

SAS® Helps Those Who Help Themselves: Creating Tools to Aid in Your Application Development

Pete Lund, Looking Glass Analytics, Olympia, WA

Abstract

Writing code, organizing output, managing datasets--all these things and more go into development of a successful application. SAS provides a wonderful environment to develop code, but there are times when the Enhanced Editor or SAS Explorer just isn't enough to develop your application in the most efficient manner.

This paper presents some tips and tricks for simplifying development tasks. Some are user-written macros to aid in code development, data management, and process tracking. Some are little known or little used SAS tools.

The goal of the paper is not intended to go into line-by-line detail on all these tips, but to inspire you to think of ways to improve your own processes. I feel kind of cheated if I make it through a whole project without acquiring a new utility macro to add to my toolbox. I hope this paper leaves you looking at your work in the same way. Besides, I'll send you the source code for all the examples so you can play with them and learn on your own!

SAS Tools

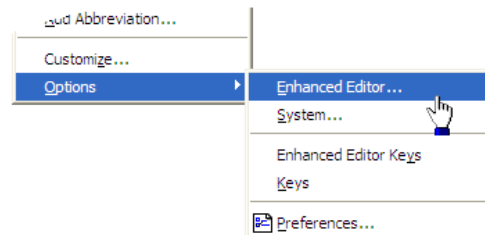
The SAS tools mentioned here are specific to the Windows V8+ Enhanced Editor ("the editor"). I realize that many of us develop on other platforms or using other editors. But, highlighting some of the tools built into the SAS editor might convince some that the

SAS environment can support efficient application development.

Syntax Highlighting

By default the editor will highlight SAS program syntax for you. This is extremely helpful in eliminating non-matching quotes and misspelled keywords. Consider a few changes that make the programming environment even friendlier.

To edit editor options, go to the Tools menu and select Options...Enhanced Editor.



From there go to the Appearance tab. On this tab you can change the color and style of a number of program elements. One change that I strongly urge is adding a background color to comments. It makes them stand out so much more on the screen.

```

91
92 array D00days(&BD:&ED) $2 _temporary_;
93 array Exclude(&BD:&ED) $1 _temporary_;
94
95 if first.BA then Status = '';
96
97 * If the charge has not been released yet, set
98 * Also, bound the add and release dates to the
99
100 if ChargeRelDate1 eq . or ChargeRelDate1 gt &EndDate
101 ChargeAddDate = max(ChargeAddDate, &BeginDate);
102
103 * One more adjustment needs to be made to the r
104 * 1:45 am or was released before 1:45 am that
105 * dates here.
106
107 if ChargeAddTime gt '01:45't then ChargeAdd*
108 if ChargeRelTime lt '01:45't then Charge*

```

I've also made my comments bold so they are even more obvious.

It's important to note that the SAS editor does not print the highlighted syntax. I often use a programmer's editor, such as UltraEdit, to print code for production documentation. A number of these editors support onscreen and printable syntax highlighting.

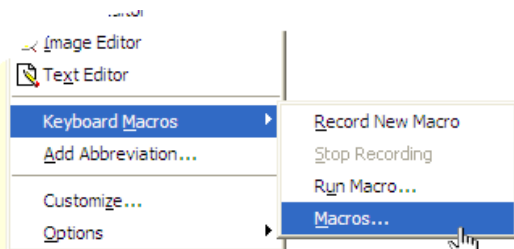
One of the categories of file elements to highlight is "User defined keyword." One place to take advantage of this is to "register" your user-written macros. From the "General" tab of the Enhanced Editor options click "User Defined Keywords." Enter the names of macros that you've written (without the leading "%"). Now, whenever you call your macros the macro name will be highlighted in the manner you've defined for user defined keywords.

```
%MakeProjectFiles(...
%MakeProjectFile(...
```

In the above example, it's easy for me to tell when I've spelled the macro name correctly.

Keyboard Macros

Keyboard macros can be recorded or written in a keyboard macro editor. To get to the recorder or editor, select the Tools menu and Keyboard Macros.




I would suggest that you quickly learn how to create your own, rather than record them as you have much more control how they look and act.

I use keyboard macros for a number of program documentation tasks. Let's walk through creating a simple, but useful, documentation aid and you'll begin to see the usefulness of this tool.

We want to put a standard comment header in our program whenever we make a change to production code. The header should look like this, containing the current date and your initials:

```
*****;
* Modified: 04/01/2003  PJJ      *;
* -----*;
```

The comment will contain the current date and your initials. Here's how we can automate that with a keyboard macro.

1. Go to Tools...Keyboard Macros...Macros
2. Select Create
3. Give the macro a name (ModifyHeader)
4. From the list of commands:
 - a) Select "Move cursor to column 1" and click the little double arrow () to place it in your contents list.
 - b) Select "Insert a string"
 - Enter "*****;"
 - c) Select "Insert a carriage return"
 - d) Select "Insert a string"
 - Enter "* Modified: "
 - e) Select "Insert current month index"
 - f) Select "Insert a character" and enter "/"
 - g) Select "Insert current day of month"
 - h) Select "Insert a character" and enter "/"
 - i) Select "Insert current year"
 - j) Select "Insert a string"
 - Enter " PJJ *;"
 - k) Select "Insert a carriage return"
 - l) Select "Insert a string"
 - Enter "* -----*;"
 - m) Select "Insert a carriage return"
5. Select OK
6. Click "ModifyHeader" in the macro list and select "Assign Keys"
7. Highlight "None" in the "Press new shortcut key" box and press a key combination, like alt-M. It will tell you if that combination is already assigned to another function.

8. Select Assign
9. Select OK
10. Select Close

Now, back in the editor, hit alt-M and you will see your comment appear in your program. It will contain the current date and your initials every time you use it.

If you spend any time looking at the list of available commands you see that there is a great deal you can do with keyboard macros. Some of the things I use keyboard macros for are:

- Program header template – this is about 40 lines long
- Comment template – so all my comments are the same width and style
- Macro call template – I now don't have to remember the syntax of some of my macros

There are a couple tips I'll pass along. First, play around with the use of carriage returns, moving to column 1, moving to the end of the current line, etc. These type of commands give you a lot of control as to how your text will be inserted in the midst of existing text.

Second, you can export macros so that others can use them (by importing them). If you have an office standard for program headers or comment style this can save a lot of duplicated effort.

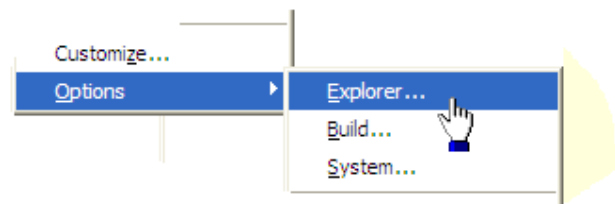
A note on exporting keyboard macros: the exported files, with an extension of .kmf, look a lot like plain text files. You might be tempted to edit them in your favorite editor, save them, and import them back into SAS. Don't! SAS will probably lock up and most likely require a reboot. Maintain and edit your keyboard macros in SAS.

SAS Explorer Tools

There are some options available in the SAS Explorer which can be used to facilitate your application development.

If you work on more than one project at a time, as we all do, you may want to take advantage of the "Favorite Folders" feature of SAS Explorer. If the "Favorite Folders" icon does not appear in your SAS Explorer list you'll need to initialize it.

To initialize this option, go to the Tools menu and select Options...Explorer. (You'll only see this menu if SAS Explorer is the current active window).



From there, select Initialize from the drop-down list at the top of the screen. Select Favorite Folders and Add. When you go back to your SAS Explorer window you will see a "Favorite Folders" icon.

Double-click on the icon and then right-click in the empty window and select "New Favorite Folder." You can now select a directory and give it a meaningful name. Double-clicking on that entry brings up a list of all files in that directory. I find it much simpler to move around between all my SAS project code directories if they're in the favorite folders list than navigating up and down through File...Open.

Another feature available for SAS Explorer is to view and manage all your file references (filerrefs). These are available under the "File Shortcuts" icon. (Follow the same initialization steps above if you do not see this item.) From here you can see all filerrefs you've set up with FILENAME statements.

You can also set up filerefs directly from here and add an option to have them initialized at startup.

I'd encourage you to explore all the features of the SAS Explorer – there are a number of features and tools that can make your programming tasks just a bit smoother.

Develop Your Own Tools

I have written a number of SAS macros that are solely for the purpose of enhancing the development process. We'll look at a few of them here – hopefully, it will spur you on to bigger and better things. (The code for all macros referenced here is available upon request – as long as you promise to send me any improvements you make!)

Don't Forget Your Syntax

It doesn't take too many macros before you begin to forget the syntax you wrote for calling them. I've gotten in the habit of writing a little piece of code at the top of the macro that checks for the existence of a required parameter. If there is no value a log note is created that reminds me of the syntax. After the log note is created control jumps to a tag, %Quit:, at the bottom of the macro.

```
%macro Overlap(BD1,ED1,BD2,ED2);
  %if &BD1 eq %str() %then
    %do;
      %put %nrstr(  %%)Overlap parameters;;
      %put %str(      %())Span 1 Begin Date,;
      %put %str(      )Span 1 End Date,;
      %put %str(      )Span 2 Begin Date,;
      %put %str(      )Span 2 End Date%str(%);
      %goto Quit;
    %end;
  :
  :
```

If I call the above macro with a null parameter string, %Overlap(), the following appears in the SAS log:

```
103 %Overlap()
      %Overlap parameters:
          (Span 1 Begin Date,
           Span 1 End Date,
           Span 2 Begin Date,
           Span 2 End Date)
```

Now all I have to remember is the name of the macro and it self-reports the syntax.

Setting Up Project Files

I always set up a common directory structure for my projects. It took a while, but I finally wrote a little macro to create those directories for me and to set up macro variables referencing them.

```
%MakeProjectFiles(ClientName=KC Jail,
                  ProjectName=SRA Inmates)
```

There is another parameter, Path, that contains the path where all these directories should be located. I assign this a default value, so I rarely have to change it. Let's assume that Path = d:\projects. The call above would create the following directories:

- d:\projects\CK Jail\SRA Inmates\SAS Code
- d:\projects\CK Jail\SRA Inmates\Results
- d:\projects\CK Jail\SRA Inmates\Data

A series of system MD (make directory) commands are used to create the directories:

```
x md "&Path\&ClientName\&ProjectName\SAS Code";
```

The macro also creates three global macro variables, &_CodePath, &_DataPath and &_ResultsPath. These make it easier to set up code that references file locations, such as FILENAME and LIBNAME statements and the OUTFILE parameter of PROC EXPORT.

The FILEEXIST function is used to determine if the directories already exist.

```
%if %sysfunc(fileexist(&Path\&ClientName\
&ProjectName\SAS Code)) ne 1 %then...
```

If the directories already exist only the macro variables are created. The macro can easily be modified to create whatever directory structure you find most useful.

Checking for Duplicates

A very common development task is checking a dataset for duplicates on a particular variable(s). A very simple process is to sort a dataset by the key variables and then look for observations that are not the first and last observation in the by-group. But even simple processes can get tedious when you've done it enough.

This macro simply automates the process by taking a dataset name and a list of key variables and creates a duplicates dataset. It also reports back how many observations were in the original dataset and how many duplicate values were found.

For example, this call

```
%getdups(dsn=kc.Bookings,
keyvars=arrestdate arrestlocation)
```

creates the dataset DUPS and generates the following log note:

```
NOTE: There were 93 duplicates records on
ARRESTDATE and ARRESTLOCATION in dataset
KC.BOOKINGS.
NOTE: (There are 12,346 observations in the
original dataset)
NOTE: Dataset DUPS created.
```

Nothing fancy, but some helpful information.

Reduced to its simplest form the code in the macro is very simple. The incoming dataset is sorted by the key variables into a temporary dataset called `__temp`. This dataset is then searched for duplicate values

on the key (log reporting has been removed from the code).

```
data &dupfile
set __temp end=done;
by &KeyVars;

if not (first.&LastKey and last.&LastKey);
run;
```

The only little trick is to get the last variable in a list of multiple variables. In the above example the `&KeyVars` parameter was "ArrestDate ArrestLocation." We need to extract ArrestLocation from that string in order for the first. and last. processing to work correctly.

The `%SCAN` function is used to get the last "word" from the value:

```
%let LastKey = %scan(&KeyVars,-1,%str( ));
```

This breaks `&KeyVars` into chunks delimited by spaces (`%str()`) and returns the last one (-1).

Formatted Log Comments

A good format of process documentation is having good comments in you run log about what's happening to your data. The SAS log notes are pretty good about that, but I often have code that is `%INCLUDE`d or generated in a macro. I often have the `NOTES` and `SOURCE2` options turned off and it can be really tricky to follow a problem.

I developed a macro that allows for putting nicely formatted comments in the SAS log. Macro variables can be passed and will be resolved in the comment.

For example, suppose I want to write a note to the log that I'm starting a particular step and give the date, time and value of a couple parameters. The call to the macro might look like this:

```
%comment(/C,Beginning Booking Report.sas,/L,
/D,Run Date: %sysfunc(date(),mmdyy10.),
Run Mode: &mode,
Output Type: &OutputType);
```

The following log note would be produced:

```
*****
* Beginning Booking Report.sas *
*-----*
* Run Date: 12/16/2002      *
* Run Mode: PROD           *
* Output Type: HTML        *
*****
```

As many lines as needed can be passed to the macro. Multiple lines are separated by commas. There are special line values for controlling the appearance of the comment:

- /C – centers the text
- /L – left justifies the text (default)
- /R – right justifies the text
- /D – prints a line of dashes
- /H – prints a line of asterisks
- /In – indents text n lines

A box of asterisks is produced around the comment, sized to the length of the longest line. The line break character can be changed from a comma to any other character if commas are included in the text. This is done by adding DLM=x as the last portion of the macro call , where x is the character to use in place of a comma as the line delimiter.

More detail on the functionality and design of this macro is available in Lund (2000).

Managing Formats

I can't tell you how many times I've spent 30, 40, 60 minutes trying to figure out why my formats are not being found or the right format are not being used only to discover a problem with my FMTSEARCH option value.

The FMTSEARCH option is necessary if you ever want to create your own permanent SAS formats. By default, SAS will only look in the WORK.FORMATS and LIBRARY.FORMATS catalogs when a format or informat is referenced. FMTSEARCH allows you to specify other format catalogs to search so that your own permanent formats can be found.

There are a couple features of the FMTSEARCH option that can make it a bit of a trap.

- It is all inclusive - With the exception of the WORK and LIBRARY libraries, which are always searched, 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.
- It 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.
- It 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 if you have a slight typo in a libref or catalog name.

All of these things can bite you when it's the least convenient. To lessen the speed at which my gray hairs were coming in I wrote a little macro to manage the FMTSEARCH option. It addresses all the issues above:

- It will add a catalog to the list, without removing any catalogs that were already referenced.
- Placement of the catalog in the list can be specified: at the beginning, end or

after WORK and LIBRARY, but before other catalogs.

- The status of all catalogs currently reference in the FMTSEARCH option is reported to the log – i.e., do they exist or not.

Here's an example call to the macro and the log note that is produced:

```
%fmtsearch(cat=kc,action=e)
```

The log contains:

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

  WORK* LIBRARY* KC

  *implicitly included by default.

=====
Status of current catalogs:

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

With this log note, you can see the order in which format catalogs will be searched and the status of their existence.

One other helpful feature is the inclusion of WORK and LIBRARY in the list, if they have not been explicitly referenced. There is a feature of the FMTSEARCH option value that can be problematic when trying to look at the catalog sequence. Try this at home:

Start a new SAS session and submit the following code:

```
%put %sysfunc(getoption(fmtsearch));
```

You'll see this in the log:

```
1      %put %sysfunc(getoption(fmtsearch));
      (WORK LIBRARY)
```

Just as expected. However, now add a catalog with the FMTSEARCH option and check the option value again.

```
options fmtsearch=(MyFmts);
%put %sysfunc(getoption(fmtsearch));
```

The value returned is very misleading:

```
2      options fmtsearch=(MyFmts);
3      %put %sysfunc(getoption(fmtsearch));
      (MYFMTS)
```

The only catalog shown for the FMTSEARCH value is MyFmts. In reality, WORK and LIBRARY are both still in the list, at the beginning. This can be a real bugaboo when trying to figure out why the "right" formats are not being used.

The %FMTSEARCH macro accepts two parameters. CAT names the catalog and can be either a one- or two-level name. One level names imply .FORMATS as the catalog name. ACTION tells the macro what to do with the catalog:

- D – deletes the catalog from the list
- E – adds (or moves) the catalog to the end of the search list
- B – adds (or moves) the catalog to the beginning of the list
- M – adds (or moves) the catalog to the middle of the list, after WORK and LIBRARY and before any other catalogs.
- X – resets FMTSEARCH to the default value

More detail on the functionality and design of this macro is available in Lund (2003).

A Simple Data Dictionary

In many projects there are a great number of datasets produced. There came a need to have an easy way of displaying the contents and properties of all the datasets for a project in one place.

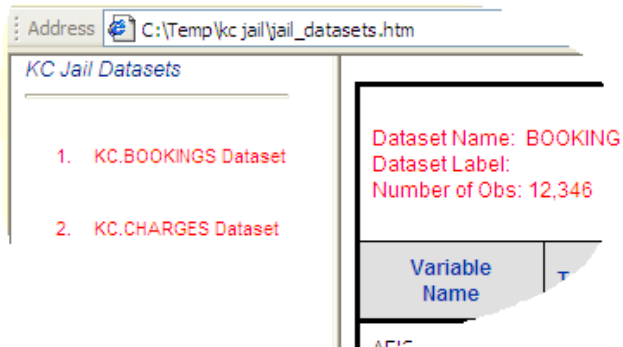
A simple data dictionary macro was written to handle this. A set of HTML pages is generated by the macro. The call is very straightforward:

```
%DataDictionary(project=KC Jail,
               htmlpath=c:\temp\kc jail,
               framename=jail_datasets,
               dslist=kc.bookings kc.charges);
```

- Project – simply gives a title for the table of contents
- HTMLpath –directory where the files will be stored
- FrameName – name of the HTML page to view
- DSlist – list of datasets, separated by spaces

In the sample call above an HTML page named `jail_datasets.htm` would be created in `c:\temp\kc jail`. If the directory does not exist, it will be created automatically.

When you load that page you'll see a list of the datasets you referenced in a table of contents on the left side of the page. Notice that the table of contents header contains the value of the "Project" parameter.



By default, information about the first dataset in the list will be displayed on the right side of the screen. The information listed contains much of the information you would get from PROC CONTENTS including the variable name, type, length, format and label. However, there are a couple important pieces that PROC CONTENTS does not report.

First, the format catalog for any user-defined formats is given. The FMTSEARCH option must be set to the same value it will have when using the datasets. The catalogs in the FMTSEARCH option are searched for the formats associated with variables in the datasets and the first catalog found for each format is listed in the Format Catalog column.

Variable Name	Type	Length	Format	Format Catalog	Index Usage
AFISNumber	Char	8			AFI
AFISstatus	Char	1			AF
Age	Num	3			
Alcohol	Char	1	\$YNU.	KC.FORMATS	
ArrestAgency	Char	9	\$JURFMT.	KC.FORMATS	
ArrestDate	Num	4	MMDDYY10.		

Also, you'll notice that the user-defined formats are underlined. They are actually hyperlinks to pages containing the format values. Clicking [\\$JURFMT](#) in the page above will display the following page, listing all the values of that format.

Contents of format: [\\$JURFMT](#) - (in catalog: KC.FORMATS)

Range Start	Range End	Label
WA0171400	WA0171400	KC - Algona Police Dept
WA0170100	WA0170100	KC - Auburn Police Dept
WA0172900	WA0172900	KC - Beaux Arts Police Dept
WA0170200	WA0170200	KC - Bellevue Police Dept
WA0171500	WA0171500	KC - Black Diamond Pd
WA0170300	WA0170300	KC - Bothell Police Dept
WA0174100	WA0174100	KC - Burien Police Dept
WA0174800	WA0174800	KC - Coe Police Dept

This macro provides a very handy method for presenting the information about a number of datasets in a single place with access to most relevant information.

More detail on the functionality and design of this macro is available in Lund (2002).

Conclusion

I sincerely hope that this paper has inspired you to look for places in your own application development process that can be streamlined or routinized. Development of applications is often a daunting task. Any tools that can take away some of the tedium and smooth some of the rough spots find a spot in my toolbox.

Acknowledgements

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

UltraEdit is a registered trademark of IDM Computer Solutions, Inc.

References

Lund, Pete A Macro for Generating Formatted Log Comments, *Proceedings of the Twenty-Fifth Annual SAS Users Group International Conference*, Cary, NC: SAS Institute Inc., 2000.

Lund, Pete A Quick and Easy Data Dictionary Macro, *Proceedings of the Twenty-Seventh Annual SAS Users Group International Conference*, Cary, NC: SAS Institute Inc., 2002.

Lund, Pete Keep Those Formats Rolling: A Macro to Manage the FMTSEARCH= Option, *Proceedings of the Twenty-Eighth Annual SAS Users Group International Conference*, Cary, NC: SAS Institute Inc., 2003.

Note: Past SUGI proceedings are available in PDF format on the web at:

www.sas.com/usergroups/sugi/proceedings/index.html

Author Contact Information

Pete Lund
Looking Glass Analytics
215 Legion Way SW
Olympia, WA 98501
(360) 528-8970 voice
(360) 570-7533 fax
pete.lund@lgan.com
www.lgan.com

All code referenced in this paper is available in electronic format upon request.