Paper AD01

An Introduction to SAS/SHARE®, By Example

Larry Altmayer, U.S. Census Bureau, Washington, DC

ABSTRACT

SAS/SHARE software is a useful tool for allowing several users to access and edit the same data set at the same time in their individual SAS® sessions. Without this software, data sets are only available for simultaneous use if the data set is set to browse mode in the properties window of a SAS frame application. With the software, if the data set properties are set to an edit mode, different users can edit different rows of the same table at the same time.

This paper gives an introduction to using SAS/SHARE to access and edit a variation of the HOUSES SAS data set in a small SAS/AF® application to be used by several users at the same time. We demonstrate how to start and stop a SAS/SHARE server, and assign a libname to the server. We also show how the application works while different users are simultaneously editing the data. From the software viewpoint, we touch on the requirement for having the server name in the /etc/services file. Finally, we look at SAS/SHARE macros as a way to condense code.

THE APPLICATION

The application we use is a SAS/AF screen containing a table viewer control and SAS data set model. It also contains a few supplemental screen objects, including graphic text and desktop icon controls. The screen represents an application used by realtors at the ABC Real Estate Company to enter bids clients offer for houses for sale. Thus, the data set seen in the table viewer is the HOUSES data set[1], with a numeric field, BID, added at the end of the observation. The screen is shown in Figure 1 below.

Figure 1

The main points of interest in the application itself are the data set attributes which allow the users to edit its observations, and the fact that all of the screen properties are specified in the properties window. Thus, there is no screen control language (SCL) behind the frame. We describe how we set the screen properties, including data set attributes allowing user edits, in the following sections.

SPECIFYING DATA SET ATTRIBUTES TO ALLOW USER EDIT

The above application contains a SAS data set model in a table viewer control. The default value of the data set model data attribute editMode is 'Browse', which does not allow the user to edit the observations of the data set. If we change this to 'RowLevelEdit' in the properties window, as shown in Figure 2 below, the user can now edit observations.

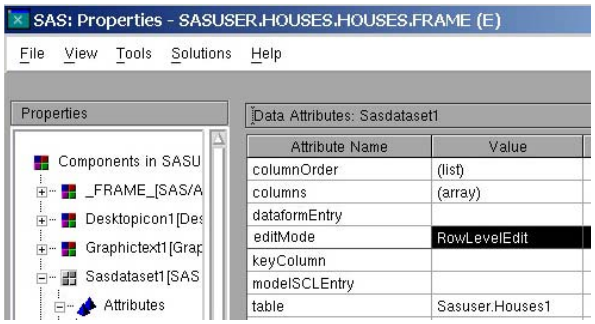

Figure 2

SPECIFYING SCREEN ATTRIBUTES IN THE PROPERTIES WINDOW

In order to avoid the use of SCL behind the frame, we specify all screen attributes in the properties window. In this paper, we describe how to change the default data set fonts to those seen here. We also show how to remove the command prompt from the frame, as well as the initial message the user sees on running the frame, indicating that the frame does not have a program. Details on other properties, such as size and location of screen objects, and those behind the graphic text and desktop icon controls, will be provided by the author upon request.

Changing the Default Data Set Fonts

The default font for the data set row and column labels, and cell entries may vary from system to system. On my system, it is a form of adobe-helvetica, point size 12, weight medium. In this paper, I have changed the fonts to point size 24, weight bold.

We change the font in two steps: first, those for the row labels; and second, those for the column labels and cell entries. In the first step, we use the properties window for the table viewer control. We change the font for labelFont under appearance attributes as shown in Figure 3 below.

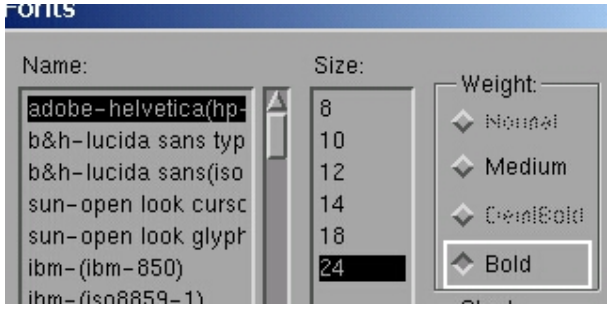

Figure 3

In the second step, we change the column labels and cell entries to this font, using the columns attribute under data attributes in the properties window for the SAS Data Set Model. We change both the **label** and **data** fonts, after highlighting all variable names first. This is illustrated in figure 4 below.

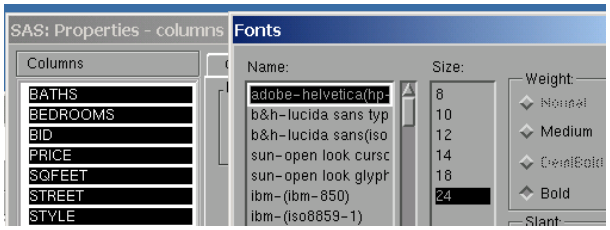

Figure 4

Removing the Command Prompt and Initial Message

Removing the command prompt and initial screen message immediately below it are done in a similar way to the changes we have made previously. To remove the command prompt, we specify bannerType=NONE under FRAME

appearance attributes, as shown in figure 5.

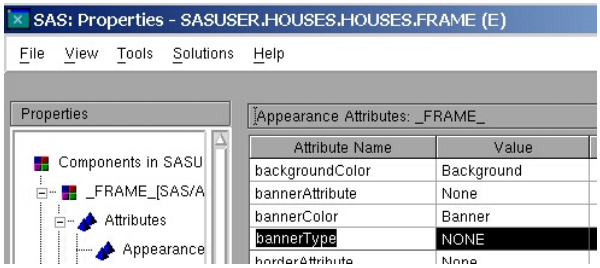

Figure 5

Removing the initial screen message immediately below the command prompt, indicating the frame does not have a program, is done by specifying a null value under the SCLEntry attribute under FRAME Source Attributes in the properties window, as shown in figure 6.

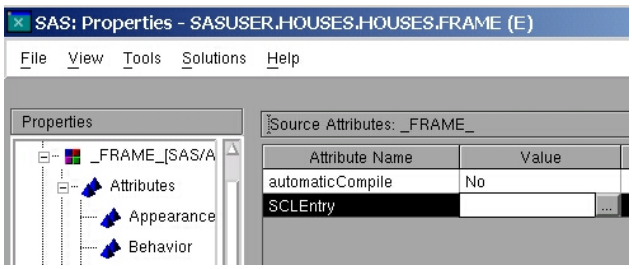

Figure 6

USING THE APPLICATION WITHOUT SAS/SHARE

There are several ways to run the application. In this paper, we assume we are running it on a UNIX® platform, although the principles described here apply to any plaform. We can run it either from the UNIX prompt or using a program in the display manager. We run it using programs submitted from the display manager. Assume we have the application located in the following frame entry: sasuser.houses.houses.frame. Then, the following code runs the program from the display manager.

```
dm 'af c=sasuser.houses.houses.frame' continue;
```

Figure 7

The continue option allows us to run the application without the program editor window coming up over the application to begin with.

If we run the application using the above program, the screen shown in figure 1 appears. As mentioned above in the section on specifying data set attributes, the default SAS data set model data attribute editMode is 'Browse', which allows us only to browse the data. When using this mode, several users can browse the data at the same time, but as the name indicates, no one can edit it.

If we change the editMode attribute to 'RowLevelEdit' as shown in figure 2, and in addition, run the application in a second session, we get a screen with a blank table viewer in the second session. Furthermore, we get a message similar to the following in the log:

ERROR: A lock is not available for SASUSER.HOUSES1.DATA,
lock held by process 13887.

Why is this? Without SAS/SHARE, only one user can edit a data set at a time. We need SAS/SHARE to allow multiple users to edit different observations of the data set at the same time. In the following sections, we illustrate how SAS/SHARE allows us to do this, and give an overview of the basics of the software.

RUNNING THE APPLICATION WITH SAS/SHARE

In order to run the application with SAS/SHARE, we must first start a SAS/SHARE server, either in a separate session, or the same one, where the user will be accessing the data. Unless we run it in the background, we cannot do anything else while the server is running. Therefore, here, we run it in the background.

```
data _null_;
rc=system('sas82 start_server.sas &');
run;
data _null_;
  x=sleep(5,1);
run;
```

Figure 8

The above code runs a program, start_server.sas, which starts the server in the background, as indicated by the '&'. The system function provides the equivalent of running the sas82 command at the UNIX prompt. The sleep function in the second null data step suspends the running of the data step, to give the server time to start. We discuss the contents of the start_server program in the next section, where we describe the basics of SAS/SHARE.

Now that the server is started, we can run the application as we do above in a single session, using the program code from figure 1. The application comes up, and we click on a cell as if we are about to make a change in the observation. This implicitly locks the row, as shown in figure 9.

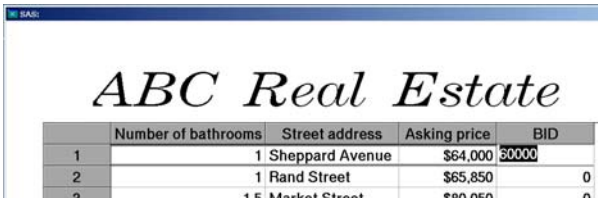

Figure 9

Next, we start another SAS session, and also run the application in this session. Again, we click on an observation (other than the observation being used in the first session), and we can make a change if we want. If we click on the same observation as that being accessed in the first session, however, we receive a warning that we cannot do this, because the observation is being used in the first session. This is illustrated on the screen by figure 10.

Figure 10

SAS/SHARE BASICS

Using SAS/SHARE software to enable multiple users to simultaneously access data requires just a couple basic steps. First, we must take into account the communications access method (CAM) for interfacing between SAS software and the network protocol for connecting between platforms. Next, we use BASE and SAS/SHARE SAS program code to specify the communications access method, assign relevant libnames and start and stop the server. We describe how to do this in this section.

COMMUNICATIONS ACCESS METHODS

There are several types of CAMs. The one we use is based on the client and server host operating system involved. If the server host operating system is UNIX, we must use the TCP/IP method. To do this, we must configure the server in the SERVICES file. This is as simple as having an entry (line) for the server in this file. In UNIX, this file is located in **/etc/services**. If the name for our server is server1, this line would look similar to the following:

```
server1      5012/tcp        # SAS/SHARE server 1
```

Figure 11

In the above figure, the first field is the service or server name, the second, the port number/protocol name, and the third field is an optional comment describing the service.

BASE AND SAS/SHARE PROGRAM CODE

After we have taken the CAM for our host and client server platforms into account, we must specify this using the SAS program code to stop and start the server. This involves specifying the type of access method in an OPTIONS statement, specifying a libref for the physical SAS data library being accessed with the server, and using the SAS/SHARE procedures SERVER and OPERATE to start and stop the server, respectively. We describe how to do these things in this section.

The OPTIONS and LIBNAME statements indicate the type of access method, and SAS data library, and possibly the server we are using. An OPTIONS statement is used to specify the type of access method we are using to connect between platforms. We use the COMAMID= option, with TCP as the option type. Next, we use a LIBNAME statement to associate a libref with the physical SAS data library we are connecting to, as the remote host. If we are connecting to the server in the same session in which we start the server, we do not need a SERVER= option on this LIBNAME statement. If we are connecting to the server in a **different** session from which we start the server, however, we need this option, so the system 'knows' to connect to the server in another session. If the system we use sets up default librefs at initialization, we may also need to clear these beforehand, to allow the SHARE server to work correctly. This is done with another LIBNAME statement with the arguments _all_ (for all libnames) and clear. The OPTIONS and LIBNAME statements for a server running in the present session are illustrated in figure 12.

```
OPTIONS comamid=tcp nostimer;
LIBNAME sasuser1 '/sasuser';
```

Figure 12

To start the server, we use the SAS/SHARE procedure, PROC SERVER. For our example, the code for running this procedure is given in figure 13.

```
proc server id=server1
  authenticate=opt
  alloc log=message;
run;
```

Figure 13

In this code, the ID= option specifies the server name given in the previously mentioned /etc/services file. The AUTHENTICATE= option is for server security. It indicates that the server does not require the user to specify a password to connect to the server. The default value for this option is REQUIRED, so we need to specify this option as above, for unrestricted user access. The ALLOC option allows the user to specify the user to define additional SAS libraries to a server after it has started. We do have to specify this option, as the default value for this option is ALLOC. Finally, the LOG= option tells the server to indicate the number of messages which are exchanged with each client. More details on this option will be given in the section on server log information.

We stop the server by using the SAS/SHARE procedure, PROC OPERATE, as shown in the following code.

```
proc operate;
  stop server server1;
quit;
```

Figure 14

To stop the server using this procedure, we use the server management statement STOP with the serverid of the server we are using, as given in the /etc/services file. This procedure can also be used for other types of server control and give information about the server. More details on this are given in the SAS OnlineDoc for version 8.

Finally, if we are in a subsequent session (the server having been started in a prior session), we must only issue the OPTIONS and relevant LIBNAME statements. We do not issue the PROC SERVER. As mentioned above, in this case, the LIBNAME statement must have a SERVER= option, so the libref defined in this session points to the physical SAS data library associated with the server. This is illustrated in the following code.

```
LIBNAME sasuser1 '/sasuser' SERVER=server1;
```

Figure 15

USING UNIX TO VERIFY A RUNNING SERVER

Once we have started the server, we can issue UNIX commands to verify the server is running, provided we have started the server in a background process. This can be done with the command shown below.

```
ps -f|grep start_
```

Figure 16

This is the familiar 'ps' command, which indicates the processes the user has running. The –f option indicates that only the user's own processes should be given. The grep command searches for the following argument (here, 'start_') in program file names. Thus, we assume here that the program to start the server begins with the characters 'start_'. The name may also contain additional characters.

When we issue the above command, we get the following output.

```
user1  24754     1  0.0 09:29:38 ttyp9   0:00.31   sas82 start_server.sas
```

Figure 17

The first field indicates the process belongs to the current user, and the last field shows the subprocess containing the running server.

SERVER LOG INFORMATION

The server log, the SAS/SHARE version of the usual SAS log, gives several pieces of information involving the performance of the server. It is located in the directory the system sends logs to by default, unless we specify otherwise in the sas command we use to call the program. The complete log is available for viewing after we have issued PROC OPERATE to shut the server off.

The log contains such information as when:

- A server starts
- A user session connects to a server
- A server library libref is accessed by a user session
- A data set is opened for input by a user
- An attempt is made to change an observation in use in another session
- A data set is closed
- A user disconnects from the server
- A server is shut off
- Messages are processed, via usage statistics for the server and user sessions

When the server starts, and after the first few of the above operations, messages similar to the following appear in the SAS log.

```
04AUG2003:09:18:08.281 SAS SERVER SERVER1 STARTED.
04AUG2003:09:18:20.114 USER USER1(1) HAS CONNECTED TO SERVER SERVER1.
04AUG2003:09:18:20.126 SERVER LIBRARY SASUSER1 ('/SASUSER' V8) ACCESSED AS SASUSER1 BY "DMS PROCESS"(1) OF
USER USER1(1).
```

Figure 18

The first line tells us when the server server1 started, to fractions of a second. The second indicates the time the user connects to the server. The third line gives the time at which the server libref sasuser1 is assigned to the SAS data library '/sasuser'. We see that it is '…accessed as sasuser1 by "dms process"(1) of user user1(1)'. The dms process is a result of running the statement in the display manager. If we had run the programs at the UNIX prompt, this phrase would have looked something like:

```
…ACCESSED AS SASUSER1 BY "PROGRAM XXXXX.SAS"(1) OF  USER  USER`1(1).
```

Figure 19

where xxxxx.sas is the SAS program assigning the libname, started at the UNIX prompt using a command similar to the system function argument in figure 8.

Opening and closing the data set, along with attempts to make changes to the observations, result in the following messages to the log:

```
04AUG2003:09:18:30.408 SASUSER1.HOUSES1.DATA(3) OPENED FOR INPUT/2 VIA ENGINE V8 BY "AF"(2) OF USER USER1(1).

04AUG2003:09:19:22.516 WHEN USER1(2) IN "AF"(2) TRIED TO UNLOCK RECORD 1 IN SASUSER1.HOUSES1.DATA(3):
WARNING: OBSERVATION IS NOT LOCKED BY YOU.

04AUG2003:09:19:42.992 SASUSER1.HOUSES1.DATA(3) CLOSED BY "AF"(2) OF USER USER1(2).
```

Figure 20

The first line in figure 20 tells us that the data set SASUSER1.HOUSES1 is opened for input using the 'af' command given above in figure 7. The second line comes after we click on an observation in the table viewer. Since the data set attributes are set for RowLevelEdit at this point, and our session is the only one trying to access the data, we are allowed to edit the data, and this is the log message. The third line in the log is issued after we close the data set, by exiting the application.

Finally, when we disconnect from the server, and shut the server off, the following messages are issued to the log.

```
04AUG2003:09:19:47.521 USER USER1(2) HAS DISCONNECTED FROM SERVER SERVER1.
04AUG2003:09:20:07.093 NORMAL TERMINATION OF SAS SERVER SERVER1 HAS OCCURRED.
04AUG2003:09:20:07.104 USAGE STATISTICS FOR SERVER SERVER1:  MESSAGES PROCESSED: 100
```

Figure 21

The first line indicates when the user disconnects from the server. The second tells when the server has been shut off, by issuing PROC OPERATE. The final line provides information on the number of messages processed between the host (server) and client session. More information on analyzing the server log can be obtained using the set of server log analysis programs described in the SAS OnlineDoc.

SAS/SHARE MACRO BASICS

As with several other areas within the SAS System, macros provide a means for more efficient use of code and other system resources. One of the primary reasons for using macros is code reduction. This reason also applies to SAS/SHARE macros. We use several procedures within SAS/SHARE to manage servers; macros provide a way to use less code while we do this. From an administrator point of view, SAS/SHARE macros can also improve program maintenance efficiency. This can be especially useful in environments where several servers are in use simultaneously. In this case, we need an adaptable server environment for best performance. We may need to quickly stop and start servers, and switch libraries and users between servers.[4] Macros can be a more efficient way to do this. Macros defined through the autocall function of the SAS macro facility provide an efficient way to do this.

We use SAS/SHARE macros in the same way we normally use macros. We must define them first. As just mentioned, it is most efficient to define them in a central location, and then call them using the SAS autocall library. We can specify the existence and location of an autocall library, using the following OPTIONS statement in our autoexec file, so the members will be available when we start our SAS session or run our program.

```
OPTIONS mautosource
  (sasautos='!SASROOT/sasautos');
```

Figure 22

The mautosource option makes the macro processor search the autocall library for a macro when it is not found in the WORK library. The sasautos= option defines the physical location of the autocall library. The !SASROOT directory is the one in which the SAS System is installed. On a UNIX operating system, this is usually in /usr/local/sas82, for SAS version 8.2. The program within this directory, /usr/local/sas82/sasautos, which contains the macro definitions is shrmacs.sas.

The shrmacs.sas program contains a macro shrmacs(), which has the definitions of the individually available SAS/SHARE macros. All of the macros fall into three overall categories: utility, user and operator macros. The individual macros we use here, to illustrate starting and stopping the server, and assigning libnames, fall into more than one of these categories. If we call this macro with the argument 'all' (i.e., shrmacs(all)), we compile all the macros at once, and they are available for use in individual calls.

The macros we use here are:

- %servlib
- %libdef
- %strtsrv
- %shutsrv

The %servlib macro associates a server name with a library. The %libef macro generates a LIBNAME statement to define a SAS data library to be accessed through the server. The %strtsrv and %shutsrv macros start and stop the server, repectively.

To illustrate the use of these macros, we start the server in the background again, This time, we use a calling and target program. The calling program is as follows.

```
data _null_;
  rc=system('sas82 start_server.sas &');
run;
data _null_;
  x=sleep(5,1);
run;
%shrmacs(all);
%servlib(/sasuser,server1);
%libdef(sasuser1,/sasuser);
```

Figure 23

The initial two data steps in the above figure are the same as those in figure 8, where started the server without using macros. These are followed by the macro call, %shrmacs(all), which defines all the necessary macros. As mentioned above, the %servlib macro associates a server name with a physical SAS data library. This server name is used in the subsequent %libdef macro call, which generates the LIBNAME statement, defining the SAS data library accessed through the server. Using these statements, we first associate server1 with the SAS data library /sasuser, and then generate a LIBNAME statement, defining a libref SASUSER1, pointing to this data library.

The target program is given in the following figure.

```
libname _all_ clear;
%shrmacs(all);
%strtsrv(server1,%str(authenticate=opt));
dm 'bye';
```

Figure 24

The first line in the above code clears all previous libnames in case there are some in use. This is necessary to get the libref to get correctly associated with the server. Note that this target program is called before the libname is set up in the calling program, so clearing is done before a libname for the server is actually set up. In the second line, we define the necessary macros again by calling the shrmacs macro. This is necessary, since this target program is being run in background, in a separate process from the calling program. Finally, we start the server, using the strtsrv macro, which generates a PROC SERVER run. The first argument in the strtsrv macro call is the required server name. The second argument generates the authenticate=opt option, as we had done in the previous example. Note the use of the %str function. This function is necessary to allow the 'authenticate=opt' character string to be applied in the macro at compilation time. The macro processor does not receive the function or argument at macro execution time. Finally, the last statement, a 'dm' statement, ends the SAS program, by submitting usual 'bye' command to end a SAS session, as a statement in the program.

We shut the server off by, again, submitting a calling and target program, as we did above. The calling and target programs follow, separated by two lines.

```
data _null_;
  rc=system('sas82 stop_srv.sas &');
run;


%shrmacs(all);
%shutsrv(server1);
```

Figure 25

The calling program is a repeat of the format of the calling program without macros from figure 8. The target program consists of defining the macros as we have previously, follwed by a call of the shutsrv macro, which generates a run of PROC OPERATE, with the argument 'server1' as the required server id.

CONCLUSIONS

SAS/SHARE software provides a useful way to allow multiple users to simultaneously edit different observations in the same SAS data set, when data set properties are set to allow edits. We start with a SAS/AF application consisting of a table viewer containing a SAS data set, with frame properties set in the properties window of the application. Without SAS/SHARE, multiple users can only browse the data, if properties are set accordingly.

With SAS/SHARE, multiple users can edit the data in their individual sessions. The share server log also provides information about the operation of the server. Finally, SAS/SHARE macros provide a way to reduce the amount of programming code, and increase program maintenance efficiency.

REFERENCES

1. HOUSES SAS Data Set. Created with SAS software. Copyright 1999, SAS Institute Inc., Cary, NC, USA. All rights reserved. Reproduced with permission of SAS Institute Inc., Cary, NC.
2. SAS OnlineDoc, Version 8. SAS/SHARE User's Guide. Copyright© 1999 SAS Institute Inc., Cary, NC, USA. All rights reserved.
3. SAS OnlineDoc, Version 8. SAS/SHARE User's Guide. Usage. A Getting Started Exercise. Copyright© 1999 SAS Institute Inc., Cary, NC, USA. All rights reserved.
4. SAS OnlineDoc, Version 8. SAS/SHARE User's Guide. Reference. SAS/SHARE Macros. Introduction. Using SAS/SHARE Macros for Server Library Access. Why Use Macros for Server Library Access? Copyright© 1999 SAS Institute Inc., Cary, NC, USA. All rights reserved.
5. SAS OnlineDoc, Version 8. Communications Access Methods for SAS/CONNECT and SAS/SHARE. Copyright© 1999 SAS Institute Inc., Cary, NC, USA. All rights reserved.

CONTACT INFORMATION

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

Larry Altmayer
U.S. Census Bureau
ESMPD, Room 1223-4, MS 6200
Washington, DC 20233-6200
(301) 763-2569
lawrence.w.altmayer@census.gov