Administering SAS/SHARE® Software in an Information Center

Harriet Janes, SAS Institute Inc., Cary, NC
Debbie Penland, SAS Institute Inc., Cary, NC

ABSTRACT

Administering SAS/SHARE® software in an information center is a matter of understanding the functions necessary for creating and maintaining the SAS® server. Once the information center understands these functions, decisions need to be made on how these functions will be delegated. This paper discusses how the information center at SAS Institute delegated these functions, the tools that were developed to help in the administration, and the user support provided by the administrators. It is not the purpose of this paper to present a comprehensive discussion of the technical details of SAS/SHARE software.

INTRODUCTION

This paper presents a brief overview of SAS/SHARE software and how it is used. The functions of the server administrator are outlined, and two different approaches to server administration are described. The OS operating system is used to exemplify a centralized approach to server administration, while the CMS operating system is used to present a decentralized approach. The delegation of responsibility for each method at this information center is discussed, along with the tools that were developed and the user support that is provided by the administrators.

How SAS/SHARE Software is Used

SAS/SHARE software allows *concurrent update access* to a SAS data library under OS and CMS operating systems. In other words, two or more of your users can have access to a SAS data library to read and write at the same time. The following activities are possible when using SAS/SHARE software:

- While you are creating a member in a library, your other users can create, read, and update members in the same library.
- While you are using the FSEDIT procedure, two or more of your other users can update the same SAS data set at the same time, but they cannot update the same observation.
- While you and your other SAS users are using the FSEDIT procedure, your other users can simultaneously do the following:
  - use PROC APPEND to add observations to the end of the SAS data set
  - run PROC CONTENTS on the SAS data set as well as on other members of the library
  - access the library using PROC DATASETS and then browse the data set
  - use PROC COPY to copy the data set to a work file so that reports can be run on the data set.

All of these activities are controlled through a *SAS server*.

SAS Server

The SAS server is a separate SAS execution that controls the execution of input and output requests to your SAS data library. Your SAS execution reads from and writes to your SAS data library through the server. **Figure 1** shows two SAS executions within a single CPU and shows the way a data library can be shared without the SAS server.

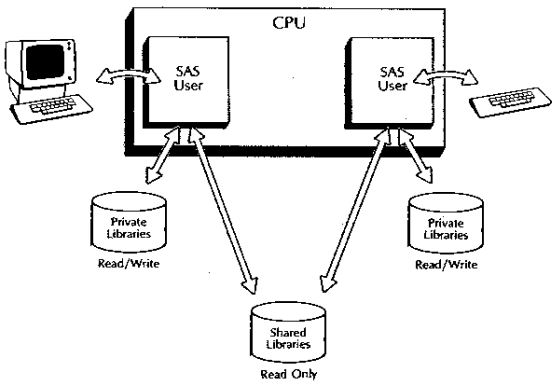

Figure 1 SAS Data Library Access Provided by the Operating System

Shared data libraries are available for read-only access without the use of the server. Your operating system does not support users having read and write access to a SAS data library simultaneously.

Figure 2 shows the SAS data library access that the server provides.

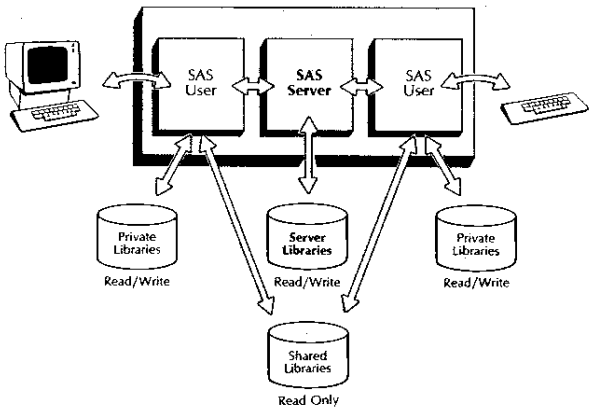

Figure 2 SAS Data Library Access Provided by the Server

The server coordinates the read and write access to the shared library to protect the integrity of the library. A SAS data library that is shared through the server is called a *server library*. The SAS server is created in a separate SAS execution invoked by the *server administrator*.

Server Administrator

The functions of the server administrator include

- creating the SAS server
- controlling the server's execution and providing tools for its execution
- giving the server access to your users' SAS data libraries and giving your users information about using the server
- evaluating server performance
- providing tools so that the server is easily accessible to your users who have permission to use it.

Your site may choose to delegate some of these functions.

SAS/SHARE Procedures

Two procedures are available in SAS/SHARE software:

PROC SERVER	creates the SAS server in a separate SAS execution.
PROC OPERATE	controls the execution of one or more SAS servers.

You can restrict the use of these procedures by setting a password when the software is installed. If you need a more tailored access control, you can implement a user exit that provides an interface between the SAS System and your system security package to limit access.

SAS/SHARE Macros for the Administrator

If you choose a centralized approach to server administration, your administration procedures may center around a series of macros that have been provided on the installation package. These macros are stored in the SAS autocall macro source library. The purposes of the following macros are to provide associations between SAS data libraries and servers and to provide server name aliases. The following is a brief description of the three macro members that you should become familiar with:

SHRMACS	compiles the LIBDEF macro and several other macros that LIBDEF needs and invokes the SHRIDSU and SHRLIBU autocall members.
SHRIDSU	defines aliases for server names that can be used by the SERVERID macro to generate SERVER=*serverid*.
SHRLIBU	defines access to your users' SAS data libraries through specified servers. The definitions are used by the LIBDEF macro.

The comment sections of each of these members fully describe their potential use in your system.

SERVER ADMINISTRATION UNDER OS

Delegating Functions: a Centralized Approach

The SAS Institute information center has chosen to delegate some of its functions. In choosing a centralized approach to server administration, the server administrator is responsible for giving the users access to the server, providing them information about how to use the server, and providing tools for using the server. Most of the server administrator's functions center around the use of the macros described earlier. By making use of these macros, all user applications are given access to the server through contact with the server administrator.

The function of creating the server has been delegated to the information center's data center systems support. They also provide the tools for the operations personnel to control the starting and stopping of the server. Systems support is also responsible for evaluating the server performance and deciding how the work load should be distributed among the servers.

Creating the Server

The server at the SAS Institute data center is executed as a *started task*. The started task name is the name by which the data center operators and most users refer to the server. It may also be the alias for the internal server name that SAS/SHARE software uses.

The macros %SHRMACS and %SHRIDSU are used in the execution of the started task. The autocall macro member %SHRMACS is used to generate SERVER=*server_name* from a table located in the autocall macro member %SHRIDSU. The server administrator maintains this macro member that contains all server names and their aliases.

Controlling the Server's Execution

The data center operations personnel can issue commands to start and stop the server whenever necessary. The commands for stopping and starting the server use macros located in the %SHRMACS member. The %STRTSRV macro is used to start the server, and the %SHUTSRV macro is used to stop the server. The server writes a SAS log, a trace file, and a SYSMDUMP file to DASD data sets. Since these files are crucial to problem determination, several generations are maintained both on DASD and on tape. The DASD files have been allocated with disposition of SHR so that they can be read by other jobs or browsed interactively while the server is executing. Whenever the server is started, new generations of these files are created and archived.

Evaluating Server Performance

If a server is heavily used, users may have a problem with performance. Since the SAS System imposes no limit on the number of servers you can run simultaneously, you can solve this problem by running multiple servers. For example, you may want to create a server for different applications, departments, and so on to spread the workload across servers and thereby improve performance. If the macros are being used, this task can be simple. The %SHRIDSU macro allows the server administrator to add additional servers, and the %SHRLIBU macro links each SAS data library with any server that the administrator chooses. If your users are using the %LIBDEF macro, they will not even be aware that a change has been made.

Functions of the Server Administrator in SAS Institute's Information Center

As server administrator, you are responsible for educating your users on how SAS/SHARE software can be used in their applications. You also need to provide a link between your users' data sets and the server. Helpful hints need to be given to your users to prevent future problems with their applications.

Education of Users You need to provide your users some type of handout or examples to follow describing what the server is and how to convert applications to use SAS/SHARE software. You should stress that it can be used both in batch as well as in interactive SAS sessions. This can be useful if you need update access to more than one data set in the same data library. The user must use the permit command to give the server update access. By issuing other RACF commands, you can choose the type of access different groups of users can have to the data library.

SAS/SHARE Macros for Users In a centralized SAS/SHARE environment, use of the macros described below easily facilitates the use of SAS/SHARE software:

- %SHRMACS(USER,NOMSG);

  This macro makes available other macros that can be used with SAS/SHARE software. Use it once at the beginning of your SAS session or program. The NOMSG parameter suppresses the display of the messages issued by the macro.

- %LIBDEF(*libref,dsname,disp*,RETRY);

  libref is the SAS data library reference. Use the libref in your SAS program as you would a DDname: to refer to a SAS data library accessed through the server. For example, use the libref as the first level of a two-level SAS data set name.

  dsname names a SAS data library that you want to access through the server. The data set name must be of the form *userid.qual1.qual2...qualn*. Although you must use the fully qualified form for the data set name, do not enclose it in single quotes.

  disp is the parameter to determine READ/WRITE access in the user environment.

  RETRY allows the system to issue a local libname if it cannot get the library through the server.

  The %LIBDEF macro is expanded to issue the LIBNAME statements needed to associate a libref with the SAS data set accessed through the server. When using the %LIBDEF macro, you do not need to identify the server name since this is already done in the %SHRLIBU macro. Also, if the form of the LIBNAME statement changes in the future, no applications will need to be changed since the %LIBDEF macro will take care of any changes for you. Note that the macro generates a LIBNAME statement for local allocation of the data library if it is not registered in the %SHRLIBU macro.

To release the library, use the LIBNAME statement

```
LIBNAME libref CLEAR
```

where *libref* is the data library reference associated with the SAS data set in the previous %LIBDEF macro.

To receive help on the %SHRMACS and %LIBDEF macros, type

```
%SHRMACS(HELP);
```

or

```
%LIBDEF(HELP);
```

An OS Interactive Example Using Macros Here is an example of a CLIST before conversion to SAS/SHARE software:

```
SAS OPTIONS('NONOTES NONEWS CLIST') SHARE
DATA
   %TSO ALLOC F(USAGE) DA('SDC.SASCURR.USAGE') OLD;
   PROC FSEDIT DATA=USAGE.USAGE SCREEN=USAGE.SCREEN;
   RUN;
   %TSO FREE F(USAGE);
   ENDSAS;
ENDDATA
```

After conversion this same CLIST looks like this:

```
SAS OPTIONS('NONOTES NONEWS CLIST') SHARE
DATA
   %SHRMACS(USER,NONSG);
   OPTIONS NONOTES;
   %LIBDEF(USAGE,SDC.SASCURR.USAGE,OLD,RETRY);
   %LIBDEF(SCREEN,SDC.SASCURR.SCREENS,SHR);
   PROC FSEDIT DATA=USAGE.USAGE SCREEN=SCREEN.USAGE;
   RUN;
   LIBNAME USAGE CLEAR;
   %TSO FREE F(SCREEN);
   ENDSAS;
ENDDATA
```

Both the %SHRMACS and %LIBDEF macros are used in the CLIST. Since the data library SDC.SASCURR.USAGE is being accessed through a server, the user does not use operating system control language.

Note that two %LIBDEF macros are used in this CLIST. The first %LIBDEF is used to access the data that are being controlled by a server. The second %LIBDEF is used to define the screens in a local environment. It is more efficient to have all the read-only data stored in a separate data library and locally accessed.

Suppose users are accessing the data member USAGE and you want to access another data member in that same library. Before SAS/SHARE software was developed, you could not allocate another data member in a batch job if a user had already allocated with write access (disposition of OLD in MVS).

The following example shows how this problem can now be solved with SAS/SHARE software.

An OS-Batch Example Using Macros

```
// EXEC SAS
//SYSIN DD *
 %SHRMACS(USER,NOMSG);
 %LIBDEF(TEST,SDC.SASCURR.USAGE,SHR,RETRY);
 PROC PRINT DATA=TEST.DATES;
    TITLE 'TEST DATES';
 RUN;
```

Note that the %SHRMACS and %LIBDEF macros are used again in this batch job. Since the data library SDC.SASCURR.USAGE is accessed by users through a server, this batch job does not use operating system control language.

Providing the Link between Data Libraries and the Server If your users are using the macros, they do not need to know

through which server their data library is accessed. In order to use these macros, you need to store information in the macro source member SHRLIBU. By modifying the member SHRLIBU in the SAS autocall macro source library, you can control which data libraries are assigned to which server. By adding lines to the end of SHRLIBU, you can maintain a list of all data sets that your site has using SAS/SHARE software. Not only does it make it easy for your users to use SAS/SHARE software, but you can also maintain information about when the data library is added and who requests that the library be added. You can also easily switch a data library from one server to another server by changing the second parameter in the %SHRLIBU macro call. Your users do not even know when their application has been moved.

Helpful Hints to Give Your Users

- If PROC FSEDIT is being used and you receive the message that says "observation is locked," press the HELP key to identify the userid of the person who is already on that observation.
- To find out who is in a particular data set, PROC OPERATE statements can be issued to see which of your users has a particular library in use through the server. See the **APPENDIX** for an example of a CLIST.
- To run a program on a data set that only requires read access and that may already be in update use by another user, use PROC COPY to copy that data set into a work file so that there are no messages issued that certain observations are locked.
- If you only want certain variables or observations, use the DATA step with the CNTLLEV=BLK option in the SET statement to subset a copy.
- The macro %SHRMACS turns the NOTES option on, so you may want to issue the NONOTES option in your session.
- These macros can all be used for data sets that cannot be defined to the server. In this case, local access is made to the data set.

SERVER ADMINISTRATION UNDER CMS

Delegating Functions: a Decentralized Approach

At SAS Institute's information center, the VM SAS representative is ultimately responsible for the administration of SAS/SHARE software, including installing the product and providing assistance and examples to the users. Because of the design of MIS applications, a decentralized approach in the delegation of the remaining administrative functions was chosen. A separate server is allocated for each application using SAS/SHARE software, and the programmer responsible for the application is also responsible for its server. These responsibilities include requesting an appropriate server userid from systems administration, writing the programs that access the server, and defining libraries to the server.

Creating a Server

Follow these steps to create one server on CMS:

1. Request that a virtual machine (for example, SASSHARE) be configured and defined in the VM system directory by systems administration, using the sample directory shown below. As with most CMS userids, it consists of a 191 minidisk. In addition, a 192 minidisk can be added for maintaining a trace data set.

```
USER SASSHARE XXXXXXX 4M 4M G 31
**** GORDON MIS DEBBIE PENLAND
IPL CMS PARM AUTOCR
ACCOUNT  MIS E375
OPTION MAXCONN 255  REALTIMER
IUCV ALLOW PRIORITY MSGLIMIT 255
CONSOLE 009 3215
SPOOL   00C 2540 READER *
SPOOL   00D 2540 PUNCH  A
SPOOL   00E 1403 A
LINK SPMAINT  19E 19E RR
LINK SASAIDS  191 49E RR
LINK CMSMAINT 19D 19D RR
LINK SPMAINT  190 190 RR
MDISK 191 3380 842 5 VM0181 MR ALL
MDISK 192 3380 237 3 VM018C MR ALL
```

2. Format the minidisks. The following is an example console file for formatting the minidisks defined in the sample directory above:

```
FORMAT 191 A
DMSFOR603R FORMAT WILL ERASE ALL FILES ON DISK 'A(191)'.
DO YOU WISH TO CONTINUE? (YES|NO):
yes
DMSFOR605R ENTER DISK LABEL:
shr191
DMSFOR733I FORMATTING DISK 'A'.
DMSFOR732I '5' CYLINDERS FORMATTED ON 'A(191)'.
R; T=0.01/0.05 10:59:43

FORMAT 192 D
DMSFOR603R FORMAT WILL ERASE ALL FILES ON DISK 'D(192)'.
DO YOU WISH TO CONTINUE? (YES|NO):
yes
DMSFOR605R ENTER DISK LABEL:
shr192
DMSFOR733I FORMATTING DISK 'D'.
DMSFOR732I '3' CYLINDERS FORMATTED ON 'D(192)'.
R; T=0.01/0.04 10:59:58
```

3. Write the PROFILE EXEC. Notice that the REXX EXEC displayed below does the following:
   - spools the console file to the RDR of another userid to keep a record of server activity.
   - uses the FILEDEF command to direct the trace data set to the 192 minidisk. CMS automatically accesses this minidisk with the filemode "D."
   - links to the minidisks where the data sets to be used are located and then accesses the minidisks.
   - executes the SAS program to invoke the server so that the server starts automatically when the userid is logged on.

```
/* REXX EXEC */
trace all

/* Spool console */
'cp spool console to snodlp2 start'

/* Direct trace  */
'filedef sassrvt1 disk trace dataset d (recfm f'

/* Link to minidisks where data sets are located */
/* and access them.                              */
cp link snodlp2 206 206 w password
acc 206 h
cp link snodlp2 207 207 w password
acc 207 i
cp link snodlp2 208 208 w password
acc 208 k

/* Turn off some line editing */
'cp term chardel off'
'cp term linedel off'

/* Start server  */
'exec sas runshare (lt ld mprint source'
```

4. Write the SAS program to execute PROC SERVER. The program for SASSHARE is illustrated below. The actual SAS data sets to be shared are ALLSUS.DATA, ALLMIS.DATA, and ALLWEEK.DATA. Notice on the LIBNAME statement that the filemode must be the same filemode defined in the PROFILE EXEC; thus, ALLWEEK.DATA is located on the SNODLP2 208 minidisk. In addition, since the LIBNAME statement is in local form, the form used to dynamically allocate a SAS data library to the SAS session in which it executes, the SERVER= option is not used.

```
/* RUNSHARE SAS */

libname allsus  'h1';
libname allmis  'i1';
libname allweek 'k1';
run;
proc server password=xxxxxxx;
run;
```

5. Place the shared data sets on the appropriate minidisks using SAS programming statements or the CMS COPYFILE command.

6. The server, SASSHARE, is now ready for use by simply logging on the userid. One of two methods can be used for logging on this server:

   - Follow the usual CMS logon procedure; then enter #CP DISC. The server continues execution in disconnect mode.

   - Autolog the server from a CMS userid with class A or B authority.

A CMS Example

It is easy to change your applications to utilize SAS/SHARE software. The programs below exemplify changing an FSEDIT procedure so that the data sets can be shared. The statement to link to the minidisk where the data set is located is changed to the server form of the LIBNAME statement, the form the user issues to access a server library. The SERVER= option must be used to identify the server. Finally, the statement to release the minidisk is changed to clear the LIBNAME defined to the server.

The following is an FSEDIT application before converting to SAS/SHARE software:

```
/*--------------------------------------*/
/*                                      */
/* THIS DATASET IS NOT SHARED           */
/*                                      */
/*--------------------------------------*/

cms link snodlp2 208 208 w password;
cms acc 208 k;
run;
proc fsedit data=allweek.data;
run;
cms rel k(det;
run;
```

The following is an FSEDIT application after converting to SAS/SHARE software:

```
/*--------------------------------------*/
/*                                      */
/* THE DATASET CAN NOW BE SHARED        */
/*                                      */
/*--------------------------------------*/
libname allweek server=sasshare;
run;
proc fsedit data=allweek.data;
run;
libname allweek clear;
run;
```

Helpful Hints for CMS

- If your site uses DIRMAINT, request DIRMAINT authority for the server on another userid so that the logon password for the server can be easily changed.

- Have the CMS systems programmers include the server in their table of userids to be autologged automatically when the system is reinvoked with the IPL command.

- Locate FSEDIT screens on a separate minidisk from the shared data sets. Since most user applications only need to read these screens, it is more efficient to locate them on a minidisk accessed in read mode. Also, you can easily modify a screen without interrupting the work of anyone using the screen by linking to the minidisk where the old screen is located, copying it to your A-disk, making the needed modifications, renaming the old screen, and then copying the new one back to the original minidisk.

- Write EXECs to invoke the SAS System and issue server commands from userids other than the actual server. For example, the EXEC shown below is executed by entering the EXEC name and server name. It displays the server libraries in use and the userids that are accessing the server. The console output from this EXEC is shown following the EXEC.

```
/* REXX EXEC to find out who is using a server */

arg servname
makebuf
  queue 'proc operate pw=xxxxxxx server=' servname ';'
  queue 'display library _all_; run;'
  queue 'proc operate pw=xxxxxxx server=' servname ';'
  queue 'display user _all_; run;'
  queue '/*'
 'exec sas (nodms nonotes 1t'
dropbuf
```

The console output from the EXEC example is as follows:

```
operate sasshare


  1?

  2?

LIBREF        STATUS        NUMBER OF USERS        MINIDISK
ALLWEEK       ACTIVE                6                 K1
ALLNIS        INACTIVE              0                 I1
ALLSUS        INACTIVE              0                 H1

  3?

  4?

USER ID       STATUS        NUMBER OF LIBRARIES
SNODLP        ACTIVE                0
SASSES3       ACTIVE                1
SASKLH2       ACTIVE                1
SNOOLR2       ACTIVE                1
SASBAK2       ACTIVE                1
SASJJH2       ACTIVE                1
SASSUS3       ACTIVE                1

  5?

R; T=0.17/0.34 16:44:14
```

NEW FEATURES IN RELEASE 5.18

With Release 5.18, there will be new options added to the PROC SERVER statement:

- CPUTIME tells the server to record its utilization of the CPU resource while it is doing work on behalf of the users connected to it.
- LOG=*value* | (*value* ...) specifies that a SAS server should record the resources it consumes on behalf of users so those users can be fairly charged for their share of the server's operation. The resources that can be specified in the LOG= options are defined below:

MESSAGE|MSG	tells the SAS server to print the number of messages exchanged with each user on its SAS log.
IO|IOEVENT|EXCP	tells the SAS server to print on its log the number of device I/O operations executed to satisfy users' requests to read and write data.
CPUTIME|CPU	tells the SAS server to print on its log the number of CPU seconds consumed to satisfy users' requests.

- You can use the automatic macro variable SYSERR to determine whether a DATA or PROC step was successful.
- You can specify the CNTLLEV= data set option with the DATA= option of the PROC PRINT, PROC FSPRINT, and PROC QPRINT statements.

CONCLUSION

Once the functions for creating and maintaining the SAS server are understood, administering SAS/SHARE software in an information center is a rather simple task. The flexibility of the software allows an administrative approach to be selected that best suits the needs of your site and your applications. If functions are shared, procedures understood, and example materials made available to users, the administration is not a time-consuming task. Whether your site chooses a centralized or decentralized approach to administration, you will find the SAS/SHARE software product a very powerful and easy-to-use tool.

APPENDIX

LISTSHR CLIST

```
PROC 0
CONTROL NOMSG
WRITE ENTER THE NAME OF THE DATA SET.
WRITE (COMPLETE DATA SET NAME WITHOUT QUOTES)
READ DB
DELETE XXX1
DELETE XXX2
DELETE XXX3
DELETE XXX4
DELETE XXX5
DELETE XXX6
WRITE NOTE: THIS CLIST WILL TAKE A LITTLE WHILE
            TO EXECUTE.
ALLOC F(TEMP2) DA(XXX2) SP(1 1) TRACK NEW REUSE
SAS OPTIONS('LOG=TEMP2 NONEWS CLIST') SHARE
DATA
   PROC OPERATE SERVER=C02SHARE;
      DISPLAY LIBRARY _ALL_;
      DISPLAY USER _ALL_;
      RUN;
   PROC OPERATE SERVER=C02DEVSV;
      DISPLAY LIBRARY _ALL_;
      DISPLAY USER _ALL_;
      RUN;
      ENDSAS;
ENDDATA
ALLOC F(TEMP1) DA(XXX1) SP(1 1) TRACK NEW REUSE
ALLOC F(TEMP2) DA(XXX2) OLD
ALLOC F(TEMP3) DA(XXX3) SP(1 1) TRACK NEW REUSE
ALLOC F(TEMP4) DA(XXX4) SP(1 1) TRACK NEW REUSE
ALLOC F(TEMP5) DA(XXX5) SP(1 1) TRACK NEW REUSE
ALLOC F(TEMP6) DA(XXX6) SP(1 1) TRACK NEW REUSE
SAS OPTIONS('LOG=TEMP5 NONOTES  NONEWS CLIST') SHARE
DATA
   ************************************************
   *                                              *
   * THIS FIRST DATA STEP FINDS ALL SYSREF AND    *
   * ASSOCIATED DATA SET NAMES.                   *
   *                                              *
   ************************************************;
  DATA TEMP4.DATASETS;
     INFILE TEMP2 LENGTH=L;
     KEEP DATASET LIBREF;
     LENGTH LIBREF $ 10;
     INPUT TEXT $VARYING72. L;
     IF _N_=1 THEN KEEPIT=.;
     IF INDEX(TEXT,'LIBREF')¬=0 THEN KEEPIT=0;
     KEEPIT=KEEPIT+1;
     RETAIN KEEPIT;
     IF KEEPIT>1 AND TEXT¬=' ' THEN DO;
        LIBREF=SCAN(TEXT,1,' ');
        I=INDEX(LIBREF,'SYS');
        LIBREF=SUBSTR(LIBREF,I);
        DATASET=SCAN(TEXT,4,' ');
        NUSERS=SCAN(TEXT,3,' ')*1;
        IF LENGTH(LIBREF)=8 THEN OUTPUT;
     END;
     IF KEEPIT>1 AND TEXT=' ' THEN KEEPIT=.;
  RUN;
  PROC SORT;
     BY LIBREF;
   ************************************************
   *                                              *
   * THIS DATA STEP FINDS ALL THE USERS ASSOCIATED *
   * WITH ALL THE SYSREFS.                        *
   *                                              *
   ************************************************;
  DATA _NULL_;
     INFILE TEMP2 LENGTH=L;
     INPUT XTEXT $VARYING72. L;
     LENGTH TEXT $ 72;
     TEXT=SCAN(XTEXT,1);
     IF LENGTH(TEXT)<3 THEN
        TEXT=SCAN(XTEXT,2);
     IF _N_=1 THEN KEEPIT=.;
     RETAIN KEEPIT;
     IF INDEX(TEXT,'USER')>0 THEN KEEPIT=0;
     KEEPIT=KEEPIT+1;
     IF KEEPIT>1 AND TEXT¬=' ' THEN DO;
        FILE TEMP1;
        PUT TEXT $9.;
     END;
     IF KEEPIT>1 AND TEXT=' ' THEN KEEPIT=.;
  RUN;
  DATA _NULL_;
     INFILE TEMP1;
     INPUT TEXT $10.;
     FILE TEMP3;
     IF _N_=1 THEN
     PUT 'PROC OPERATE SERVER=C02SHARE;';
     PUT 'DISPLAY USER ' TEXT ';';
     FILE TEMP6;
     IF _N_=1 THEN
     PUT 'PROC OPERATE SERVER=C02DEVSV; ';
     PUT 'DISPLAY USER ' TEXT ';';
  RUN;
  OPTIONS NOTES;
  %INC TEMP3/SOURCE2;
  %INC TEMP6/SOURCE2;
  RUN;
  ENDSAS;
ENDDATA
ALLOC F(TEMP1) DA(XXX1) OLD
ALLOC F(TEMP2) DA(XXX2) OLD
ALLOC F(TEMP3) DA(XXX3) OLD
ALLOC F(TEMP4) DA(XXX4) OLD
ALLOC F(TEMP5) DA(XXX5) OLD
SAS OPTIONS('NONOTES NOSOURCE NONEWS CLIST') SHARE
```

```
DATA
   ********************************************
   *                                          *
   * THIS DATA STEP FINDS ALL NAMES AND LIBREFS *
   * ASSOCIATED DATA SET NAMES.               *
   *                                          *
   ********************************************;
  DATA TEMP4.NAMES;
     INFILE TEMP5 LENGTH=L;
     KEEP SERVER LIBREF USER ;
     LENGTH LIBREF $ 10 SERVER USER $ 8;
     INPUT TEXT $VARYING72. L;
     I=INDEX(TEXT,'SERVER=');
     IF I>0 THEN SERVER=SUBSTR(TEXT,I+7);
     RETAIN SERVER;
     SERVER=COMPRESS(SERVER,';');
     IF _N_=1 THEN KEEPIT=.;
     IF INDEX(TEXT,'ACCESSING')¬=0 THEN KEEPIT=0;
     KEEPIT=KEEPIT+1;
     RETAIN KEEPIT USER;
     IF KEEPIT=1 THEN USER=SCAN(TEXT,2,' ');
     IF KEEPIT>1 AND INDEX(TEXT,'SYS')¬=0 THEN DO;
        LIBREF=SCAN(TEXT,2,' ');
        IF LIBREF¬=' ' THEN OUTPUT;
     END;
  RUN;
  PROC SORT;
     BY LIBREF;
  DATA SET1;
     MERGE TEMP4.DATASETS TEMP4.NAMES;
     BY LIBREF;
     IF DATASET='&DB' ;
  RUN;
  %MACRO RUNIT;
  DATA _NULL_;
     I=NOBS;
     SET SET1 POINT=I NOBS=NOBS;
     CALL SYMPUT('NOBS',NOBS);
     STOP;
  RUN;
  %IF &&NOBS¬=0 %THEN %DO;
     OPTIONS NOTES;
  PROC PRINT DATA=SET1;
     VAR DATASET SERVER USER;
     TITLE3 'ALL THOSE USERS THAT HAVE THIS DATASET
             THROUGH THE SERVER';
  RUN;
  %PUT DO YOU WANT TO SEND A MESSAGE TO EACH ONE OF THESE USERS;
  %PUT ASKING THEM TO PLEASE GET OUT OF THE DATA SET?;
  %INPUT ANS;
  OPTIONS NONOTES;
  %IF %UPCASE(&&ANS)=Y %THEN %DO;
  DATA SET1;
     ARRAY XNAME $ 8 XNAME1-XNAME30;
     KEEP XNAME1-XNAME30 DATASET;
     DO OVER XNAME;
        SET SET1 END=EOF;
        XNAME=USER;
        IF EOF THEN OUTPUT;
     END;
  RUN;
  DATA _NULL_;
     SET SET1;
     CALL SYMPUT('DATASET',PUT(DATASET,$CHAR40.));
     %DO I=1 %TO 30;
         CALL SYMPUT("NAME&&I",XNAME&&I);
     %END;
  RUN;
  OPTIONS NOTES;
     %DO I=1 %TO 30;
        %IF &&&&NAME&&I NE  %THEN %DO;
     X SEND 'PLEASE GET OUT OF THIS DATASET -'
             USER( &&&&NAME&&I );
        %END;
     %END;
   %PUT MESSAGES SENT.;
  %END;
  %END;
  %ELSE %DO;
     OPTIONS NOTES;
  RUN;
     %PUT THERE ARE NO DATA SETS BY THAT NAME
     BEING USED BY THE SERVER.;
  %END;
  %MEND;
  %RUNIT;
  RUN;
  OPTIONS NONOTES;
  ENDSAS;
  ENDDATA
  DELETE XXX1
  DELETE XXX2
  DELETE XXX3
  DELETE XXX4
  DELETE XXX5
  DELETE XXX6
```

REFERENCES

SAS Institute Inc. (1987), *Installation and Tuning Guide for CMS SAS/SHARE Software, Release 5.16*, Cary, NC: SAS Institute Inc.

SAS Institute Inc. (1986), *SAS/SHARE Administrator's Guide, Version 5 Edition*, Cary, NC: SAS Institute Inc.

SAS Institute Inc. (1986), *SAS/SHARE User's Guide, Version 5 Edition*, Cary, NC: SAS Institute Inc.

SAS Institute Inc. (1988), Technical Report P-175, *Changes and Enhancements to the SAS© System, Release 5.18, under OS and CMS*, Cary, NC: SAS Institute Inc.