FORMATS),%str()));

❸  %if &Action eq X %then
     %do;
       options fmtsearch=();
     %end;

❹  %if &Action eq D or &Action eq M or &Action eq B or &Action eq E %then
     %do;
       %let _NewFMS = %sysfunc(tranwrd(%str( &_FMS ),%str( &cat ),%str()));
       %if &Action eq D %then %str(options fmtsearch=(&_NewFMS););
     %end;

❺  %if &Action eq M %then
     %do;
       options fmtsearch=(&Cat &_NewFMS);
     %end;

❻  %if &Action eq B %then
     %do;
       %let _NewCat = &Cat;
       %if %sysfunc(indexw(&_NewFMS,WORK)) eq 0 and &Cat ne WORK %then %let _NewCat = &_NewCat WORK;
       %if %sysfunc(indexw(&_NewFMS,LIBRARY)) eq 0 and &Cat ne LIBRARY %then
         %let _NewCat = &_NewCat LIBRARY;
       options fmtsearch=(&_NewCat &_NewFMS);
     %end;

❼  %if &Action eq E %then
     %do;
       options fmtsearch=(&_NewFMS &Cat);
     %end;

❽  %if &Status eq Y %then
     %do;
       %put;
       %put %str(===========================================================);
       %let _FMS = %upcase(%sysfunc(compress(%sysfunc(getoption(fmtsearch)),%str(%(%)))));
       %if %sysfunc(indexw(%upcase(&_FMS),LIBRARY)) eq 0 %then %let _FMS = &_FMS LIBRARY*;
```

```
      %if %sysfunc(indexw(%upcase(&_FMS),WORK)) eq 0 %then %let FMS = WORK* &FMS;
      %put Current FmtSearch Option value:;
      %put ;
      %put %str(     ) &_FMS;
      %put ;
      %if %index(&_FMS,*) ne 0 %then
        %do;
          %put %str(   )*implicitly included by default.;   %put;
        %end;

   ❾  %let _FMS = %sysfunc(compress(&_FMS,%str(*)));
      %put %str(============================================================);
      %put Status of current catalogs:;
      %put ;
      %let i = 1;
      %do %while(%scan(&_FMS,&i,%str( )) ne %str( ));
        %let ThisCat = %scan(&_FMS,&i,%str( ));
        %if %index(&ThisCat,.) eq 0 %then %let ThisCat = &ThisCat..FORMATS;
        %if %sysfunc(cexist(&ThisCat)) eq 1 %then %put NOTE:    &ThisCat EXISTS;
        %else %put WARNING: &ThisCat DOES NOT EXIST;
        %let i = %eval(&i + 1);
      %end;
      %put %str(============================================================);
      %put ;
    %end;

  %Finish:
%mend;
```