Paper 237-31

The Personal Touch: Control Your Environment as a SAS® User

Peter Crawford, Crawford Software Consultancy Limited

ABSTRACT

Most software works for you "straight out of the box." Equally, most software feels a bit bland when used that way. Just as we are all different, we work and respond, better when the environment is adjusted to suit the way we want to work. I find that an important part of this is knowing what in our environment we can control and how we can control it. This tutorial introduces the main opportunities in SAS client environments for giving them the "personal touch".

INTRODUCTION AND SCOPE

This tutorial describes customization of your SAS IDE (integrated development environment) under headings: SAS Display Manager (basic, gui, user-defined), SAS Enterprise Guide® (briefly), other clients, and servers

So far I haven't enough experience on SAS Enterprise Guide to adequately cover the need/wish/opportunity for customization of that interface. (Hopefully I might soon). However, I have extensive exposure to SAS Display Manager on z/OS, Unixand Windows - over "more than 10 years".

The rate of use of clients on z/OS and UNIXis lower than on Windows. However, these clients each add extra layers for customization: - like session-type and keyboard mapping in 3270-terminal emulation, and through Xresources on uUnix. Rather than here, or inside your SAS session, these layers for customization "outside SAS" should be an area for your client support group. You may find useful discussion and guidance in the SAS® 9.1 Companion for Unixand SAS® 9.1.3 Companion for z/OS.

The windows platform prevails. It has been a long time since I was offered a Unixworkstation, even longer since a real 3270 terminal. It's three years since I last used SAS through a mainframe terminal emulator. So I will limit this tutorial to customizing on Windows.

IDE

We want to work in an integrated development environment (IDE), with all we need within easy reach.

Not all we need can be controlled by SAS - some must be controlled by the operating system.

Both Unixand windows use environment variables to simplify relationship with applications. SAS uses these too - and more. There are many environment variables used by SAS. Fortunately, the handling is almost the same in both Unixand windows environments. A short SAS program works in both to collect a table of the names and values:

```
DATA evars( KEEP= name value  COMPRESS= yes ) ;
      LENGTH name $40  value  $2000 ;
FILENAME     envs PIPE 'set ' LRECL= 2000;
      INFILE envs TRUNCOVER DSD DLM= '=' COL= col  ;
      INPUT  name;
      IF     name ne ' ' ;
      value = SUBSTR( _infile_ || ' ', col ) ;
    RUN;
FILENAME envs CLEAR;
```

One distinction is that handling of environment variable names on Unixplatforms is case-sensitive.

SASROOT is a very common example. We can create, customize and use many more.

The traditional SAS client user interface is SAS Display Manager. It is not a modern IDE. On the plus side it has enjoyed two decades of improvements by SAS Institute product developers. So it is a hard act to follow.

It is even harder to break that tradition, to create designs and environments never considered when SAS Display Manager's internal architecture and interface were determined. Now SAS® Enterprise Guide® seems to be a strategic direction that is replacing SAS Display Manager as the new SAS programmers' interface – *the* modern SAS IDE.

Unlike new and non-SAS clients, the rich history that developed SAS Display Manager has provided strong integration of client layers with the core of a SAS session. As a result, SAS Display Manager provides a rich access to SAS data and programs. It also has many ways to customize its appearance and ways to save that customization.

OTHER SAS CLIENTS

Other (perhaps less-obvious) SAS clients can be considered. SAS/Connect® is one example. You can customize that session in several places, but as it has no visual component, I do not cover it in this tutorial.

This tutorial could also point out customization of SAS as a server. There are many facilities available that simplify administration – and enhance the "out-of-the-box" state. All of these features are documented in on-line help and various SAS publications.

(But a tutorial is probably the best introduction)

.

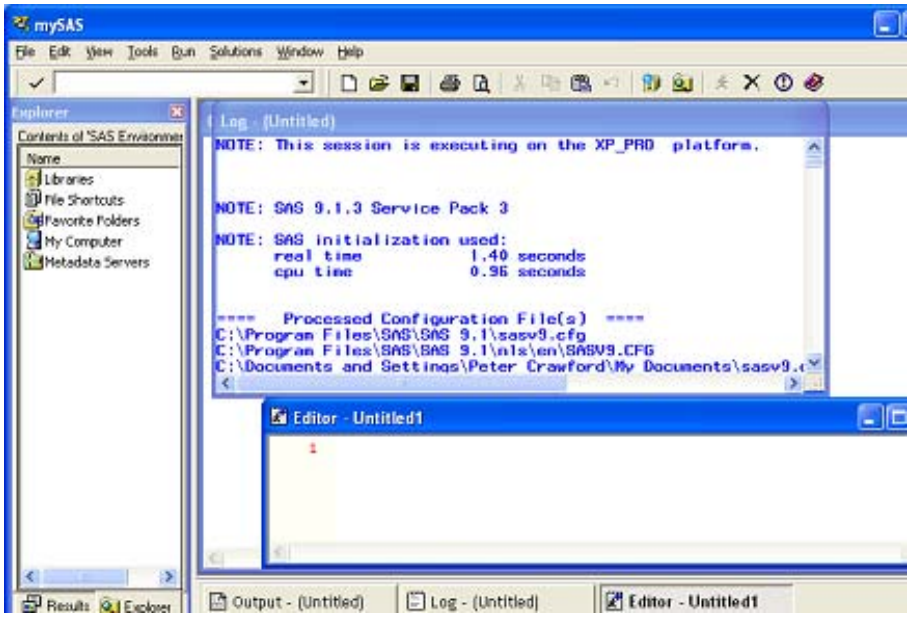
USER INTERFACES

What follows is a walk through some facilities for making customizations – first in SAS Display Manager. Later sections review how to save and restore these customizations if/when they are lost, or you wish to apply them where your normal session is not available. A final section refers to non-interface customization.

CLASSIC PGM/LOG/LISTING

Size and position can be defined (drag the edges) and many features can have particular colors chosen (on the menu in the classic window, navigate to /tools/options/colors and choose your preferences.

There are command line commands but the dialog is more convenient.

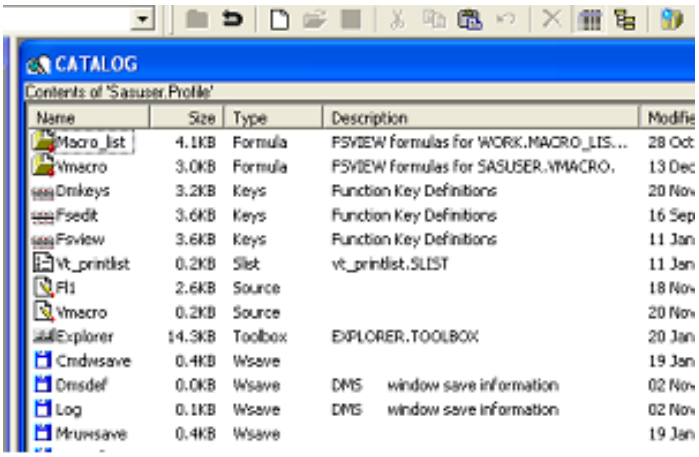


then, through the menus (tools/ options/ preferences - save settings on exit and close).

Alternatively, issue command "WSAVE ALL".

Another feature of the classic SAS Display Manager windows that this will save, is whether a command line is shown. I prefer to use the command bar.

The color, sizes and positions of the classic windows, and general preferences, are saved in the catalog sasuser.profile as wsave-type entries.



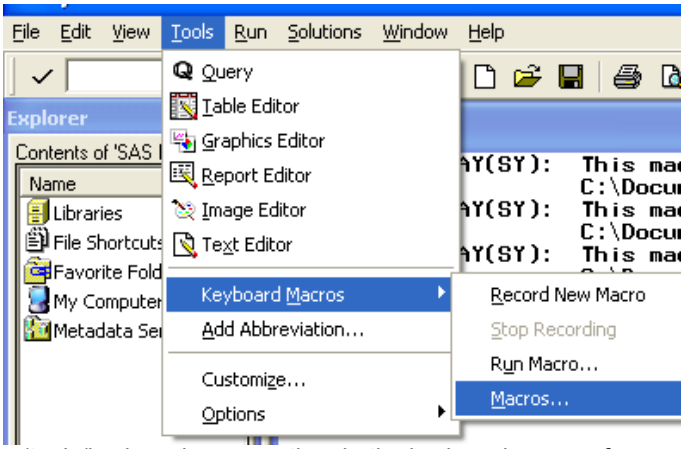
Before we finish with classic windows, remember that these are not the only windows where you can customize/define commands on function keys. Keys F1-F12 in combinations with shift, cntl and alt. With the alt- and cntl shift keys, many alpha-character keys can be used as function keys too. Keys can be defined in the classic way for any window of a SAS

Display Manager session. As demonstrated here, function key definitions are also saved in the sasuser profile. The toolbox (or toolbar) supports these classic windows too. Generally, any toolbox customization will be saved in the sasuser.profile.

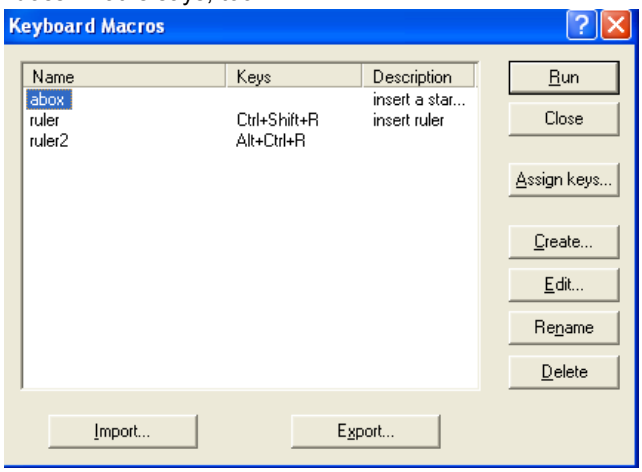
GUI

The windows of SAS Display Manager I refer to as GUI, are The Enhanced Editor and the SAS Explorer. As GUI interfaces, they customize in different ways from classic SAS windows.
For Enhanced Editor windows:

- Size and position vary because many can be open at the same time.
- Styles and colors for code or text being edited are controllable and many schemes can be saved. Perhaps you would review code with different emphasis on comments. Just pick your preferred color scheme.
- The keyboard commands available greatly exceed those allocated to keys as supplied "out-of-the-box". I particularly like being able to sort the lines of text in a selection – just customize the keys!
- Among the many excellent papers on the SAS website see how you can build a template with all the options for procedure like PROC SUMMARY at http://support.sas.com/sassamples/papers/0303_saseditor.pdf
- Enhanced editor macros, abbreviations and shortcut keys are stored in the windows registry, but can also be backed-up and shared by saving them to external files. Via the enhanced editor menu select



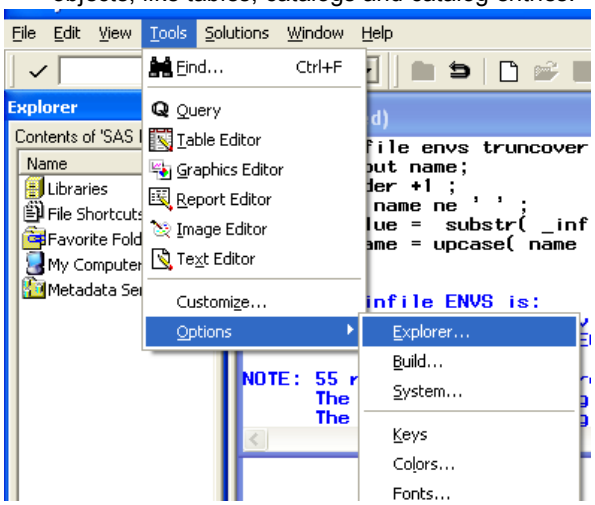
/tools/keyboard macros, then in the keyboard macros frame, select the item to save, and click on the "export" button. The corresponding "import" button does what is says, too.

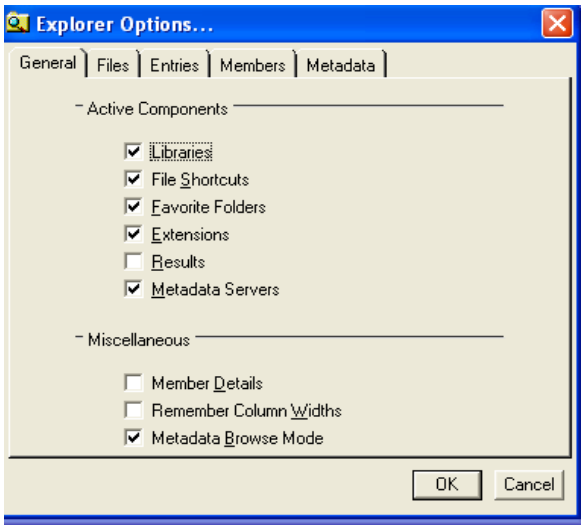


- Most settings for enhanced editor are saved outside the SAS environment, in the user's windows profile/registry.

SAS EXPLORER

- There are two forms of the SAS Explorer window, docked, and undocked.
- More than one (undocked) SAS Explorer can be open at the same time. Saving window size and position seem less important as more than one may be open.
- However, SAS Explorer provides a rich support for customizing behaviour you might want on the SAS objects, like tables, catalogs and catalog entries.



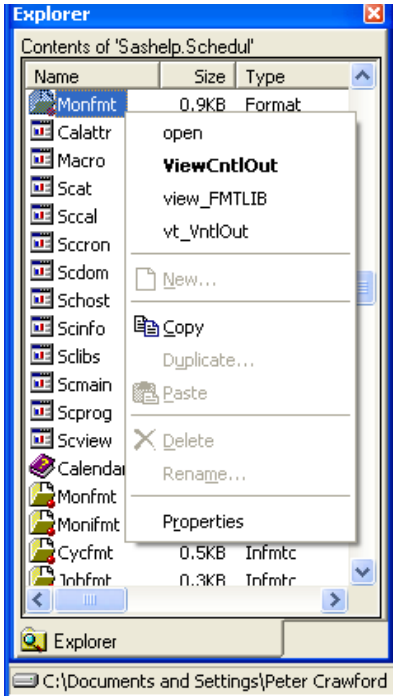


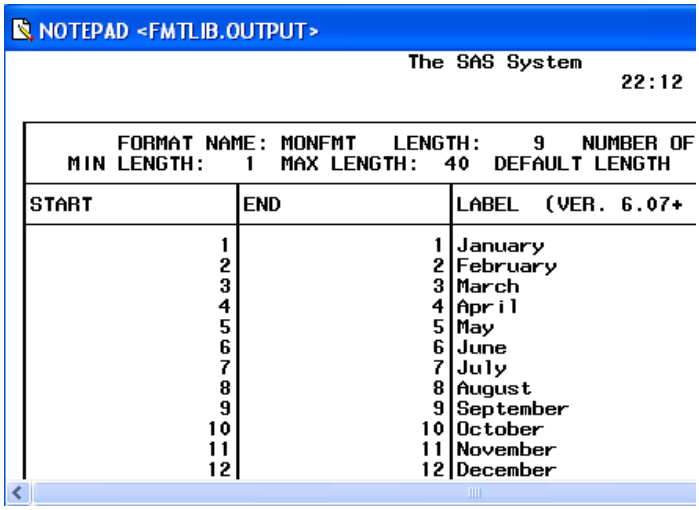
- These customizations are stored in the SAS registry, which can be exported from and imported to, with ease.

```
PROC REGISTRY EXPORT= "my_reg_&sysdate9..txt"; RUN;
PROC   FSLIST   FILE= "my_reg_&sysdate9..txt"; RUN; *to review;
```

- Because these exports are plain text in external files, as well as reviewing them with PROC FSLIST code like above, they can be copied, and shared. Individual behaviors can be imported. A collection of these may be found on Richard DeVenezia's website at http://www.devenezia.com/#sas and follow the link for actions.

- So if you like the way my SAS Explorer behaves when I click on a format, in an unusual formats catalog, you can import just that behaviour (see Appendix), while picking other behaviour from separate sources.





- For a tip on SAS Explorer behaviour on the SAS Samples web site, see: http://support.sas.com/sassamples/quicktips/05mar/viewtable.html

VIEWTABLE

This provides a sophisticated interface to data tables (and mddb). The layouts support a lot of customization; unfortunately saving these customizations seems almost impossible.

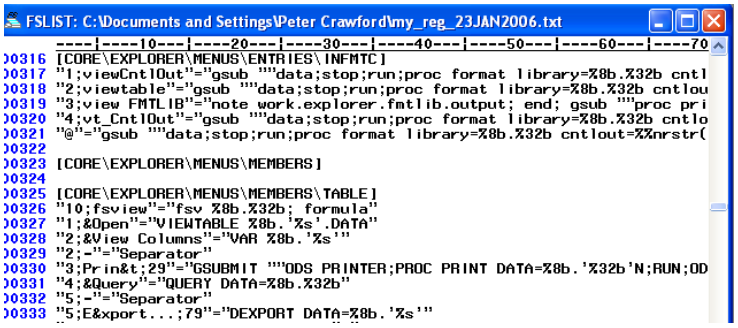
OTHER USER INTERFACES IN SAS DISPLAY MANAGER

V6LIB, V6DIR, V6VAR, V6CAT

In sas9 there are windows just like the old windows on SAS6 - V6lib, v6dir, v6var, V6Cat
Now, I can truly say, I prefer SAS Explorer (it took all the time of v8).

FSLIST (PART OF BASE SAS, SINCE SAS8)

The procedure and window FSLIST provide a read-only view of an external file. There isn't much customization. As with other old windows of SAS Display Manager, colors can be redefined: foreground, background, marked text and numbers. The feature I find convenient to customize is a function key to issue command cols on; nums on. It also forms part of my habit of reviewing the SAS



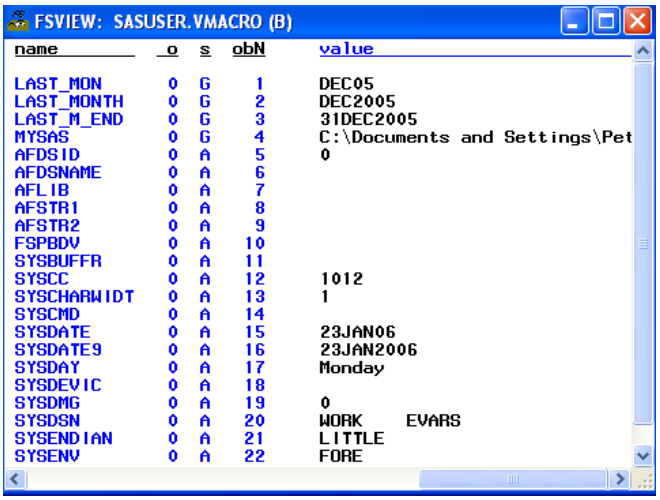
registry after exporting it for backup.

SAS/AF

The application development environment needs SAS Display Manager to deliver its user interface. Fully customizable by application developers, any customization is only under control of that app.

SAS/FSP FSBROWSE (FSEDIT), FSVIEW

These pre-gui applications provide very effective user-customizable views of data as tables or forms. The customization is stored in SAS catalogs. By default FSVIEW stores layouts in sasuser.profile, and FSBROWSE doesn't store customization.



The invocation of FSBROWSE allows you to define a location for storing customization. Unlike VIEWTABLE, the customization of FSVIEW and FSBROWSE can be saved and reused.

When integrated with SAS Explorer, this customization is easily built-in and becomes personal and transparent. Here is an example that uses a customized FSVIEW layout to provide a quick view of the current settings of all the automatic macro variables.

I think the form view that FSBROWSE supports is particularly useful for data sets and tables like data marts with many columns, because it provides more than one way to arrange the information, without needing separate copies of the mart. I would have liked to demonstrate with a recent client data mart that holds around 200 columns. No one organization of the columns will satisfy all users, but each user can organize their personal layout preference(s)

JAVA

(Sadly, outside of my experience inside SAS).

As yet I have seen no need nor scope for customizing the layouts within the SAS Management Console.

SERVER SIDE

The following sections are relevant also on SAS Display Manager clients.

ENVIRONMENT VARIABLES

Add a lot of environment variables and use them as needed as librefs or even filerefs. .

Unless you want to add engine information, just use the environment variable as a SAS library. The name needs to conform to name conventions for librefs. On Unix they also have to be in uppercase. The traditional alternative - LIBNAME statements in the autoexec - can grow into overheads we don't need. Once a librefis known to the SAS environment, it slows down the SAS Explorer, perhaps just a little. Imagine 300+ libname statements invoked by the autoexec. It makes the use of dictionary tables and columns far slower than necessary. Environment variables would only cause this kind of impact once they are used as librefs. The example I found with more than 300 LIBNAME statements in the autoexec, seldom used more than 15 in one SAS job. I was unable to find anyone who new a routine that used more than a few. It would always take noticeable time to start a session with so many LIBNAME statements. It took only fractions of a second to start a session using the 300+ corresponding environment variables, prepared and ready to be librefs.

SASAUTOS

The facility immediately to use macros (prepared earlier) allows us as user, more quickly to continue to use our user interfaces. So I think SASAUTOS deserves special mention here. If you use macro libraries that you want to share, consider supporting applications, by establishing -SASAUTOS in its earliest form as

```
-SASAUTOS ( sasautos )
```

Then application configs can append the path(s) to search for macro, with a line in the config file, like:

```
-APPEND SASAUTOS 'mymacros'
```

or, with a line in the config file like:

```
-INSERT SASAUTOS 'path to test macros'
```

we can insert the path(s) to macro libraries (providing precedence) to additional paths for application specific and project specific and test specific macro pools

THE SEARCH FOR _ALL_ THE CONFIGS

When starting up, a SAS session seeks its environment definition from config files in several ways. If a command line defines `-config` then these are used. Otherwise the platform is searched for any files of the name sasv9.cfg (that 9 is the major number in the SAS version) in 3 areas:
!sasroot (or the path of the SAS executable)
userprofile /home in uUnix, %userprofile% on windows
current folder
As well as these, the executables investigate environment variables
SAS_SYS_CONFIG
USER_SYS_CONFIG
which are presumed to hold the path/filename of a config file
Finally, environment variable
SAS_OPTIONS
is examined assuming it is a string of SAS options, and when it contains (one or more) -config settings, these will be used too.

KNOWING WHAT IS GOING ON

There is merit in brevity, but that probably doesn't help diagnosis of any problems. To discover the invocation-stage settings of system options, add option (valid at startup only)

```
-verbose
```

Chief among the information provided, is a list of the configuration files used. It seems to be the only way to obtain this information.
Once a session is started, PROC OPTIONS can reveal lots of information. To display the definition and values of options at "environment setup", try:

```
PROC OPTIONS GROUP= ENVFILES DEFINE VALUE;
RUN;
```

KEEPING TRACK

```
-ALTLOG
```

This (invocation) option defines a file into which all log lines are copied even when PROC PRINTTO is redirecting the log. To make this more useful, you want a different altlog file for each session. Either the calling routine defines this on the command line, or the value needs to change in a config file. I use the latter technique, and SAS-L has seen my %upalts() macro which performs this update.
.

SECURING CUSTOMIZATION

This paper has drawn attention to the following areas of customization: Classic SAS Display Manager Windows, Enhanced Editor Editing Schemes, Enhanced Editor Macros And Abbreviations, SAS Explorer behaviour and SAS/FSP windows FSBROWSE and FSVIEW. The rest of this section provides commentary and code or procedures for back up and restore of their customization.

CLASSIC SAS DISPLAY MANAGER WINDOWS

These save your customization in sasuser.profile. That can be copied, or more "cleanly" PROC CPORT can export most of the entry types from this catalog. Use PROC CATALOG to secure the rest, like:

```
PROC CPORT CATALOG= sasuser.profile FILE= "bu_prof_&sysdate9..stc" DATECOPY ;
RUN;
```

Use PROC CATALOG to secure the rest, like:

```
PROC CATALOG; /* copy entries that cannot be cport-ed */
 Libname hs '<somewhere safe>' ;
  COPY IN= sasuser.profile OUT= hs.profsaves_&sysdate9 ET= wsave   ;
  COPY IN= sasuser.profile OUT= hs.profsaves_&sysdate9 ET= pgsetup ;
 RUN;
 LIBNAME hs CLEAR /* try not to use it by accident */ ;
QUIT;
```

When you need to reinstall your environment and preferences, these copied files will help. You may want to reinstate directly with something like

```
%LET savedate =  <whenever that was, in ddmmmyyyy format>;
PROC CIMPORT CATALOG= sasuser.profile FILE= "bu_prof_&savedate9..stc" ;
RUN;
PROC CATALOG; /* copy entries that cannot be cport-ed */
 LIBNAME hs '<that safe folder>' ;
  COPY OUT= sasuser.profile IN= hs.profsaves_&savedate;
 RUN;
 LIBNAME hs CLEAR ;
QUIT;
```

I hesitate to write directly into sasuser.profile like that and usually import to work.profile:

```
PROC CIMPORT LIB= work FILE="bu_prof_&savedate9..stc" ;
RUN;
```

After that I use the SAS Explorer to select and reinstate the entries needed.

ENHANCED EDITOR EDITING SCHEMES

The editing schemes are kept in the windows registry. Securing that is a bit outside my SAS-level-of-confidence. However, I have needed so little customization, that I'm happy to re-apply that and avoid interfering with the windows registry.

ENHANCED EDITOR MACROS AND ABBREVIATIONS

These can be exported and imported through the dialog referred earlier

SAS EXPLORER BEHAVIOUR

Saving these settings is so simple, it provides the best example:

```
PROC REGISTRY EXPORT= "my_reg_&sysdate9..txt"; RUN;
```

These export files are plain text, and sections or nodes can be re-used on their own.

Reinstating SAS Explorer behaviour, or installing new behaviour copied from other users, is just as simple as the export to back up. The settings can be the whole set, or only selected sections. For example, you could reduce the content to just the node for table member behaviour which starts

```
[CORE\SAS EXPLORER\MENUS\MEMBERS]
[CORE\SAS EXPLORER\MENUS\MEMBERS\TABLE]
```

Here is the SAS code for importing a set of registry nodes.

```
PROC REGISTRY IMPORT= "<name of file containing stuff to reinstate";
RUN;
```

SAS/FSP FSBROWSE (FSEDIT), FSVIEW

These windows integrate easily with the SAS Explorer behaviour. The pseudo command strings I use are:

```
FSVIEW   %8b.%32b; FORMULA
FSBROWSE %8b.%32b sasuser.%8b.%32b.screen
```

The `%8b.%32b` will be filled by SAS Explorer with <libname>.<memname> for the selected data set

The FSVIEW table layout customizations are stored in the sasuser.profile catalog as entries of type formula. Each has the member name as the entry name in the catalog. These will be backed up or reinstated as sasuser.profile is backed-up for the classic windows of SAS Display Manager.

To keep the form-style FSBROWSE layouts (catalog entries of type: screen) my naming convention for a SAS data set like <lib ref>.<memname> is sasuser.<libref>.<memname>.screen. Therefore to backup these layouts, I use code:

```
PROC CPORT LIB= sasuser MT= catalog ET= screen     DATECOPY
          FILE= "<somewhere safe>\bu_my_scrs_&sysdate9..stc"   ;
RUN;
```

To reinstate these screen data sets, the code is:

```
PROC CIMPORT LIB= sasuser FILE= "<somewhere safe>\bu_my_scrs_&savedate..stc"   ;
RUN;
```

CONCLUSION

I believe you should expect, easily, to be able to save and reinstate any customization the IDE offers. Not all is necessary and in SAS, not all is possible. So it seems a good idea to examine the range and areas of the interface that do allow user customization to persist across sessions. My objective for this tutorial was to demonstrate the substantial core of the SAS IDE that achieves this objective.

Of course all the code here should be validated in your own environments to ensure that it achieves what you hope for.

ACKNOWLEDGMENTS

Support and help from Pete Lund is very much appreciated. Richard DeVenezia has produced a very useful website and generously allowed me to reference his section on customizing SAS Explorer actions.

RECOMMENDED READING

Locate the online help for SAS Explorer customization through the help buttons on those frames.
There is also an extensive page in SAS online Doc page on customizing your SAS Session at
http://support.sas.com/onlinedoc/913/getDoc/en/hostwin.hlp/customizing.htm
I hope this provides more information on saving and restoring these customizations.

CONTACT INFORMATION

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

Peter Crawford
Crawford Software Consultancy Limited
31 Sefton Road,
Croydon, Surrey, CR0 7HS, UK
+44 7802732254
CrawfordSoftware@gmail.com

APPENDIX

EXPLORER BEHAVIOUR FORMATS, INFORMATS

The following source code and the small text provide the behaviour that allows the SAS Explorer to drill down into format entries.
The text that appears below in very small font size, should be saved to a text file.
If you named that text file 'crawfords.format.behaviour.txt', then the following code would import it. You may need to take care to ensure that the folder where it is stored, is the current folder for your SAS session.

```
proc registry import= 'crawfords.format.behaviour.txt' ;
run;
```

The contents of that text file should be

```
CORE\EXPLORER\MENUS\ENTRIES]

[CORE\EXPLORER\MENUS\ENTRIES\FORMAT]
"11;open"="postmessage 'format entries like %8b.%32b.%32b.%8b need code submitted'"
"12;viewCntlOut"="gsub ""data;stop;run;proc format library=%8b.%32b cntlout=%%nrstr(&syslast); select %32b ;run;quit;""; fsv; formula cntlout"
"13;view_FMTLIB"="notepad work.explorer.fmtlib.output;end;gsub ""proc printto file=work.explorer.fmtlib.output new; run;proc format fmtlib library=%8b.%32b; select %32b; run;proc printto;run;""; notepad work.explorer.fmtlib.output"
"14;vt_VntlOut"="gsub ""data;stop;run;proc format library=%8b.%32b cntlout=%%nrstr(&syslast); select %32b ;run;quit;""; vt _last_"
"@"="postmessage 'format entries like %8b.%32b.%32b.%8b need code submitted'"

[CORE\EXPLORER\MENUS\ENTRIES\FORMATC]
"11;open"="postmessage 'format entries like %8b.%32b.%32b.%8b need code submitted'"
"12;viewCntlOut"="gsub ""data;stop;run;proc format library=%8b.%32b cntlout=%%nrstr(&syslast); select $%32b ;run;quit;""; fsv; formula cntlout"
"13;vt_CntlOut"="gsub ""data;stop;run;proc format library=%8b.%32b cntlout=%%nrstr(&syslast); select $%32b ;run;quit;""; vt _last_"
"14;view_fmtlib_print"="notepad work.explorer.fmtlib.output;end;gsub ""proc printto file=work.explorer.fmtlib.output new; run;proc format fmtlib library=%8b.%32b; select $%32b; run;proc printto;run;""; notepad
work.explorer.fmtlib.output"
"@"="postmessage 'format entries like %8b.%32b.%32b.%8b need code submitted'"

[CORE\EXPLORER\MENUS\ENTRIES\INFMT]
"11;open"="postmessage 'format entries like %8b.%32b.%32b.%8b need code submitted'"
"12;viewCntlOut"="gsub ""data;stop;run;proc format library=%8b.%32b cntlout=%%nrstr(&syslast); select @%32b ;run;quit;""; fsv; formula cntlout"
"13;view FMTLIB"="notepad work.explorer.fmtlib.output;end;gsub ""proc printto file=work.explorer.fmtlib.output new; run;proc format fmtlib library=%8b.%32b; select @%32b; run;proc printto;run;""; notepad work.explorer.fmtlib.output"
"14;vt_CntlOut"="gsub ""data;stop;run;proc format library=%8b.%32b cntlout=%%nrstr(&syslast); select @%32b ;run;quit;""; vt _last_"
"@"="postmessage 'format entries like %8b.%32b.%32b.%8b need code submitted'"

[CORE\EXPLORER\MENUS\ENTRIES\INFMTC]
"11;open"="postmessage 'format entries like %8b.%32b.%32b.%8b need code submitted'"
"12;viewCntlOut"="gsub ""data;stop;run;proc format library=%8b.%32b cntlout=%%nrstr(&syslast); select @$%32b ;run;quit;""; fsv; formula cntlout"
"13;view FMTLIB"="notepad work.explorer.fmtlib.output;end;gsub ""proc printto file=work.explorer.fmtlib.output new; run;proc format fmtlib library=%8b.%32b; select @$%32b; run;proc printto;run;""; notepad work.explorer.fmtlib.output"
"14;vt_CntlOut"="gsub ""data;stop;run;proc format library=%8b.%32b cntlout=%%nrstr(&syslast); select @$%32b ;run;quit;""; vt _last_"
"@"="postmessage 'format entries like %8b.%32b.%32b.%8b need code submitted'"
```

That text was created with `PROC REGISTRY EXPORT=` .