CHAPTER

1

Getting Started with SAS/SHARE

SAS/SHARE: Learning to Use

Introduction

If you're a new user of SAS/SHARE, this section provides answers to frequently asked questions (FAQs). A step-by-step example exercise shows the different types of activities that are involved when using SAS/SHARE. Where applicable, operating environment specifics are provided.

Note: The following exercise is an example only and should not be used to set up production applications. △

If you have some experience with SAS/SHARE and choose not to perform this exercise or read the FAQs, proceed to Chapter 2, "Using SAS/SHARE Software," on page 19.

Setting Up Your Operating Environment

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 also supports other communications access methods, which are described in detail in *Communications Access Methods for SAS/CONNECT and SAS/SHARE*.

To use the TCP/IP access method, you must verify that a SAS/SHARE server ID has been added to the TCP/IP SERVICES file. To find the location of the SERVICES file for your operating environment, see *Communications Access Methods for SAS/CONNECT and SAS/SHARE* and the documentation for your TCP/IP software.

- □ If a SAS/SHARE server ID has already been added to the SERVICES file, proceed to the next section "Invoking SAS for Client/Server Sessions (All New Users)" on page 5, and use an existing server ID from the SERVICES file in place of **&servername** in the remainder of this exercise.
- □ If a SAS/SHARE server ID 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*, specify a number that is not already specified in the SERVICES file.

Execute the following statement in the server, the client, and the operator sessions:

```
%let servername=demoserv;
```

- ☐ If you do not have authority to edit the SERVICES file, ask your server administrator to add **demoserv** to the SERVICES file. A server administrator ensures that SAS/SHARE servers are identified in the SERVICES file on each operating environment that accesses SAS/SHARE.
- ☐ The TCP/IP access method allows you to specify syntax that uses two consecutive underscores with a port number, in place of a server ID that has been defined in the client TCP/IP SERVICES file. As an alternative to editing the TCP/IP SERVICES file, execute the following statement in the server, the client, and the operator sessions:

  ```
  %let servername=_ _port-number;
  ```

  for *port-number* specify a number that is not already used in the TCP/IP SERVICES file. Do not space after the first underscore or the second underscore.

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

Invoking SAS for Client/Server Sessions (All New Users)

You need to invoke three SAS sessions for this example 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.

Note: Arrange your SAS sessions so that you can see and use all of them while you are doing this exercise, because you will perform specific tasks in the Program Editor window of the user or server sessions. △

In this example, one user logs on to the same machine three times

USER1 is john(1)

USER2 is john(2)

SERVER is demoserv.

Note: Be sure to issue the SAS statements that are appropriate for the specific SAS session. Each step in this example clearly identifies the session for which the instruction is intended. △

Starting a SAS/SHARE Server (All New Users)

Note: Usually, a server administrator starts the server so that it is available when end users and applications developers need to share SAS files. It is recommended that the server be run in non-interactive mode. For the z/OS operating environment, the server should be run in line mode. △

Note: In this example exercise, the data is logged for a UNIX operating environment, and the TCP/IP communications access method is used. If you choose a different method, replace **tcp** in every occurrence of the COMAMID= option with the appropriate access method value. △

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

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

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

The omission of the 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.

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 users with valid access permission to connect to a server without requiring verification. See "Ensuring That User IDs Are Valid" on page 35. For more information about the AUTHENTICATE= option see the PROC SERVER statement.

2 Examine the SERVER Log window, which now will contain the following information:

```
NOTE: Libref DEMO was successfully assigned as follows:
      Engine:        V9
      Physical Name: /local/u/john
1    options comamid=tcp;
2    libname demo (work);
3    proc server id=&servername authenticate=optional;
4    run;

30Apr2003:15:12:09.095 SAS server DEMOSERV started.
```

Defining a SAS Data Library to a Server (All New Users)

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, which specifies a name for the 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 from the Program Editor window:

```
options comamid=tcp;
libname demo server=&servername;
```

Note: If you are connecting to a server on a remote operating environment, you must specify the network node name in the SERVER= option as follows:

```
server=network-node-name.&servername
```

△

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

Examine the USER1 Log window, which will contain the following information:

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

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

2 Examine the SERVER Log window, which now contains the following information about your connection to the server. The messages include the server name, the name of the server library that you specified, and the user identification in the form *user-ID*(*n*), where *n* is the server connection number.

```
30Apr2003:15:16:46.521 User john(1) has connected to server demoserv.
30Apr2003:15:16:52.566 User john(1) has created "DMS Process"(1)
  under "Kernel"(0).
30Apr2003:15:16:59.079 Server library ('/local/u/john' V9) accessed as
  DEMO by user john(1).
```

Creating a SAS Data Set (All New Users)

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

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

This DATA step creates a SAS data set that contains 5 observations and 1 variable that you will use in the remainder of this example. The Log window displays information about the DATA step and the name of the SAS data set that is opened for output and then closed.

2 Examine the SERVER Log window again.

```
30Apr2003:15:23:17.110 User john(1) has created "DATASTEP"(2)
  under "DMS Process"(1).
30Apr2003:15:23:20.719 DEMO.TEST.DATA(1) opened  for output via
  engine V9 by "DATASTEP"(2) of user john(1).
30Apr2003:15:23:26.835 DEMO.TEST.DATA(1) closed by "DATASTEP"(2)
  of user john(1).
30Apr2003:15:23:27.194 User john(1) has terminated "DATASTEP"(2)
  (under "DMS Process"(1)).
```

Locking an Observation (All New Users)

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

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

An FSEDIT window appears in the center of the screen. It shows the value **1** in the first observation.

```
i:     1
```

Examine the USER2 Log window, which contains the following information:

```
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, to which the following information was added:

```
30Apr2003:15:29:39.116 User john(2) has connected to server demoserv.
30Apr2003:15:29:42.483 User john(2) has created "Process"(1)
  under "Kernel"(0).
30Apr2003:15:29:48.155 Server library ('/local/u/john' V9) accessed as
  DEMO by user john(2).
30Apr2003:15:29:54.124 User john(2) has created "FSEDIT"(2)
  under "DMS Process"(1).
30Apr2003:15:29:56.109 DEMO.TEST.DATA(1) opened for input/2 via
  engine V9 by "FSEDIT"(2) of user john(2).
30Apr2003:15:29:56.933 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 FSEDIT window in the USER2 session, change the value in the first observation by placing the cursor over the value **1** and type **5**, but do not save it.

The FSEDIT window of USER2 now looks like this:

```
i:    5
```

Accessing a Locked Observation (All New Users)

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

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

PROC FSEDIT 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 the PROC FSEDIT statement in USER2's 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 did not save the change in the previous step.

Releasing a Locked Observation (All New Users)

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 a Locked Observation (All New Users)

1 After the FSEDIT window in the USER2 session closes, return to the USER1 FSEDIT session, re-read the observation by selecting **`View -> Observation Number`** from the pull-down menu and type **`1`** in the resulting pop-up window, click OK and the USER1 FSEDIT window now looks like this:

```
i:    5
```

Notice that the observation was updated to reflect USER2's change from 1 to 5.

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

The USER1 FSEDIT window now looks like this:

```
i:    4
```

Stopping the Server (All New Users)

Note: In the real world, servers are usually stopped by server administrators, not by end users. △

For this example exercise, if you are an end user, stop the server and close all SAS sessions.

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

2 Also, in the USER1 session, stop the server by submitting the following from the Program Editor window:

```
proc operate server=&servername;
stop server;
quit;
```

Examine the USER1 Log window, which now contains the following information:

```
16   proc operate server=&servername;
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 in the PROC OPERATE statement,

```
proc operate server=network_node_name.&servername;
```

△

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

```
endsas;
```

4 On the command lines of both the USER1 and USER2 Program Editor windows, close the user sessions by issuing the following command:

```
bye
```

For SAS/SHARE end users, you have finished the example exercise. See “Frequently Asked Questions (FAQs) about SAS/SHARE” on page 13.

Identifying the Server (Server Administrators and Applications Developers)

Note: Usually, the OPERATE procedure is used by a server administrator; sometimes an applications developer has responsibilities that include server administration. △

The remainder of the steps in this section are mainly here for applications developers and server administrators who are continuing this exercise. In the USER2 session, submit the following from the Program Editor window:

```
proc operate server=&servername;
```

PROC OPERATE is an interactive procedure that is terminated by a QUIT statement. A RUN statement is not used or needed with a PROC OPERATE statement.

PROC OPERATE manages the execution of a SAS/SHARE server. You must identify which SAS/SHARE 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 in the PROC OPERATE statement:

```
proc operate server=network_node_name.&servername;
```

Examine the USER2 Log window, which contains the following information:

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

Usually, 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 operating environment, 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 in an earlier step. See “Locking an Observation (All New Users)” on page 8.

Viewing the Server Libraries (Server Administrators and Applications Developers)

PROC OPERATE has several commands. You will use some of the commands in the next steps. All output generated by PROC OPERATE is displayed in the Log window.

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

```
display library _all_;
```

The DISPLAY LIBRARY command in the PROC OPERATE step 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.

Examine the USER2 Log window.

Viewing Information about Clients (Server Administrators and Applications Developers)

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

```
display user _all_;
```

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

Examine the USER2 Log window, which now contains the following information:

```
USER                  NUMBER OF
ID        STATUS      LIBRARIES
-------------------------------
john     ACTIVE         0
john     ACTIVE         1
john     ACTIVE         1
==============================
7    display user _all_;
```

Disconnecting Clients from the Server (Server Administrators and Applications Developers)

In the USER2 session, submit the following 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 terminates or after the window closes.

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

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

Examining the Server Log (Server Administrators and Applications Developers)

1 In the USER1 session, because the FSEDIT window is still opened, USER1 can still edit the data set that was created in an earlier step. See "Creating a SAS Data Set (All New Users)" 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 the following information:

```
30Apr2003:15:56:57.207 PROC OPERATE command from user john(3):
  QUIESCE USER 1 2;
30Apr2003:15:59:52.065 DEMO.TEST.DATA(1) closed by "FSEDIT"(3)
  of user john(1).
30Apr2003:15:00:02.161 User john(1) has terminated "FSEDIT"(3)
  (under "DMS Process"(1)).
```

Accessing a Closed Server (Server Administrators and Applications Developers)

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

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

Examine the USER1 Log window, which contains the following information:

```
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 messages in the Log window tell you that the attempt by USER1 to communicate with the server is rejected. Because USER1 is stopped, you cannot access the data set.

Stopping the Server (Server Administrators and Applications Developers)

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

```
stop server;
quit;
```

The STOP SERVER command in the PROC OPERATE step terminates a server as quickly 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 (Server Administrators and Applications Developers)

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

```
endsas;
```

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

```
bye
```

Frequently Asked Questions (FAQs) about SAS/SHARE

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 server without having to use a separate SAS/CONNECT remote log in for each user.

Where can I read about SAS/SHARE software?

You can read about SAS/SHARE in:

- This document, which explains SAS/SHARE software, describes the parts of the software, and applies to all operating environments. It also includes basic and 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* for information about using a communications access method to connect from a client session to a server session and for 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: PROC FSEDIT, PROC APPEND, PROC FSVIEW, and so forth.

Instead of requiring users to change the SAS tools they already know and use, SAS/SHARE 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/SHARE server. A *SAS/SHARE server* allows many users to read and update data in one or multiple SAS files, concurrently, by tracking locks on observations, catalog entries, and SAS files.

Does each person responsible for maintaining data have to run an individual SAS/SHARE server?

No. There are three roles that users assume with respect to SAS/SHARE:

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/SHARE servers are available to the end users.

The three roles can be performed by the same person, or one person may perform two roles, or each role may be assigned to a separate group of people.

It's not unusual for the same person to perform the tasks of an applications developer and a server administrator, for example, when the person who develops an application is responsible for the SAS/SHARE 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. The three roles help organize the efforts so that shared maintenance of data is possible. In real life, the responsibilities of the various people involved in a project might overlap. Often, the same person who develops an application also maintains a SAS/SHARE server.

To help you keep track of how responsibilities usually are divided when multiple users need to update a SAS file at the same time, the remainder of this section answers the questions most frequently asked by end users, application developers, and server administrators.

FAQs by End Users

How do I get started with SAS/SHARE?

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

How can I find out if I'm accessing a SAS library through a SAS/SHARE server?

The SERVER= option is required in a 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 in the Log window about 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. This information includes:

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

FAQs by Applications Developers

How do I get started with SAS/SHARE?

See "SAS/SHARE: Learning to Use" on page 4.

Topics of importance for applications developers include SAS library access, locking data objects, and SAS programming considerations. See Chapter 4, "Writing End-User Applications to Access Shared Data," on page 41. For a sample SCL application, see Appendix 4, "SAS Component Language (SCL) Application," on page 255. For complete details about locking, see Chapter 5, "Locking SAS Data Objects," on page 57.

You might also find helpful information about how server administrators manage SAS/SHARE servers. See Chapter 3, "Managing a SAS/SHARE Server (Server Administrators)," on page 27. For specific details about creating a SAS/SHARE server and setting SAS options to enhance performance and to establish logging parameters, by operating environment, see Appendix 2, "Creating the SAS/SHARE Server Environment," on page 219.

What do I have to do to a SAS library so that users can access it through a SAS/SHARE server?

You have to 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 executing 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, "Managing a SAS/SHARE Server (Server Administrators)," on page 27. For details about the LIBNAME statement, see Chapter 13, "Remote Library Services," on page 137.

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 that you might be familiar with, it is still a process that generates a lot of disk I/O. That's especially true because a server generates 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 that is 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. Usually, file servers are not aware of the content of the files they manage, but a SAS/SHARE server allows several users to update the same copy of a SAS file at the same time.

SAS/SHARE 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 enable 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 that is updated by the users of my application?

No. A server can share many files in the same SAS library and 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 limits 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 the same as any other process on a computer; as it is asked to handle greater workloads it takes longer to do the work. 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. As with 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 a period of time and then re-started. Servers are not immune to such interruptions.

FAQs by Server Administrators

How do I get started with SAS/SHARE software?

Read Chapter 1, "Getting Started with SAS/SHARE," on page 3, giving special attention to when PROC SERVER is started and stopped. Occasionally, you might need to use PROC OPERATE, so you should read those tasks in this exercise.

Usually, a SAS/SHARE server is started when initialization of the operating environment is completed, and it continues to run until the computer is shut down or the server is terminated. You should be familiar with creating and managing those types of processes. Of course, a server only generates I/O or uses the processor while users are accessing data through it; a server doesn't process a residual amount of work when it is not processing work on behalf of other users.

Because a server executes within a SAS session, you need to know how to invoke SAS 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 operating environment represents data differently from the server operating environment.

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

No! SAS/SHARE 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 does not restrict who can connect to a server or 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, which the server administrator must supply (in the OPERATE procedure), to connect to the server. The UAPW= option specifies a password that the user must supply in the LIBNAME statement to connect to the server. Of course, your file system restricts a server's access to files based on the access permission set for 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 users can access through a server?

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

Remember that passwords can be used to restrict access to individual SAS files. See the PW= data set option in *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 disconnects a user from a server when the user ends the SAS program step currently being executed or closes the window currently being used. The STOP USER command immediately terminates a user's connection to a server and might cause loss of updates that have not been communicated to the server.

In either instance, the user cannot re-connect to the server until a START USER command is executed, which lets the user re-connect, or until the server is re-started. Servers do not retain a list of stopped users when they are terminated and re-started.

See "Quiescing User Access to a Server" on page 159 and "Terminating User Connections to a Server" on page 160.

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 might cause loss of updates that have not been communicated to the server.

See "Quiescing a Server" on page 154 and "Stopping a Server" on page 157.

Can a server use more than one communications access method?

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

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 environment management tools, you might 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, consider moving some of the work from that server to another, possibly new, server.

Distributing the workload among servers must be a cooperative effort between server administrators and applications developers. SAS Institute provides a set of autocall macros that assign resources to servers symbolically. These macros can make moving resources from one server to another much easier. See Chapter 6, "SAS/SHARE Macros for Server Access," on page 73.