CHAPTER 29

Windows: TCP/IP Access Method

SAS Support for TCP/IP on Windows **416**
Tasks That Are Common to SAS/CONNECT and SAS/SHARE **416**
- *System and Software Requirements for SAS/CONNECT and SAS/SHARE* **416**
  - *Windows NT, Windows 95, and Windows 98 Requirements* **416**
  - *Windows NT Only* **416**
  - *Windows 32s Requirements* **416**
- *Setting SAS Options and Variables* **417**
  - *Displaying SAS System Option Settings* **417**
- *Setting Security for SAS/CONNECT and SAS/SHARE* **418**
  - *Providing Client Identification in a Version 8 Session* **418**
  - *Providing Client Identification in a pre-Version 8 Session* **419**
  - *Providing Userid-Based Security for a SAS/SHARE Server* **420**
- *SAS/CONNECT and SAS/SHARE Options* **420**
- *SAS/CONNECT Only Options and Variables* **421**
- *SAS/SHARE Only Options* **423**

SAS/CONNECT **424**
- *Local Host Tasks* **424**
- *Remote Host Connection Considerations* **424**
- *Configuring the Spawner Service in the SERVICES File* **425**
- *Setting Security for Local Hosts* **425**
- *Configuring Local and Remote Host Names and Internet Addresses* **425**
- *Specifying the TCP/IP Communications Access Method* **426**
- *Specifying the Remote Node Name* **426**
- *Identifying a Script File for Signing On and Signing Off* **427**
- *Signing On to the Remote Host* **428**
- *Local Host Example* **428**
- *Remote Host Tasks* **428**
- *Starting the PC Spawner Program* **428**
- *Remote Host Example* **429**

SAS/SHARE **429**
- *Client Tasks* **429**
- *Configuring the Server in the SERVICES File* **429**
- *Assigning the Appropriate Rights for Connecting Clients* **429**
- *Setting Security for Connecting Clients* **430**
- *Specifying the TCP/IP Communications Access Method* **430**
- *Specifying a Server Name* **431**
- *Client Example* **432**
- *Server Tasks* **432**
- *Configuring the Server in the SERVICES File* **433**
- *Setting Server Security* **433**
- *Enforcing Server Userid and Password Encryption* **433**

Assigning the Appropriate Rights for a Secure Server **433**
Specifying the TCP/IP Access Method **433**
Specifying a Server Name **434**
Server Example **435**

SAS Support for TCP/IP on Windows

Note: The TCP/IP communications access method can be used with a Windows NT, a Windows 95, a Windows 98, and a Windows 32s host. For SAS/CONNECT, the Windows 32s can be used as a local host only.

However, beginning with Version 7, the Windows 32s platform is not supported. Information about Windows 32s is included here for Version 6 users. △

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

System Administrator or User

To use the TCP/IP access method with a Windows 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 the resources for the TCP/IP access method have been defined.
3. Verify that you know how to set options in SAS software.
4. Set the SAS/CONNECT and SAS/SHARE options that you want.

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

Ensure that the following conditions have been met:

1. A supported TCP/IP package must be installed at both the local and remote hosts at your site.
2. SAS software has been installed on both the local and remote hosts.

Windows NT, Windows 95, and Windows 98 Requirements

To use the TCP/IP access method with Windows NT, Windows 95 and Windows 98, install and configure:

1. the Microsoft TCP/IP System Driver, which is provided with Windows NT, Windows 95, and Windows 98.

Windows NT Only

Ensure that Windows NT 4.0 or a subsequent release has been installed.

Windows 32s Requirements

To use the TCP/IP access method with Windows 32s, install and configure one of the following emulation packages:

1. the Novell LAN Workplace for DOS Version 4.2 or subsequent version.
2. the Microsoft LAN Manager Version 2.1 or subsequent version.

3 any TCP/IP package that provides a Winsock V1.1 API and subsequent versions of API.

Setting SAS Options and Variables

You may need to set specific options or variables 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 options must be set and what values to assign to them.

You may specify an option in any of several forms, as follows:

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

  OPTIONS SET=*variable-name value*;

  Example:

  ```
  options set=tcpsec _secure_;
  ```

- □ SAS configuration file or at SAS invocation:

  -SET *variable-name value*

  Example:

  ```
  -set tcpsec _secure_
  ```

- □ SAS macro variable:

  %LET *variable-name=value*;

  Example:

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

- □ DOS operating system environment variable:

  SET *variable-name=value*

  Example:

  ```
  set tcpsec=_secure_
  ```

Values for these options or 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 underscore (_).

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

SAS macro variable

OPTIONS statement

AUTOEXEC file

SAS invocation

SAS configuration file

DOS 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

Note: In the Windows environment, SAS/SHARE server security is supported on the Windows NT platform only. △

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 option is still accepted but is not recommended in Version 8. The USER= and PASSWORD= options take precedence over the client TCPSEC option 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 option setting in the same client SAS session.

CAUTION:
In order to make a SAS/SHARE server secured, the TCPSEC option must be set at a SAS/SHARE server that can run on any host. △

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 option. 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 option.

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: If you are using the Windows NT native authentication facility, you may not necessarily need to set TCPSEC at the client. See "Setting Security for Connecting Clients" on page 430 for more details about setting security for connecting clients. △

  When supplying username and password information to a Version 8 SAS session on Windows NT, you may specify *username* in the form

Windows-NT-domain-name\username. Here is an example of how you might specify this information in the TCPSEC option:

```
options set=tcpsec "apex\bass.time2go";
```

Domain name **apex** identifies the location of the username and password database. Username **bass** and password **time2go** will be verified against those in the identified domain's username and password database.

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 option is shared among many users.

Examples:

```
options set=tcpsec _prompt_;
options set=tcpsec bass.time2go;
options set=tcpsec "apex\bass.time2go";
```

Providing Userid-Based Security for a SAS/SHARE Server

Note: SAS/SHARE server security is supported on the Windows NT platform only. △

The TCPSEC option also specifies whether the TCP/IP access method performs user authentication before connecting to a SAS/SHARE server. The TCPSEC option must be set on the server before you start the SAS/SHARE server.
Here is the syntax and description of this option.

TCPSEC=_SECURE_ | _NONE_

SECURE
The _SECURE_ value for the TCPSEC option 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 option 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:

```
options set=tcpsec _secure_;
options set=tcpsec _none_;
```

SAS/CONNECT and SAS/SHARE Options

TCPSEL=NONE | BLOCK | POLL

The algorithm used by the TCP/IP access method changed between Release 6.08 and Release 6.10 TS040 of SAS software to make the TCP/IP access method compatible with other access methods that might be used concurrently in a SAS program. For Windows NT, Windows 95, and Windows 32s, however, some SAS programs experienced a less efficient performance. An application's performance depends on which Windows TCP/IP package is used.

The Release 6.08 TCP/IP access method for Windows uses a polling algorithm that consumes CPU cycles even when a SAS program is idle. Although this consumption is acceptable with most single-user PC systems, such as Windows, this polling behavior is an unacceptable drain of resources for multi-user PC systems, such as Windows NT .

The Release 6.10 TCP/IP access method replaced the polling algorithm with one that uses asynchronous data notification with Windows messaging. For some applications and TCP/IP packages, the performance change between Release 6.08 to Release 6.10 is negligible. However, for some applications and TCP/IP packages, the Release 6.10 TCP/IP access method is slower than with Release 6.08.

By setting the option TCPSEL, you can use one of two other algorithms that may improve the performance of your application. You may specify any of the following:

TCPSEL=NONE
: provides the default Release 6.10 behavior. This method will not consume unnecessary CPU cycles and will quickly recognize network failures. The disadvantage of this method is that it may slow down some SAS programs.

  Setting the TCPSEL option to an explicit value of NONE is the same as not setting it at all.

TCPSEL=BLOCK
: does not poll the CPU but attempts to read TCP/IP messages with blocking sockets. This method will be faster than the Release 6.10 algorithm for some programs. The disadvantage of using the BLOCK method is that it may take longer for SAS to recognize a network failure than it would with the default method.

TCPSEL=POLL
: restores the Release 6.08 behavior of polling. This method should elicit quicker notification of a network failure than the BLOCK method does. The disadvantage of this method is that it will consume CPU cycles even when SAS is idle.

The value (BLOCK, POLL, or NONE) that you choose for TCPSEL depends on your application and which Windows TCP/IP package you are running. You should try your applications with each value to determine which one will give you the best performance.

You must set the option before you sign on to a SAS/CONNECT remote host or before you define a libref to a SAS/SHARE server. You cannot change the value during an active session. You do not need to exit SAS to change the value, but you must be signed off from all SAS/CONNECT sessions, and you must clear all libref statements that are assigned to SAS/SHARE servers.

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 at a SAS invocation or in the configuration file. △

TCPTN3270

TCPTN3270 is an environment variable that is set on the local host to support a connection to an OS/390 or a CMS host that uses 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 427 for information about these script files.

Set TCPTN3270 to the value of 1 at the Windows local host in the SAS configuration file or in an OPTIONS statement.

Examples:

```
-set tcptn3270 1

options set=tcptn3270 1;
```

The TCP/IP sample scripts TCPTSO32.SCR and TCPCMS32.SCR allow you to sign on and sign off in these environments.

TCPMSGLEN *n*

defines the size of the buffer in bytes that the TCP/IP access method uses for breaking up a message that it sends to or receives from the SAS/CONNECT application layer during a SAS/CONNECT session. The application layer uses a message size that is stored in the TBUFSIZE option (default 32K) that you may specify as an option in the SIGNON statement or as a SAS option. See *SAS/SHARE User's Guide* for information about the TBUFSIZE option.

If TBUFSIZE is larger than TCPMSGLEN, the TCP/IP access method breaks the message into a buffer whose size is defined by TCPMSGLEN and issues the number of send and receive messages that are necessary to complete the message transaction.

The platform-specific default size of TCPMSGLEN must be set at both the local and remote hosts. Default values by platform are shown in the following table.

Table 29.1 Windows TCP/IP Default TCPMSGLEN Values by Remote Host

Platform	TCPMSGLEN Default Value
OS/390	8K
Windows	16K

Platform	TCPMSGLEN Default Value
UNIX	32K
OpenVMS	32K

If the values that are set for TCPMSGLEN at the local host and at the remote host are different, the smaller value of the two is used during the SAS/CONNECT session.

Use the following syntax to set these variables at SAS invocation or in the configuration file at the local and the remote hosts:

```
-set tcpmsglen 16384
-set tcptn3270 1
```

SAS/SHARE Only Options

CAUTION:

Windows NT only Server security is supported on the Windows NT platform only. The following options are supported in a secure environment only. △

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 that are running older releases of SAS/SHARE .

To require only encrypted userids and passwords, you must set the AUTHENCR option as an environment variable or 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 controls encryption for clients connecting to a secured server.

OPTIONAL
: means that a client can optionally encrypt the username and the 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 the password that it sends to the server.

See "Setting SAS Options and Variables" on page 417 for examples of the forms that you can use to specify the AUTHENCR option.

Note: You must use either the environment variable or the SAS macro variable form to set AUTHENCR.

AUTHSERVER *NT-domain-or-NT-server-name*

valid for Windows NT hosts only, specifies the location of the database that contains the username and password pairs that are used for validation. Specify the name of either an NT domain or an NT server at which the database resides. △

Note: You may specify the AUTHSERVER option in an OPTIONS statement in a SAS session or in an AUTOEXEC file, in a SAS configuration file, at SAS invocation, or as a SAS macro variable. △

For Version 8, you are not limited to specifying a single NT domain by means of the -AUTHSERVER option. Instead, you may bypass this option and specify the domain name in the form *domain\username* when you supply your username to the Windows NT environment. Here is an example of how you might specify this information in the SIGNON statement:

```
signon user=apex\bass password=time2go;
```

Domain name **`apex`** identifies the location of the username and password database. Username **`bass`** and password **`time2go`** will be verified against those in the identified domain's username and password database.

SAS/CONNECT

Local Host Tasks

User and Applications Programmer

To connect a Windows 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 program, configure the spawner in the SERVICES file, as necessary.
3. If you connect to a remote host by means of a spawner, you may need to set security, as necessary.
4. Configure the local and remote hosts' names and Internet addresses in the local HOSTS file or by means of the domain server.
5. Specify the communications access method.
6. Specify the remote node name.
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, then 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, then 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 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.

If you sign on to a host that is not a Windows NT, a Windows 95, a Windows 98, an OS/2, a UNIX, or an OS/390 host, 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

It is assumed that the local host and the remote host both run Windows NT 4.0 or a subsequent release and are included in an NT domain. Also, it is assumed that the user was authenticated through the domain controller.

A local host can use the same user context or a different user context when accessing a remote host. Users can establish user context by logging in to a remote host with their userids and passwords to access files that they have permission to access. However, users can also establish a different user context by logging in to a remote host with someone else's userid and password. Supplying someone else's userid and password gives permission to access files that they may be otherwise denied access to. A system administrator's userid and password is an example of a different context. Such a context does not belong to the user but can be granted to the user for access to particular files.

In order for a local host to connect to a remote host *in the same user context*, do *not* establish security by means of the USER= and PASSWORD= options in applicable statements or the TCPSEC option. The Windows NT native authentication facility transmits the user's context to the remote host when the user makes a connection.

In order for a local host to access a remote host *with a different context*, you *must* set security.

Set security at the local host by using either of the methods explained in "Setting Security for SAS/CONNECT and SAS/SHARE" on page 418. 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 418.

For Version 7 security behavior, if you set the TCPSEC option 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 option, see "Providing Client Identification in a pre-Version 8 Session" on page 419.

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 by means of the name server. A name server program supplies name-to-address translation, mapping from domain names to IP addresses. The name server process 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

Note: TCP/IP is the default communications access method on the Windows 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 stands for Transmission Control Protocol/ Internet Protocol) is an example of an *access-method-id*.

Example:

```
options comamid=tcp;
```

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

Specifying the Remote Node Name

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

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

The value of *node-name* that you specify 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 35, "PC Spawner Program," on page 471 for more information.
- □ If you are connecting to a UNIX, an OS/390, or an OpenVMS Alpha host that is running the spawner, 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 name assigned to 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 see "Configuring the SERVICES File" on page 485 for information about configuring the spawner in the SERVICES file.
- □ If you are connecting to any other remote host platforms, use the node name of the remote host.

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

Example:

```
options remote=rmtnode;
```

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, as illustrated here:

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

Example:

```
%let rmtnode=149.999.228.6;
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* **\connect\saslink**. You must customize the sample scripts to accurately reflect your site logon process. Failure to do so will produce errors.

The fileref syntax follows:

```
FILENAME RLINK '!sasroot\connect\saslink\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 29.2 Windows 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

Example:

```
filename rlink '!sasroot\connect\saslink\tcpcms.scr';
```

Note: If you connect to a spawner program, 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 will not define an RLINK fileref. See Chapter 32, “Spawner Programs,” on page 457 for details about the spawner programs. △

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 418.

You can also SIGNON and identify a specific port number that a spawner monitors for a server. The specified port is used to pass data to and receive data from the server. The port number is specified with two preceding underscore (_) characters. For example, you can specify the port as 5000 using the SIGNON statement:

```
signon rmtnode.__5000;
```

Local Host Example

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

```
filename rlink '!sasroot\connect\saslink\tcpcms.scr';
options comamid=tcp remote=rmtnode;
signon;
```

The first line identifies the script file that you use to sign on to a CMS remote host. The script file prompts 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

1 Start the PC spawner program at the remote host.

Starting the PC Spawner Program

You must invoke the PC spawner program on a Windows NT, a Windows 95, or a Windows 98 remote host to enable local hosts to connect to it. The spawner program resides on a remote host, listening for SAS/CONNECT client requests for connection to the remote host. After the spawner program receives a request, it invokes the remote SAS session.

For Windows NT only, setting the -SECURITY option in the PC spawner invocation command secures the server.

The spawner then verifies the userid and the password against those that are assigned to the USER= and PASSWORD= options in the SIGNON statement or to the TCPSEC option. For information about setting security, see “Setting Security for SAS/CONNECT and SAS/SHARE” on page 418.

See Chapter 32, “Spawner Programs,” on page 457 for information about starting the spawner on the remote host.

Note: If you set the -SECURITY option at the PC spawner invocation, inform local users that they may need to set security on the local host. △

Remote Host Example

You may set the following variables in a Windows NT, a Windows 95, or a Windows 98 remote host AUTOEXEC file to restrict port access:

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

These statements restrict access to ports 5020 through 5050.

The following example shows how the PC spawner is invoked on a Windows NT, a Windows 95, or a Windows 98 remote host:

```
c:\sas\connect\sasexe\spawner -comamid tcp -file mysas.cmd
```

The spawner is invoked and the TCP/IP access method is specified. The -FILE option executes the MYSAS.CMD file, which invokes a SAS session.

See Chapter 35, "PC Spawner Program," on page 471 for information about the contents of a command file and about executing the PC spawner. Options that are set by means of the spawner may override options that are set in a remote host configuration file.

SAS/SHARE

Client Tasks

User and Applications Programmer

Note: Server security is supported on the Windows NT platform only. △

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

1. Configure the server in the client SERVICES file.
2. For a Windows NT client only, assign the appropriate rights to each connecting client.
3. For a Windows NT client only, set security for connecting clients.
4. Specify the TCP/IP access method.
5. 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 "Configuring the SERVICES File" on page 485 for information about editing the SERVICES file.

Assigning the Appropriate Rights for Connecting Clients

CAUTION:

Windows NT only Server security is supported on the Windows NT platform only. △

The account in which a connecting client runs must have the appropriate rights. To assign these rights

1 Click on the Administrative Tools icon.
2 Click on the User Manager icon.
3 From the Policies pull-down menu, select User Rights.
4 Click the Show Advanced User Rights box.
5 Assign “Log on as a batch job” rights to the appropriate users.

Setting Security for Connecting Clients

CAUTION:
Windows NT only Server security is supported on the Windows NT platform only. △

It is assumed that the client and the server both run Windows NT 4.0 or a subsequent release and are included in an NT domain. Also, it is assumed that the user was authenticated through the domain controller.

A client can use the same user context or a different user context when accessing a server. Users can establish user context by logging in to a server with their userids and passwords to access files that they have permission to access. However, users can also establish a different user context by accessing a server with someone else’s userid and password. Supplying someone else’s userid and password gives permission to access files that they may be otherwise denied access to. A system administrator’s userid and password is an example of a different context. Such a context does not belong to the user but can be granted to the user for access to particular files.

In order for a local host to connect to a remote host *in the same user context*, do *not* establish security by means of the USER= and PASSWORD= options in applicable statements or the TCPSEC option. The Windows NT native authentication facility transmits the user’s context to the remote host when the user makes a connection.

In order for a local host to access a remote host *with a different context*, you *must* set security.

Set security at the client using either of the methods explained in “Setting Security for SAS/CONNECT and SAS/SHARE” on page 418. For Version 8 security behavior, specify the USER= and PASSWORD= options in the appropriate statement. For details, see “Providing Client Identification in a Version 8 Session” on page 418.

For Version 7 security behavior, if you set the TCPSEC option at the client, either specify a userid and a password that are valid on the server or specify _PROMPT_ to supply the userid and password when connecting to a server. For information about setting the TCPSEC option, see “Providing Client Identification in a pre-Version 8 Session” on page 419.

For Windows NT only that runs Version 8, you may qualify *username* in the form *Windows-NT-domain-name\username*. Here is an example of how you might specify this information in the LIBNAME statement in SAS/SHARE:

```
libname test 'prog2 a' user=apex\bass.time2go server=share1;
```

Domain name **apex** identifies the location of the username and password database. Username **bass** and password **time2go** will be verified against those in the identified domain’s username and password database.

Specifying the TCP/IP Communications Access Method

Note: TCP/IP is the default communications access method on the Windows 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 at each connecting client, before you access a server, 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 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 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. *access method-id* is 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 a Windows NT client connecting to an OS/390 server follows:

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

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

Note: Additionally, a Windows 32s client supports the CPIC 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, and 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 would cause 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: Concepts* for details about SAS naming rules. See the *SAS/SHARE User's Guide* for details about the LIBNAME and PROC OPERATE statements.

Client Example

The following example illustrates the statements that you specify in a Windows NT client SAS session to connect in a different context to a server with the TCP/IP access method:

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

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

Note: Server security is supported on the Windows NT platform only. △

Server Administrator

To set up a secure server and to make it accessible to a client, perform the following tasks:

1. Configure SAS/SHARE servers in the SERVICES file.
2. For a server that is running on a Windows NT host, optionally set server security by using the TCPSEC option.
3. For a server that is running on a Windows NT host, optionally enforce client userid and password encryption.
4. For a server that is running on a Windows NT host, assign the appropriate rights for a secure server.
5. Specify the TCP/IP access method.
6. 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 remote host node from which a client session will access the server. This file usually is located in the directory in which the TCP/IP product is installed. See “Configuring the SERVICES File” on page 485 for information about editing 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 on which the server is running. A secured server uses a validated userid and password to verify a user’s authority to access a SAS library or a file.

Requiring connecting clients to supply a valid userid and password enforces server security. From a server session, set the TCPSEC option to the value _SECURE_. See “SAS/CONNECT and SAS/SHARE Options” on page 420 for more information about setting this option.

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 Options” on page 423 for details about setting the AUTHENCR option.

Assigning the Appropriate Rights for a Secure Server

CAUTION:
Windows NT Only This process is supported on the Windows NT platform only. △

The account in which a secure server runs must have the appropriate rights. To assign these rights

1. Click on the Administrative Tools icon.
2. Click on the User Manager icon.
3. From the Policies pull-down menu, select $\boxed{\text{User Rights}}$.
4. Click the $\boxed{\text{Show Advanced User Rights}}$ box.
5. Assign “Act as part of the operating system” rights to the appropriate users.

Specifying the TCP/IP Access Method

Note: TCP/IP is the default communications access method on the Windows 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 before you can create and access a SAS/SHARE server, use the following syntax at the server:

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

where COMAMID is an acronym for Communications Access Method Identification. *access-method-id* identifies 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 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, at a 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 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
```

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

```
-comamid tcp
-comaux1 spx
-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 TCP/IP access method as well as to clients that use the SPX and NetBIOS access methods.

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=__5000;
  ```

See *SAS Language Reference: Concepts* 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 Windows NT host:

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

The value _SECURE_ for the TCPSEC option specifies that clients supply a userid and a password that are valid on the server. The value REQUIRED for the AUTHENCR option specifies that only encrypted userids and passwords from clients are accepted.

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

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

The TCP/IP access method is declared and the server SHARE1 is started on the Windows NT 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