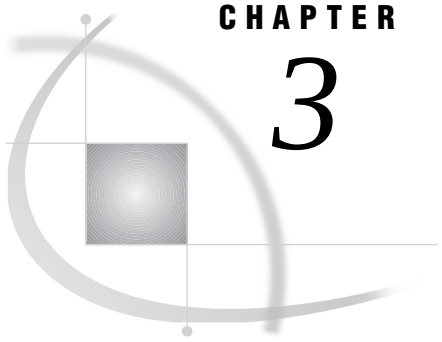

CHAPTER

3

Starting and Managing a SAS/SHARE Server

Audience

This chapter is recommended primarily for server administrators.

Starting a Server: A Fast-Track Approach

You may need to use the fast-track approach to start a new server quickly if

- you are installing SAS/SHARE for the first time.
- a new application presents a sudden demand.

- system resources become strained and require that the workload be shifted to a new server.

If you are new to SAS/SHARE software, there are a number of different issues to consider as you fine-tune your server environment: system options, pre-defined libraries, automated start-up, and so on. However, you might want to defer some of these issues and start a server right away so that your applications developers can begin to create new applications or migrate old ones to the multi-user environment.

To get a server up and running with minimal effort, perform the following steps in a SAS session on the server machine (that is, the host at which you start a server):

1. After the requisite configuration has been done for the communications access method, declare the communications access method to be used between a SAS/SHARE server and its clients.

   An example of declaring the TCP/IP communications access method follows:

   ```
   options comamid=tcp;
   ```

2. Start the server, assigning it the name that you just configured. An example of starting a server na
   ```
   run;
   ```

   The following message appears in the server SAS log with a default time stamp:

   ```
   18Feb1999:09:47:30.000 SAS server SHARE1 started.
   ```

The SAS/SHARE server is now available for SAS clients and other clients to use.

CAUTION:

Be aware of these limitations to the fast-track approach to starting a server. Server security was not addressed. It may be possible for a client that accesses the server to have the same privilege to access data as the server does. Also, any SAS client that can access the server can manage the server with PROC OPERATE statements. Thus, such a client can stop the server or stop access to any library through the server. For details about server and library security, see “Server Security” on page 33.

If you run a server SAS session interactively, the SAS session assumes that you can resolve any problems that it encounters by using a requestor window. While the SAS session waits for a response to its query, the server may not be able to continue to service client requests until the query is answered. However, you might not be aware that a response is required if the window in which the server is running is not visible or is not being monitored. Therefore, it is recommended that you specify the SAS system option NOTERMINAL so that SAS does not display requestor windows, and it handles the situation without prompting. △

Specifying a Communications Access Method

A communications access method is the interface between SAS software and the network protocol that you use to connect two hosts. The access method that you choose is determined by the network protocols that you have available at your site and the host types that you are connecting. Supported access methods for Version 8 are:

APPC (Advanced Program-to-Program Communications)

DECnet (the Digital Equipment Corporation Networking architecture)

NetBIOS (the IBM Network Basic Input/Output System)

TCP/IP (Transmission Control Protocol/Internet Protocol)

IUCV (the VM/CMS Inter-User Communications Vehicle)

XMS (Cross Memory Services)

For a complete list of valid connections among platforms and access methods for SAS/SHARE, see *Communications Access Methods for SAS/CONNECT and SAS/SHARE Software*.

Hosts and access methods typically require the configuration of access method resources on the host at which the server will run.

Consult with your network administrator about the configuration of access method resources on the host at which you start a server. For details about configuring access method resources, see *Communications Access Methods for SAS/CONNECT and SAS/SHARE Software*.

Furthermore, server administrators should be aware of special considerations and limitations when the server is accessed by clients on a host type that is different from that of the server. For information about cross-architecture limitations and considerations, see "Cross-Architecture Access" in Appendix 1.

After you verify that access method resources have been configured but before you start a server, you must specify the access method to use. Use the following syntax to specify an access method:

OPTIONS COMAMID=*access-method*;

where *access-method* is the abbreviation for the method that is used by the server to communicate with a client. See the preceding list for acceptable access-method identifiers.

For a server that runs on a host on which only one access method is available, use only the COMAMID option. An example follows:

```
options comamid=tcp;
```

You may specify the COMAMID= option in an OPTIONS statement, at SAS invocation, or in a SAS configuration file.

If the host on which the server runs supports multiple access methods, you may specify up to two auxiliary access methods by which clients may access the server by using the COMAUX1 and COMAUX2 options.

All of the access methods initialize when the server initializes. The activation of multiple access methods makes a server available to several groups of clients, each using a different access method simultaneously.

COMAUX options can be specified only at a SAS invocation or in a SAS configuration file. The syntax for the COMAUX options follows:

COMAUX1=*alternate-method*

COMAUX2=*alternate-method*

An example of configuration file entries for a server that is running on an OS/2 host follows:

```
-comamid appc
-comaux1 tcp
-comaux2 netbios
```

When the server starts, all of the communications access methods are initialized. The server is simultaneously available to client sessions that use the APPC access method as well as to clients that use the TCP/IP and NetBIOS access methods.

For more information about access methods, see *Communications Access Methods for SAS/CONNECT and SAS/SHARE Software*.

Pre-Defining SAS Data Libraries to the Server

Although it is not required, you can pre-define one or more selected SAS data libraries to the server for use by client applications. Advantages of pre-defining a library are

- Applications can use the libref that you define to identify the library. This protects them against future changes to the library's physical pathname.
- The library is available to all clients that use the server's SQL services, including SAS clients that use the Remote SQL Pass-Through (RSPT) facility, and other clients that use the SAS ODBC driver, the JDBC driver, or the SQL library for C.
- The server administrator can refer to the library with a libref in PROC OPERATE commands.

Be aware that the userid from which the server runs must have the appropriate permissions to access the SAS data library that you want to pre-define to a server. You should be familiar with the permissions requirements of the operating environment, the network, and the security software before you pre-define a library. For information about server security, see "Server Security" on page 33.

Methods for Pre-Defining a Server Library

Server administrators can pre-define a SAS data library to a server in the following ways:

- in your server program, include a LIBNAME statement before specifying the PROC SERVER step.
- in a server administrator session, use an ALLOCATE LIBRARY command in the OPERATE procedure to pre-define a library.

Note: In addition to these recommended methods, you can also define a libref externally to SAS. For this information, see the SAS documentation for your operating environment. △

Pre-Defining a Server Library by Using the LIBNAME Statement

In your server program, before specifying the PROC SERVER step, you may pre-define one or more SAS data libraries to the server. The following syntax is typical:

LIBNAME *libref 'SAS-data-library' <options>*;

How you specify the physical pathname of the SAS data library and options are host-specific. For details about the LIBNAME statement and host-specific options, see *SAS Language Reference: Concepts* and the SAS documentation for your operating environment.

Here is an example of how to pre-define a library to the server with the LIBNAME statement for a UNIX host:

```
libname mylib '/payroll/div2/emp';
```

After you define a library to the server, a message is displayed to the SAS log. An example of a message for a server on a UNIX host follows:

```
1     libname mylib '.';
NOTE: Libref MYLIB was successfully assigned as follows:
Engine: V8
Physical Name: /payroll/div2/emp
```

Pre-Defining a Server Library by Using the ALLOCATE LIBRARY Command in the OPERATE Procedure

From a server administrator session, while the server is running, pre-define one or more SAS data libraries to the server by using the following syntax:

PROC OPERATE SERVER=*serverid*;

ALLOCATE LIBRARY *libref 'SAS-data-library'* <*options*>;

How you specify the physical pathname of the SAS data library and options are host specific. For full details about the ALLOCATE LIBRARY command, see "Allocating a Library to a Server That Is Already Running" on page 121. For details about host-specific options, see *SAS Language Reference: Dictionary* and the SAS documentation for your operating environment.

Here is an example of how to use the ALLOCATE LIBRARY command of the OPERATE procedure for a Windows host:

```
proc operate server=share1;
  allocate library mylib '\payroll\dev2\emp';
```

Starting a Server

To start a server, you invoke the PROC SERVER statement, which enables multiple clients to access and to use SAS data libraries and members in those libraries at the same time. As part of server start-up, you must assign a server name, and you may optionally specify operational attributes.

Use the following syntax to start a server with the selected options:

PROC SERVER <ID=>*serverid* <ALLOC|NOALLOC>
<AUTHENTICATE=OPT|REQ>
<LOG=*value*><DTF=*SAS-datetime-format*>;

The following sections explain each of these options for the SERVER procedure. For complete information about server options, see Chapter 8, "The SERVER Procedure," on page 95. For an example of a typical log, see "Server Log Reporting All Logging Statistics" on page 39.

Identifying the Server

You assign to the server a valid SAS name that can contain up to eight characters, including alphanumeric characters and the following special characters: dollar sign ($), at sign (@), and pound sign (#). See *SAS Language Reference: Concepts* for details about SAS naming rules.

Server naming is also constrained by the type of host that the server runs on and the access method that it uses. For complete information about serverids by host, see *Communications Access Methods for SAS/CONNECT and SAS/SHARE Software*.

Here is an example of how to specify a serverid:

```
proc server id=share1;
```

Limiting Users to Pre-Defined Libraries

You can use the NOALLOC option to limit users to accessing only libraries that you pre-define to the server. This option offers a broad method for controlling what data users can access through the server.

Authenticating Users

The AUTHENTICATE= option controls whether the server requires connecting clients to supply valid userid and password pairs when they connect to the server. See "Ensuring That Userids Are Authentic" on page 35 for details about user authentication.

Logging Server Usage Statistics

You can log usage statistics for each client that accesses a server. This is useful for debugging and tuning server applications. It also allows you to charge users for their share of server resource consumption. The server, by default, writes a client's usage statistics to the server log when the client disconnects from the server. Here are some of the statistics that you may request:

Number of messages processed
: Number of messages (requests and replies) that a client exchanges with a server in a single session.

Bytes transferred
: Cumulative number of bytes that are received from a client and that are sent to a client in a single session.

Active time
: Cumulative elapsed time that the server processed requests on behalf of a client in a single session. Although this figure is not CPU time, it is related to CPU time. Whereas CPU time for a given operation normally is relatively independent of other server usage, this figure increases with increased server level of activity. However, it should give a good relative indication of the CPU usage by the client compared with other clients' values that are tracked during similar levels of server activity. Note that this value can exceed the elapsed time value, especially in the server totals, because many server requests can be active (hence, being timed) concurrently.

Examples of how to set logging follow:

```
proc server id=share1 log=message;
proc server id=share1 log=bytecount;
proc server id=share1 log=(message bytecount active
  elapsedtime);
proc server id=share1 log=all;
```

A typical example of a client log for all statistics follows:

```
Usage statistics for user mike(1):
   Messages processed:       5,143
   Bytes transferred:       10,578K
   Active time:       1:47:23.6148
   Elapsed time:      3:28:64.7386
```

For complete details about the logging options, see Chapter 8, "The SERVER Procedure," on page 95. For a more complete example of a SAS log, see "Server Log Reporting All Logging Statistics" on page 39.

To charge users for their share of server resource consumption, allocate that consumption proportionately according to the utilization statistics. You may allocate consumption based on a single statistic or on a combination of statistics. The most useful statistics for this purpose are

number of messages processed

bytes transferred

active time.

You will probably need to experiment with the relative weights of these in your charge-back formula. They are at sensitive to a particular host, access method, server level of activity, and application mix.

Number of messages processed represent actual, billable work by the server. The MESSAGE, BYTECOUNT, and ACTIVETIME values of the LOG= option report data that, together, characterize the work that a user asked the server to perform. Here are a few examples:

- Small BYTECOUNT and MESSAGE values but a large ACTIVETIME value might indicate that a small amount of data was selected from a large file by sequentially searching the file or interpreting a complex view.
- Moderate or large BYTECOUNT with a large MESSAGE value might indicate that a small amount of data is being read by the user on each message that is exchanged with the server. This might be caused by random access or exceptionally long observations and might suggest taking a snapshot of the data for the user's analysis.
- Small ACTIVETIME with a large BYTECOUNT suggests that the user is exchanging data "in bulk" with the server. That does not ordinarily indicate a problem, but if the server is overloaded, you might be able to suggest that the user try another bulk-data-transfer technique.

Specifying the Format for the Server Log Timestamp

You can prepend a datetime stamp of a particular format to each message that is written to the server log or you can suppress the datetime stamp. The DATETIME22.3 default format presents the date and time in the form *DDMMMYYYY:hh:mm:ss.ddd*.

Examples of how to set the timestamp follow:

```
proc server id=share1 alloc log=cpu dtformat=time11.2;
proc server id=share1 noalloc log=io dtformat=_NODTS_;
```

An example of a datetime format is 18JAN1999:14:02.39.186.

Server Security

Controlling Administrator Access to a Server

By default, any user can send administrator requests to a SAS/SHARE server to stop the server or to display and modify its characteristics, such as who can access the server and what SAS libraries it can serve. To prevent unauthorized users from sending these types of requests, you can assign an administrator password to a server when you

start it. To do this, use the OAPW= option in the PROC SERVER statement. Here is an example:

```
proc server id=share1 oapw=blue31;
```

When you use PROC OPERATE to send administrator commands to the server, you must use the SAPW= option to specify the administrator password. For example,

```
proc operate id=share1 sapw=blue31;
```

You can also specify the administrator password in the individual PROC OPERATE server management commands. For details, see "Displaying Information about a Server" on page 126.

Controlling Access to Data through a Server

A SAS/SHARE server must have appropriate permissions to read and write any data that any of the server's clients will access through it. Usually this means that a server is authorized to access more libraries and data sets than any one of its clients. For this reason, you may want to employ security measures to prevent unauthorized access to data through the server by using one or more of the following:

- a single password on the server
- limitations on what libraries the server can access
- passwords on SAS data sets, views, and catalogs
- host or file system protections on libraries and data sets
- userid authentication.

Assigning a User Password to a Server

The simplest method that you can use to limit access to data through a server is to assign a user password to the server when you start it. This password must be supplied whenever a client connects to the server. This method offers the broadest control over who can access data through the server. It applies to all users regardless of the data that they access or the operations that they perform.

To assign a user password to a server, use the UAPW= option in the PROC SERVER statement. Here is an example:

```
proc server id=share1 uapw=hotwings;
run;
```

When a user specifies the server for the first time in a LIBNAME statement or in a PROC SQL CONNECT TO statement, the SAPW= option must be used to specify the user password. Here is an example:

```
libname invoice server=share1 sapw=hotwings;
```

Limiting Which Libraries the Server Can Access

If there are libraries or data sets that should not be accessed through a particular server, you can prevent such unauthorized access by using the host or file system security software to deny the server access to those libraries or files.

You can also use the NOALLOC option in the PROC SERVER statement to limit users to accessing only those libraries that you pre-define. For more information, see "Pre-Defining SAS Data Libraries to the Server" on page 30 and "Limiting Users to Pre-Defined Libraries" on page 32.

Host or File System Protections

When a user reads or writes SAS libraries and SAS files, most hosts and file systems verify the user's authority to read or write that data.

Because a server is interposed between a user and the data, the authority checking that is normally performed by the host or the file system security software must be done in the server's session to protect access to that data through the server. For this reason, a SAS/SHARE server calls the host or the file system to check a user's authority whenever an attempt is made to read or write a library through the server.

In Version 7, by default, a SAS/SHARE server required each communications access method to present, for each user that connects to the server, an authenticated userid for the host on which the server runs. In Version 8, the provision of an authenticated userid and password as arguments to either the LIBNAME statement, the ALLOCATE LIBRARY command of the PROC OPERATE statement, or the Remote SQL Pass-Through statement preempts the Version 7 method of supplying a userid and password by way of a communications access method. Regardless of the method used for collecting a userid and password, the server uses the authenticated userid in making the authority check. To permit access by unauthenticated userids, you can use the AUTHENTICATE=OPTIONAL option in the PROC SERVER statement for both Version 7 and Version 8 behaviors. See Chapter 8, "The SERVER Procedure," on page 95 for details about setting options in the PROC SERVER statement. For details about implementing Version 8 userid and password behavior, see the syntax for Chapter 9, "The LIBNAME Statement," on page 103, Chapter 10, "Remote SQL Pass-Through (RSPT) Facility," on page 109, or Chapter 13, "The OPERATE Procedure," on page 119.

In order to authenticate a userid, most access methods require the user to use an access method-specific mechanism to supply the userid and corresponding password for the server host. The access method encrypts the userid and password and transmits them to the server session to be authenticated. For details about the mechanisms that are available to control whether an access method authenticates connecting users and the mechanisms by which users can supply their userids and passwords, see *Communications Access Methods for SAS/CONNECT and SAS/SHARE Software*.

On most hosts, to authenticate a userid and to make an authority check, a SAS/SHARE server calls the host or the file system directly. However, on CMS and UNIX, the server performs these functions by calling an external program that you can modify.

Ensuring That Userids Are Authentic

Because a SAS/SHARE server calls the host or the file system to check a user's authority to access data, the accuracy of the userid that the server presents to the host or to the file system determines whether a user is allowed to access data.

When all communications access methods are configured (see *Communications Access Methods for SAS/CONNECT and SAS/SHARE Software*), on most hosts each userid is authenticated by the communications access methods when a user connects to the server. That authentication is valid for the duration of the connection, and the server presents the userid to the host or to the file system whenever the user attempts to access data through the server.

The option AUTHENTICATE=OPTIONAL specified in a PROC SERVER statement is used to bypass the requirement that the access methods authenticate connecting userids. This illustrates the usefullness of the option AUTHENTICATE=OPTIONAL for getting a server up and running quickly with an absolute minimum of work. In addition, this use shows the value of configuring production servers to run without the option AUTHENTICATE=OPTIONAL, so that production servers can run unattended yet only allow users to access data that they have permission to access.

If you are running a SAS/SHARE server on a platform that does not have a security facility, which controls users' access to files, or if you are running a SAS/SHARE server

in an environment in which every user should have full access to every file available to the server, you may want to use the option AUTHENTICATE=OPTIONAL instead of implementing security for each of the access methods that is used by the server. An example of this situation is a SAS/SHARE server that is running under OS/2 without a security facility installed. In that situation, it is not possible for communications access methods to validate connecting userids, nor is it possible for the server to verify access to files on a per-userid basis.

There is another situation in which a communications access method used by a SAS/SHARE server runs in a secure, distributed network of computers, but not all of the users have userids on the computer on which the SAS/SHARE server runs. In this case, users without userids who are connecting by means of that access method can be "trusted" by the server because this type of connection can come only from computers in the secure network. If all of the files that are available to the server can be safely accessed by all of the userids in the secure network, the convenience of "bypassing" security checking for that access method outweighs the trouble of creating userids on the server's computer just so the access method can re-validate each user's identity.

An example of this situation is a SAS/SHARE server that is running on a UNIX platform that can be accessed by either the TCP/IP or the APPC access method. If all connections by means of the APPC access method can come only from mainframe computers in a secure network but it is impractical to duplicate the mainframe userids on the UNIX computer, trusting the connections made by way of the APPC access method might be the most practical decision. Meanwhile, connections to the specific server that is using the TCP/IP access method would not be trusted because (at this hypothetical installation) the TCP/IP network is not secure. In this case, specifying TCPSEC=_SECURE_ and APPCSEC=_NONE_ to control the access methods and using the option AUTHENTICATE=OPTIONAL in the PROC SERVER statement allows each APPC user to access any file available to the server, but limits TCP/IP users to the files available to their userids on the computer on which the server is running.

Using Encryption Services

As another form of security, you may choose to implement encryption services that protect data that is sent between hosts across a network. See Appendix 5, "Encryption Services," on page 207 for more information.

Writing a SAS Program to Start a Server

Using the information that is presented in this chapter, you are ready to write a simple SAS program to start a server. You may set SAS options in either a configuration file that is processed during SAS invocation, or you can set the options explicitly in a server session. To enable communication between the server and clients on different host platforms, you must declare a communications access method. Also, you may choose to pre-define SAS data libraries to the server. Finally, you start the server and specify the options that you want.

The following example shows how to start a server on a UNIX host. Exact syntax varies according to host.

The following sample program starts a server:

```
options comamid=tcp;
libname payable '/dept/acct/pay';
proc server id=share1 msgnumber;
```

Automating Server Start-up

Instead of starting a server in interactive mode, you can automate server start-up by creating a command file that runs during system initialization. The steps that you perform vary by host platform. For complete details about how to automate server start-up on a specific host, see Appendix 2, “Creating the SAS/SHARE Server Environment,” on page 155.

Managing a Server and Its Libraries

You can use commands in the OPERATE procedure to manage an *active server*, that is, a server that is running. You may display information about a server, quiesce a server, re-set a server for subsequent management operations, re-start a quiesced server, and stop a server. For details about how to manage a server, see “Server Management Commands” on page 126.

You can also use commands in the OPERATE procedure to manage server libraries. You may allocate a library to an active server, display information about a library, free a library, quiesce a library, or stop a library. For details about how to manage a library, see “Library Management Commands” on page 121.

Finally, you can use commands in the OPERATE procedure to manage users. You may display information about a user, quiesce a user’s access to a server, re-start a quiesced or a stopped user, or terminate a user’s connection to a server. For details about how to manage users, see “User Management Commands” on page 128.

Server Log Reporting OPERATE Procedure

Output 3.1 on page 38 shows a portion of the server administrator’s log, which collects the PROC OPERATE command and its output. The log reports client/server transactions for users John and Maria who are working on a server named SHARE1.

Output 3.1 Administrator's Log for the OPERATE Procedure

```
LOG
Command ===>

1    proc operate serverid=share1;
=============================================================================
2    display user _all_;

                                        NUMBER OF
                 USER ID      STATUS    LIBRARIES
                 ---------------------------------
                 john(1)      ACTIVE         1
                 maria(2)     ACTIVE         2
                 myid(3)      ACTIVE         0
=============================================================================
3    stop user maria;
User maria(2) stopped from active state.
User maria is now disallowed from connecting to server SHARE1.
=============================================================================
4    display user maria;

                                        NUMBER OF
                 USER ID      STATUS    LIBRARIES
                 ---------------------------------
                 maria(2)     STOPPED        0
=============================================================================
5    quiesce user 1;
User john(1) quiesced from active state.
=============================================================================
6    display user 1;
                                        NUMBER OF
                 USER ID      STATUS    LIBRARIES
                 ---------------------------------
                 john(1)      QUIESCED       1

User john(1) is accessing these libraries:

       USER LIBREF    SERVER LIBREF    LIBRARY NAME
       ------------------------------------------------------------
       DATALIB        SYSUSE           <SAS-library-name>

User john(1) is accessing these data sets:

USER LIBREF    SERVER LIBREF    MEMBER       TYPE         OPEN MODE
---------------------------------------------------------------------
DATALIB        SYSUSE           USAGE        CATALOG       UPDATE
=============================================================================
7    start user maria;
User maria started from stopped state and therefore has become unknown
to server SHARE1.
=============================================================================
8    quit;
```

The OPERATE procedure sends commands 2 through 8 (shown in Output 3.1 on page 38) to be executed. The DISPLAY USER command requests general information about all users who are currently accessing SHARE1. The users are listed by userid along with their current status and the number of library assignments to the server.

Next, the STOP USER command immediately disconnects user MARIA(2) from the server. If Maria is updating an observation by using the FSEDIT procedure when the

STOP command is issued, she loses the updates on her display, but she does not lose previous updates. The STOP command terminates all attachments to that server, and she is prohibited from accessing that server until the administrator issues a START command.

After stopping Maria, the server administrator issues the DISPLAY USER command to check her current status. No libraries are listed for Maria because the STOP command released them.

The QUIESCE USER command gradually terminates user JOHN(1) to allow John to finish work in the data sets that he currently has open. The subsequent DISPLAY USER command reports that John is still accessing member USAGE in library DATALIB. When John closes USAGE in that library, the server releases the library and disconnects John. Because John's session was quiesced by its connect number, John can re-connect to the server when he wants to. He receives a new connection number at that time.

Next, the administrator issues the START USER command to allow Maria to access the server again. Note that the START USER command does not establish a communication path between the server and Maria. It simply enables Maria to re-establish a path by using a LIBNAME statement or an SQL CONNECT TO statement. She must explicitly re-access the server.

Finally, the QUIT statement terminates PROC OPERATE.

Server Log Reporting All Logging Statistics

Output 3.2 on page 39 shows a typical server log with all logging statistics reported for the server SHARE2 that is running on a UNIX host.

Output 3.2 Sample Log for SAS/SHARE Server SHARE2

```
Command ===>
1?  PROC SERVER ID=share2 LOG=(ACTIVETIME BYTECOUNT ELAPSEDTIME MESSAGE) msgnumber;
2?  run;
07JAN1999:07:15:36.690 043131 SAS server SHARE2 started.
07JAN1999:07:16:20.048 043021 User john(1) has connected to server SHARE2.
07JAN1999:07:16:20.442 043143 User john(1) has created "Line Mode Process"(1)
                              under "Kernel"(0).
07JAN1999:07:16:21.206 043069 Server library TESTDATA
                              ('/local/u/john/server' V8) accessed as
                              TESTDATA by user john(1).
07JAN1999:07:16:31.593 043021 User maria(2) has connected to server SHARE2.
07JAN1999:07:16:31.846 043143 User maria(2) has created "Line Mode Process"(1)
                              under "Kernel"(0).
07JAN1999:07:16:31.923 043069 Server library DEMOTEST
                              ('/local/u/john/server' V8) accessed as
                              DEMOTEST by user maria(2).
07JAN1999:07:17:32.462 043143 User maria(2) has created "PRINT"(2) under
                              "Line Mode Process"(1).
07JAN1999:07:17:33.537 043100 DEMOTEST.X.DATA(1) opened for input/S via
                              engine V8 by "PRINT"(2) of user maria(2).
07JAN1999:07:17:40.361 043102 DEMOTEST.X.DATA(1) closed by "PRINT"(2)
                              of user maria(2).
07JAN1999:07:17:40.422 043144 User maria(2) has terminated "PRINT"(2)
                              (under "Line Mode Process"(1)).
07JAN1999:07:18:05.575 043143 User maria(2) has created "DATASTEP"(3)
                              under "Line Mode Process"(1).
07JAN1999:07:18:05.668 043100 DEMOTEST.DEMO.DATA(1) opened for output via
                              engine V8 by "DATASTEP"(3) of user maria(2).
```

```
07JAN1999:07:18:06.016 043102 DEMOTEST.DEMO.DATA(1) closed by "DATASTEP"(3)
                              of user maria(2).
07JAN1999:07:18:06.096 043144 User maria(2) has terminated "DATASTEP"(3)
                              (under "Line Mode Process"(1)).
07JAN1999:07:18:48.262 043143 User john(1) has created "PRINT"(2)
                              under "Line Mode Process"(1).
07JAN1999:07:18:48.313 043100 TESTDATA.DEMO.DATA(1) opened for input/S via
                              engine V8 by "PRINT"(2) of user john(1).
07JAN1999:07:18:49.734 043102 TESTDATA.DEMO.DATA(1) closed by "PRINT"(2)
                              of user john(1).
07JAN1999:07:18:49.765 043144 User john(1) has terminated "PRINT"(2)
                              (under "Line Mode Process"(1)).
07JAN1999:07:18:58.322 04306A Server library DEMOTEST (accessed as DEMOTEST)
                              released by user maria(2).
07JAN1999:07:18:58.338 043144 User maria(2) has terminated "Line Mode
                              Process"(1) (under "Kernel"(0)).
07JAN1999:07:18:58.909 043022 User maria(2) has disconnected from server SHARE2.
07JAN1999:07:18:59.886 043151 Usage statistics for user maria(2):
                                  Messages processed:                  24
                                  Bytes transferred:                8,028
                                  Active time:               0:00:02.7525
                                  Elapsed time:              0:02:28.3527
07JAN1999:07:19:06.298 04306A Server library TESTDATA (accessed as TESTDATA)
                              released by user john(1).
07JAN1999:07:19:06.319 043144 User john(1) has terminated "Line Mode
                              Process"(1) (under "Kernel"(0)).
07JAN1999:07:19:06.411 043022 User john(1) has disconnected from server SHARE2.
07JAN1999:07:19:06.425 043151 Usage statistics for user john(1):
                                  Messages processed:                  14
                                  Bytes transferred:                3,133
                                  Active time:               0:00:01.8139
                                  Elapsed time:              0:02:46.6840
07JAN1999:07:19:16.018 043021 User john(3) has connected to server SHARE2.
07JAN1999:07:19:16.569 0430A9 PROC OPERATE command from user john(3): STOP SERVER;
07JAN1999:07:19:16.603 043132 Normal termination of SAS server SHARE2 has occurred.
07JAN1999:07:19:17.212 043022 User john(3) has disconnected from server SHARE2.
07JAN1999:07:19:17.246 043151 Usage statistics for user john(3):
                                  Messages processed:                   2
                                  Bytes transferred:                  102
                                  Active time:               0:00:01.0691
                                  Elapsed time:              0:00:01.9230
07JAN1999:07:19:17.642 043150 Usage statistics for server SHARE2:
                                  Messages processed:                  40
                                  Bytes transferred:               11,263
                                  Active time:               0:00:05.6355
                                  Elapsed time:              0:03:42.8948
NOTE: PROCEDURE SERVER used:
      real time           3:45.51
      cpu time            0.74 seconds
```

In the server log, a message is posted for each significant client/server transaction. A log message is presented in the form:

dtformat msgnumber message

The *dtformat* and *msgnumber* fields are controlled by options that you provide in the PROC SERVER statement. See Chapter 8, “The SERVER Procedure,” on page 95 for explanations of these options.

In the server log, users JOHN(1), MARIA(2), and JOHN(3) started and terminated three separate server sessions. Upon termination, usage statistics were generated for each of the three sessions. In addition, cumulative usage statistics were reported for the server SHARE2. The usage statistics are controlled by values that you provide for

the LOG option in the PROC SERVER statement. See Chapter 8, “The SERVER Procedure,” on page 95 for explanations of usage statistics for messages processed (MESSAGE), bytes transferred (BYTECOUNT), active time (ACTIVETIME), and elapsed time (ELAPSEDTIME).

To make the raw data in the server log meaningful, you can use a set of server log analysis programs to examine specific data resources and to create usable reports. See Chapter 7, “Analyzing the Server Log,” on page 87 for details about analyzing the server log.

The correct bibliographic citation for this manual is as follows: SAS Institute Inc., *SAS/SHARE User's Guide, Version 8*, Cary, NC: SAS Institute Inc., 1999. pp. 247.

SAS/SHARE User's Guide, Version 8

ISBN 1–58025–478–0

SAS Institute Inc., SAS Campus Drive, Cary, North Carolina 27513.

1st printing, September 1999