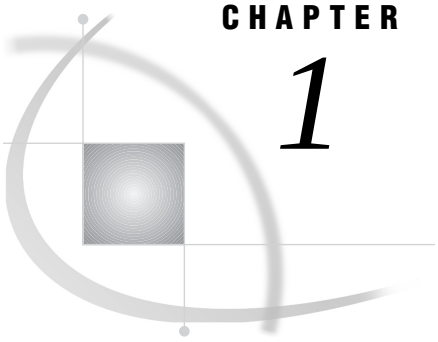

CHAPTER

1

Getting Started with SAS/SHARE Software

Introduction

This chapter is optional. It consists of an exercise to help you get started using SAS/SHARE software and a section that contains answers to frequently asked questions.

The exercise provides a sample of the different types of activities that are involved when using SAS/SHARE software at your site. Where applicable, operating environment specifics are provided.

Note: This exercise is for the purpose of example only and should not be used to set up production applications. △

If you choose not to perform the exercise or to read the frequently asked questions, see Chapter 2 for an introduction to the services that are provided by SAS/SHARE.

Audience

This exercise is recommended for all SAS/SHARE software users.

A Getting Started Exercise

Setting Up Your System

The SAS sessions that you use in this exercise exchange data by using a communications access method. For this exercise, the TCP/IP communications access method is used for all operating environments. SAS/SHARE software also supports other communications access methods. These access methods are described in detail in *Communications Access Methods for SAS/CONNECT and SAS/SHARE Software*.

To use the TCP/IP access method, you must make sure that a SAS/SHARE serverid has been added to the TCP/IP SERVICES file. Refer to the documentation for your TCP/IP software and *Communications Access Methods for SAS/CONNECT and SAS/SHARE Software* to determine the location of the SERVICES file for your operating environment.

- If a SAS/SHARE serverid has already been added to the SERVICES file, skip to the next section "Invoking SAS Software for Client and Server Sessions" on page 5, and use an existing serverid from the SERVICES file in place of **demoserv** in the rest of this exercise.
- If a SAS/SHARE serverid has not already been added to the SERVICES file, edit the SERVICES file and add a line similar to the following:

  ```
  demoserv port-number/tcp  # SAS/SHARE server
  ```

  For *port-number*, supply a number that is not already used in the SERVICES file.
- If you do not have authority to edit the SERVICES file, ask your server administrator to add **demoserv** to the SERVICES file for you. A server administrator makes sure that SAS/SHARE servers are identified in the SERVICES file on each system from which SAS/SHARE software is accessed.

Note: If you choose to use a communications access method that is different from TCP/IP, some configuration of your operating environment may be required. For more information, see *Communications Access Methods for SAS/CONNECT and SAS/SHARE Software.* △

Invoking SAS Software for Client and Server Sessions

You need to invoke three SAS sessions for this exercise. You can run these SAS sessions by logging on to three different machines or by logging on to the same machine three times. To invoke a SAS session for two clients and the SAS/SHARE server, use the commands that are specific to your operating environment. For OS/390 and CMS, the server should be run in line mode. Arrange your SAS sessions so that you can see and use all of them while you are doing this exercise.

Note: Throughout this exercise you are instructed to perform specific tasks in the Program Editor window of the user or server sessions. To illustrate the example here, one user logs on to the same machine three times. The following conventions are used in this exercise:

USER1 is john(1)
USER2 is john(2)
SERVER is demoserv.

Be sure to issue the SAS statements that are appropriate for the particular SAS session. Each step clearly identifies the session for which the instruction is intended. △

Starting a SAS/SHARE Server

Note: As mentioned earlier, this exercise uses the TCP/IP communications access method. If you use a different access method, substitute the appropriate value for the COMAMID= option wherever you see "tcp". △

Note: This exercise shows logged data for a UNIX host. △

1 In the SERVER session, submit the following lines from the Program Editor window:

```
options comamid=tcp;
libname demo(work);
proc server id=demoserv authenticate=optional;
run;
```

The omission of either the Version 7 TCPSEC= option or the Version 8 USER= and PASSWORD= options in the LIBNAME statement means that the SAS/

SHARE client/server session is running unsecured. The COMAMID= option specifies the access method that is used to communicate between a client SAS session and the server. You must specify the COMAMID= option before you invoke PROC SERVER.

A LIBNAME statement associates a SAS library reference (*libref*) with a SAS data library.

PROC SERVER manages concurrent update access to SAS data libraries and the members in those libraries. PROC SERVER runs in its own SAS session, which serves client SAS sessions by executing input and output requests to SAS data libraries. The value OPTIONAL for the AUTHENTICATE= option allows trusted users to connect to a server without requiring authentication. See "Ensuring That Userids Are Authentic" on page 35 and Chapter 8, "The SERVER Procedure," on page 95 for more information about the AUTHENTICATE= option to the PROC SERVER statement.

2 Examine the SERVER Log window:

```
NOTE: Libref DEMO was successfully assigned as follows:
      Engine:        V8
      Physical Name: /local/u/john
1     options comamid=tcp;
2     libname demo(work);
3     proc server id=demoserv authenticate=optional;
4     run;

21JAN1999:15:12:09.095 SAS server DEMOSERV started.
```

Typically, a server administrator starts the server or takes steps to have it started so that it is available when the end users and applications developers need to share SAS files. It is recommended that the server be run in non-interactive mode.

Defining a SAS Data Library to a Server

A LIBNAME statement associates a SAS library reference (libref) with a SAS data library. When you access a SAS data library through a server, your SAS session reads from and writes to that data library through the server instead of reading and writing directly to the library.

The first LIBNAME statement that specifies a particular server connects your SAS session to that server. For a client session, you must specify the COMAMID= option before you try to connect to the server.

1 In the USER1 session, submit the following lines from the Program Editor window:

```
options comamid=tcp;
libname demo server=demoserv;
```

Note: If you are connecting to a server on a remote host, you must specify the network node name in the SERVER= option in the form:

```
server=network-node-name.demoserv
```

△

See the TCP/IP chapter for your particular operating environment in *Communications Access Methods for SAS/CONNECT and SAS/SHARE Software* for more information.

Examine the USER1 Log window:

```
NOTE: Libref DEMO was successfully assigned as follows:
      Engine:        REMOTE
      Physical Name: /local/u/john
1    options comamid=tcp;
2    libname demo server=demoserv;
```

For convenience in this exercise, the libref DEMO is associated with the server library WORK. This means that the data files that are created in the exercise are temporary.

2 Examine the SERVER Log window:

```
21JAN1999:15:16:46.521 User john(1) has connected to server demoserv.
21JAN1999:15:16:52.566 User john(1) has created "DMS Process"(1)
  under "Kernel"(0).
21JAN1999:15:16:59.079 Server library ('/local/u/john' V8) accessed as
  DEMO by user john(1).
```

The log displays messages about your connection to the server. The messages include the server name, the name of the server library that you assigned, and the user identification. The user identification has the form *userid*(*n*), where *n* is the server connection number.

Creating a SAS Data Set

1 In the USER1 session, submit the following lines from the Program Editor window:

```
data demo.test;
   do i=1 to 5;
      output;
   end;
run;
```

This creates a SAS data set that contains 5 observations and 1 variable that you will use in the remaining steps.

2 Examine the SERVER Log window:

```
21JAN1999:15:23:17.110 User john(1) has created "DATASTEP"(2)
  under "DMS Process"(1).
21JAN1999:15:23:20.719 DEMO.TEST.DATA(1) opened  for output via
  engine V8 by "DATASTEP"(2) of user john(1).
21JAN1999:15:23:26.835 DEMO.TEST.DATA(1) closed by "DATASTEP"(2)
  of user john(1).
21JAN1999:15:23:27.194 User john(1) has terminated "DATASTEP"(2)
  (under "DMS Process"(1)).
```

The log displays information about the DATA step and the name of the SAS data set that is opened for output and then closed.

Locking an Observation

1 In the USER2 session, submit the following lines from the Program Editor window:

```
options comamid=tcp;
libname demo server=demoserv;
proc fsedit data=demo.test;
run;
```

An FSEDIT window appears and contains the following information centered on the screen:

```
i:      1
```

It shows the value 1 in the first observation.

Examine the USER2 Log window:

```
NOTE: Libref DEMO was successfully assigned as follows:
      Engine:        REMOTE
      Physical Name: /local/u/sasvcl
1     options comamid=tcp;
2     libname demo server=shr9;
3     proc fsedit data=demo.test;
4     run;
```

2 Examine the SERVER Log window:

```
21JAN1999:15:29:39.116 User john(2) has connected to server demoserv.
21JAN1999:15:29:42.483 User john(2) has created "Process"(1)
  under "Kernel"(0).
21JAN1999:15:29:48.155 Server library ('/local/u/john' V8) accessed as
  DEMO by user john(2).
21JAN1999:15:29:54.124 User john(2) has created "FSEDIT"(2)
  under "DMS Process"(1).
21JAN1999:15:29:56.109 DEMO.TEST.DATA(1) opened for input/2 via
  engine V8 by "FSEDIT"(2) of user john(2).
21JAN1999:15:29:56.933 DEMO.TEST.DATA(1) reopened for update/R by
  "FSEDIT"(2) of user john(2).
```

The FSEDIT procedure accesses the data set that was created by USER1 in the previous section. The first observation is currently locked by USER2 for update access.

3 In the USER2 FSEDIT window, change the value in the first observation by positioning the cursor over the value **1** and overwriting it with **5**, but do not save it.

The FSEDIT window now appears like this:

```
i:    5
```

Attempting to Modify the Locked Observation

In the USER1 session, submit the following lines from the Program Editor window:

```
proc fsedit data=demo.test;
run;
```

This procedure also accesses the data set that was created by USER1.

When the FSEDIT window opens, the following message is displayed because the first observation is already locked by USER2's FSEDIT session:

```
WARNING: User john(2) (server connection 2) is using this observation.
```

USER1 cannot update the observation until after USER2 releases it. Notice that the value of **i** is still **1** because USER2 had not saved the change in the previous step.

Releasing the Observation

In the USER2 session, close the FSEDIT window by selecting **File** -> **Close** from the pull-down menu. This action releases the observation that was locked by USER2.

Retrying Access to the Observation

1 After the FSEDIT window in the USER2 session is closed, in the USER1 FSEDIT session, re-read the observation by selecting **View** -> **Observation Number** from the pull-down menu and then enter **1** in the pop-up and click OK.

   The USER1 FSEDIT window now looks like this:

```
i:     5
```

   Notice that the observation was updated to reflect USER2's change to 5.

2 In the USER1 FSEDIT window, change the value from **5** to **4**.

   The USER1 FSEDIT window now looks like this:

```
i:     4
```

Stopping the Server

Note: If you are an applications developer or a server administrator, skip to the next section. "Identifying the Server" on page 10. △

If you are an end user, you can stop the server and close all SAS sessions in this section. (See *Note:* immediately following this paragraph.) After you complete the steps in this section, proceed to "Frequently Asked Questions" on page 12.

Note: Servers are typically stopped by server administrators, not by end users. △

1 In the USER1 session, close the FSEDIT window by selecting **File** -> **Close** from the pull-down menu.

2 In the USER1 session, stop the server by submitting the following lines from the Program Editor window:

```
proc operate server=demoserv;
stop server;
quit;
```

   Examine the USER1 Log window:

```
16   proc operate server=demoserv;
PROC OPERATE is set to default server DEMOSERV.
================================================
17   stop server;
Default server DEMOSERV is now stopped.
PROC OPERATE was previously set to default server
DEMOSERV but is not set to any server now.
================================================
18   quit;
```

   Note: If you are not on the same machine as the server, you must specify the network node name in the SERVER= option:

```
server=network_node_name.demoserv
```

   △

3 In the SERVER Program Editor window, close the server session by entering the following line:

```
endsas;
```

4 On both the USER1 and USER2 command lines in the Program Editor window, close the user session by submitting the following command:

```
bye
```

For SAS/SHARE end users, you have finished the exercise.

Identifying the Server

For applications developers and server administrators to continue with this exercise, in the USER2 session, submit the following line from the Program Editor window:

```
proc operate server=demoserv;
```

A RUN statement is not used or needed with this statement.

The OPERATE procedure manages the execution of a SAS/SHARE server. You must identify which SAS server you want to manage, even if there is only one server executing. If you are not on the same machine as the server, you must specify the network node name in the SERVER= option:

```
server=network_node_name.demoserv
```

PROC OPERATE is an interactive procedure and is terminated by a QUIT statement. PROC OPERATE has several commands. You will use a few of them in the next steps. All output generated by PROC OPERATE is displayed in the Log window.

Examine the USER2 Log window:

```
proc operate server=demoserv;
PROC OPERATE is set to default server DEMOSERV.
```

In general, you should specify the COMAMID= option before using PROC OPERATE to connect to a server. If you know that you will use the default access method on your host, you may omit the COMAMID= option. You do not need to specify a value for the COMAMID= option in this step because it was already specified for this SAS session as shown in “Locking an Observation” on page 7.

Note: PROC OPERATE is usually used by a server administrator; sometimes an applications developer has responsibilities that include server administration. △

Viewing the Server Libraries

In the USER2 session, submit the following line from the Program Editor window:

```
display library _all_;
```

Examine the USER2 Log window. The DISPLAY LIBRARY command displays information about the libref, status, the number of users, and the library name of all SAS data libraries that have been defined to the server.

Viewing Information about Clients

In the USER2 session, submit the following line from the Program Editor window:

```
display user _all_;
```

Examine the USER2 Log window.

```
USER                    NUMBER OF
ID         STATUS       LIBRARIES
-------------------------------
```

```
john      ACTIVE          0
john      ACTIVE          1
john      ACTIVE          1
==============================
7    display user _all_;
```

The DISPLAY USER command displays information about the userid, the status, and the number of libraries that have been defined by each connected client.

Disconnecting Clients from the Server

In the USER2 session, submit the following line from the Program Editor window:

```
quiesce user 1 2;
```

The QUIESCE USER command gradually terminates a user's access to a server by denying new user requests for resources and moving the user from active status to stopped status. The user can continue the SAS program step or window that is currently in use but will not be able to use the server after that step or window terminates.

Users can be identified by userids or connection numbers. For example, user JOHN(1) can be quiesced by either of the following statements:

```
quiesce user 1;
quiesce user john;
```

Examining the Server Log

1. In the USER1 session, because the FSEDIT window is still displayed, USER1 could still edit the data set that was created in "Creating a SAS Data Set" on page 7.

   Close the USER1 FSEDIT window by selecting **File** -> **Close** from the pull-down menu.

2. Examine the SERVER Log window, which displays information about disconnecting the quiesced users from the server.

```
21JAN1999:15:56:57.207 PROC OPERATE command from user john(3):
  QUIESCE USER 1 2;
21JAN1999:15:59:52.065 DEMO.TEST.DATA(1) closed by "FSEDIT"(3)
  of user john(1).
21JAN1999:15:00:02.161 User john(1) has terminated "FSEDIT"(3)
  (under "DMS Process"(1)).
```

Retrying Access to the Server

In the USER1 session, re-submit the following lines from the Program Editor window:

```
proc fsedit data=demo.test;
run;
```

Examine the USER1 Log window:

```
10   proc fsedit data=demo.test;
You cannot open data set DEMO.TEST.DATA because user JOHN(1)
is quiesced on server DEMOSERV.
11   run;
```

```
NOTE: The SAS System stopped processing this step because of errors.
```

The log displays information indicating that the attempt to communicate with the server was rejected. Because USER1 is stopped, you are not able to communicate with the server to access the data set.

Stopping the Server

In the USER2 session, submit the following lines from the Program Editor window:

```
stop server;
quit;
```

The STOP SERVER command terminates a server as rapidly as possible. If users are connected to the server when you execute a STOP SERVER command, changes that they have not saved are lost. The QUIT command terminates PROC OPERATE in interactive mode.

Closing the SAS Sessions

1 In the SERVER Program Editor window, close the server session by submitting the following line:

```
endsas;
```

2 On both the USER1 and USER2 command lines of the Program Editor window, close the user session by submitting the following command:

```
bye
```

Frequently Asked Questions

The following are answers to questions that are frequently asked by new users of SAS/SHARE software.

General Questions

What is SAS/SHARE software? Why would I use it?

You use SAS/SHARE software when

- more than one user needs to update a SAS file (or several SAS files) at the same time.
- users want to access SAS files on a remote host without having to use a separate SAS/CONNECT remote login for each user.

Where can I read about SAS/SHARE software?

- The book you're reading now applies to all operating environments and explains what SAS/SHARE software is, describes the parts of the software, and includes

other basic material. This book includes detailed reference material for PROC SERVER, PROC OPERATE, the LIBNAME statement, and the LOCK statement.

- Also see *Communications Access Methods for SAS/CONNECT and SAS/SHARE Software* for information about using a communications access method to make a connection from a client session to a server session. This book also contains instructions to configure the access method.

Do people have to use a new SAS procedure to share their data?

No. The users who add and maintain data continue to use the SAS procedures and windows that they already know: FSEDIT, PROC APPEND, FSVIEW, and so forth.

Instead of requiring users to change the SAS tools they know and use, SAS/SHARE software takes advantage of the SAS Multiple Engine Architecture (MEA) to allow those SAS tools to access data through a "traffic cop" that's formally known as a SAS server. A *SAS server* allows many users to read and update the data in one or more SAS files, all at the same time by keeping track of locks on observations, catalog entries, and SAS files.

Does each person who is responsible for maintaining data have to run his or her own SAS server?

No. There are really three roles that people assume with respect to SAS/SHARE software. All of the roles can be performed by the same person, or one person may take two of the roles, or each role may be assigned to a separate group of people. Those roles are:

end user
: The end user reads, adds, and updates data.

applications developer
: The applications developer writes SAS programs used by the end users.

server administrator
: The server administrator makes sure SAS servers are available to the end users.

It's not unusual for an applications developer and a server administrator to be the same person. This happens when someone who develops an application is responsible for the SAS server(s) used by that application.

Three people? I hope this software doesn't require the effort of a large team of people.

No, three roles. We use the three roles to help organize the efforts that make possible shared maintenance of data. In real life, the responsibilities of the various people involved in a project tend to overlap; many times the person who develops an application and maintains a SAS server is the same individual.

To help you keep track of how responsibilities usually are divided when more than one user needs to update a SAS file at the same time, the remainder of the questions and answers are divided according to the roles described above:

- Questions end users ask
- Questions application developers ask
- Questions server administrators ask.

Questions Asked by End Users

How do I get started with SAS/SHARE software?

There isn't much you need to know.

You use an application that someone else developed to read, add, or update data in one or more SAS files. Occasionally you will find that an observation, a catalog entry, a file, or a library is locked by another user. When that happens, a message appears on your display and you cannot modify the data. SAS/SHARE software keeps track of which users have which data locked, so users cannot cause each other's changes to become mysteriously "lost."

How can I determine whether I am accessing a SAS library through a SAS/SHARE server?

The SERVER= option is required in any LIBNAME statement (or, in SCL programs, any LIBNAME() function) for a library to be accessed through a SAS/SHARE server.

When a library is accessed through a server, the information that is displayed by the LIBNAME statement shows you that the engine, which was used to access the library, is named REMOTE, and the physical name is a subdirectory accessed by the server SAS session.

Use the LIST option in a LIBNAME statement to obtain information about how a SAS library is defined to a SAS session, including:

- the name of the server through which the library is accessed
- the libref used by the server to refer to the library (which may be the same as or different from the user's libref for the library)
- the engine used in the server SAS session to read and write files in the library
- the operating system and machine type on which the server is running.

Questions Asked by Applications Developers

How do I get started with SAS/SHARE software?

Read Chapter 1, "Getting Started with SAS/SHARE Software," on page 3. To understand your applications in perspective, it is very helpful to see how end users experience locking conflicts and how server administrators create and terminate SAS/SHARE servers.

Also, read through the sample SCL program in Appendix 5 of this book. It illustrates a variety of programming techniques for sharing data among users.

There is advice about the performance of your applications in Appendix 4.

What do I have to do to a SAS library to have it accessed through a SAS/SHARE server?

The one thing you have to do is add a SERVER= option to each LIBNAME statement that a user will use to access the library.

You might want to pre-define one or more libraries to a server. To do that, include a LIBNAME statement for each library before entering the PROC SERVER statement. This removes the requirement for a physical name in each user's LIBNAME statement that accesses any of those libraries. This can make it easier to maintain your application.

For more information about server libraries, see Chapter 3, "Starting and Managing a SAS/SHARE Server," on page 27. See Chapter 9, "The LIBNAME Statement," on page 103 for reference information about server libraries.

Can a SAS/SHARE server access a SAS library across a network?

Yes, but you usually do not want to organize it that way.

Even though a SAS/SHARE server is not exactly like other file servers you might be familiar with, it is still a process that does a lot of disk I/O. That's especially true because a server does I/O to files on behalf of a large number of users. For that reason, you want the path length between the server SAS session and the physical disk to be as short as possible. That means storing data on the same computer as the server used to access that data, whenever possible.

I've used servers before. A SAS/SHARE server is similar to the file servers we have on our network, isn't it?

Not really. Ordinarily, file servers are not aware of the content of the files they manage, but a SAS/SHARE server allows several users to update a single copy of a SAS file at the same time.

SAS/SHARE software is tuned to manage locking conflicts within SAS files, for example, two users attempting to update the same observation of a SAS data file or two users attempting to modify the same entry in a SAS catalog. SAS/SHARE is not optimized to provide the bulk data transfer services at which many file servers excel.

Can a server use more than one communications access method?

Yes. A server administrator uses SAS options to do this.

If your application requires the use of more than one communications access method, ask your server administrator to set up the server for your application with the access methods that you need. For more information about access methods, see "Specifying a Communications Access Method" on page 28.

Do I have to use a different SAS/SHARE server for each file updated by the users of my application?

No. A server can share many files in the same SAS library as well as sharing files in many different SAS libraries at the same time.

Is there a limit on how many users or libraries a SAS/SHARE server can support?

No. There are no quotas coded into the software, and you do not need to use SAS options to specify how many users or files a server will support at one time.

However, a server is like any other process on a computer; as it is asked to do a great deal of work it takes longer to get that work done. It is possible to put so much traffic through a server that users complain about response time. If any of your servers become that busy, you should consider creating one or more additional servers and dividing the files among the servers.

See your server administrator about creating additional servers.

Do I need to ask my server administrator to start and stop my application's server each day?

Probably not. Like other processes on a computer, SAS/SHARE servers can usually run for long periods of time without intervention. Sometimes periodic maintenance or

backup activity requires processes to be stopped for awhile and then restarted. Servers are certainly not immune to such interruptions.

Questions Asked by Server Administrators

How do I get started with SAS/SHARE software?

Read Chapter 1, "Getting Started with SAS/SHARE Software," on page 3, paying particular attention to when PROC SERVER is started and stopped. You may occasionally need to use PROC OPERATE, so you should at least read over those steps of the exercise.

A SAS/SHARE server is usually started when the operating system has completed its initialization, and it continues to run until the computer is shut down or someone decides that the server should be terminated. You should be familiar with creating and managing those kinds of processes. Of course, a server only does I/O or uses the processor while users are accessing data through it; a server doesn't do a residual amount of work while it is not doing work on behalf of other users.

Because a server executes within a SAS session, you need to know how to invoke SAS software on each computer on which a server will run.

I've used servers before. A SAS/SHARE server is similar to the file servers we have on our network, isn't it?

Not really. Ordinarily, file servers are not aware of the content of the files they manage, but a SAS/SHARE server allows several users to update a single copy of a SAS file at the same time. Also, SAS/SHARE servers automatically translate transmitted data when the client host represents data differently from the server host.

Is being a SAS/SHARE server administrator a full-time job?

No! SAS/SHARE software is designed to require no regular maintenance or other administrative activity.

Can a server administrator control access to a server?

Yes. By default, SAS/SHARE software does not restrict who can connect to a server nor which files they can access, but you can restrict access to a server with the OAPW= and UAPW= options in the PROC SERVER statement. The OAPW= option specifies a password that the server administrator must supply (using the OPERATE procedure) to connect to the server. The UAPW= option specifies a password that the user must supply in the LIBNAME statement to establish communication with the server. Of course, your file system restricts a server's access to files according to the privileges of the files and the server's process. You can also set up a secured server. For more information, see "Server Security" on page 33.

Can a server administrator control which libraries can be accessed through a server?

Yes. For each server, you can prevent users from defining libraries to the server and restrict them to only those libraries that you define. To do this, use the NOALLOC option in the PROC SERVER statement. See "Limiting Which Libraries the Server Can Access" on page 34.

Remember that passwords can be used to restrict access to individual SAS files. See *SAS Language Reference: Dictionary* for more information about data set passwords.

How can I terminate a user's connection to a SAS/SHARE server?

First, decide whether you want the access terminated immediately or as soon as it is convenient for the user.

The QUIESCE USER command causes a user to be disconnected from a server when the user finishes the SAS program step or window currently being executed. The STOP USER command immediately terminates a user's connection to a server and may cause loss of updates that have not been communicated to the server.

In both cases, the user cannot re-connect to the server until a START USER command is executed to give the user permission to re-connect or until the server is re-started. Servers do not remember a list of stopped users when they are terminated and re-started.

See "Quiescing a User's Access to a Server" on page 130 and "Terminating a User's Connection to a Server" on page 130.

How can I stop a SAS/SHARE server?

The QUIESCE SERVER command causes a server to stop when all users have disconnected from the server. The STOP SERVER command immediately stops the server and may cause loss of updates that have not been communicated to the server.

See "Quiescing a Server" on page 126 and "Stopping a Server" on page 128.

Can a server use more than one communications access method?

Yes, if your host supports more than one communications access method. See "Specifying a Communications Access Method" on page 28 for information about the communication access methods available on your host.

Refer to the Appendix "Configuration Instructions for SAS/SHARE Software" in your installation instructions for information about configuration requirements for the communication access methods.

How can I determine when I need to create a second SAS/SHARE server?

You need to create a second server when the traffic on a server becomes so heavy that an application's performance is less efficient.

Just as you periodically check the resource consumption of the other service processes on a computer, you should, from time to time, take a look at how much CPU, I/O, and virtual storage the servers are using. Using operating system management tools, you may notice that a server is executing a very large number of disk I/O operations or needs a very high percentage of the processor. When you observe those conditions, you should consider moving some of the work being done by that server to another, possibly new, server.

Distributing load among servers must be a cooperative effort between server administrators and application developers. SAS Institute supplies a set of autocall macros that assign resources to servers symbolically, which can make moving resources from one server to another much easier. See Chapter 6, "Using SAS/SHARE Macros for Server Access," on page 75.