CHAPTER

11

OpenVMS: TCP/IP Access Method

Tasks That Are Common to SAS/CONNECT and SAS/SHARE

System Administrator or User

To use the TCP/IP access method with an OpenVMS host for SAS/CONNECT and SAS/SHARE, perform these tasks:

1. Verify that you have met all your site and software requirements.
2. Verify that you know how to set variables in SAS software.
3. Set the SAS/CONNECT and SAS/SHARE variables.

System and Software Requirements for SAS/CONNECT and SAS/SHARE

The following sections describe the system requirements for using the TCP/IP access method, which supports two particular host platforms: VAX and AXP.

TCP/IP with OpenVMS on VAX

To use the TCP/IP access method with OpenVMS on a VAX system, you must have Version 5.3 or a subsequent version of OpenVMS and one of the following packages installed on any OpenVMS node that is used as a local or a remote host.

1. DEC TCP/IP Services for OpenVMS, Version 2.0.
2. Version 1.3a of VMS/ULTRIX Connection with the latest patch kit from Digital Equipment Corporation Customer Support Center or a subsequent version of VMS/ULTRIX Connection.
3. Wollongong PathWay Runtime Version 1.1 through Version 3.5.
4. TGV MultiNet Software with UCX compatibility.
5. Process Software TCPware for OpenVMS with UCX compatibility.
6. any package that provides an interface that is compatible with DEC TCP/IP Services for OpenVMS, formerly known as UCX.

TCP/IP with OpenVMS on AXP

To use the TCP/IP access method with OpenVMS on an AXP system, your system must be running Release 6.09 or a subsequent release of SAS/CONNECT or SAS/SHARE and one of the following:

1. DEC TCP/IP Services for OpenVMS, Version 3.0 or a subsequent release.
2. TGV MultiNet Software with UCX compatibility.
3. Wollongong PathWay with UCX compatibility, Version 1.1 through Version 3.5.
4. Process Software TCPware For OpenVMS with UCX compatibility.
5. any package that provides an interface that is compatible with DEC TCP/IP Services for OpenVMS, Version 3.0 or a subsequent release.

Setting SAS Options and Variables

You may need to set specific SAS options and variables to allow the desired connections with SAS/CONNECT or SAS/SHARE when using the TCP/IP communications access method.

Consult with your network administrator to determine what variables must be set and what values to assign to them.

You may specify a SAS variable in one of these forms:

- an OPTIONS statement in a SAS AUTOEXEC file or in a SAS session:

  OPTIONS *variable=value*;

  Example:

```
OPTIONS COMAMID=TCP;
```

- in a SAS configuration file or at a SAS invocation:

  /variable-name=value

  Example:

```
/COMAMID=TCP
```

- as a SAS macro variable:

  %LET *variable:==value*;

  Example:

```
%let TCPSEC=_SECURE_;
```

- as an environment variable in DCL:

  variable:==value

  Example:

```
TCPSEC:==_SECURE_
```

Values for these variables can contain up to eight characters, consisting of alphanumeric characters, the percent sign (%), the dollar sign ($), the pound sign (#), the at sign (@), and the underscore (_).

Note: If you set the same variable using different forms, typically the last variable setting will take precedence and override an earlier variable setting. △

Displaying SAS System Option Settings

To display the settings of the SAS system options in the SAS log, use the OPTIONS procedure. The following statement produces a list of options with a brief explanation of what each option does:

```
proc options; run;
```

Setting Security for SAS/CONNECT and SAS/SHARE

For SAS/CONNECT, you must supply identifying information to sign on without a script to a remote host running a spawner program. A SAS/SHARE server, running secured, requires identification from each connecting client. The next two sections outline the version-specific methods for specifying client identification for SAS/CONNECT and SAS/SHARE. The third section describes how to configure your SAS/SHARE server to either require or not require connecting clients to supply user identification.

Providing Client Identification in a Version 8 Session

In Version 8, you provide client identification to a SAS/CONNECT remote host or a SAS/SHARE server using the USER= and PASSWORD= options. These options are valid in the following statements:

SIGNON

RSUBMIT

LIBNAME

PROC SQL
Connect to Remote

PROC OPERATE
(in the PROC statement)
set server
stop server
quiesce server
start server
display server

Specifying client identification in the TCPSEC variable is still accepted but is not recommended in Version 8. The USER= and PASSWORD= options take precedence over the client TCPSEC variable when both are specified. For example, a SAS/SHARE client's execution of a LIBNAME statement with values assigned to the USER= and PASSWORD= options would override a TCPSEC variable setting in the same client SAS session.

Here is the syntax and definitions for these options:

USER | **USERNAME** | **USERID** | **UID**=*username* | _PROMPT_

PASSWORD | **PASSWD** | **PASS** | **PWD** | **PW**=*password* | _PROMPT_

Specifying these options allows a user on the local host whose username and password have been verified to access the remote host.

username
is a valid userid for the remote host and is thus host-dependent in form. If the value contains blanks or special characters, it must be enclosed in quotes.

password
is the password, if any, required for authentication of the supplied username. This value will not be echoed in the SAS log. If the value contains blanks or special characters, it must be enclosed in quotes.

PROMPT
specifies that the SAS System prompts the client for *username* and *password*.

Note: The values provided when prompted must NOT be quoted. △

Specifying USER=_PROMPT_ and omitting the PASSWORD= specification will cause SAS to prompt you for both userid and password.

This is especially useful for allowing the SAS statements containing the USER= and PASSWORD= options to be copied and otherwise effectively reused by others.

For SAS/SHARE, the values supplied for the USER= and PASSWORD= options are valid for the duration of the remote host connection. Additional accesses of the remote host while the connection to that host is still in effect do not require re-supplying of the USER= and PASSWORD= options. For example, while the first connecting library assign to a SAS/SHARE server may require specification of the options, subsequent assigns to the same server will not need specification of these options as long as the original connection is in effect. A subsequent re-connect to the same server or connect to a different server would require re-supplying of the USER= and PASSWORD= options.

Here is a Version 8 example for SAS/SHARE:

```
libname test 'prog2 a' user=joeblue password="2muchfun" server=share1;
```

For SAS/CONNECT, these values are valid until SIGNOFF.

Here is a Version 8 example for SAS/CONNECT:

```
signon rmthost user=joeblack password=born2run;
```

As a security precaution, PASSWORD= field entries echoed in the log are replaced with Xs. If _PROMPT_ was specified for entering the password, the entry would not be displayed on the screen as it is typed.

Providing Client Identification in a pre-Version 8 Session

In Version 6 and Version 7, you provide client identification to a SAS/CONNECT remote host or a SAS/SHARE server using the TCPSEC variable. TCPSEC must be defined on the local host before you connect to the remote host (using the SIGNON statement) or access a SAS/SHARE server (using the LIBNAME statement).

Here is the syntax and description of this variable.

TCPSEC:==*userid.password* | _PROMPT_

userid.password
specifies the remote host userid and password and is thus host-dependent in form. If either the userid or password contains blanks or special characters, it must be enclosed in quotes. A period (.) is used as a delimiter between the userid and password; and therefore is not a valid character.

PROMPT
specifies that the SAS system prompt the client for the userid and password.

Note: The values provided when prompted must NOT be quoted. △

This technique is especially useful when the configuration file specifying this variable is shared among many users.

Examples:

```
TCPSEC:==bass.time2go
TCPSEC:==_PROMPT_
```

Providing Userid-Based Security for a SAS/SHARE Server

The TCPSEC variable also specifies whether the TCP/IP access method performs user authentication before connecting to a SAS/SHARE server. The TCPSEC variable must be set on the server before you start the SAS/SHARE server.

Here is the syntax and description of this variable.

TCPSEC:==_SECURE_

SECURE
The _SECURE_ value for the TCPSEC variable causes the TCP/IP access method to attempt to authenticate connecting SAS/SHARE clients. Each client connecting using TCP/IP is required to supply a userid and password valid for the host on which the server is running.

Examples:

```
TCPSEC:==_SECURE_
```

SAS/CONNECT Only Options and Variables

TCPPORTFIRST

TCPPORTLAST

The TCPPORTFIRST and TCPPORTLAST options restrict the range of TCP/IP ports through which local hosts can remotely connect to remote hosts.

These options must be set at the SAS/CONNECT remote host.

Define the range of TCP/IP ports by assigning a beginning range value to TCPPORTFIRST and an ending range value to TCPPORTLAST, within the range of 0 through 32767.

Consult with your network administrator for advice about these settings.

Use the following syntax for the configuration file or SAS invocation:

```
/TCPPORTFIRST=n /TCPPORTLAST=n
```

In the following example, the local host is restricted to TCP/IP ports 4020 through 4050 when making a remote host connection:

```
/tcpportfirst=4020;
/tcpportlast=4050;
```

To restrict the range of ports to only one port, you may set the TCPPORTFIRST and TCPPORTLAST options to the same number.

Note: On the remote host, you may set TCPPORTFIRST and TCPPORTLAST at a SAS invocation or in the configuration file. △

TCPTN3270

TCPTN3270 is an environment variable that is set on the local host to support connections to CMS and OS/390 remote hosts that use the full-screen 3270 TELNET protocol. The following sample script files are provided with SAS/CONNECT:

CMS	TCPCMS32.SCR
OS/390	TCPTSO32.SCR

See “Identifying a Script File for Signing Off and Signing On” on page 178 for more information about these script files.

Note: You must use the environment variable form to set TCPTN3270. △

To set the TCPTN3270 variable, enter the following command on the OpenVMS local host:

```
TCPTN3270:==1
```

If this variable is not set, the TCP/IP access method will use the TELNET line mode protocol by default.

SAS/SHARE AUTHENCR Variable

AUTHENCR:==OPTIONAL | REQUIRED

or

%let AUTHENCR=OPTIONAL | REQUIRED;

By default, a secure server accepts userids and passwords from clients in either encrypted or plain text form. The option to accept either form ensures compatibility with client sessions running older releases of SAS software.

To require only encrypted userids and passwords, you must set the AUTHENCR environment variable or a SAS macro variable. Requiring encryption requires that all clients have been upgraded to Release 6.11 or the 6.09 Enhanced Release of SAS software.

Setting this variable in a server session enables encryption for clients connecting to a secured server. The values for this variable follow:

OPTIONAL
means that a client can optionally encrypt the username and password that it sends to the server. This is the default. When using the default, the server allows connections from clients that are incapable of using encryption because they are running earlier versions of SAS that do not support encryption (releases prior to the 6.09 Enhanced Release as well as Release 6.11) and from clients that are capable of encryption.

REQUIRED
means that each client must encrypt the username and the password that it sends to the server.

See "Setting SAS Options and Variables" on page 170 for examples of the forms that you can use to specify the AUTHENCR variable.

SAS/CONNECT

OpenVMS VAX and Alpha Platforms

Digital Equipment Corporation offers the OpenVMS operating system on VAX and Alpha platforms. In most ways, SAS runs the same on both hardware platforms. However, certain differences in the hardware require SAS to operate somewhat differently on the two platforms.

The OpenVMS Alpha spawner program that is documented in Chapter 33, "OpenVMS Alpha Spawner Program," on page 459 runs on the OpenVMS Alpha platform only and cannot be run on the OpenVMS VAX platform.

Local Host Tasks

User or Applications Programmer
To connect an OpenVMS local host to a remote host, perform these tasks at the local host:

1. Consider the requirements of the remote host that you are connecting to.
2. If you connect to a UNIX, an OS/390, or an OpenVMS Alpha remote host by means of the spawner, configure the spawner service in the SERVICES file, as necessary.
3. If you connect to a remote host by means a spawner program, set security, as applicable.
4. Configure the local and the remote hosts' names and Internet addresses in the local HOSTS file or through the domain server.
5. Specify the communications access method.
6. Specify a remote host to connect to.
7. Identify the script file to be used for signing on and signing off, as necessary.
8. Sign on to the remote host.

Remote Host Connection Considerations

If you are connecting to a Windows 95, Windows 98, or Windows NT remote host, you must connect by means of a spawner program that is already running on the remote host. If you are connecting to an OS/2, a UNIX, an OS/390, or an OpenVMS Alpha remote host, you optionally may connect by means of a spawner program, which also must already be running on the remote host. A spawner program allows the encryption of userids and passwords when passed through the network. Without a spawner, readable userids and passwords are passed through the network, which may present a security risk. See Chapter 32, "Spawner Programs," on page 457 for information about starting the spawner on the remote host.

You may also sign on to the remote host with a script file. If you do not sign on with a script file, as a security measure, set the USER= and PASSWORD= options in the SIGNON statement, which is passed to the remote host, allowing a local host connection.

Note: Setting the Version 7 TCPSEC variable at the local host will also work. △

If the -NOSCRIPT option is set at the spawner invocation, sign on with a script is prohibited. Ask your network administrator whether the -NOSCRIPT option is set at the spawner invocation.

For all other hosts, you will sign on with a script.

Configuring the Spawner Service in the SERVICES File

To prepare for local hosts that connect to a UNIX, an OS/390, or an OpenVMS remote host with the spawner program, configure the spawner service in the SERVICES file at the local host. See "Configuring the SERVICES File" on page 485 for more information.

Setting Security for Local Hosts

If you are not using a script file to sign on to the remote host, set security at the local host using either of the methods explained in "Setting Security for SAS/CONNECT and SAS/SHARE" on page 171. For Version 8 security behavior, specify the USER= and PASSWORD= options to the SIGNON statement. For details, see "Providing Client Identification in a Version 8 Session" on page 171.

For Version 7 security behavior, if you set the TCPSEC variable at the local host, either specify a userid and a password that are valid on the remote host or specify _PROMPT_ to supply the userid and password when connecting to a remote host. For information about setting the TCPSEC variable, see " Providing Client Identification in a pre-Version 8 Session" on page 173.

Configuring Local and Remote Host Names and Internet Addresses

You must specify the names and Internet addresses of the local and the remote hosts in the HOSTS file or through a name server. A name server supplies name-to-address translation, mapping from domain names to IP addresses. The name server often runs on a dedicated processor, and the host itself is referred to as the name server.

The format for a HOSTS file entry follows:

Internet-address host-name optional-alias

Example:

```
172.20.10.200      monarch       local 172.20.10.201
  omega          remote
```

Specifying the TCP/IP Communications Access Method

You must specify the TCP/IP communications access method to make a remote host connection. Use the following syntax:

```
OPTIONS COMAMID=access-method-id;
```

where COMAMID is an acronym for Communications Access Method Identification. *access-method-id* identifies the method used by the local host to communicate with the remote host. **TCP**. TCP (short for TCP/IP, which is an abbreviation for Transmission Control Protocol/Internet Protocol) is as example of *access-method-id*.

Example:

```
options comamid=tcp;
```

Alternatively, you may set this option in a SAS command or in a SAS configuration file.

Specifying the Remote Node Name

To make a connection from an OpenVMS local host to a remote host, use the following syntax:

```
OPTIONS
REMOTE=node-name.<service-name>;
```

The value of *node-name* that you specify depends on the type of remote host that you are connecting to.

- If you are connecting to a Windows NT, a Windows 95, a Windows 98, or an OS/2 remote host that is running the PC spawner program, use the name of the node on which the PC spawner is running. See Chapter 32, “Spawner Programs,” on page 457 for more information.
- If you are connecting to a UNIX, an OS/390, or an OpenVMS Alpha host that is running the spawner program, use a two-level name in the form of *node-name.service-name*, where *node-name* specifies the node on which the spawner program is running and *service-name* specifies the port on which the spawner is listening for a connection request.

  See Chapter 32, “Spawner Programs,” on page 457 for more information about the spawner program and “Configuring the SERVICES File” on page 485 for details about configuring the spawner service in the SERVICES file.
- If you are connecting to any other remote host platforms that use a sign-on script, use the node name of the remote host. The remote host must be defined in a local HOSTS file or in a domain name server.

The value of the REMOTE option must be a valid SAS name. See *SAS Language Reference: Dictionary* for details about SAS naming rules.

Example:

```
OPTIONS REMOTE=node-name;
```

If you use an Internet address or some other invalid SAS name, you must assign the address to a macro variable and specify the macro variable for the REMOTE value, illustrated as follows:

```
%let node=Internet-address;
options remote=node;
```

Do not choose a macro name that is also a valid host name on your network. SAS first attempts to reach a network host with the REMOTE option value (in this example, MYNODE).

Example:

```
%let mynode=172.20.10.200; options remote=mynode;
```

Identifying a Script File for Signing Off and Signing On

To use one of the sample script files that is supplied with SAS/CONNECT for signing on and signing off, assign the RLINK fileref to the appropriate script file, depending on which remote host you are connecting to. The sample script files are located in *!SAS$ROOT*:[TOOLS]. You must customize the sample scripts to accurately reflect your site logon process. Failure to do so will produce errors.

The fileref format follows:

```
FILENAME
RLINK '!SAS$ROOT:[TOOLS]script-name';
```

where *script-name* identifies the script that corresponds to the remote host that you want to connect to. Names of scripts that are supplied by SAS Institute follow:

Table 11.1 OpenVMS TCP/IP SAS/CONNECT Sign-on Scripts

Remote Host	Script Name
CMS	TCPCMS.SCR
CMS (using full-screen 3270 TELNET protocol)	TCPCMS32.SCR
OS/390 (with TSO)	TCPTSO.SCR
OS/390 (without TSO)	TCPMVS.SCR
OS/390 (using full-screen 3270 TELNET protocol)	TCPTSO32.SCR
OpenVMS	TCPVMS.SCR
OS/2	TCPOS2.SCR
UNIX	TCPUNIX.SCR
Windows NT, Windows 95, and Windows 98	TCPWIN.SCR

Note: You will not use a script file if you are signing on to a UNIX or a PC spawner program that was invoked with the -NOSCRIPT option. Consult with your network administrator to find out whether the spawner was invoked with this option. △

Signing On to the Remote Host

To complete your sign on to the remote host, enter the SIGNON statement, as follows:

```
signon user=_prompt_;
```

To set security at the remote host, specify valid values for the USER= and PASSWORD= options in the SIGNON statement. For details, see "Providing Client Identification in a Version 8 Session" on page 171.

Local Host Example

The following example illustrates the statements that you specify in an OpenVMS local host SAS session to connect to a remote host running the spawner program configured for the TCP/IP access method.

In the OpenVMS local host SAS session, specify the following:

```
filename rlink '!sasroot:[tools]tcpunix.scr';
options comamid=tcp remote=rmtnode.unxspawn;
signon;
```

The first line identifies the script file that you use to sign on to the UNIX remote host. The script file includes a prompt for a userid and a password that are valid on the remote host. The TCP/IP communications access method is declared with a connection to a remote UNIX spawner, which is identified by the two-level name RMTNODE.UNXSPAWN. The SIGNON statement performs the sign-on process.

Remote Host Example

You may set the following variables to restrict port access in the remote host configuration file:

```
/tcpportfirst=5020
/tcpportlast=5050
```

These statements restrict access to ports 5020 through 5050.

SAS/SHARE

Client Tasks

User and Applications Programmer

To prepare for accessing a SAS/SHARE server, perform the following tasks:

1. Configure the server in the client SERVICES file.
2. Set security for connecting clients, as desired.
3. Specify the TCP/IP access method.
4. Specify the server name.

Configuring the Server in the SERVICES File

Each server must be defined as a service in the SERVICES file on each host node from which a client session will access the server. This file usually is located in the directory in which the TCP/IP software is installed. See Chapter 37, "TCP/IP SERVICES File," on page 485 for information about editing the SERVICES file.

Setting Security for Connecting Clients

Requiring connecting clients to supply a valid userid and password enforces server security. At the client, set the preferred security method for relaying a userid and password that are valid on the server host. For details, see “Setting Security for SAS/CONNECT and SAS/SHARE” on page 171.

Specifying the TCP/IP Access Method

You must specify the TCP/IP communications access method at the client before you access a server.

Use the following syntax to specify the TCP/IP access method at each connecting client:

```
OPTIONS COMAMID=access-method-id;
```

Where COMAMID is an acronym for Communications Access Method Identification. *access-method-id* is an abbreviation for the method used by the client to communicate with the server. TCP (which is short for TCP/IP, is an abbreviation for Transmission Control Protocol/Internet Protocol) is an example of an *access-method-id*.

Example:

```
options comamid=tcp;
```

The server is accessed using the TCP/IP access method.

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

Additionally, you may use the COMAUX1 and COMAUX2 options to designate auxiliary communications access methods. See Table 1.3 on page 10 for the supported access methods by host. If the first method fails to access a server, the second method is attempted, and so on. You can specify up to two auxiliary access methods, depending on the number of methods that are supported between client and server hosts.

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 an OpenVMS client connecting to a Windows NT server follows.

```
/comamid=tcp /comaux1=decnet
```

If the server cannot be reached using the TCP/IP method, a second attempt is made with the DECnet access method.

Specifying a Server Name

The server name that you specify in the PROC SERVER, PROC OPERATE, and LIBNAME statements must be a defined SAS/SHARE TCP/IP service on the hosts where the server and the user session run.

Note: Do not use the pound sign (#) in a server node name. △

If the client and server sessions are running on different network nodes, you must include the TCP/IP node in the server identifier in the PROC OPERATE SET SERVER statement or in the LIBNAME and PROC OPERATE statements as follows:

```
SERVER=node.server
```

This representation is known as a two-level server name.

node must be a valid TCP/IP node name. If the server and the client sessions are running on the same node, you may omit the node name. If the TCP/IP node name is not a valid SAS name, you must assign the name of the server node to a SAS macro variable, then use the name of the macro variable for *node* in the two-level server name, or assign the node name to a DCL symbol, then use the DCL symbol for *node* in the two-level server name.

The access method evaluates the node name, using this order of precedence:

- acceptable node name
- SAS macro variable
- DCL symbol.

server can represent either a *server-id* or a *port* number.

- *server-id* must be identical to the service name specified in the SERVICES file. See "Configuring the SERVICES File" on page 485 for more information on specifying the *server-id* in the SERVICES file.
- *port* is the location for passing data to and receiving data from the server. The port number is specified with two preceding underscore (_) characters. For example, you can specify the server port as 5000 using the SERVER= option in a LIBNAME statement:

```
libname mylib '.' server=srvnode.__5000;
```

The following examples show the use of a SAS macro variable and a DCL symbol, respectively, to contain a server node name:

Example 1:

```
%let srvnode=mktserve.acme.com;
libname sales server=srvnode.server1;
```

The macro variable SERVNODE is assigned to the fully qualified node name and is then used in the two-level node name.

Example 2:

```
$srvnode:==mktserve.acme.com
```

The DCL symbol SRVNODE is assigned to the fully qualified node name

Note: Do not use an ampersand (&) in a two-level server name. An ampersand causes the macro variable to be resolved by the SAS parser prior to syntactic evaluation of the SERVER= option. The access method evaluates the node name in a two-level server name. △

See *SAS Language Reference: Dictionary* for details about SAS naming rules. See the *SAS/SHARE User's Guide* for details about the PROC SERVER, PROC OPERATE, and LIBNAME statements.

Client Example

The following example illustrates the statements that you specify in an OpenVMS client SAS session to connect to a server by using the TCP/IP access method:

```
options comamid=tcp; libname sasdata
[edc.prog2.sasdata] user=_prompt_ server=rmtnode.share1;
```

The COMAMID option declares the TCP/IP access method. The LIBNAME statement specifies the data library that is accessed through the server, which is specified by the two-level server name RMTNODE.SHARE1 by means of a prompt for a username and a password that are valid on the server.

Server Tasks

Server Administrator

To prepare for starting a SAS/SHARE server, perform the following tasks:

1. Configure SAS/SHARE servers in the SERVICES file.
2. Set server security through the TCPSEC variable.
3. Set the AUTHENCR environment variable to enforce userid and password encryption, as needed.
4. Specify the TCP/IP access method.
5. Specify the server name.

Configuring the Server Service in the SERVICES File

Each server must be defined as a service in the SERVICES file on each remote host node on which a server runs and on each node from which a client will access the server. This file is located in the directory in which the TCP/IP software is installed. Find out the correct location of the TCP/IP software on your host system. See "Configuring the SERVICES File" on page 485 for information about configuring a server in the SERVICES file.

Setting Server Security

You may use file permissions to restrict a user's access to libraries and files through a server. A secured server allows connections only from those clients that provide valid userids and passwords for the host at which the server is running. A secured server uses a valid userid and password pair to verify a user's authority to access a SAS library or file.

Requiring connecting clients to supply both a valid userid and password enforces server security. From a server session, set the TCPSEC variable to the value _SECURE_. See " Providing Client Identification in a pre-Version 8 Session" on page 173 for more information about setting this variable.

Enforcing Server Userid and Password Encryption

As a security measure, you may set the AUTHENCR variable to enforce the encryption of userids and passwords when they are passed from the client to the server. See "SAS/SHARE AUTHENCR Variable" on page 174 for details about setting the AUTHENCR variable.

Specifying the TCP/IP Communications Access Method

You must specify the TCP/IP communications access method at the server before you create a SAS/SHARE server.

Use the following syntax to specify the TCP/IP access method at the server:

```
OPTIONS COMAMID=access-method-id;
```

where COMAMID is an acronym for Communications Access Method Identification. *access-method-id* is an abbreviation for the method used by the server to communicate

with the client. TCP (short for TCP/IP, which is an abbreviation for Transmission Control Protocol/Internet Protocol) is an example of an *access-method-id*.

For a server that is running on a host on which only one communications access method is available, use the COMAMID option.

Example:

```
options comamid=tcp;
```

The server will be available only to SAS/SHARE sessions that use the TCP/IP access method.

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

If, however, the host on which a 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. See Table 1.3 on page 10 for the supported access methods by host.

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 communications 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
```

Here is an example of configuration file entries for a server that is running on an OpenVMS host.

```
/comamid=tcp /comaux1=decnet
```

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

Specifying a Server Name

The server name that you specify in the PROC SERVER statement must be a defined SAS/SHARE TCP/IP service on the hosts where the server and the user session run. Use the following syntax:

```
SERVER=server
```

server can represent either a *server-id* or a *port* number.

- *server-id* corresponds to the service that was configured in the SERVICES file. See “Configuring the SERVICES File” on page 485 for more information.
- *port* is the location for passing data to and receiving data from the server. The port number is specified with two preceding underscore (_) characters. For example, you can specify the server port as 5000 using the SERVER= option in a LIBNAME statement:

  ```
  libname mylib '.' server=__5000;
  ```

Server Example

The following example illustrates the statements that you specify in a SAS session on the OpenVMS host at which you start a server:

```
%let tcpsec=_secure_;
%let authencr=required;
options comamid=tcp;
proc server id=share1;
run;
```

The _SECURE_ value for the TCPSEC macro variable requires clients to supply a userid and a password that are valid at the server. The AUTHENCR variable enforces encryption of userids and passwords when passed from the client to the server.

The TCP/IP access method is declared and the server SHARE1 is started on the OpenVMS host.

The correct bibliographic citation for this manual is as follows: SAS Institute Inc., *Communications Access Methods for SAS/CONNECT and SAS/SHARE Software, Version 8*, Cary, NC: SAS Institute Inc., 1999. pp. 643.

Communications Access Methods for SAS/CONNECT and SAS/SHARE Software, Version 8

ISBN 1–58025–479–9

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

1st printing, September 1999