CHAPTER 20

UNIX: 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 a UNIX 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 desired SAS/CONNECT and SAS/SHARE variables.

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

Ensure that the following conditions have been met:

1. SAS software has been installed on both the local and remote hosts.
2. Any TCP/IP package that comes with the operating system has been installed.

The kernel layer in all types of UNIX systems provides the necessary software to communicate across a network.

Setting SAS Options and Variables

You may need to set specific options or variables in SAS to establish the connections that you want with SAS/CONNECT and 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 variable in any of several forms, as follows:

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

  OPTIONS *variable=value*;

  Example:

```
OPTIONS TCPPORTFIRST=4020;
```

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

  -SET *variable-name value*

  Example:

```
-set tcpsec _secure_
```

- as a SAS macro variable:

  %LET *variable-name=value*;

  Example:

```
%let tcpsec=_secure_;
```

- as a UNIX environment variable in a shell or a profile file:

  Korn shell: export *VARIABLE-NAME=value*

  C shell: setenv *VARIABLE-NAME value*

  Examples:

```
export TCPSEC=_SECURE_
setenv TCPSEC _SECURE_
```

Note: Variable names must be in uppercase. △

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

If you set multiple forms of the same variable, here is the order of precedence that is followed:

SAS macro variable

AUTOEXEC file

SAS invocation

SAS configuration file

UNIX environment variable.

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

Note: The value of TCPSEC will be set and displayed like any other macro variable. Thus if the %LET statement that is used to set the value of TCPSEC appears in the SAS log, the password will also appear in plain text. Similarly, a %PUT statement will also print the password in plain text. △

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:

```
%let tcpsec=bass.time2go;
%let 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_ | _NONE_

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.

NONE
The _NONE_ value for the TCPSEC variable causes the TCP/IP access method to NOT attempt to authenticate connecting SAS/SHARE clients. This is the default action when TCPSEC has not been set.

Examples:

```
%let tcpsec=_secure_;
%let tcpsec=_none_;
```

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:

```
-TCPPORTFIRST n
-TCPPORTLAST n
```

Use the following syntax for the AUTOEXEC file:

```
OPTIONS TCPPORTFIRST=n;
OPTIONS TCPPORTLAST=n;
```

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

```
options tcpportfirst=4020;
options 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: At the remote host, you may set TCPPORTFIRST and TCPPORTLAST in an OPTIONS statement, at a SAS invocation, in the configuration file, or in the AUTOEXEC 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 script files are provided:

CMS	TCPCMS32.SCR
OS/390	TCPTSO32.SCR

See “Identifying a Script File for Signing On and Signing Off” on page 302 for information about these script files.

At the UNIX local host, set the TCPTN3270 variable in the SAS configuration file as follows:

```
-set TCPTN3270 1
```

If you do not set this variable, the TCP/IP access method uses TELNET line mode protocol by default.

SAS/SHARE Only Option

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

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

Setting this option in a server session enables encryption for clients that are connecting to a secured server. The forms of this option follow:

AUTHENCR=OPTIONAL | REQUIRED

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 releases of SAS/SHARE 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 password that it sends to the server. See "Setting SAS Options and Variables" on page 294 for examples of the forms that you can use to specify the AUTHENCR option.

SAS/CONNECT

Local Host Tasks

User or Applications Programmer
To connect a UNIX local host with 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 program, then configure the spawner in the SERVICES file, as necessary.
3. If you connect to a remote host by means of a spawner program, then set security, as necessary.
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 the remote node name.
7. Specify a sign-on script, 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 or VAX remote host, you *may* connect by means of a spawner program that also is already 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 to the SIGNON statement, which is passed to the remote host, allowing a local host connection.

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 Alpha 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 295. For Version 8 security behavior, specify the USER= and PASSWORD= options in the SIGNON statement. For details, see “Providing Client Identification in a Version 8 Session” on page 295.

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 297.

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 **/etc/hosts** file or through the name server. A server program supplies name-to-address translation, mapping from domain names to IP addresses. The server processor often runs on a dedicated processor, and the host itself is referred to as the name server.

The format for an **/etc/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

Note: TCP/IP is the default communications access method on the UNIX platforms. You may omit specifying the access method in a COMAMID statement and the TCP/IP access method is assumed, by default. △

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

Example:

```
options comamid=tcp;
```

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

Specifying the Remote Node Name

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

```
OPTIONS REMOTE=node-name[.service-name];
```

The value that is specified for *node-name* is based 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 information about the spawner program and “Configuring the SERVICES File” on page 485 for information 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=rmthost;
```

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 value of the REMOTE= option, illustrated as follows:

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

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 value of the REMOTE= option (in this example, MYNODE).

Example:

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

Identifying a Script File for Signing On and Signing Off

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, based on the remote host that you are connecting to. The sample scripts are installed at *!sasroot* **/ misc/connect.** You must customize the sample scripts to accurately reflect your site logon procedures. Failure to do so will produce errors.

The FILEREF syntax follows:

```
FILENAME RLINK '!sasroot/misc/connect/script-name';
```

where *script-name* specifies the appropriate script file for the remote host.

The following table lists the scripts that are supplied by SAS Institute:

Table 20.1 UNIX TCP/IP SAS/CONNECT Sign-on Scripts

Remote Host	Script Name
CMS	**tcpcms.scr**
CMS (using full-screen 3270 TELNET protocol)	**tcpcms32.scr**
TSO under OS/390	**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**

Example:

```
filename rlink '!sasroot/connect/saslink/tcptso.scr';
```

Note: If you connect to a PC or UNIX spawner program, then you may optionally sign on with a script file unless the spawner is invoked with the -NOSCRIPT option. In this case, you cannot use a script file. With no script file, you do not define an RLINK fileref statement. See Chapter 32, “Spawner Programs,” on page 457 for more information. △

Note: If you connect to an OS/2 remote host by means of a TELNET daemon, then your script must invoke the SASDMR program in the TYPE statement that starts the remote SAS session. △

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 295.

If the NOSCRIPT option was set when the UNIX spawner program was invoked, then you cannot use a script file. Ask the network administrator if sign ons with scripts are allowed. An example of signing on without a script follows:

```
signon noscript;
```

Local Host Example

The following example illustrates the statements that you specify in a UNIX local host SAS session to connect to a remote host with the TCP/IP access method:

```
filename rlink '!sasroot/misc/connect/tcptso.scr';
options comamid=tcp remote=rmtnode;
signon;
```

The first line identifies the script file that you use to sign on to an OS/390 remote host. The script file contains 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 the remote host RMTNODE.

Remote Host Tasks

System Administrator

Before a user on a local host can sign on to a UNIX remote host by means of a UNIX spawner program, perform the following tasks:

1. If you connect to the UNIX spawner program, configure the UNIX spawner service in the **/etc/services** file at the remote host.
2. Start the UNIX spawner program at the remote host, as necessary.

Configuring the UNIX Spawner Service in the SERVICES File

To prepare a local host to connect to a UNIX remote host by means of the UNIX spawner program, configure the spawner service in the **/etc/services** file on the remote host. See "Configuring the SERVICES File" on page 485 for more information

Starting the UNIX Spawner Program at the Remote Host

Note: Running the UNIX spawner program on a UNIX remote host is optional. △

You optionally invoke the UNIX spawner program on the UNIX remote host to enable local hosts to connect to it. The spawner program resides on a remote host and listens for SAS/CONNECT client requests for connection to the remote host. After the spawner program receives a request, it invokes the remote SAS session. See Chapter 36, "UNIX Spawner Program," on page 479 for information about starting the UNIX spawner program.

Note: If you set the -NOSCRIPT option at the UNIX spawner invocation, inform local users that they cannot sign on with a script file. △

If you connect to a UNIX spawner without a script file, you must invoke the spawner with the -SASCMD option and an executable file on the remote host. The executable file starts the remote SAS session.

Remote Host Example

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

```
options tcpportfirst=5020;
options tcpportlast=5050;
```

These statements restrict access to ports 5020 through 5050.

A typical example of how to invoke the UNIX spawner program at the UNIX remote host follows:

```
!sasroot/utilities/bin/sastcpd -service unxspawn
```

The UNIX spawner program UNXSPAWN is invoked.

SAS/SHARE

Client Tasks

User or Applications Programmer

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

1. Configure the server in the client SERVICES file.
2. Set security for connecting clients.
3. Specify the TCP/IP access method.
4. Specify a 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 295.

Specifying the TCP/IP Communications 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* identifies the method that is used by the client to communicate with the server. TCP (short for TCP/IP, which 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 option to designate an auxiliary communications access method. See Table 1.3 on page 10 for the supported access methods by host. If the COMAMID method fails to access a server, the second method is attempted. You can specify only one auxiliary access method.

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

```
-COMAUX1 alternate-method
```

An example of configuration file entries for an UNIX client connecting to an OS/390 server follows:

```
-comamid tcp
-comaux1 appc
```

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

Specifying a Server 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 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.

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;
```

If the TCP/IP node name is not a valid SAS name, assign the name of the server node to a SAS macro variable, then use the name of that macro variable for *node* in the two-level server name.

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

- □ acceptable node name
- □ SAS macro variable
- □ environment variable.

The following example shows how to use a SAS macro variable to relay a server node name:

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

Note: Do not use an ampersand (&) in a two-level 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 *SAS/SHARE User's Guide* for details about the LIBNAME and PROC SERVER statements.

Client Example

The following example illustrates the statements that you specify in a UNIX client SAS session to access a server with the TCP/IP access method:

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

The TCP/IP access method is declared. 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 set up a secure server, perform the following tasks at the server:

1. Configure SAS/SHARE servers in the **/etc/services** file.
2. Set the TCPSEC variable for server security.
3. Set the AUTHENCR variable to enforce client userid and password encryption.
4. Configure the authorization of users on remote hosts.
5. Ensure that ***!sasroot*/utilities/bin/sasauth** is owned by ROOT and that the "Set-user-id" bit is set for the file (**chmod 4755 *!sasroot*/utilities/bin/sasauth**).
6. Ensure that ***!sasroot*/utilities/bin/sasperm** is owned by ROOT and that the "Set-user-id" mode bit is set for the file (**chmod 4755 *!sasroot*/utilities/bin/sasperm**).
7. Specify the TCP/IP access method.
8. Specify the server name.

Note: Optional tasks apply to setting up server security. △

Configuring the Server in the /etc/services File

Each server must be defined as a service in the **/etc/services** file on each remote host node from which a client session will access the server. A typical entry follows:

```
sassrv2   5011/tcp  # SAS/SHARE server 2
```

See "Configuring the SERVICES File" on page 485 for information about editing the **/etc/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 on which the server is running. A secured server uses a validated userid and password pair to verify a user's authority to access a SAS library or a SAS file.

Requiring connecting clients to supply 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 297 for more information about setting this variable.

Enforcing Server Userid and Password Encryption

As a security measure, you may set the AUTHENCR option to enforce the encryption of userids and passwords when passed from the client to the server. See "SAS/SHARE Only Option" on page 298 for details about setting AUTHENCR.

Configuring User Authorization

If SAS was installed from the root account, then it can be assumed that this task has already been performed. Otherwise, a root user must configure resources on the remote host on which a server runs in order to authenticate a remote user's identity and to check the user's authority to access resources.

Perform these tasks through the SAS Setup menu or by issuing the corresponding UNIX commands at a shell prompt:

From the root account, at the command line, initialize **!*SASROOT*/sassetup**, and from the SAS Setup Primary Menu, select:

```
Run setup Utilities -> Perform SAS System Configuration ->
Configure User Authorization
```

where **!*SASROOT*** is the directory where SAS was installed.

Alternatively, issue these UNIX commands at a shell prompt:

```
su root
cd !SASROOT/utilities/bin
chown root sasauth sasperm sastcpd objspawn
chmod 4755 sasauth sasperm sastcpd objspawn
exit
```

Validating Client Userid and Password Pairs with the Authenticate Program

Note: This feature applies to a server that is running on a UNIX host only. △

You may use a built-in authentication program named **sasauth**, which is invoked automatically when a client accesses a server that is running in secure mode. This program authenticates userid and password pairs, which allow client access to the server.

To secure a server, the server administrator sets the TCPSEC environment variable to _SECURE_. See "Providing Client Identification in a pre-Version 8 Session" on page 297 for information about TCPSEC.

Note: For Version 6, the **authenticate** program is used to validate server userid and passwords. See Chapter 39, "Authenticate Program," on page 495 for details about using this program. △

Allowing Client Access to SAS Libraries or Files with the Permission Program

Note: This feature applies to a server that is running on a UNIX or a CMS host only. △

When presented with a validated userid, the server uses a default program named **sasperm** to verify the following attributes:

- username
- file or directory path for a SAS library or file
- file or directory access permissions (read or write).

The **sasperm** program determines whether the requesting user has access to the file or directory.

Note: For Version 6, the **permission** program enables clients to access SAS libraries or files. See Chapter 40, "Permission Program," on page 499 for details about using this program. △

Specifying the TCP/IP Access Method

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

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* identifies the method that is 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 only 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, on the SAS invocation, or in a SAS configuration file.

However, if the host on which a server is running supports multiple access methods, you may specify an auxiliary access method by which clients may access the server. 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.

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

```
-COMAUX1 alternate-method
```

An example of configuration file entries for a server that is running on a CMS host follows:

```
-comamid tcp
-comaux1 appc
```

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 APPC access method.

Specifying a Server Name

You must specify the server name in the PROC SERVER statement. 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=srvnode.__5000;
```

The following example shows how to use a SAS macro variable to relay a server node name:

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

See *SAS Language Reference: Dictionary* for details about SAS naming rules. See the *SAS/SHARE User's Guide* for details about the PROC SERVER statement.

Server Example

The following example illustrates the statements that you specify in the server configuration file on a UNIX host:

```
-set tcpsec _secure_
-set authencr required
```

The value _SECURE_ for the TCPSEC variable requires clients to supply a userid and password that are valid on the server. The value REQUIRED for AUTHENCR allows only encrypted userids and passwords from clients.

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

```
options comamid=tcp;
proc server id=share1 authenticate=req;
run;
```

The TCP/IP access method is declared and the server SHARE1 is started on the UNIX host. The additional options in the PROC SERVER statement allow only validated clients to access the server.

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