SAS Publishing

Communications Access Methods for

SAS/CONNECT® 9.1 and SAS/SHARE® 9.1

The Power to Know®

The correct bibliographic citation for this manual is as follows: SAS Institute Inc. 2004. *Communications Access Methods for SAS/CONNECT® 9.1 and SAS/SHARE® 9.1*. Cary, NC: SAS Institute Inc.

Communications Access Methods for SAS/CONNECT® and SAS/SHARE® 9.1

ISBN 1-59047-229-2

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

1st printing, January 2004

Contents

What's New ***vii***
Overview **vii**
Details **viii**

PART 1 Introduction 1

Chapter 1 △ Communications Access Methods 3
Communications Access Method: Definition **3**
Communications Access Methods Supported by SAS/CONNECT and SAS/SHARE **3**
Supported Communications Access Methods by Operating Environment **3**
Operating Environments Supported in SAS 9.1 **4**
Finding Information in This Documentation **5**
Additional SAS Documentation **5**
SAS Syntax Conventions **5**

PART 2 z/OS Operating Environment 7

Chapter 2 △ z/OS: TCP/IP Access Method 9
Prerequisites for Using TCP/IP under z/OS **10**
SAS/CONNECT Client Tasks **14**
SAS/CONNECT Server Tasks **18**
SAS/SHARE Client Tasks **20**
SAS/SHARE Server Tasks **22**
System Configuration for TCP/IP **25**

Chapter 3 △ z/OS: XMS Access Method 45
Prerequisites for Using XMS under z/OS **46**
SAS/CONNECT Client Tasks **47**
SAS/CONNECT Server Tasks **49**
SAS/SHARE Client Tasks **49**
SAS/SHARE Server Tasks **51**
System Configuration for the XMS Access Method **52**

PART 3 OpenVMS Alpha Operating Environment 55

Chapter 4 △ OpenVMS Alpha: TCP/IP Access Method 57
Prerequisites for Using TCP/IP under OpenVMS Alpha **58**
SAS/CONNECT Client Tasks **60**
SAS/CONNECT Server Tasks **66**
SAS/SHARE Client Tasks **67**
SAS/SHARE Server Tasks **70**

PART 4 **UNIX Operating Environments 73**

Chapter 5 △ UNIX: TCP/IP Access Method 75
Prerequisites for Using TCP/IP under UNIX **76**
SAS/CONNECT Client Tasks **78**
SAS/CONNECT Server Tasks **84**
SAS/SHARE Client Tasks **84**
SAS/SHARE Server Tasks **87**

PART 5 **Windows Operating Environments 91**

Chapter 6 △ Windows: TCP/IP Access Method 93
Prerequisites for Using TCP/IP under Windows **94**
SAS/CONNECT Client Tasks **97**
SAS/CONNECT Server Tasks **103**
SAS/SHARE Client Tasks **104**
SAS/SHARE Server Tasks **107**
Data Security for SAS/CONNECT or SAS/SHARE Servers **109**

PART 6 **Spawners and Files 111**

Chapter 7 △ SAS/CONNECT Spawners 113
Spawner Definition **113**
Benefits of Using a Spawner to Sign On to a Server **114**
Support for Spawners by Operating Environment **114**
Client Connection to a Spawner **114**
Spawner Connection Examples **115**

Chapter 8 △ OpenVMS Alpha Spawner 119
OpenVMS Alpha Spawner Requirements **119**
Starting the OpenVMS Alpha Spawner **120**
Ending the OpenVMS Spawner **121**

Chapter 9 △ z/OS Spawner 123
z/OS Spawner Requirements **123**
Starting the z/OS Spawner **124**
Defining the Shell Script for Starting SAS **125**
Ending the z/OS Spawner **126**

Chapter 10 △ UNIX Spawner 127
UNIX Spawner Requirements **127**
Starting the UNIX Spawner **127**
Ending the UNIX Spawner **129**

Chapter 11 △ Windows Spawner 131
Windows Spawner Requirements **131**
Starting the Windows Spawner **131**

Ending the Windows Spawner **135**

Chapter 12 △ Encryption Options 137

Security Services **137**

Security Service Requirements **138**

Encryption Options **138**

SAS/SECURE Options **139**

SAS Proprietary Options **141**

SSL Options **141**

Chapter 13 △ TCP/IP SERVICES File 145

Configuring the SERVICES File **145**

PART 7 Configuring SAS/CONNECT for Use with Firewalls 147

Chapter 14 △ Configuring SAS/CONNECT for Use with a Firewall 149

Definitions **149**

Requirements for Using a Firewall **149**

Firewall Configuration Example **150**

PART 8 SAS/CONNECT Scripts 153

Chapter 15 △ Sign-On Scripts 155

Script Rules **155**

Sample Scripts **156**

PART 9 Error Messages 175

Chapter 16 △ z/OS Error Messages 177

z/OS: TCP/IP Access Method **177**

Chapter 17 △ OpenVMS Alpha Error Messages 179

OpenVMS Alpha: TCP/IP Access Method **179**

Chapter 18 △ UNIX Error Messages 181

UNIX: TCP/IP Access Method **181**

Chapter 19 △ Windows Error Messages 183

Windows: TCP/IP Access Method **183**

PART 10 Appendix 185

Appendix 1 △ Recommended Reading 187

Recommended Reading **187**

Glossary 189

Index 193

What's New

Overview

SAS now supports

- the TCP/IP communications access method for network connections between these operating environments: OpenVMS Alpha, UNIX, Windows, and z/OS. Also, the XMS communications access method can be used between address spaces under z/OS.
- the network security protocol Secure Sockets Layer (SSL), which encrypts connections between client and server.
- a new shell script for starting SAS is provided for the z/OS spawner.
- new instructions for configuring TCP/IP that runs under the OS/390 and z/OS operating environments.
- the new Windows spawner option -NAME.
- new options for the OpenVMS spawner, UNIX spawner, Windows spawner, and z/OS spawner are -OMRCONFIGFILE and -SASSPAWNERCN. The -INSTALLDEPENDENCIES option is valid only in the Windows spawner.

Note:

- This section describes the features of the SAS communications access methods that are new or enhanced since SAS 8.2.
- z/OS is the successor to the OS/390 operating system. Throughout this document, any reference to z/OS also applies to OS/390, unless otherwise stated.

△

Details

Access Methods and Operating Environments

- □ SAS now supports the TCP/IP and XMS communications access methods. For details, see "Supported Communications Access Methods by Operating Environment" on page 3.
- □ SAS now supports the OpenVMS Alpha, UNIX, Windows, and z/OS operating environments. For details, see "Operating Environments Supported in SAS 9.1" on page 4.
- □ SAS no longer supports the APPC, DECnet, EHLLAPI, and NETBIOS communications access methods or the CMS, OpenVMS VAX, OS/2, Windows 95, and Windows 98 operating environments.

SSL Protocol

SAS/CONNECT and SAS/SHARE support the Secure Sockets Layer (SSL) protocol, which provides network security and protects the privacy of information by encrypting client/server transfers under the UNIX and Windows operating environments. For details, see Chapter 12, "Encryption Options," on page 137.

OS/390 and z/OS Operating Environments

- □ A new shell script for starting SAS is provided for the z/OS spawner. For details, see "Defining the Shell Script for Starting SAS" on page 125.
- □ New instructions are provided to configure TCP/IP that runs under the OS/390 and z/OS operating environments. For details, see "System Configuration for TCP/IP" on page 25.

Spawners

- □ The new -NAME option for the Windows spawner is used to assign a name to the spawner that is installed and started as a service. A specified name overrides the default name that is automatically assigned when the -INSTALL option is used. For details, see "Starting the Windows Spawner" on page 131.
- □ New options for the OpenVMS spawner, UNIX spawner, Windows spawner, and z/OS spawner are -OMRCONFIGFILE and -SASSPAWNERCN. The new -INSTALLDEPENDENCIES option is valid only in the Windows spawner. For details about these options, see the sections about the spawners.

PART 1

Introduction

Chapter 1. . . . . **Communications Access Methods** *3*

CHAPTER

1

Communications Access Methods

Communications Access Method: Definition **3**
Communications Access Methods Supported by SAS/CONNECT and SAS/SHARE **3**
Supported Communications Access Methods by Operating Environment **3**
Operating Environments Supported in SAS 9.1 **4**
Finding Information in This Documentation **5**
Additional SAS Documentation **5**
SAS Syntax Conventions **5**

Communications Access Method: Definition

A communications access method is the interface between SAS and the network protocol that you use to connect two operating environments.

You must use a communications access method with both SAS/CONNECT and SAS/SHARE.

The communications access method that you choose is determined by the network protocols that you have available at your site and the operating environments that you are connecting.

Communications Access Methods Supported by SAS/CONNECT and SAS/SHARE

SAS/CONNECT and SAS/SHARE support the following communications access methods:

TCP/IP (Transmission Control Protocol/Internet Protocol)
: is a program-to-program interface that is supported on hardware from multiple vendors.

XMS (Cross-Memory Services)
: is an interface that is part of the z/OS operating environment and is used by programs that run within a single z/OS environment.

Supported Communications Access Methods by Operating Environment

Note: The following table shows only the communications access methods that are supported in SAS/CONNECT 9.1 and SAS/SHARE 9.1. △

Table 1.1 SAS/CONNECT and SAS/SHARE: Valid Communications Access Methods between a Client and a Server

Server Operating Environments	Client Operating Environments			
	OpenVMS Alpha	z/OS	UNIX	Windows
OpenVMS Alpha	TCP/IP	TCP/IP	TCP/IP	TCP/IP
z/OS	TCP/IP	TCP/IP XMS	TCP/IP	TCP/IP
UNIX	TCP/IP	TCP/IP	TCP/IP	TCP/IP
Windows	TCP/IP	TCP/IP	TCP/IP	TCP/IP

Operating Environments Supported in SAS 9.1

Table 1.2 Operating Environments Supported in SAS 9.1

Machine	Baseline Operating Environment	Size (in bits)
	OpenVMS Alpha	
OpenVMS Alpha	7.2	64
	z/OS	
OS/390	V2R10	32
z/OS	V1R1 (and later)	64
	UNIX	
AIX 64	5.1	64
Compaq Digital UNIX	5.1	64
HP 64	11.0 PA	64
HP/UX for Itanium Platform Family	11i	64
RedHatLinux on Intel	2.4	32
Solaris 64	8	64
	Windows	
Windows NT/2000/XP	4.0	32
Windows for IPF	XP	64

Finding Information in This Documentation

To find the information that you need to perform tasks at the client:

1 Find the number of the Part for the client operating environment that you will use.

2 In that Part, find the chapter for the communications access method that you will use.

3 In that chapter, find the section for the SAS product (SAS/CONNECT or SAS/SHARE) that you will use.

To find the information that you need to perform tasks at the server:

1 Find the number of the Part for the server operating environment.

2 In that Part, find the chapter for the communications access method that you will use.

3 In that chapter, find the section for the SAS product (SAS/CONNECT or SAS/SHARE) that you will use.

Table 1.3 Finding Information for the Server

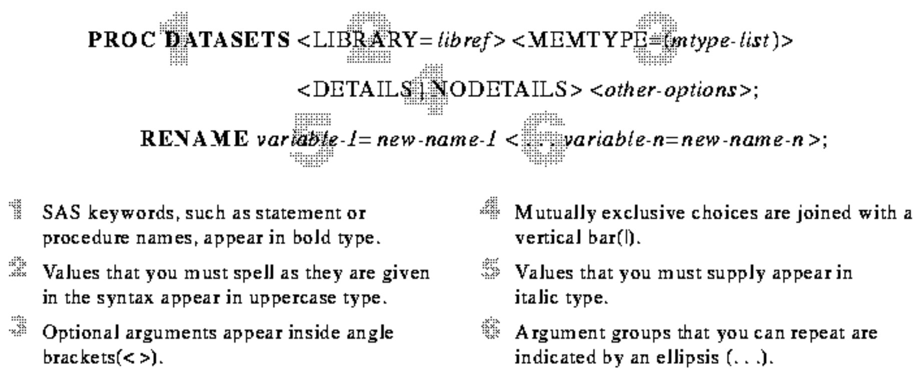
Question	Answer
What is the server operating environment?	z/OS, Part 2
What communications access method am I using?	TCP/IP, Part 2, Chapter 2,
What SAS product am I using?	SAS/CONNECT

Additional SAS Documentation

If you use the TCP/IP communications access method under the OS/390 and z/OS operating environments, the following SAS publication might be helpful.

□ *SAS/C Library Reference, Third Edition, Volume 2, Release 7.00.*

SAS Syntax Conventions



PROC DATASETS <LIBRARY=*libref*> <MEMTYPE=(*mtype-list*)>
<DETAILS|NODETAILS> <*other-options*>;
RENAME *variable-1*=*new-name-1* <. . . *variable-n*=*new-name-n*>;

1 SAS keywords, such as statement or procedure names, appear in bold type.

2 Values that you must spell as they are given in the syntax appear in uppercase type.

3 Optional arguments appear inside angle brackets(< >).

4 Mutually exclusive choices are joined with a vertical bar(|).

5 Values that you must supply appear in italic type.

6 Argument groups that you can repeat are indicated by an ellipsis (. . .).

P A R T 2

z/OS Operating Environment

Chapter ***2***. . . . .**z/OS: TCP/IP Access Method** *9*

Chapter ***3***. . . . .**z/OS: XMS Access Method** *45*

CHAPTER
2

z/OS: TCP/IP Access Method

Prerequisites for Using TCP/IP under z/OS **10**
- *Task List* **10**
- *Software Requirements* **11**
- *TCP/IP Access Method Terminology* **11**
- *SAS/CONNECT and SAS/SHARE Network Security* **11**
- *SAS/CONNECT and SAS/SHARE Options* **12**
- *SAS/CONNECT Options Only* **13**
- *SAS/SHARE Options Only* **13**

SAS/CONNECT Client Tasks **14**
- *Task List* **14**
- *Specifying TCP/IP as the Communications Access Method* **15**
- *Encrypting Data in Client/Server Transfers* **15**
- *Choosing a Method to Use to Sign On* **15**
- *Signing On Using a Spawner* **15**
  - *Ensuring That the Spawner Is Running on the Server* **15**
  - *Specifying the Server and the Spawner Service* **16**
  - *Specifying a Sign-On Script or a User ID and Password* **17**
  - *Specifying a Sign-On Script* **17**
  - *Specifying a User ID and Password* **17**
  - *Signing On Using the Spawner* **17**
- *Signing On Using a Telnet Daemon* **18**
  - *Specifying the Server* **18**
  - *Specifying a Sign-On Script* **18**
  - *Signing On to the Server Session* **18**

SAS/CONNECT Server Tasks **18**
- *Task List* **18**
- *Installing the Logon Procedure on the Server* **19**
- *SAS/CONNECT Server Example* **19**

SAS/SHARE Client Tasks **20**
- *Task List* **20**
- *Configuring the Server Service* **20**
- *Specifying TCP/IP as the Communications Access Method* **20**
- *Accessing a Secured Server* **20**
- *Encrypting Data in Client/Server Transfers* **21**
- *Specifying the Server* **21**
- *SAS/SHARE Client Example* **22**

SAS/SHARE Server Tasks **22**
- *Task List* **22**
- *Installing the SAS SVC Routine* **23**
- *Configuring the Server Service* **23**
- *Setting the TCPSEC Option to Require Client Authentication* **23**

Encrypting Data in Server/Client Transfers **23**
Specifying TCP/IP as the Communications Access Method **24**
Specifying the Server **24**
SAS/SHARE Server Example **25**
System Configuration for TCP/IP **25**
Using a SAS TCP/IP Configuration Plan **25**
TCP/IP Overview **26**
TCP/IP: Software Requirements **26**
SAS Transient Library **27**
Processing the CTRANSLOC Option **27**
TCP/IP Stacks **29**
TCP/IP Communication Stack: Definition **29**
Sample Definitions of TCP/IP Stacks **29**
System and Process Limits **30**
TCP/IP Host Name Configuration **31**
IP Addresses **31**
TCP/IP Host Name Configuration for Communications Servers **31**
TCP/IP Stack Configuration Files **33**
IBM and CA TCP/IP Stack Configuration Files **33**
IBM PROFILE.TCPIP File **33**
IBM TCPIP.DATA File **34**
CA TCPCPGnn TCPIP.PARM Member **34**
TCP/IP Name Resolver Configuration **35**
Name Resolver: Definition **35**
Supported Name Resolvers **35**
IBM OS/390 Name Resolvers: Overview **35**
IBM z/OS Name Resolver: Overview **35**
SAS/C Name Resolver: Overview **37**
SAS/C Environment Variables and SAS 9.1 System Options **40**
SAS/C Environment Variables and SAS System Options: Definitions **40**
The Default TCPIP Prefix **41**
Changing the Default TCPIP Prefix **42**
SAS/C Environment Variables in the SASCTCPV Data Set **42**
The UNIX System Services (USS) Shell **42**
Shell Configuration Requirements **42**
Specifying the SAS Transient Library **43**
Configuring TCP/IP Using SAS/C Environment Variables **43**
The Shell Profile File **43**
The Services File **44**
Services File: Overview **44**
The Services File Search Order **44**
References **44**

Prerequisites for Using TCP/IP under z/OS

Task List

- □ Verify that the software requirements are met.
- □ Become familiar with the TCP/IP access method terminology.

- If using network security, set the appropriate SAS system options.
- Set the appropriate SAS/CONNECT and SAS/SHARE options.

Software Requirements

- Base SAS software and either SAS/CONNECT or SAS/SHARE must be installed on both the client and the server.
- SAS/CONNECT and SAS/SHARE require the SAS Transient Library that is provided with SAS/CONNECT 9.1 and SAS/SHARE 9.1.

  Note: If your site has installed a previous release of the SAS Transient Library, you must replace it with the transient library that is included with SAS 9.1. For details, see "SAS Transient Library" on page 27. △
- SAS/CONNECT and SAS/SHARE also require one of the following Communications Servers:
  - IBM OS/390 V2R10 Communications Server
  - IBM z/OS Communications Server
  - CA Unicenter TCPaccess Communications Server (formerly named Interlink SNSTCP), Version 5.3 or later.
- SAS/CONNECT or SAS/SHARE require the definition of TCP/IP resources for the z/OS system. For details, see "System Configuration for TCP/IP" on page 25.

TCP/IP Access Method Terminology

Familiarity with the following terms will help you when you set SAS options:

name resolution
: The process of mapping a server name to an address. The domain name system provides a facility for naming servers in which programs use remote name servers to resolve server names into IP addresses.

name server
: The server program that supplies name-to-address translation, that is, mapping from server names to IP addresses. The server program often runs on a dedicated processor, and the operating environment itself is referred to as the name server.

name resolver
: The client software that uses one or more name servers when translating a server name.

For a complete discussion about DNS and name resolution, see *DNS and BIND, 4th Ed., by Paul Albitz & Cricket Liu, O'Reilly and Associates, Inc*.

SAS/CONNECT and SAS/SHARE Network Security

Encryption is the process of transforming plaintext into a less readable form (called ciphertext) by using a mathematical process. The ciphertext is translated back to plaintext for anyone who can supply the appropriate key, which is necessary for decrypting (or unlocking) the ciphertext.

SAS/CONNECT and SAS/SHARE support the following network security services in the z/OS operating environment:

SASproprietary
: a fixed encoding algorithm that is included with Base SAS software and is available in all SAS supported operating environments. It requires no additional SAS product licenses.

SAS/SECURE
: an add-on product that uses the encryption algorithms RC2, RC4, DES, and tripleDES.

For complete details about setting up and using network security, see the *SAS/CONNECT User's Guide*. After network security is set up in your environment, you set SAS encryption options that are appropriate to the network security service and to the requirements of the client or the server session.

SAS/CONNECT and SAS/SHARE Options

TCPIPMCH=*value*

Setting TCPIPMCH= is highly recommended for sites that run multiple versions of TCP/IP.

Note: Do not use this option if your site uses only one version of TCP/IP. △

The TCPIPMCH= option identifies which version of TCP/IP to use at sites that simultaneously run multiple versions of TCP/IP. Each version is referred to as a specific TCP/IP stack or transport driver.

The IBM Communications Server and the CA Unicenter TCPaccess Communications Server are implemented as IBM UNIX System Services Physical File Systems (PFS) and are defined in the system's SYS1.PARMLIB(BPXPRM*nn*) file. A PFS is defined by using a type of CINET, and each TCP/IP stack is defined as a SUBFILESYSTYPE under this PFS and has a NAME(*value*) associated with it.

The value for TCPIPMCH should be identical to the NAME(*value*) for the TCP/IP stack. If TCPIPMCH is not defined, SAS will use the value that is specified in either the TCPIPJOBNAME statement or the TCPIPUSERID statement in a TCPIP.DATA file, if available. Otherwise, SAS will use the default name TCPIP.

You can set TCPIPMCH in a SAS configuration file, in a SAS start-up command, or in a CLIST variable.

TCPIPPRF=*name*

enables you to specify a naming convention for a data set at your site by attaching a descriptive prefix to data set names. For example, to attach the descriptive prefix SYS2.VER2.TCP to the configuration file ETC.HOSTS, set

```
TCPIPPRF=SYS2.VER2.TCP
```

This option setting produces the data set name SYS2.VER2.TCP.ETC.HOSTS.

Note: The TCPIPPRF option initializes a data set prefix for the current SAS session. You must set this option each time you start a SAS session on the client and the server in a SAS/CONNECT session and at the SAS/SHARE server and client. △

You can set TCPIPPRF in a SAS configuration file, in a SAS start-up command, or in a CLIST variable.

SAS/CONNECT Options Only

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 32768) that you can specify in the SIGNON statement or as a SAS option. For details, see the TBUFSIZE= system option in the *SAS/CONNECT User's Guide*.

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 value for TCPMSGLEN (default=32768) must be set at both the client and the server. If the values that are set for TCPMSGLEN at the client and at the server are different, the smaller value of the two is used during the SAS/CONNECT session.

TCPPORTFIRST=*port-number* (set at the server)
TCPPORTLAST=*port-number* (set at the server)

restrict the range of TCP/IP ports that clients can use to access servers.

Within the range of 0 through 32767, assign a beginning value to TCPPORTFIRST and an ending value to TCPPORTLAST. To restrict the number of ports to only one port, set the values for both the TCPPORTFIRST and TCPPORTLAST options to the same number. Consult with your network administrator for advice about setting these values.

At the server, you can set TCPPORTFIRST and TCPPORTLAST in the AUTOEXEC file or in the SAS configuration file.

In the following example, the client is restricted to TCP/IP ports 4020 through 4050 when connecting to a server:

```
options tcpportfirst=4020;
options tcpportlast=4050;
```

TCPTN3270 (set at the client)

supports connections to a z/OS server that uses the full-screen 3270 Telnet protocol. The script file TCPTSO32 is provided. See Table 2.1 on page 17 for a complete list of sign-on scripts.

You can set the TCPTN3270 variable only in the SAS CLIST.

To set the TCPTN3270 variable,

- set the TCPTN3270 CLIST variable at the client.
- add TCPTN3270(1) to the SAS CLIST.

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

SAS/SHARE Options Only

TCPSEC=_SECURE_ | _NONE_ (set at the server)

specifies whether the TCP/IP access method verifies user access authority before allowing clients to access the server. The TCPSEC option must be set at the server before the server session is started.

SECURE
requires the TCP/IP access method to verify the authority of clients that attempt to access the server. Each client must supply a user ID and a password that are valid at the server.

NONE
specifies that the TCP/IP access method does *not* authenticate SAS/SHARE clients that attempt to access the server.

Default: _NONE_

SECPROFILE=*name* (set at the client and the server)
specifies the name of a RACF (Resource Access Control Facility) secured sign-on function profile. SAS uses the secured sign-on function to permit a SAS/SHARE client to access a SAS/SHARE server without specifying a password. Successful signon without a password requires that the following conditions are met:

- Both the client and the server run under z/OS operating environments that are secured by RACF or by another security product that supports PassTickets.
- The RACF security administrator has activated the PTKTDATA class, and has defined at least one PTKTDATA profile for use by SAS/SHARE.

  If the client and server run under different z/OS operating environments, the RACF security administrator must activate the PTKTDATA class and define identical PTKTDATA profiles in both z/OS operating environments.
- TCP/IP is the communications access method.
- At the server, the SECPROFILE= option is assigned the name of a valid PTKTDATA profile.
- At the client, the SECPROFILE= option is assigned the same name as that assigned at the server.
- The client's user ID is specified in either of these ways:
  - The USER= option in a LIBNAME or a PROC OPERATE statement specifies the client's RACF user ID.
  - If the USER= option in a LIBNAME or a PROC OPERATE statement is omitted, the client's user ID is used by default.

SAS/CONNECT Client Tasks

Task List

1. Specify TCP/IP as the communications access method.
2. Specify encryption of client/server data transfers (optional).
3. Sign on to the server.

Note: SAS/CONNECT enables TCP/IP connections from clients outside a firewall to spawners that run on servers inside a firewall. For details, see Chapter 14, "Configuring SAS/CONNECT for Use with a Firewall," on page 149. △

Specifying TCP/IP as the Communications Access Method

TCP/IP is the default communications access method for all the SAS supported operating environments, except z/OS. Therefore, you do not have to explicitly specify the default.

If you choose to explicitly specify TCP/IP, you can use the following syntax:

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

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*. You can set this option in an OPTIONS statement, in a SAS start-up command, or in a SAS configuration file.

Example:

```
options comamid=tcp;
```

Encrypting Data in Client/Server Transfers

If network security is available and is configured at the client, you can specify SAS options to encrypt all data that is transferred between a client and a server. In the following example, the NETENCRYPTALGORITHM= option specifies the RC4 encryption algorithm.

```
options netencryptalgorithm=rc4;
```

For complete details about network security options, see the *SAS/CONNECT User's Guide*.

Choosing a Method to Use to Sign On

Based on your operating environment, you can use one of the following methods to sign on:

- a spawner
- a Telnet daemon.

Signing On Using a Spawner

1. Ensure that the spawner is running on the server.
2. Specify the server and an optional service.
3. Specify the sign-on script (if you are signing on using a script), or specify a user ID and password (if you are signing on without a script).
4. Sign on to the server through the spawner.

Ensuring That the Spawner Is Running on the Server

Before you can access the spawner, the spawner program must be running on the server. For information about the spawner that you are connecting to, see Chapter 7, "SAS/CONNECT Spawners," on page 113.

Note: The system administrator for the machine that the spawner runs on must start the spawner. The spawner program on the server cannot be started by the client. △

Specifying the Server and the Spawner Service

The name of the server can be specified either in an OPTIONS statement:

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

or directly in the SIGNON statement or command:

```
SIGNON node-name[.service-name | .port-number];
```

node-name is based on the server that you are connecting to. *node-name* must be a valid SAS name that is 1 to 8 characters in length and is either:

- □ the short machine name of the server you are connecting to. This name must be defined in the HOSTS file in the client operating environment or in your Domain Name Server (DNS).
- □ a macro variable that contains either the IP address or the name of the server you are connecting to.

The process for evaluating *node-name* follows:

1. If *node-name* is a macro variable, the value of the macro variable is passed to the operating environment's GETHOSTBYNAME function.
2. If *node-name* is not a macro variable or the value of the macro variable does not produce a valid value, *node-name* is passed to the GETHOSTBYNAME function.
3. If GETHOSTBYNAME fails to resolve *node-name*, an error message is returned and the signon fails.

   Note: The order in which the GETHOSTBYNAME function calls the DNS or searches the HOSTS file to resolve *node-name* varies based on the operating environment implementation. △

You specify *service-name* when connecting to a server that runs a spawner program that is listening on a port other than the Telnet port. If the spawner was started using the -SERVICE spawner option, you must specify an explicit *service-name*. The value of *service-name* and the value of the -SERVICE spawner option must be identical. Alternatively, you can specify the explicit port number that is associated with *service-name*.

Example 1:

In the following example, REMHOST is the name of the node that the spawner is running on. PORT1 is the name of the service that is defined at the client. The client service PORT1 must be assigned to the same port that the spawner is listening on.

```
signon remhost.port1;
```

Example 2:

In the following example, the macro variable REMHOST is assigned to the fully-qualified name of the machine that the server runs on. This server has a spawner running that is listening on port 5050. The server session that is specified in the SIGNON statement uses the node-name REMHOST and the port number 5050.

```
%let remhost=pc.rem.us.com;
signon remhost.5050;
```

You can also assign a specific port number by including the port number in the definition of the macro variable, for example,

```
%let remhost=pc.rem.us.com 5050;
signon remhost;
```

Specifying a Sign-On Script or a User ID and Password

You can use a sign-on script to sign on to the spawner, or you can sign on to a spawner without a script. If you do not use a sign-on script and if the spawner is running secured, you must supply a user ID and password to sign on to the spawner.

Note: If you connect to a spawner, you can sign on by using a script unless the spawner is started using the -NOSCRIPT option. If the -NOSCRIPT option is set, you cannot use a script. If there is no script, you do not assign the fileref RLINK in a FILENAME statement. For information about the spawner that you are connecting to, see Chapter 7, "SAS/CONNECT Spawners," on page 113. △

Specifying a Sign-On Script

If you are signing on by using a script, you must specify the script that you want to use. The script file is executed by the SIGNON statement or command. By default, the script prompts for user ID and password.

To use one of the sample script files that are supplied with SAS/CONNECT for signing on and signing off, assign the default fileref RLINK to the appropriate script file. The script is based on the server that you are connecting to. The sample scripts are installed at

```
prefix.CTMISC
```

To specify a script, use the FILENAME statement. For example,

```
FILENAME RLINK 'prefix.CTMISC/script-name';
```

script-name specifies the appropriate script file for the server.

Table 2.1 on page 17 lists the scripts that are supplied in SAS software:

Table 2.1 SAS/CONNECT Sign-on Scripts for Using TCP/IP under z/OS

Server	Script Name
TSO under OS/390	**tcptso.scr**
TSO under z/OS, SAS 9 or later	**tcptso9.scr**
z/OS (without TSO)	**tcpmvs.scr**
z/OS (using full-screen 3270 Telnet protocol)	**tcptso32.scr**
OpenVMS Alpha	**tcpvms.scr**
UNIX	**tcpunix.scr**
Windows	**tcpwin.scr**

Specifying a User ID and Password

If you are signing on to the spawner without using a script and the spawner is running secured, you must submit the SIGNON statement and provide a user ID and a password in order to log on to the server. For example,

```
SIGNON USER=user-ID | _PROMPT_ [ PASSWORD=password | _PROMPT_ ];
```

Signing On Using the Spawner

To start SAS, sign on to the server by using the spawner.

In the following example, a client connects to a UNIX server through a spawner without using a script file. In the SIGNON statement, RMTHOST.SPAWNER specifies the node RMTHOST and the service SPAWNER. This server specification presumes that a spawner is running on the node RMTHOST, and that the spawner was started with the service SPAWNER. Specifying USER=_PROMPT_ causes a log-on dialog box to appear so that a user ID and a password can be provided.

```
options comamid=tcp;
signon rmthost.spawner user=_prompt_;
```

Signing On Using a Telnet Daemon

1 Specify the server.
2 Specify a sign-on script.
3 Sign on to the server session.

Specifying the Server

The name of the server can be specified either in an OPTIONS statement:

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

or directly in the SIGNON statement or command:

```
SIGNON node-name;
```

Specifying a Sign-On Script

If you are signing on by using a script, you must specify the script that you want to use. The script file is executed by the SIGNON statement or command. By default, the script prompts for user ID and password. For details, see "Specifying a Sign-On Script" on page 17.

Signing On to the Server Session

In the following example, you specify the statements at a z/OS client to use the TCP/IP access method to connect to a server. The FILENAME statement identifies the script file that you use to sign on to the server. The script file contains a prompt for a user ID and a password that are valid on the server. The COMAMID= option specifies the TCP/IP communications access method for connecting to the server RMTNODE, which is specified in the REMOTE= option.

```
filename rlink 'prefix.CTMISC/tcptso.scr';
options comamid=tcp remote=rmtnode;
signon;
```

SAS/CONNECT Server Tasks

Task List

If you are signing on to a z/OS server with TSO, there are no server tasks.

Otherwise, follow either of these tasks, as appropriate:

1 To allow a client to connect to a z/OS server that is running a z/OS spawner, start the spawner program at the z/OS server. For details, see Chapter 9, “z/OS Spawner,” on page 123.

2 To allow a client to connect to a z/OS server without running a TSO terminal monitor program, install the logon procedure on the z/OS server.

Installing the Logon Procedure on the Server

For z/OS server connections, you can eliminate the need for TSO by replacing the terminal monitor program (also called logon procedure) with a procedure that starts SAS with the options that you want. The benefits of this method are that signing on and signing off a z/OS server is much faster than running with TSO, and you eliminate the overhead consumed by running TSO. However, a disadvantage of running without TSO is that you cannot execute any X commands or TSO commands.

In the following example, the logon procedure starts SAS with the DMR and the COMAMID=TCP options. When you log on to the z/OS server, this procedure is immediately run so that the current z/OS account is limited to running SAS each time that the current z/OS account user logs on.

```
//JOBDL   PROC ENTRY=SASHOST,
//             OPTIONS=,
//             WORK='500,200'
//JOBDL   EXEC PGM=&ENTRY,
// PARM='&OPTIONS DMR COMAMID=TCP',REGION=4096K
//STEPLIB   DD DISP=SHR,DSN=&prefix.TS450.LIBRARY
//CONFIG    DD DISP=SHR,DSN=&prefix.TS450.CNTL(TSOXA)
//SASAUTOS  DD DISP=SHR,DSN=&prefix.TS450.AUTOLIB
//SASHELP   DD DISP=SHR,DSN=&prefix.TS450.SASHELP
//SASMSG    DD DISP=SHR,DSN=&prefix.TS450.SASMSG
//WORK      DD UNIT=SYSDA,SPACE=(6144,(&WORK),,,ROUND),
//             DCB=(RECFM=FS,DSORG=PS,LRECL=6144,BLKSIZE=6144)
//SASPARM   DD UNIT=SYSDA,SPACE=(400,(100,300)),
               DCB=(RECFM=FB,LRECL=80,BLKSIZE=400,BUFNO=1)
```

A script file is still required at the client for signon. However, a SAS start-up command is not included in the script file because the logon procedure already executes the SAS start-up command.

For the content of the script file, see “TCPMVS.SCR Script” on page 164.

SAS/CONNECT Server Example

The following command starts the spawner O390SPAWN on a z/OS machine. The absence of the -SASCMD option in the spawner start-up command implies that the client will use a script file to specify the SAS command that starts SAS on the z/OS machine.

```
sastcpd -service o390spawn
```

SAS/SHARE Client Tasks

Task List

1 Configure the server service.
2 Specify TCP/IP as the communications access method.
3 Access a secured server.
4 Specify encryption of client/server data transfers (optional).
5 Specify the server.

Configuring the Server Service

Each server must be defined as a service in the SERVICES file on each machine that a client will access the server from. For details about editing the SERVICES file, see "Configuring the SERVICES File" on page 145.

Specifying TCP/IP as the Communications Access Method

You must specify the TCP/IP communications access method at the server before you can start a server. You can use the COMAMID= option in an OPTIONS statement. For example:

```
options comamid=tcp;
```

The COMAMID= option specifies the communications access method. TCP specifies the TCP/IP access method.

Alternatively, you can specify the COMAMID= option in a SAS configuration file or in a SAS start-up command.

The COMAUX1= specifies an auxiliary communications access method and can be specified only in a SAS configuration file or in a SAS start-up command. The syntax for the COMAUX1= option is:

```
COMAUX1=alternate-method
```

If the first method that you specify in the COMAMID= option fails to access a server, the second method is used. You can specify one auxiliary access method.

Example:

```
comamid=tcp
comaux1=xms
```

For details about the supported access methods, see "Supported Communications Access Methods by Operating Environment" on page 3.

Accessing a Secured Server

Requiring clients to supply a valid user ID and password when attempting to access a server enforces server security. The values for a user ID and a password are provided in the USER= and PASSWORD= options in the LIBNAME statement and the PROC

OPERATE statement. For details, see the LIBNAME statement and the OPERATE procedure in the *SAS/SHARE User's Guide*.

Example:

```
libname sasdata 'edc/prog2/sasdata' server=rmtnode.share user=_prompt_ ;
```

The value _PROMPT_ requires the client to provide a user ID and password when a client attempts to access the server.

Encrypting Data in Client/Server Transfers

If network security is configured at the client, you can specify SAS options to encrypt data that a client transfers to a server. For example,

```
options netencrypt netencryptalgorithm=rc4;
```

The NETENCRYPT option specifies that all data transactions between a client and a server will be encrypted. The RC4 encryption algorithm is assigned in the NETENCRYPTALGORITHM= option. For general information about security services, see "SAS/CONNECT and SAS/SHARE Network Security" on page 11.

Specifying the Server

If the client and server sessions are running on different network nodes, you must include the TCP/IP node in the server ID in the LIBNAME or in the PROC OPERATE statement by using a two-level server name as follows:

```
SERVER=node.server
```

node must be specified by using either a *server-ID* or a *port* number.

If the server and the client sessions are running on the same node, you can omit the node name.

server can be either a *server-ID* or a *port*.

The *server-ID* must be identical to the service name that is specified in the SERVICES file. For details, see Chapter 13, "TCP/IP SERVICES File," on page 145.

The value for *port* is the unique number that is associated with the service that is used for passing data to and receiving data from the server.

Precede the port number with two consecutive underscores.

Note: Do *not* space after the first underscore or the second underscore. △

Example:

```
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 priority:

1 SAS macro variable
2 acceptable node name.

Example 1:
You might assign the node name and the server ID to a macro variable:

```
%let srvnode=mktserver.acme.com 5000;
libname sales server=srvnode;
```

or

```
%let srvnode=12.34.56.78 5000;
libname sales server=srvnode;
```

Example 2:

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

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

For details about SAS naming rules, see *SAS Language Reference: Concepts*. For details about LIBNAME, see the LIBNAME statement, and for details about PROC OPERATE, see the OPERATE procedure, in the *SAS/SHARE User's Guide*.

SAS/SHARE Client Example

The following example shows the statements that are used at a z/OS client to access 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 specifies the TCP/IP access method. The LIBNAME statement specifies the SAS data library that is accessed through the server. The value _PROMPT_ in the USER= option specifies that the client must provide a valid user ID and password. The SERVER= option specifies the two-level server name RMTNODE.SHARE1.

SAS/SHARE Server Tasks

Task List

1. Verify that the SAS SVC routine has been installed.
2. Configure SAS/SHARE servers in the SERVICES file.
3. Configure SAS options (optional).
   - □ Set the TCPSEC= option to require client authentication.
   - □ Set security service options to encrypt data that is transferred between a server and a client.

4 Specify TCP/IP as the communications access method.
5 Specify the server name.

Installing the SAS SVC Routine

The SAS SVC control program routine is an interface between the z/OS operating environment and a specific request, such as "third-party checking." This facility provides verification in the form of calls for authentication of user IDs and passwords and of library authority.

1 Install the SAS SVC routine, if necessary.

   If you have already installed the SAS SVC routine for Release 6.09 of SAS software, do not repeat the step here. If you need to perform the installation, see the *Installation Instructions and System Manager's Guide, The SAS System under MVS* for details.

   Because SAS SVC in Release 6.09 is backward compatible, it replaces the SAS SVC routines from previous releases. You might continue using previous releases of Base SAS and SAS/SHARE with the Release 6.09 SAS SVC that is installed on your system.

2 Verify the SVC routine SAS system options.

   Verify that the SAS system options for the SVC routine accurately reflect the way that the SAS SVC is installed. The SAS system option SVC0SVC should be set to the number at which the SAS SVC is installed (for example, 251 or 109). If the SAS SVC is installed at 109 as an ESR SVC, the SAS system option SVC0R15 should be set to the ESR code (for example, 4).

3 Verify installation on all CPUs, as needed.

   If you have more than one CPU, verify that the SAS SVC is installed on the systems that will be running SAS/SHARE at your site.

Configuring the Server Service

Each server must be defined as a service in the TCP/IP SERVICES file on each node that a client will connect to. This file usually is located in the directory that the TCP/IP software is installed in. For details about editing the SERVICES file, see “Configuring the SERVICES File” on page 145.

Example:

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

Setting the TCPSEC Option to Require Client Authentication

To authenticate connecting clients, you must specify the value _SECURE_ in the TCPSEC= option to require that clients provide a user ID and a password that are valid on the server. For details about the TCPSEC= option, see “SAS/SHARE Options Only” on page 78.

Encrypting Data in Server/Client Transfers

If network security is configured at the server, you can specify SAS options to encrypt data that a server transfers to a client, for example,

```
options netencrypt netencryptalgorithm=rc4;
```

The NETENCRYPT option specifies that all data transfers between a server and a client will be encrypted. The RC4 security algorithm is assigned in the NETENCRYPTALGORITHM= option. For general information about security services, see “SAS/CONNECT and SAS/SHARE Network Security” on page 11.

Specifying TCP/IP as the Communications Access Method

You must specify TCP/IP as the communications access method at the server before a client can access it. You can use the COMAMID= option in an OPTIONS statement. For example:

```
options comamid=tcp;
```

The COMAMID= option specifies the communications access method. TCP specifies the TCP/IP access method.

Alternatively, you can specify the COMAMID= option in a SAS start-up command or in a SAS configuration file.

The COMAUX1= option specifies an auxiliary communications access method and can be specified only in a SAS start-up command or in a SAS configuration file.

The syntax for the COMAUX1= option is:

```
COMAUX1=alternate-method
```

If the first method fails to access a server, the second method is used. You can specify one auxiliary access method.

Example:

```
comamid=tcp
comaux1=xms
```

For details about the supported access methods, see “Supported Communications Access Methods by Operating Environment” on page 3.

Specifying the Server

You must specify the name of the server in the SERVER= option in the PROC SERVER statement.

```
SERVER=server
```

server can be either a *server-ID* or a *port* number. The *server-ID* corresponds to the service that was configured in the SERVICES file. For details, see “Configuring the Server Service” on page 23. The value for *port* is the unique number that is associated with the service that is used for transferring data between a client and a server. Precede the port number with two consecutive underscores.

Note: Do *not* space after the first underscore or the second underscore. △

Examples:

```
proc server server=apex;
proc server server=_ _5000;
```

For details about creating valid SAS names, see *SAS Language Reference: Concepts*. For details about PROC SERVER, see the SERVER procedure in the *SAS/SHARE User’s Guide*.

SAS/SHARE Server Example

The following example shows the statements that you specify in the server configuration file on a machine that runs the z/OS operating environment:

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

The value _SECURE_ that is specified for the TCPSEC option requires clients to provide a user ID and a password that are valid on the server.

The COMAMID= option specifies the TCP/IP access method. The PROC SERVER statement specifies the server SHARE1.

System Configuration for TCP/IP

Using a SAS TCP/IP Configuration Plan

You are encouraged to use this plan to collect information that is necessary to configure (or to verify your configuration of) TCP/IP communication servers that will run under the OS/390 operating environment or the z/OS operating environment.

1. Is your operating environment running multiple TCP/IP stacks?
   - ☐ If *yes*, go to Step 2.
   - ☐ If *no*, go to Step 3.
2. What is the name of the TCP/IP stack that you want SAS to use?

   Set the **TCPIP_MACH** SAS/C environment variable or the **TCPIPMCH** SAS system option.
3. Which vendor's TCP/IP stack will SAS be using?
   - ☐ *IBM IP Communications Server*

     Configure or verify the host name configuration by adding a **HOSTNAME** statement in the appropriate IBM TCPIP.DATA file.
   - ☐ *CA Unicenter TCPaccess Communications Server*

     Configure or verify the host name configuration by adding a **SYSUNIQ SYSNAME()** statement in the appropriate CA TCPCFG*nn* PARM member.
4. Which operating environment is SAS running under?
   - ☐ *OS/390*

     Configure SAS to use the *SAS/C Name Resolver*.
   - ☐ *z/OS*

     Which DNS Name Resolver do you want to use?
     - ☐ *IBM z/OS Name Resolver*

       Apply the zap in Usage Note 2143 to the SAS Transient Library.

       Configure SAS to use the IBM z/OS Name Resolver.
     - ☐ *SAS/C Name Resolver*

       Configure SAS to use the SAS/C Name Resolver.

5 Verify that appropriate services are configured for SAS/CONNECT or SAS/SHARE in the SERVICES file. For details, see “The Services File” on page 44.

❶ and ❷ For information about TCP/IP stacks and how to determine whether a system is running using single or multiple TCP/IP stacks, see “TCP/IP Stacks” on page 29. For details about the SAS/C environment variable and the SAS system option, see “SAS/C Environment Variables and SAS 9.1 System Options” on page 40.

❸ Consult the person who installed SAS on your system. For information about host names, see “TCP/IP Host Name Configuration” on page 31. For information about configuring TCP/IP stack files, see “TCP/IP Stack Configuration Files” on page 33.

❹ Consult your system administrator. For information about configuring the appropriate TCP/IP Name Resolver, see “TCP/IP Name Resolver Configuration” on page 35. Also, see this topic for details about applying the zap in Usage Note 2143 to the SAS Transient Library.

If you are using the SAS/C Name Resolver, see “SAS/C Environment Variables and SAS 9.1 System Options” on page 40, “SAS/C Environment Variables in the SASCTCPV Data Set” on page 42, and “The UNIX System Services (USS) Shell” on page 42.

TCP/IP Overview

TCP/IP is a set of layered protocols that enable cooperating computers to perform tasks and to share resources across a network. TCP/IP is comprised of TCP and IP.

TCP is a set of routines that applications use to communicate with another computer over a network. All applications do *not* use TCP. However, all network applications require the services that are provided in IP. IP is a set of routines that TCP calls, but the IP routines are also available to applications that do *not* use TCP. SAS 9.1 uses both TCP and IP, and requires that certain types of information be made available to the operating environment.

Although you might refer to a computer by using its host name, TCP/IP applications refer to computers by using their IP addresses. To facilitate the use of host names in a network, the Domain Name System translates host names to IP addresses. This Domain Name System provides host-to-IP address mapping through network server hosts, which are called *domain name servers*. The Domain Name System also provides other information about server hosts and networks, such as the TCP/IP services that are available to the server host and the location of the domain name servers in the network.

TCP/IP: Software Requirements

Verify that these software requirements have been met:

- □ The 7.50C version of the SAS Transient Library that is shipped with SAS 9.1 has been installed.
- □ One of the following TCP/IP packages has been installed:
  - □ IBM OS/390 IP Communications Server
  - □ IBM z/OS IP Communications Server
  - □ Computer Associates (CA) Unicenter TCPaccess Communications Server 5.3 or later.

    Note: The Unicenter TCPaccess Communications Server 5.3 must be running at maintenance level SP0208 and requires fix TP09208 from Computer Associates. △
- □ The UNIX System Services (USS) file system is available.
- □ A default OE segment (or an individual OE segment for each user ID) is required and must be defined in the security software (such as RACF).

- ☐ Ensure that the appropriate software zaps have been applied to SAS, as necessary. Zaps are included on the SAS software install tape and are documented in usage notes.

  UN2143
  : To use the SAS Transient Library with the z/OS Name Resolver instead of the SAS/C Name Resolver, apply this zap.

  M7504151
  : To change the default prefix for a data-set name from TCPIP to another name, apply this zap.

SAS Transient Library

The SAS Transient Library contains various modules and routines that SAS 9.1 uses during execution. The SAS Transient Library is also required for communication between SAS 9.1 and the TCP/IP Communications Servers. The library is automatically unloaded from tape during the installation process into a data set that is named *&prefix*.SASC.TRANSLIB. The prefix is a *high-level qualifier* of SAS 9.1 installation libraries. *&prefix*.SASC.TRANSLIB is copied to a link-list library or to the LPA.

The SAS Transient Library is made available to SAS 9.1 in one of the following ways:

1. The CTRANSLOC option that is specified in the SAS config file. This is the default.

   The CTRANSLOC option is generated in the BATCH and TSO members of the CNTL data set during installation. For example, CTRANSLOC=*&prefix*.SASC.TRANSLIB. For details, see "Processing the CTRANSLOC Option" on page 27.
2. *&prefix*.SASC.TRANSLIB is copied to a link-list library of the LPA.
3. *&prefix*.SASC.TRANSLIB is added to the STEPLIB or the TASKLIB concatenations in the SAS cataloged procedure and to the SAS CLIST, respectively.
4. An allocation for the CTRANS DD is added to the *&prefix*.SASC.TRANSLIB data set in the SAS cataloged procedure and SAS CLIST.

   An example of an allocation for BATCH follows:

   ```
   //CTRANS DD DISP=SHR,DSN=&prefix.SASC.TRANSLIB
   ```

   An example of an allocation for TSO follows:

   ```
   ALLOC F(CTRANS) DA('&prefix.SASC.TRANSLIB') SHR
   ```

Processing the CTRANSLOC Option

CTRANSLOC=CTRANS

If CTRANS is already allocated, a note is posted to the job log:

Note: A note is not posted to the SAS log because the SAS log is not yet available when this message is generated. △

```
NOTE: C TRANSIENTS WILL BE LOADED FROM location
```

If CTRANS has not been allocated, a note is posted to the job log:

Note: A note is not posted to the SAS log because the SAS log is not yet available when this message is generated. △

```
NOTE: C TRANSIENTS WILL BE LOADED FROM LIBRARIES IN THE NORMAL SEARCH PATH.
```

CTRANSLOC= DDname or Data Set Name

If CTRANS is already allocated:

1 CTRANSLOC obtains the name of the data set that was allocated to CTRANS.
   - If DDname is specified as the value of CTRANSLOC, CTRANSLOC obtains the name of the data set that is associated with the DDname.
   - If no data set is associated with the DDname, a warning is posted to the job log:

     Note: A warning is not posted to the SAS log because the SAS log is not yet available when this message is generated. △

     ```
     WARNING: NO DATA SET ALLOCATED TO DDNAME name.
     C TRANSIENTS WILL BE LOADED FROM LIBRARIES IN THE NORMAL SEARCH PATH.
     ```

2 CTRANSLOC compares the names of the data sets.
   - If the names of the data sets match, a note is posted to the job log:

     ```
     NOTE: C TRANSIENTS WILL BE LOADED FROM location.
     ```

   - If the names of the data sets do *not* match, a warning is posted to the job log:

     ```
     WARNING: THE VALUE SPECIFIED IN CTRANSLOC IS IGNORED
     BECAUSE CTRANS IS ALREADY ALLOCATED TO DATA SET name.
     ```

If CTRANS has not been allocated:

1 If DDname is specified as the value for CTRANSLOC, CTRANSLOC obtains the name of the data set that is associated with the DDname.

2 If no data set is associated with the DDname, a warning is posted to the job log:

   Note: A warning is not posted to the SAS log because the SAS log is not yet available when this message is generated. △

   ```
   WARNING: NO DATA SET ALLOCATED TO DDNAME name.
   C TRANSIENTS WILL BE LOADED FROM LIBRARIES IN THE NORMAL SEARCH PATH.
   ```

3 The data set is allocated to CTRANS.
   - If there is an error in allocation, a warning is posted to the job log:

     Note: A warning is not posted to the SAS log because the SAS log is not yet available when this message is generated. △

     ```
     WARNING: DATA SET name COULD NOT BE ALLOCATED TO CTRANS.
     C TRANSIENTS WILL BE LOADED FROM LIBRARIES IN THE NORMAL SEARCH PATH.
     ```

   - If the data set is allocated successfully, a note is posted to the job log:

     ```
     NOTE: C TRANSIENTS WILL BE LOADED FROM location
     ```

CTRANLSLOC=Null

If CTRANS is already allocated or has not been allocated, a note is posted to the job log:

Note: A note is not posted to the SAS log because the SAS log is not yet available when this message is generated. △

```
NOTE: C TRANSIENTS WILL BE LOADED FROM LIBRARIES IN THE NORMAL SEARCH PATH.
```

TCP/IP Stacks

TCP/IP Communication Stack: Definition

TCP/IP stack is a term for the set of protocols that comprise TCP/IP. A TCP/IP Communication stack that runs under the OS/390 and z/OS operating environments is implemented as a UNIX System Services (USS) physical file system (PFS). An operating environment can run using one or more TCP/IP stacks.

Note: A *TCP/IP stack* is also referred to as a *transport driver*. △

The IBM INET physical file system type supports a single TCP/IP stack. The IBM CINET physical file system type supports multiple stacks.

Note: If you will configure only one TCP/IP stack and you have access to both INET and CINET, it is advisable to configure the stack under INET because of its efficiency over CINET. △

Sample Definitions of TCP/IP Stacks

USS physical file systems are configured in the IBM parmlib member BPXPRM*nn*. The following examples show typical entries in the BPXPRM*nn* parmlib member for INET and CINET physical file systems.

- Defining a Single IBM TCP/IP Stack

  The following example shows the statements in the IBM parmlib member BPXPRM*nn* that define an INET physical file system for a single IBM TCP/IP stack system.

```
FILESYSTYPE TYPE(INET) ENTRYPOINT(EZBPFINI)
   NETWORK DOMAINNAME(AF_INET)
      DOMAINNUMBER(2)
      MAXSOCKETS(64000)
      TYPE(INET)
```

- Defining a Single CA TCPaccess Stack

  The following example shows the statements in the IBM parmlib member BPXPRM*nn* that define an INET physical file system for a single CA TCPaccess stack system.

```
FILESYSTYPE TYPE(INET) ENTRYPOINT(T01OPFSA)
 PARM('SYSID(ACSS)')
   NETWORK DOMAINNAME(AF_INET)
      DOMAINNUMBER(2)
      MAXSOCKETS(64000)
      TYPE(INET)
```

- Defining Multiple IBM TCP/IP Stacks

  The following example shows the statements in the IBM parmlib BPXPRM*nn* member that define a multiple TCP/IP stack system. This example includes three stacks:

  - two IBM stacks, TCPIP and TCPIP2
  - one CA TCPaccess stack, SNSTCP.

  The values of the FILESYSTYPE substatements, TYPE and ENTRYPOINT, define the PFS type and the entry point for the CINET PFS. CINET requires a SUBFILESYSTYPE statement for each stack. The NAME substatement names the TCP/IP stack that is being defined. This name is also used as the name of the Started Task that invokes the TCP/IP stack.

Note: CINET requires the NAME substatement. △

An optional SUBFILESYSTYPE substatement named DEFAULT defines the default TCP/IP stack for a multiple stack system. If DEFAULT is not specified or if the default stack is not active, the first stack that is activated is the default stack.

Note: The entry point for CA's TCPaccess TCP/IP stack must be T010PFSA. △

```
FILESYSTYPE TYPE(CINET) ENTRYPOINT(BPXTCINT)
   NETWORK DOMAINNAME(AF_INET)
      DOMAINNUMBER(2)
      MAXSOCKETS(64000)
      TYPE(CINET)
      INADDRANYPORT(63000)
      INADDRANYCOUNT(1000)

   SUBFILESYSTYPE NAME(TCPIP)
      TYPE(CINET)
      ENTRYPOINT(EZBPFINI)
      DEFAULT

   SUBFILESYSTYPE NAME(TCPIP2)
      TYPE(CINET)
      ENTRYPOINT(EZBPFINI)

   SUBFILESYSTYPE NAME(SNSTCP)
      TYPE(CINET)
      ENTRYPOINT(T010PFSA)
      PARM('SYSID(ACSS)')
```

For complete details, see the IBM documentation *z/OS UNIX System Services Planning*, GA22-7800.

System and Process Limits

The following IBM system values are set in the parmlib member BPXPRM*nn* and affect the number of TCP/IP sockets that SAS can use.

MAXSOCKETS
: system limit; specifies the maximum number of sockets that can be obtained for a given file system type. IBM recommends that this value be set to 64000.

MAXFILEPROC
: process limit; specifies the maximum number of file descriptors that a single process can have open concurrently, such as all open files, directories, sockets, and pipes. This value is usually set to 256. However, for heavy server use, it is advisable to set this number to 2000.

Note: You can use the RACF ALTUSER or ADDUSER system commands to set MAXFILEPROC on a per-user basis. △

For complete details about MAXSOCKETS and MAXFILEPROC, see the IBM documentation *z/OS UNIX System Services Planning*, GA22-7800.

TCP/IP Host Name Configuration

IP Addresses

In order for a process to connect to a host machine via TCP/IP, the process must know the IP address of the host machine. To obtain the IP address, the process calls the following functions:

gethostname()
: retrieves a string that contains its host name.

gethostbyname()
: resolves the host name string to its IP address.

Because each host name is associated with a TCP/IP stack, it is critical that the host name be configured correctly for each TCP/IP stack.

TCP/IP Host Name Configuration for Communications Servers

Configuration for TCP/IP host names varies according to the communications server that is used.

- □ TCP/IP Host Names for IBM Communications Servers

  When an IBM TCP/IP stack starts, the configuration process searches the TCPIP.DATA data set, which contains configuration statements, to locate its host name. For details, see "TCP/IP Stack Configuration Files" on page 33.

  - □ Search Order to Locate Stack Host Name

    1. If the IBM stack reads a TCPIP.DATA HOSTNAME configuration statement, it saves this value as the stack's host name.
    2. If a TCPIP.DATA HOSTNAME configuration statement is not read, the TCP/IP stack searches for the Virtual Machine Communication Facility (VMCF) node name from VMCF and uses its node name as the stack's host name.

       Note: VMCF should be running before any TCP/IP stacks are started. △
    3. If VMCF is not running when the TCP/IP stack is started, the TCP/IP stack's host name is determined by the release of the operating environment.
       - □ Under OS/390 2.10 and earlier releases, the stack's host name is set to a NULL string.
       - □ Under z/OS 1.2 and later releases, the stack's host name is set to the CVTSNAME, which is the SYSNAME=*value* in IEASYS*nn* that was used when the system was started.

- □ Multiple Host Names in a Single File

  As an option, you can insert a prefix *system_name* in TCPIP.DATA configuration statements. Using prefixes enables you to configure multiple hosts in a single TCPIP.DATA data set. The *system_name* prefix is matched against the system name that the TCP/IP stack is started under. The *system_name* is identical to the VMCF node name. The TCP/IP stack reads and processes the TCPIP.DATA configuration statements in the order that they appear in the data set.

  The following example shows a HOSTNAME statement in a TPCIP.DATA data set:

  ```
  SDCESA:   HOSTNAME TEST
  S390DEVA: HOSTNAME DEV
  SDCMVS:   HOSTNAME PROD
  ```

The *system_name* is specified in the first column; the associated *host_name* is specified in the final column.

A TCP/IP stack that started on the system named SDCESA would set its host name to TEST. A TCP/IP stack that started on the system named S390DEVA would set its host name to DEV. A TCP/IP stack that was started on the system named SDCMVS would set its host name to PROD.

The following rules are used to process HOSTNAME statements:

1. If the *system_name* prefix does not match a host name, the configuration statement is ignored.
2. If the *system_name* prefix is not located, the configuration statement is applied to all hosts.
3. If the *system_name* matches a host name, the associated configuration statement is applied to that host.
4. The final configuration statement that matches a *system_name* remains in effect.

A HOSTNAME statement is ignored under these conditions:

- the *system_name* prefix did not match the VMCF node name
- VMCF is unavailable
- The system name does not match the MVS name of the system that the TCP/IP stack started under.

- IBM System Name Considerations

  The Virtual Machine Communication Facility (VMCF) node name is used as the *system_name* prefix when processing IBM TCPIP.DATA configuration statements. The VMCF can be configured in two ways:

  - as a restartable subsystem

    If you have configured VMCF as a restartable subsystem, the node name is obtained from the value of the P= parameter in the EZAZSSI started procedure.

  - as a non-restartable subsystem

    If you configured VMCF as a non-restartable subsystem, the node name is specified in the IEFSSN*nn* member of PARMLIB.

  Note: IBM recommends that the MVS system name be used for the VMCF node name specification. △

  For details about configuring VMCF, see the IBM documentation *z/OS Communication Server: IP Configuration Guide*, SC31-8775.

- TCP/IP Host Names for the CA TCPaccess Communications Server

  When a CA TCPaccess TCP/IP stack starts, the configuration process searches the TCPCFG*nn* PARM member, which contains configuration statements, to locate its host name. The host name is defined by the SYSUNIQ SYSNAME(*host_name*), which is stored in the TCPCPG*nn* TCPIP.PARM member. For details about this file, see “CA TCPCPGnn TCPIP.PARM Member” on page 34.

  If a configuration statement is not located in the TCPCFG*nn* PARM member, the host name is obtained from the SYS1.PARMLIB(IEASY*nn*) SYSNAME=*system_name*.

  For complete details, see the CA documentation *Unicenter TCPaccess Communications Server: Customization Guide 6.0*.

  Note: Support for SYSUNIQ for releases prior to TCPaccess 6.0 requires the following maintenance:

- □ maintenance level SP0208
- □ fix TP09208.

△

TCP/IP Stack Configuration Files

IBM and CA TCP/IP Stack Configuration Files

When a TCP/IP stack is started, the TCP/IP stack reads one or more configuration files that contain statements that define its default behavior. For details, see “IBM PROFILE.TCPIP File” on page 33, “IBM TCPIP.DATA File” on page 34 and “CA TCPCPGnn TCPIP.PARM Member” on page 34.

IBM PROFILE.TCPIP File

□ Important Statements

The following PROFILE.TCPIP statements can restrict the ports that SAS servers can use:

PORT
: reserves ports for server tasks. The PORT statement specifies only the job names and PROC names that are allowed access to the port.

PORTRANGE
: same as PORT parameter but for a range of ports.

RESTRICT
: defines a list of user IDs that are prohibited from using TCP/IP.

RESTRICTLOWPORTS
: restricts the use of ports 1 to 1023 to specific job names or PROC names that are specified in the PORT or the PORTRANGE statement.

For details about the IBM PROFILE.TCPIP statements, see the IBM documentation *z/OS Communication Server: IP Configuration Reference*, IBM SC31-8776.

□ Search Order to Locate PROFILE.TCPIP

The search order that is used by the IBM TCP/IP stack to locate PROFILE.TCPIP involves both explicit and dynamic data-set allocation, as follows:

1. //PROFILE DD DSN=
2. *jobname.nodename*.TCPIP
3. hlq.*nodename*.TCPIP
4. *jobname*.PROFILE.TCPIP
5. TCPIP.PROFILE.TCPIP

Note: IBM recommends explicitly specifying the PROFILE DD statement in the TCPIPROC JCL. △

□ Neither SAS nor the SAS Transient Library access the PROFILE.TCPIP file directly. However, because this file is used to configure the IBM TCP/IP stack, the statements in this file can have an indirect effect on how SAS functions.

IBM TCPIP.DATA File

- □ Important Statements

  The TCPIP.DATA file contains the following statements that are used to configure the IBM TCP/IP stack and Communication Server applications.

  TCPIPJOBNAME (or TCPIPUSERID)
  : specifies the member name of the procedure that is used to start the TCPIP address space, which is the TCP/IP stack name.

  HOSTNAME
  : is used by the TCP/IP stack to determine its host name.

  DATASETPREFIX
  : is a high-level qualifier (hlq) for the dynamic allocation of data sets in IBM TCP/IP servers and clients.

  For complete details about the IBM TCPIP.DATA statements, see the IBM documentation *z/OS Communication Server: IP Configuration Reference*, IBM SC31-8776.

- □ Search Order to Locate TCPIP.DATA

  The IBM TCP/IP stack and the IBM Communication Server applications, including its Name Resolvers, use the following search order to locate the data set that contains the TCPIP.DATA configuration statements:

  1. GLOBALTCPIPDATA value (z/OS 1.2 and later releases)
  2. RESOLVER_CONFIG environment variable
  3. **`/etc/resolv.conf`**
  4. SYSTCPD DD
  5. *jobname*.TCPIP.DATA
  6. SYS1.TCPPARMS(TCPDATA)
  7. DEFAULTTCPIPDATA value (z/OS 1.2 and later releases)
  8. TCPIP.TCPIP.DATA

Note: The SAS Transient Library also uses TCPIP.DATA under certain circumstances. For details, see “SAS/C Name Resolver: Overview” on page 37. △

CA TCPCPGnn TCPIP.PARM Member

The CA Unicenter TCPaccess Communications Server uses a TCPCFG*nn* TCPIP.PARM member to configure its TCP/IP stack at start-up.

- □ Important Statements

  The following TCPCFG*nn* statements are used to configure the TCP/IP stack and to restrict the ports that SAS servers can use:

  SYSUNIQ SYSNAME(*host_name*)
  : defines the host name that is associated with the TCP/IP stack

  TCP PORTASGN
  : restricts the range of port numbers.

  For details about the TCPaccess TCPCFG*nn* statements, see the CA documentation *Unicenter TCPaccess Communications Server: Customization Guide 6.0*.

TCP/IP Name Resolver Configuration

Name Resolver: Definition

A name resolver is a set of routines that act as a client on behalf of an application to read a local host file or to access one or more domain name servers (DNS) for name-to-address or address-to-name resolution. Name resolution occurs by calling the name resolver functions *gethostbyname()* and *gethostbyaddr()*.

A name resolver must be configured for each host. Configuration files under UNIX were traditionally placed in the **`/etc`** directory. The local host configuration data is in the **`/etc/hosts`** file. The DNS domain name and the name servers IP addresses are in the configuration file **`/etc/resolv.conf`**.

Supported Name Resolvers

- "IBM OS/390 Name Resolvers: Overview" on page 35 (not supported)
- "IBM z/OS Name Resolver: Overview" on page 35
- "SAS/C Name Resolver: Overview" on page 37
- CA TCPaccess Domain Name Resolver (DNR) (not supported).

IBM OS/390 Name Resolvers: Overview

Prior to z/OS V1R2, IBM supported six name resolvers, depending on the IBM application or TCP/IP API. These resolvers were based on versions of BIND that preceded Version 4.8.3. Some of these resolvers used different configuration data sets or files and different search orders for these data sets.

The primary file that was used for most of the IBM TCP/IP resolvers was the TCPIP.DATA data set, which

- configured the IBM TCPIP stack
- configured the TCPIP Name resolver (instead of using the RESOLV.CONF resolver configuration file).

The primary directives for the IBM TCP/IP Name Resolver are:

- DOMAINORIGIN (alias DOMAIN)
- NSINTERADDR (alias NAMESERVER)
- NSPORTADDR
- RESOLVERTIMEOUT
- RESOLVERUDPRETRIES
- RESOLVEVIA.

For details about the IBM TCPIP.DATA statements, see the IBM document *z/OS Communication Server: IP Configuration Reference*, IBM SC31-8776.

Note: The SAS Transient Library does *not* use any of the IBM z/OS Name Resolvers. Instead, it uses its own SAS/C Name Resolver that can use the IBM TCPIP.DATA file. △

IBM z/OS Name Resolver: Overview

Starting with z/OS V1R2, IBM introduced a name resolver that was designed to replace all previous IBM name resolvers. The IBM z/OS Name Resolver is based on the most recent version of the BIND Name Resolver, Version 9. Instead of running in the application's address space, the IBM z/OS Name Resolver runs in a separate address space and is executed by a started task. The IBM z/OS Name Resolver must be running before any resolver API calls can be made.

When the IBM z/OS Name Resolver is started, it reads a new IBM configuration file that is pointed to by the DD statement SETUP, which can contain the following SETUP directives.

COMMONSEARCH | NOCOMMONSEARCH

DEFAULTIPNODES

DEFAULTTCPIPDATA

GLOBALIPNODES

GLOBALTCPIPDATA.

Note: The most important SETUP directives are GLOBALTCPIPDATA and DEFAULTTCPIPDATA. △

GLOBALTCPIPDATA
: identifies a global TCPIP.DATA file. Any TCPIP.DATA directive that is specified in this file are system-wide and cannot be overridden by a local TCPIP.DATA file.

DEFAULTTCPIPDATA
: identifies a default TCPIP.DATA file, which overrides the TCPIP.DATA file that is named TCPIP.TCPIP.DATA.

If a GLOBALTCPIPDATA statement is located in the resolver setup file, the IBM z/OS Name Resolver will read any name resolver directives that are located in this global TCPIP.DATA file. The IBM z/OS Name Resolver will then search for a local TCPIP.DATA file in the following order:

1. RESOLVER_CONFIG environment variable
2. `/etc/resolv.conf`
3. SYSTCPD DD
4. *jobname*.TCPIP.DATA
5. SYS1.TCPPARMS(TCPDATA)
6. DEFAULTTCPIPDATA value (if specified in the z/OS Name Resolver setup file)
7. TCPIP.TCPIP.DATA

Some useful IBM z/OS Name Resolver Server directives include:

LOOKUP
: changes the order in which name resolution is performed between a DNS name server and a local hosts file. Using the LOOKUP directive, you can specify DNS (only), LOCAL (only), DNS LOCAL, or LOCAL DNS. By default, a DNS name server is queried first. If DNS fails, then DNS LOCAL is used.

SEARCH
: specifies a search of up to six domains, in the specified order. The first domain name that is specified is used as the value for DOMAINORIGIN. If both the SEARCH and DOMAINORIGIN statements are specified, the one that appears last is used.

SORTLIST
: specifies up to four IP addresses to use for a specific host. If DNS returns more than one IP address for a host, SORTLIST can use search masks to sort and identify which IP address the resolver returns.

OPTIONS
: specifies that for a domain name that contains *n* or more periods (.), the resolver should look up the name as is before applying the DOMAINORIGIN or SEARCH statement settings. The range of *n* is 1 to 15. The default is 2.

For complete information about these directives, see the IBM documentation *z/OS IP Configuration Guide and IP Configuration Reference*.

SAS/C Name Resolver: Overview

The SAS/C Name Resolver was ported from the BIND Name Resolver. The version of BIND that was ported, although current at the time of the port, was prior to Version 4.8.3. Because the nonstandard TCPIP.DATA file is used instead of **`/etc/resolv.conf`**, significant changes were made to the BIND code in order to support this data set, including the code to search for this file.

The SAS/C Name Resolver provides limited compatibility with the IBM resolvers by means of environment variables that are used to configure the SAS/C Name Resolver. Because the TCPIP.DATA file usually configures resolvers and TCP/IP stacks, the presence of multiple TCP/IP stacks changes the behavior of the SAS/C Name Resolver.

When the first TCP/IP function call is made to the SAS Transient Library, the library loads (by default) and initializes its "OE" socket library (LSCNOE). Initialization includes code that identifies whether one or more TCP/IP stacks is running. This code calls the UNIX System Services pfsctl(BPX1PCT), using the TCP/IP stack name ",".

- Only One TCP/IP Stack Is Detected

  If one TCP/IP stack is detected, pfsctl() returns an errno of ENXIO.

  No attempt is made to search for and read the TCPIP.DATA file.

- TCPIP_MACH Is Located

  1. If pfsctl() returns any value that is not ENXIO, it is assumed that multiple TCP/IP stacks are running.

     The SAS Transient Library searches for the SAS/C environment variable TCPIP_MACH.

  2. If TCPIP_MACH is set, the SAS Transient Library uses that value.

     No further attempt is made to search for and read the TCPIP.DATA file.

- TCPIP.DATA Is Located or Default TCP/IP Stack Name Is Used

  1. If TCPIP_MACH is not set, the SAS Transient Library searches for the TCPIP.DATA file, which contains the TCIPJOBNAME (or TCPIPUSERID) directive.

     If the TCPIPJOBNAME directive specifies the TCP/IP stack, pfsctl() is called with the value that is set by the directive.

  2. If the TCPIP.DATA file is not found or it does not contain TCPIPJOBNAME, pfsctl() is called with the default TCP/IP stack name TCPIP.

     The SAS Transient Library reads a TCPIP.DATA file.

- Name Resolver Directives Are Located

  1. When the SAS Transient Library reads a TCPIP.DATA file, it caches the values of all statements or directives that are found, including name resolver directives.

     When the first resolver function is called (gethostbyname() or gethostbyaddr()), the SAS Transient Library searches for a RESOLV.CONF name resolver configuration file, in the following order:

     a. value of ETC_RESOLV_CONF environment variable, if defined
     b. **`/etc/resolv.conf`**

   c. tso-prefix.ETC.RESOLV.CONF if under TSO
   d. ETC.RESOLV.CONF
   e. *tcpip-prefix*.ETC.RESOLV.CONF, if TCPIP_PREFIX is defined.

2. If the SAS Transient Library locates a resolver configuration file, it uses the resolver configuration file to configure the SAS/C Name Resolver and it clears its cache.

   Note: The UNIX and IBM resolver directives are recognized. △

□ Cached Resolver Data or TCPIP.DATA File Is Located

1. If a RESOLV.CONF file is not located, the SAS Transient Library uses any cached resolver data from a previously read TCPIP.DATA file.
2. If there is no cached resolver data, the SAS Transient Library searches for and reads a TCPIP.DATA file for resolver directives and uses these directives to configure the SAS/C Name Resolver.

   The SAS Transient Library searches for the IBM TCPIP.DATA file in the following order:

   a. the SAS/C environment variable TCPIP_DATA string
   b. the data set that is identified by the DDname SYSTCPD
   c. the *tso-prefix*.TCPIP.DATA, if under TSO
   d. the SYS1.TCPPARMS(TCPDATA) data set
   e. the SAS/C environment variable TCPIP_PREFIX
   f. the *tcpip_prefix*.TCPIP.DATA
   g. the default value of TCPIP_PREFIX
   h. the *default-value*.TCPIP.DATA
   i. TCPIP.DATA.

□ Hosts Files Are Located

If a TCPIP.DATA file is not located or the resolver directory is not read, the SAS Transient Library searches for and attempts to read a local hosts file; for example, ETC.HOSTS, in the following order:

1. value of ETC_HOSTS environment variable, if defined
2. **`/etc/hosts`**
3. *tso-prefix*.ETC.HOSTS if under TSO
4. ETC.HOSTS
5. *tcpip-prefix*.ETC.HOSTS, if TCPIP_PREFIX is defined

   Example excerpt from a HOSTS file:

```
# The form for each entry is:
#  <internet address>  <official hostname>  <aliases>
# For example:
# 192.1.2.34    hpfcrm  loghost
#
# See the hosts(4) manual page for more information.
# Note: The entries cannot be preceded by a space.
#       The format described in this file is the correct format.
#       The original Berkeley manual page contains an error in
#       the format description.
#

10.19.0.127     snstcp
127.0.0.1       localhost       loopback
```

□ Limitations of the SAS/C Name Resolver

The SAS/C Name Resolver does not support the IBM Name Resolver setup directives or the new resolver configuration directives. The SAS/C Name Resolver ignores any directive that it does not recognize.

In order for the SAS/C Name Resolver to correctly use a TCPIP.DATA file that was written for the IBM Name Resolver, the DOMAINORIGIN directive must be included in the TCPIP.DATA file. If one or more SEARCH directives are used in the file, the DOMAINORIGIN directive must precede them. The SAS/C Name Resolver reads the DOMAINORIGIN directive but ignores SEARCH directives, because the SAS/C Name Resolver does not recognize them. However, the IBM Name Resolver reads the DOMAINORIGIN directive, but any SEARCH directives that follow will override the behavior of the DOMAINORIGIN directive.

If your site uses any of the new features of the IBM z/OS Name Resolver, it is highly recommended that the SAS Transient Library be configured to use the IBM z/OS Name Resolver instead of the SAS/C Name Resolver.

□ Configuring the SAS Transient Library to Use the IBM z/OS Name Resolver

If you use the SAS Transient Library and want to use the z/OS Name Resolver instead of the SAS/C Name Resolver, apply a zap (see Usage Note UN2143), which causes the SAS Transient Library to call the z/OS Resolver functions instead of the SAS/C Resolver functions.

□ Configuring the CA Unicenter TCPaccess Communication Server to Use a Name Resolver

Although the CA Unicenter TCPaccess Communication Server provides the DNR (Domain Name Resolver), neither SAS nor SAS/C can use it. Therefore, if you are using the TCPaccess stack, you must use the SAS/C Name Resolver.

□ Configuring SAS to Use the SAS/C Name Resolver

The SAS/C Name Resolver must be configured before the SAS Transient Library can search for and access it. The resolver can be configured by using the following SAS/C environment variables, SAS system options, and files in the operating environment:

ETC_RESOLV_CONF environment variable
: Assigning an IBM TCPIP.DATA file to this environment variable prevents the SAS Transient Library from using a long search list. This file does *not* contain TCPIPJOBNAME or the TCPIPPREFIX statements.

TCPIP_DATA environment variable
: If access to TCPIPJOBNAME or the TCPIPPREFIX statements is needed, then TCPIP_DATA should be set.
: If SAS is running under a multiple TCP/IP stack system, and unless TCPIP_MACH is specified, the SAS Transient Library must have access to the TCPIP.DATA file, which contains a TCPIPJOBNAME statement.

TCPIP_MACH environment variable
: specifies the version of TCP/IP that is running when multiple TCP/IP stacks are being used.

TCPIPMCH system option
: This SAS system option is another implementation of the TCPIP_MACH environment variable.

TCPIP_PREFIX environment variable
: If multiple TCP/IP stacks are configured and prefixes are used to differentiate among names for TCP/IP stacks, this variable identifies the prefix name.

TCPIPPRF SAS system option
: This SAS system option is another implementation of the TCPIP_PREFIX environment variable.

SERVICES file
: specifies the services that are required by SAS/CONNECT and SAS/SHARE.

□ Configuring SAS to Use the IBM z/OS Name Resolver

In order for SAS to use the IBM z/OS Name Resolver, you must apply a zap (see Usage Note UN2143) to the SAS Transient Library that was supplied with SAS 9.1. If this zap is applied, configuration is complete.

However, if SAS runs multiple TCP/IP stacks, the SAS Transient Library must have access to resolver information that is contained in the following files:

TCPIP.DATA file
: contains a TCPIPJOBNAME statement and also contains a DATASETPREFIX statement.

TCPIP_MACH environment variable
: specifies the version of TCP/IP that is running when multiple TCP/IP stacks are being used.

TCPIPMCH system option
: This SAS system option is another implementation of the TCPIP_MACH environment variable.

TCPIP_PREFIX environment variable
: If multiple TCP/IP stacks are configured and prefixes are used to differentiate among names for TCP/IP stacks, this variable identifies the prefix name.

TCPIPPRF SAS system option
: This SAS system option is another implementation of the TCPIP_PREFIX environment variable.

SERVICES file
: specifies the ports that are available for TCP/IP stacks, and also might specify prefixes that are used to differentiate among multiple TCP/IP stacks.

SAS/C Environment Variables and SAS 9.1 System Options

SAS/C Environment Variables and SAS System Options: Definitions

If your site's TCP/IP system configuration files are not included in the search path, or the configuration files are not formatted as required by the SAS Transient Library, you can customize SAS 9.1 by using SAS/C environment variables and SAS system options.

SAS 9.1 uses the following SAS/C environment variables (and SAS system option equivalents) to alter default processing for TCP/IP initialization:

TCPIP_MACH=*name* (environment variable)
TCPIPMCH=*name* (SAS system option)
: is useful for sites that run TCP/IP from multiple vendors or multiple instances of the same vendor's TCP/IP simultaneously.

  The TCPIP_MACH environment variable can be used to specify the name of the TCP/IP stack name, which is also referred to as a *started task*. Setting TCPIP_MACH is the same as using the TCPIPJOBNAME/TCPIPUSERID configuration keywords in the IBM TCPIP.DATA file. If TCPIP_MACH is set, the SAS Transient Library does not search for these keywords.

The default name for the TCP/IP stack is TCPIP.

TCPIP_DATA=dsn:*data.set.name* (environment variable)
specifies the fully qualified name of an IBM TCPIP.DATA configuration data set. If the TCPIP_MACH environment variable is not set, the SAS Transient Library uses this file to locate the TCP/IP stack name. If a RESOLV.CONF file is not located, the SAS/C Name Resolver will also read this file to retrieve DNS name resolver directives. Specifying this environment variable will override the default search order for this file.

ETC_RESOLV_CONF=dsn:*data.set.name* (environment variable)
used by the SAS/C Name Resolver, specifies the fully qualified name of a RESOLV.CONF file that contains DNS name resolver directives for domain-name server processing. This data set name can be a TCPIP.DATA file because IBM uses this file to store resolver directives. Specifying this environment variable will override the default search order for this file.

ETC_HOSTS=dsn:*data.set.name* (environment variable)
used by the SAS/C Name Resolver, specifies the fully qualified name of the data set that contains host-name resolution information. For details, see "TCP/IP Name Resolver Configuration" on page 35.

If your site does not enable domain-name server processing and you are using the SAS/C Name Resolver, you must specify a local host file.

ETC_SERVICES=dsn:*data.set.name* (environment variable)
specifies the fully qualified name for the data set name that contains service names and port numbers. Specifying this environment variable will override the default search order for this file.

TCPIP_PREFIX=*high.level.qualifier* (environment variable)
TCPIPPRF=*high.level.qualifier* (SAS system option)
specifies a *high-level qualifier* for the various TCP/IP configuration data sets. The DATASETPREFIX keyword and the assigned value in the TCP/IP configuration file can also be used to specify the high-level qualifier.

For example, you might organize the configuration data sets under a single high-level qualifier, as follows:

```
TCPIPPRF=SYS2.TCP26
```

The TCP/IP prefix is used to name the data sets, as follows:

```
SYS2.TCP26.TCPIP.DATA
SYS2.TCP26.ETC.HOSTS
SYS2.TCP26.ETC.SERVICES
```

If a form of the TCP/IP prefix is used, the SAS Transient Library will not search for the DATASETPREFIX keyword. Also, specifying this environment variable will override the default data-set prefix.

The Default TCPIP Prefix

Unless you applied the zap to change the default prefix for a data-set name, TCPIP is the default.

If you cannot use ETC at your site and if DS names do not conflict with the ETC high-level qualifier, you can use the default prefix to produce names such as TCPIP.ETC.HOSTS.

Note: If you use the default TCPIP prefix, do not set the TCPIPPRF option to override the TCPIP prefix, and do not apply the zap to change the default value of the TCPIP prefix in the SAS Transient Library. △

Changing the Default TCPIP Prefix

To change the default DATASETPREFIX value of TCPIP, you can apply zap number M7504151, which is included on the SAS software install tape and is documented in usage notes.

For example, to change the name of the TCPIP.SERVICES file to SYS.PROD.CONFIG.ETC.SERVICES, use the provided zap. The default TCPIP prefix is changed to SYS.PROD.CONFIG.

SAS/C Environment Variables in the SASCTCPV Data Set

The following guidelines show how to store SAS/C environment variables in the data set SASCTCPV:

- Allocate the data set to the SASCTCPV DD, using RECFM=FB and LRECL=80.
- Do not enable line numbers in the SASCTCPV data set.
- If you use SAS/C environment variables, you must allocate the SASCTCPV DD in the JCL or the CLIST that executes SAS 9.1.

  For example, if the data set SAS.TCPIP.ENVIRON.DATA contains the environment variable information that you want, you would use the following allocation statements for BATCH and TSO:

  ```
   //SASCTCPV DD DISP=SHR,DSN=SAS.TCPIP.ENVIRON.DATA
  ```

  or

  ```
  ALLOC F(SASCTCPV) DA('SAS.TCPIP.ENVIRON.DATA') SHR
  ```

  Each logical record is assumed to contain an environment variable assignment in the form:

  ```
     environment_variable_name=value
  ```

Specifying SAS 9.1 system options that control configuration data sets will override the SAS/C environment variables that are set in the SASCTCPV data set.

The UNIX System Services (USS) Shell

Shell Configuration Requirements

Some SAS 9.1 applications, such as the Broker for SAS/IntrNet, that execute under the UNIX System Services (USS) shell must be customized for the shell environment, which requires using the shell **`export`** command to set SAS/C environment variables. These SAS/C environment variables control how SAS 9.1 initializes and how it uses the SAS Transient Library and TCP/IP configuration files under the shell.

Store the SAS/C environment variables in the user's shell configuration file **`.profile`**, which is located in the user's home directory, or in the system configuration file **`/etc/profile`**.

The USS shell reads these configuration files when the shell starts. The shell requires the following information:

- The location of the SAS Transient Library
- If running multiple TCP/IP stacks, the name of the TCP/IP stack
- The name of the TCP/IP stack configuration data set TCPIP.DATA
- The name of file that is used for domain name-server processing

- □ The name of the file that is used for host table lookup
- □ The name of the SERVICES file.

Specifying the SAS Transient Library

The SAS Transient Library contains various modules and routines that are used by SAS 9.1 during execution. These libraries are unloaded from tape during SAS 9.1 installation. If the SAS Transient Library is not located in LINKLIST or LPALIB, you must specify the SAS Transient Library by using the SAS/C environment variable, ddn_CTRANS, in the following shell export command:

```
export ddn_CTRANS=&prefix.SASC.TRANSLIB
```

&prefix is a high-level qualifier of SAS 9.1 installation libraries.

You are advised to store this shell command in the shell system file **`/etc/profile`** to ensure that all SAS users who are executing under a USS shell will locate the SAS Transient Library.

Configuring TCP/IP Using SAS/C Environment Variables

You can set any SAS/C environment variable by using the shell **`export`** command

For example, to set the SAS/C environment variable TCPIP_MACH to specify the TCP/IP stack name on a multiple TCP/IP stack system, use the following shell **`export`** command:

```
export TCPIP_MACH=TCPIP2
```

TCPIP2 is the name of the TCP/IP stack, which is specified in the SYS1.PARMLIB(BPXPR4M*nn*) system parmlib.

For descriptions of the SAS/C environment variables, see "SAS/C Environment Variables and SAS 9.1 System Options" on page 40.

Note: With the exception of the SAS Transient Library, all MVS data sets need to be prefixed with //dsn: and any allocated DD NAME with //ddn:.

Example:

```
export TCPIP_DATA='//dsn:SSD.TCPACC53.TCPIP.DATA'
```

△

The Shell Profile File

During USS shell initialization, the shell reads and executes any shell commands that are located in these files, in this order:

1. the system shell profile file **`/etc/profile`**
2. the user file **`.profile`**, which is located in the invoking user's home directory.

SAS/C environment variables that are specified via **`export`** commands are automatically set when the shell is initialized.

A typical user **`.profile`** follows:

```
#----------------------------------------------
# .profile - default ksh login script.
#
# items in this file are executed once at login
```

```
#
#------------------------------------------------

#------------------------------------------------
# Set SAS Transient Library
#------------------------------------------------
export ddn_CTRANS='SAS.SASC.TRANSLIB'


#------------------------------------------------
# Set SAS/C TCP/IP environment variables
#------------------------------------------------
export TCPIP_MACH=SNSTCP
export TCPIP_DATA='//dsn:SSD.TCPACC53.TCPIP.DATA'
```

The Services File

Services File: Overview

The SERVICES file defines port resources that are used when TCP/IP is used to connect client/server sessions. Examples of configured port services include the telnet port, spawner ports, MP CONNECT pipes, and SAS/SHARE servers. For more information, see “Configuring the SERVICES File” on page 145.

The Services File Search Order

The SAS Transient Library searches for the SERVICES file, in the following order:

1 value of the ETC_SERVICES environment variable
2 **/etc/services**
3 *tso-prefix*.ETC.SERVICES under TSO or *user-ID*.ETC.SERVICES under batch execution
4 ETC.SERVICES
5 TCPIP.ETC.SERVICES
6 *tcpip-prefix*.ETC.SERVICES.

References

DNS and BIND, 4th Ed., by Paul Albitz & Cricket Liu, O'Reilly and Associates, Inc.
CA Unicenter TCPaccess Communications Server: Customization Guide 6.0
SAS/C Library Reference, Volume 2, SAS Institute Inc.
z/OS Communication Server: IP Configuration Guide, IBM SC31-8775
z/OS Communication Server: IP Configuration Reference, IBM SC31-8776
z/OS UNIX System Services Planning, IBM GA22-7800

CHAPTER 3 z/OS: XMS Access Method

Prerequisites for Using XMS under z/OS **46**
Task List **46**
Software Requirements **46**
Defining Resources for the XMS Communications Access Method **46**
SAS/CONNECT and SAS/SHARE Network Security **46**
SAS/SHARE SUBSYSID= Option **47**
SAS/CONNECT Client Tasks **47**
Task List **47**
Specifying XMS as the Communications Access Method **47**
Encrypting Data in Client/Server Transfers **48**
Signing On to the Same Multi-Processor Machine **48**
Specifying the Server Session **48**
Starting SAS Using the SASCMD Option **48**
Signing On to the Server Session **49**
SAS/CONNECT Server Tasks **49**
SAS/SHARE Client Tasks **49**
Task List **49**
Specifying XMS as the Communications Access Method **49**
Specifying the Server **50**
SAS/SHARE Client Example **50**
SAS/SHARE Server Tasks **51**
Task List **51**
Installing the SAS SVC Routine **51**
Specifying XMS as the Communications Access Method **51**
Specifying a Server Name **52**
SAS/SHARE Server Example **52**
System Configuration for the XMS Access Method **52**
Installation Tasks **52**
Steps for Installing the Load Module **53**
Defining an Anchor Point **53**
Anchor Point: Definition **53**
Steps for Defining an Anchor Point **53**

Prerequisites for Using XMS under z/OS

Task List

- □ Verify that software requirements are met.
- □ Define resources for the XMS access method.
- □ If using network security, set the appropriate SAS system options
- □ Set the SAS/SHARE SUBSYSID= option, if applicable.

Software Requirements

Ensure that

- □ Base SAS and either SAS/CONNECT or SAS/SHARE are installed on both the client and the server.
- □ XMS has been installed on both the client and the server.

Defining Resources for the XMS Communications Access Method

Network Administrator
Before you can use SAS/CONNECT and SAS/SHARE with the XMS communications access method, you must first define XMS resources for the z/OS operating environment. For the tasks to define resources for SAS/CONNECT and SAS/SHARE, see "System Configuration for the XMS Access Method" on page 52.

SAS/CONNECT and SAS/SHARE Network Security

Encryption is the process of transforming plaintext into a less readable form (called ciphertext) by using a mathematical process. The ciphertext is translated back to plaintext for anyone who can supply the appropriate key, which is necessary for decrypting (or unlocking) the ciphertext.

SAS/CONNECT and SAS/SHARE support the following network security services for protecting data on a network.

SASproprietary
a fixed encoding algorithm that is included with Base SAS software and is available in all SAS supported operating environments. It requires no additional SAS product licenses.

SAS/SECURE
an add-on product that provides additional encryption algorithms in addition to the SAS proprietary algorithm.

For complete details about using network security, see the *SAS/CONNECT User's Guide*. After network security is set up in your environment, you set a SAS encryption option that is appropriate to the network security service and to the requirements of the client or the server session.

SAS/SHARE SUBSYSID= Option

SUBSYSID=*anchor-point*
stands for subsystem identifier, which specifies the cross-memory anchor point that identifies the inactive z/OS subsystem. The subsystem is defined by your network administrator during the XMS access method configuration. For details, see “System Configuration for the XMS Access Method” on page 52.

Defining an inactive subsystem causes a z/OS machine to create a subsystem communications vector table (SSCVT) at IPL time. The SSCVT chain is in common memory and is easily accessible to the XMS access method routines. The SSCTSUSE field of the SSCVT is available to these routines and is used as the anchor point for their control blocks.

The default value for SUBSYSID= is SAS0. You must set this option to enable clients to access the server with the XMS communications access method. Set this option at both the SAS/SHARE server and at each client that will access the server.

SAS/CONNECT Client Tasks

Task List

1 Specify XMS as the communications access method.

2 Specify encryption of client/server data transfers (optional).

3 Sign on to the same SMP (Symmetrical Multi-Processor) machine.

Specifying XMS as the Communications Access Method

XMS is the default communications access method to sign on to one or more sessions on the same multi-processor machine that runs the z/OS operating environment. Therefore, you do not have to explicitly specify the default.

Note: TCP/IP is the default communications access method for all other operating environments. △

If you choose to explicitly specify XMS, you can use the COMAMID= option in an OPTIONS statement. For example:

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

COMAMID is an acronym for Communications Access Method Identification. *access-method-ID* identifies the method used by the client to communicate with the server. XMS, which is an abbreviation for Cross Memory Services, is an example of an *access-method-ID*. Alternatively, you can set this option in a SAS configuration file or in a SAS start-up command.

Example:

```
options comamid=xms;
```

Encrypting Data in Client/Server Transfers

If network security is available and is configured at the client, you can specify SAS options to encrypt all data that is transferred between a client and a server. In the following example, the NETENCRYPTALGORITHM= option specifies the DES encryption algorithm.

```
options netencryptalgorithm=des;
```

For complete details about network security options, see the *SAS/CONNECT User's Guide*.

Signing On to the Same Multi-Processor Machine

If your client machine is equipped with SMP (Symmetric Multi-Processors), and if you want to run one or more server sessions on your machine, perform these steps:

1 Specify the server session.
2 Specify the SASCMD command to start SAS.
3 Sign on to the server session.

Specifying the Server Session

You can specify the server session in an OPTIONS statement:

```
OPTIONS PROCESS=session-ID;
```

or in the SIGNON statement or command:

```
SIGNON session-ID;
```

session-ID must be a valid SAS name that is 1 to 8 characters in length, and is the name that you assign to the server session on the multi-processor machine.

Note: PROCESS=, REMOTE=, CREMOTE=, and CONNECTREMOTE= can be used interchangeably. For details, see the CONNECTREMOTE= system option in the *SAS/CONNECT User's Guide*. △

For details about SIGNON, see the SIGNON statement in the *SAS/CONNECT User's Guide*.

Starting SAS Using the SASCMD Option

Use the SASCMD option to specify the SAS command and any additional options that you want to use to start SAS in the server session on the same multi-processor machine.

The SASCMD option can be specified either in an OPTIONS statement:

```
OPTIONS SASCMD=":SAS-system-options" | "!SASCMD SAS-system-options" ;
```

or directly in the SIGNON statement or command:

```
SIGNON name SASCMD=":SAS-system-options" | "!SASCMD SAS-system-options" ;
```

Example:

```
options sascmd=":memsize=64M nonumber";
```

The -DMR option is automatically appended to the command. If *!SASCMD* is specified, SAS/CONNECT starts SAS on the server by using the same command that was used to start SAS for the current (parent) session.

Note: In order to execute additional commands prior to starting SAS, you might write a script that contains the SAS start-up commands that are appropriate for the operating environment. Specify this script as the value in the SASCMD= option. △

For details, see the SASCMD= system option and the SIGNON statement in the *SAS/CONNECT User's Guide*.

Signing On to the Server Session

Example 1:

In the following example, XMS is the access method, SAS1 is the name of the server session, and the MEMSIZE= option is used when starting SAS on a multi-processor machine.

```
options comamid=xms;
signon sas1 sascmd=":memsize=64M";
```

Example 2:

In the following example, OPTIONS statements set the values for the COMAMID= , the SASCMD=, and the PROCESS= options. The SASCMD= option is a non-blank value that causes the same CLIST that was used to start the client session to be used to start the server session. The PROCESS= option identifies the server session on the same multi-processor machine. Because the SASCMD= and the PROCESS= options are defined, only a simple SIGNON statement is needed.

```
options comamid= xms sascmd="abc";
options process=sas1;
signon;
```

SAS/CONNECT Server Tasks

There are no SAS/CONNECT server tasks.

SAS/SHARE Client Tasks

Task List

1 Specify XMS as the communications access method.

2 Specify a server name.

Specifying XMS as the Communications Access Method

XMS is the default communications access method in the z/OS operating environment. You can omit specifying the access method in a COMAMID statement and the XMS access method is assumed, by default.

If you choose to specify XMS to connect to a server, you can use the COMAMID= option in an OPTIONS statement.

```
options comamid=xms;
```

The COMAMID= option specifies the communications access method. XMS is an abbreviation for the Cross Memory Services access method.

Alternatively, you can specify the COMAMID= option in a SAS configuration file or in a SAS start-up command.

The COMAUX1= option specifies an auxiliary communications access method and can be used only in a SAS configuration file or in a SAS start-up command. If the first method that you specify in the COMAMID= option fails to access the server, the auxiliary access method is used. You can specify one auxiliary access method.

The syntax for the COMAUX1= option is:

```
COMAUX1=alternate-method
```

Example:

```
comamid=xms
comaux1=tcp
```

For details about the supported access methods, see "Supported Communications Access Methods by Operating Environment" on page 3.

Specifying the Server

To use the XMS access method, a server and a client must be running under the same z/OS operating environment.

To access the server, you specify the server name in the LIBNAME and PROC OPERATE statements using this syntax:

```
SERVER=server-ID
```

server-ID is the name that you assign to the server. The name can be a maximum of 8 characters in length.

For details about creating valid SAS names, see the *SAS Language Reference: Concepts*. For details about LIBNAME and PROC OPERATE, see the LIBNAME statement and the OPERATE procedure in the *SAS/SHARE User's Guide*.

SAS/SHARE Client Example

The following example shows the statements that you specify in a z/OS client configuration file to access a server by using the XMS access method. The XMS access method is assumed by default. The LIBNAME statement specifies the SAS data library that is accessed through the server SHARE1.

```
libname sasdata 'edc.prog2.sasdata' server=share1;
```

SAS/SHARE Server Tasks

Task List

1 Ensure that the SAS SVC routine has been installed.

2 Specify XMS as the communications access method.

3 Specify a server name.

Installing the SAS SVC Routine

The SAS SVC control program routine is an interface between the z/OS operating environment and a specific request, such as "third-party checking." This facility provides verification by requiring authentication of both the user ID and password and of library authority.

1 Install the SAS SVC routine, if necessary.

If you have already installed the SAS SVC routine for Release 6.09 of SAS, do not repeat the step now.

If you need to perform the installation, see the *Installation Instructions and System Manager's Guide, The SAS System under MVS*.

Because SAS SVC in Release 6.09 is backward compatible, it replaces the SAS SVC routines from previous releases. You can continue using previous releases of SAS and SAS/SHARE with the Release 6.09 SAS SVC that is installed on your machine.

2 Verify the SAS options for the SVC routine.

You must verify that SAS for the SVC routine accurately reflects the way that the SAS SVC is installed. The SAS option SVC0SVC and SAS SVC should be set to the same number. If the SAS SVC is installed at 109 as an ESR SVC, the SAS option SVC0R15 should be set to the ESR code (for example, 4).

3 Verify installation on all CPUs, as needed.

If you have more than one CPU, verify that the SAS SVC routine is installed on the machines that will run SAS/SHARE at your site.

Specifying XMS as the Communications Access Method

XMS is the default communications access method in the z/OS operating environment. You can omit specifying the access method in a COMAMID= option and the XMS access method is assumed, by default.

If you choose to specify XMS to connect to a server, you can use the COMAMID= option in an OPTIONS statement. For example:

```
options comamid=xms;
```

Alternatively, you can specify the COMAMID= option in a SAS configuration file or in a SAS start-up command.

The COMAUX1= option specifies an auxiliary communications access method and can be specified only in a SAS configuration file or in a SAS start-up command. If the first method that you specify in the COMAMID= option fails to access the server, the auxiliary access method is used. You can specify one auxiliary access method.

The syntax for the COMAUX1= option is:

```
COMAUX1=alternate-method
```

Example:

```
comamid=xms
comaux1=tcp
```

For details about the supported access methods, see "Supported Communications Access Methods by Operating Environment" on page 3.

Specifying a Server Name

To use the XMS access method, a server and a client must be running under the same z/OS operating environment.

Specify the server name in the PROC SERVER statement using this syntax:

```
SERVER=server-ID
```

server-ID is the name that you assign to the server. The name can be a maximum of 8 characters in length.

For details about creating valid SAS names, see *SAS Language Reference: Concepts*. For details about PROC SERVER, see the SERVER procedure in the *SAS/SHARE User's Guide*.

SAS/SHARE Server Example

The following example shows the statements that you specify to start the server SHARE1 in the z/OS operating environment. The XMS access method is assumed by default.

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

System Configuration for the XMS Access Method

Installation Tasks

1 Install the SASVXMS load module. See "Steps for Installing the Load Module" on page 53

Note: The version of SASVXMS that is distributed with each maintenance release of SAS/SHARE can be used only with that maintenance release. △

2 Define an anchor point. See "Defining an Anchor Point" on page 53.

Steps for Installing the Load Module

To install the SASVXMS0 load module, perform these tasks:

1 Copy SASVXMS0 into an authorized link list library.

- □ You can copy the module SASVXMS0 into any authorized library that is part of the link list.
- □ Alternatively, you can install this module into the link pack area.

  You can use any standard utility program to copy the module SASVXMS0 from your *HLQ*.LIBRARY data set to your link list library.

2 Rename SASVXMS0.

  After you copy SASVXMS0 into the appropriate library, you must rename it. You can use any standard utility to rename the module.

- □ If you have a previous release of SAS/SHARE installed, rename SASVXMS0 to SASVXMS*n*, where *n* is the last digit of the release of SAS. Specify the communications access method as XMS*n*.

  For example, for Release 6.08, rename SASVXMS0 to SASVXMS8 and specify the access method as XMS8. For details, see “Specifying XMS as the Communications Access Method” on page 51.
- □ If you do not have a previous release of SAS/SHARE installed, rename SASVXMS0 to SASVXMS. Specify the communications access method as XMS in the SAS configuration file for batch processing and in the TSO CLIST. For details, see “Specifying XMS as the Communications Access Method” on page 49.

  When SAS/SHARE loads the module SASVXMS, it must find that module marked as authorized, re-entrant, and reusable and that it was loaded from an authorized library.

Defining an Anchor Point

Anchor Point: Definition

The *anchor point* is a place in common memory that can be located by servers and clients and that is used to store and retrieve cross-memory communication information.

The anchor point is specified by defining an inactive z/OS subsystem. Doing this causes z/OS to create a subsystem communications vector table (SSCVT) at IPL time. The SSCVT chain is in common memory and easily accessible to the cross-memory access method routines. The SSCTSUSE field of the SSCVT is available to these routines and is used as the anchor point for their control blocks.

Steps for Defining an Anchor Point

Note: If you have defined an anchor point for a previous release of SAS/SHARE, do not repeat these steps now. △

To define an anchor point, perform these tasks:

1 Define an inactive z/OS subsystem by adding the entry SAS0 to any of the following:

- □ the SCHEDULER SYSGEN macro instruction
- □ the IEFJSSNT member of 'SYS1.LINKLIB'

- □ an IEFSSN*xx* member of ’SYS1.PARMLIB’.

CAUTION:

Do not use the name SAS0 if it conflicts with standards or conventions at your site. Regardless of the method that is used, you must include the subsystem name, but you should not specify an initialization routine name. △

For details about each option, see the z/OS system initialization and tuning documentation.

Although you define a subsystem to z/OS, the subsystem will never be considered active and will provide no system services because the SSCTSSVT field of the SSCVT will never be non-zero.

2 Assign the anchor point to the SUBSYSID system option.

For details, see “SAS/SHARE SUBSYSID= Option” on page 47.

P A R T 3

OpenVMS Alpha Operating Environment

Chapter 4. . . . . **OpenVMS Alpha: TCP/IP Access Method** *57*

CHAPTER 4 OpenVMS Alpha: TCP/IP Access Method

Prerequisites for Using TCP/IP under OpenVMS Alpha **58**
  Task List **58**
  Software Requirements **58**
  SAS/CONNECT and SAS/SHARE Network Security **58**
  SAS/CONNECT Options Only **59**
  SAS/SHARE Options Only **59**
SAS/CONNECT Client Tasks **60**
  Task List **60**
  Specifying TCP/IP as the Communications Access Method **60**
  Encrypting Data in Client/Server Transfers **60**
  Choosing a Method to Use to Sign On **61**
  Signing On to the Same Multi-Processor Machine **61**
    Specifying the Server Session **61**
    Starting SAS Using the SASCMD Option **61**
    Signing On to the Server Session **62**
  Signing On Using a Spawner **62**
    Ensuring That the Spawner Is Running on the Server **62**
    Specifying the Server and the Spawner Service **62**
    Specifying a Sign-On Script or a User ID and Password **63**
    Specifying a Sign-On Script **64**
    Specifying a User ID and Password **64**
    Signing On Using the Spawner **64**
  Signing On Using a Telnet Daemon **65**
    Specifying the Server **65**
    Specifying a Sign-On Script File **65**
    Signing On to the Server Session **65**
SAS/CONNECT Server Tasks **66**
  Task List **66**
  Configuring the OpenVMS Alpha Spawner Service **66**
  Starting the OpenVMS Alpha Spawner **67**
  SAS/CONNECT Server Example **67**
SAS/SHARE Client Tasks **67**
  Task List **67**
  Configuring the Server Service **67**
  Specifying TCP/IP as the Communications Access Method **67**
  Accessing a Secured Server **68**
  Encrypting Data in Client/Server Transfers **68**
  Specifying the Server **68**
  SAS/SHARE Client Example **69**
SAS/SHARE Server Tasks **70**
  Task List **70**

Configuring the Server Service **70**
Setting the TCPSEC Option to Require Client Authentication **70**
Encrypting Data in Server/Client Transfers **70**
Specifying TCP/IP as the Communications Access Method **70**
Specifying the Server **71**
SAS/SHARE Server Example **71**

Prerequisites for Using TCP/IP under OpenVMS Alpha

Task List

- □ Verify that software requirements are met.
- □ If using network security, set the appropriate SAS options.
- □ Set the appropriate SAS/CONNECT and SAS/SHARE options.

Software Requirements

Ensure that

- □ Base SAS and either SAS/CONNECT or SAS/SHARE are installed on both the client and the server.
- □ One of the following packages must be installed on each machine that runs the OpenVMS Alpha operating environment that is used as either the client or the server.
  - □ DEC TCP/IP Services for OpenVMS, Version 3.0 or later
  - □ TGV MultiNet Software with UCX compatibility
  - □ Wollongong PathWay with UCX compatibility, Version 1.1 through Version 3.5
  - □ Process Software TCPware for OpenVMS with UCX compatibility
  - □ any package that provides an interface that is compatible with DEC TCP/IP Services for OpenVMS, Version 3.0 or later

SAS/CONNECT and SAS/SHARE Network Security

Encryption is the process of transforming plaintext into a less readable form (called ciphertext) by using a mathematical process. The ciphertext is translated back to plaintext for anyone who can supply the appropriate key, which is necessary for decrypting (or unlocking) the ciphertext.

SAS/CONNECT and SAS/SHARE support the SASproprietary network security service in the OpenVMS Alpha operating environment. SASproprietary is a fixed encoding algorithm that is included with Base SAS software and is available in all SAS supported operating environments. It requires no additional SAS product licenses.

For complete details about setting up and using network security, see the *SAS/CONNECT User's Guide*. After network security is set up in your environment, you set a SAS encryption option that is appropriate to the network security service and to the requirements of the client or the server session.

SAS/CONNECT Options Only

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 32768) that you can specify in the SIGNON statement or as a SAS option. For details, see the TBUFSIZE= system option in the *SAS/CONNECT User's Guide*.

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 value for TCPMSGLEN value (default 32768) must be set at both the client and server. If the values that are set for TCPMSGLEN at the client and at the server are different, the smaller value of the two is used during the SAS/CONNECT session.

Example:

```
TCPMSGLEN:==65536
```

TCPPORTFIRST=*port-number* (set at the server)
TCPPORTLAST=*port-number* (set at the server)

restricts the range of TCP/IP ports that clients can use to remotely access servers.

Within the range of 0 through 32767, assign a beginning value to TCPPORTFIRST and an ending value to TCPPORTLAST. To restrict the range of ports to only one port, set the values for TCPPORTFIRST and TCPPORTLAST to the same number. Consult with your network administrator for advice about setting these values.

At the server, you can set TCPPORTFIRST and TCPPORTLAST in a SAS start-up command or in the configuration file.

In the following example, the server is restricted to the TCP/IP ports 4020 through 4050:

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

TCPTN3270

supports connections to z/OS servers that use the full-screen 3270 Telnet protocol. The script file TCPTS032.SCR is provided. See Table 4.1 on page 64 for a complete list of sign-on scripts.

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

To set the TCPTN3270 variable, enter the following command at the client:

```
TCPTN3270:==1
```

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

SAS/SHARE Options Only

TCPSEC:==_SECURE_ (set at the server)

specifies whether the TCP/IP access method verifies user access authority before allowing clients to access the server. The TCPSEC option must be set at the server before the server session is started.

SECURE
: requires that the TCP/IP access method verify the authority of clients that attempt to access the server. Each client must supply a user ID and a password that are valid at the server.

Example:

```
TCPSEC:==_SECURE_
```

If you do not set this variable, the TCP/IP access method does NOT verify the authority of clients that attempt to access the server.

SAS/CONNECT Client Tasks

Task List

1 Specify TCP/IP as the communications access method.
2 Specify encryption of client/server data transfers (optional).
3 Sign on to the server.

Note: SAS/CONNECT enables TCP/IP connections from clients outside a firewall to spawners that run on servers inside a firewall. For details, see Chapter 14, “Configuring SAS/CONNECT for Use with a Firewall,” on page 149 △

Specifying TCP/IP as the Communications Access Method

TCP/IP is the default communications access method for all operating environments, except z/OS. Therefore, you do not have to explicitly specify the default.

If you choose to specify TCP/IP to connect to a server, you can use the COMAMID= option in an OPTIONS statement.

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

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 as example of an *access-method-ID*. Alternatively, you can set this option in a SAS start-up command or in a SAS configuration file.

Example:

```
options comamid=tcp;
```

Encrypting Data in Client/Server Transfers

If network security is available and is configured at the client, you can specify SAS options to encrypt all data that is transferred between a client and a server. In the following example, the NETENCRYPTALGORITHM= option specifies the SASPROPRIETARY algorithm.

```
options netencryptalgorithm=sasproprietary;
```

For complete details about network security options, see the *SAS/CONNECT User’s Guide*.

Choosing a Method to Use to Sign On

Based on your operating environment, you can use one of the following methods to sign on:

- the same multi-processor machine

  Note: This method is most useful if your client machine is equipped with SMP (Symmetric Multi-Processor) hardware. △

- a spawner
- a Telnet daemon.

Signing On to the Same Multi-Processor Machine

If your client machine is equipped with SMP (Symmetric Multi-Processors), and if you want to run one or more server sessions on your machine, perform these tasks:

1. Specify the server session.
2. Specify the SASCMD command to start SAS.
3. Sign on to the server session.

Specifying the Server Session

You can specify the server session in an OPTIONS statement:

```
OPTIONS PROCESS=session-ID;
```

or in the SIGNON statement or command:

```
SIGNON session-ID;
```

session-ID must be a valid SAS name that is 1 to 8 characters in length, and is the name that you assign to the server session on the multi-processor machine.

Note: PROCESS=, REMOTE=, CREMOTE=, and CONNECTREMOTE= can be used interchangeably. For details, see CONNECTREMOTE= system option in the *SAS/CONNECT User's Guide*. △

For details about SIGNON=, see the SIGNON statement in the *SAS/CONNECT User's Guide*.

Starting SAS Using the SASCMD Option

Use the SASCMD option to specify the SAS command and any additional options that you want to use to start SAS in a server session on the same multi-processor machine.

The SASCMD option can be specified either in an OPTIONS statement:

```
OPTIONS SASCMD="SAS-command" | "!SASCMD";
```

or directly in the SIGNON statement or command:

```
SIGNON name SASCMD="SAS-command" | "!SASCMD";
```

The -DMR option is automatically appended to the command. If *!SASCMD* is specified, SAS/CONNECT starts SAS on the server by using the same command that was used to start SAS for the current (parent) session.

Note: In order to execute additional commands prior to starting SAS, you might write a script that contains the SAS start-up commands that are appropriate for the operating environment. Specify this script as the value in the SASCMD= option. △

For details, see the SASCMD= system option and the SIGNON statement in the *SAS/CONNECT User's Guide*.

Signing On to the Server Session

Example 1:

In the following example, TCP is the access method, SAS1 is the name of the server session, and SAS_START is the command that starts SAS on the same multi-processor machine.

```
options comamid=tcp;
signon sas1 sascmd='sas_start';
```

Example 2:

In the following example, the values for the COMAMID=, SASCMD=, and PROCESS= options are set in the OPTIONS statements. The SASCMD= option identifies the command that starts SAS. The PROCESS= option identifies the server session on the same multi-processor machine. Because the SASCMD= and the PROCESS= options are defined, only a simple SIGNON statement is needed.

```
options comamid=tcp sascmd='sas_start';
options process=sas1;
signon;
```

Signing On Using a Spawner

1. Ensure that the spawner is running on the server.
2. Specify the server and an optional service.
3. Specify the sign-on script (if you are signing on using a script), or specify a user ID and password (if you are signing on without a script).
4. Sign on to the server using a spawner.

Ensuring That the Spawner Is Running on the Server

Before you can access the spawner, the spawner program must be running on the server. For information about the spawner that you are connecting to, see Chapter 7, "SAS/CONNECT Spawners," on page 113.

Note: The system administrator for the machine that the spawner runs on must start the spawner. The spawner program on the server cannot be started by the client. △

Specifying the Server and the Spawner Service

The name of the server can be specified either in an OPTIONS statement:

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

or directly in the SIGNON statement or command:

```
SIGNON node-name[.service-name | .port-number];
```

node-name is based on the server that you are connecting to. *node-name* must be a valid SAS name that is 1 to 8 characters in length and is either:

- □ the short machine name of the server that you are connecting to. This name must be defined in the **/etc/hosts** file in the client operating environment or in your Domain Name Server (DNS).

- a macro variable that contains either the IP address or the name of the server that you are connecting to.

The process for evaluating *node-name* follows:

1. If *node-name* is a macro variable, the value of the macro variable is passed to the operating environment's GETHOSTBYNAME function.
2. If *node-name* is not a macro variable or the value of the macro variable does not produce a valid value, *node-name* is passed to the GETHOSTBYNAME function.
3. If GETHOSTBYNAME fails to resolve *node-name*, an error message is returned and the sign on fails.

   Note: The order in which the GETHOSTBYNAME function calls the DNS or searches the HOSTS file to resolve *node-name* varies based on the operating environment implementation. △

You specify *service-name* when connecting to a server that runs a spawner program that is listening on a port other than the Telnet port. If the spawner was started by using the -SERVICE spawner option, you must specify an explicit *service-name*. The value of *service-name* and the value of the -SERVICE spawner option must be identical. Alternatively, you can specify the explicit port number that is associated with *service-name*.

Example 1:

In the following example, REMHOST is the name of the node that the spawner runs on. PORT1 is the name of the service that is defined at the client. The client service PORT1 must be assigned to the same port that the spawner is listening on.

```
signon remhost.port1;
```

Example 2:

In the following example, the macro variable REMHOST is assigned to the fully-qualified name of the machine that the server runs on. This server has a spawner running that is listening on port 5050. The server session that is specified in the SIGNON statement uses the node name REMHOST and the service name 5050, which is the explicit port value.

```
%let remhost=pc.rem.us.com;
signon remhost.5050;
```

You can also assign a specific port number by including the port number in the definition of the macro variable, for example,

```
%let remhost=pc.rem.us.com 5050;
signon remhost;
```

Specifying a Sign-On Script or a User ID and Password

You can use a sign-on script to sign on to the spawner, or you can sign on to a spawner without a script . If you do not use a sign-on script and if the spawner is running secured, you must supply a user ID and password to sign on to the spawner.

Note: If you connect to a spawner, you can sign on by using a script unless the spawner is started by using the -NOSCRIPT option. If the -NOSCRIPT option is set, you cannot use a script. If there is no script, you do not assign the fileref RLINK in a FILENAME statement. For information about the spawner that you are connecting to, see Chapter 7, "SAS/CONNECT Spawners," on page 113. △

Specifying a Sign-On Script

If you are signing on by using a script, you must specify the script that you want to use. The script file is executed by the SIGNON statement or command. By default, the script prompts for user ID and password.

To use one of the sample script files that are provided with SAS/CONNECT for signing on and signing off, assign the fileref RLINK to the appropriate script file. The script is based on the server that you are connecting to. The sample scripts are installed at

```
SAS$ROOT:[TOOLS]
```

To specify a script, use the FILENAME statement. For example,

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

script-name specifies the appropriate script file for the server.

Table 4.1 on page 64 lists the scripts that are provided in SAS software:

Table 4.1 SAS/CONNECT Sign-on Scripts for TCP/IP under OpenVMS Alpha

Server	Script Name
TSO under OS/390	**tcptso.scr**
TSO under z/OS, SAS 9 or later	**tcptso9.scr**
z/OS (without TSO)	**tcpmvs.scr**
z/OS (using full-screen 3270 Telnet protocol)	**tcptso32.scr**
OpenVMS Alpha	**tcpvms.scr**
UNIX	**tcpunix.scr**
Windows	**tcpwin.scr**

Specifying a User ID and Password

If you are signing on to the spawner without using a script and the spawner is running secured, you must submit the SIGNON statement and provide a user ID and a password in order to log on to the server. For example,

```
SIGNON USER=user-ID | _PROMPT_ [ PASSWORD=password | _PROMPT_ ];
```

Signing On Using the Spawner

In the following example, a client connects to a UNIX server by using a spawner without a script. In the SIGNON statement, RMTHOST.SPAWNER specifies the node RMTHOST and the service SPAWNER. This server specification presumes that a spawner is running on the node RMTHOST, and that the spawner was started with the service SPAWNER. Specifying USER=_PROMPT_ causes a dialog box to appear so that a user ID and a password can be provided.

Example:

```
options comamid=tcp;
signon rmthost.spawner user=_prompt_;
```

Signing On Using a Telnet Daemon

1 Specify the server.
2 Specify a sign-on script.
3 Sign on to the server session.

Specifying the Server

The name of the server can be specified either in an OPTIONS statement:

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

or directly in the SIGNON statement or command:

```
SIGNON node-name;
```

Specifying a Sign-On Script File

If you are signing on by using a script, you must specify the script that you want to use. The script file is executed by the SIGNON statement or command. By default, the script prompts for user ID and password. For details, see "Specifying a Sign-On Script" on page 64.

Signing On to the Server Session

In the following example, you specify the statements at an OpenVMS client to use the TCP/IP access method to connect to a server. The FILENAME statement identifies the script file that you use to sign on to the server. The script file contains a prompt for a user ID and a password that are valid on the server. The COMAMID= option specifies the TCP/IP communications access method for connecting to the server RMTNODE, which is specified in the REMOTE= option.

```
filename rlink 'SAS$ROOT:[TOOLS]tcptso.scr';
options comamid=tcp remote=rmtnode;
signon;
```

SAS/CONNECT Server Tasks

Task List

1 Configure the OpenVMS Alpha spawner service.

2 Start the OpenVMS Alpha spawner at the server.

Note: If the OpenVMS Alpha spawner is not being used, there are no server tasks. △

Configuring the OpenVMS Alpha Spawner Service

To enable clients to connect to an OpenVMS Alpha server by using the OpenVMS Alpha spawner, configure the spawner service in the **`SERVICES`** file at the server. For details, see Chapter 13, “TCP/IP SERVICES File,” on page 145.

Starting the OpenVMS Alpha Spawner

You must start the OpenVMS Alpha spawner on an OpenVMS Alpha server to enable clients to connect to it. The spawner program resides on a server and listens for SAS/CONNECT client requests for connection to the server. After the spawner program receives a request, it starts a server session. For details, see Chapter 8, "OpenVMS Alpha Spawner," on page 119.

If network security has been configured at the server, set the appropriate encryption options when starting the spawner.

SAS/CONNECT Server Example

The following command starts the spawner VMSSPAWN on an OpenVMS Alpha machine. The absence of the -SASCMD option in the spawner start-up command implies that the client will use a script to specify the SAS command that starts SAS on the OpenVMS Alpha machine.

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

SAS/SHARE Client Tasks

Task List

1 Configure the server service.

2 Specify TCP/IP as the communications access method.

3 Access a secured server.

4 Specify encryption of client/server data transfers (optional).

5 Specify the server.

Configuring the Server Service

Each server must be defined as a service in the SERVICES file on each machine that a client will access the server from. The SERVICES file usually is located in the directory in which the TCP/IP software is installed. For details about editing the SERVICES file, see "Configuring the SERVICES File" on page 145.

Specifying TCP/IP as the Communications Access Method

You must specify TCP/IP as the communications access method at the client before you access a server. An example follows:

```
/comamid=tcp
```

The COMAMID= option specifies the communications access method. TCP specifies the TCP/IP access method.

You can specify the COMAMID= option in an OPTIONS statement, in a SAS configuration file, or in a SAS start-up command.

Accessing a Secured Server

Requiring clients to supply a valid user ID and password when attempting to access a server enforces server security. The values for a user ID and a password are provided in the USER= and PASSWORD= options in the LIBNAME statement and the PROC OPERATE statement. For details about supplying a user ID and a password, see the LIBNAME statement and the OPERATE procedure in the *SAS/SHARE User's Guide*.
Example:

```
libname sasdata 'edc/prog2/sasdata' server=rmtnode.share user=_prompt_ ;
```

The value _PROMPT_ requires the client to provide a user ID and password when a client attempts to access the server.

Encrypting Data in Client/Server Transfers

If network security is configured at the client, you can specify SAS options to encrypt data that a client transfers to a server. For example:

```
options netencrypt netencryptalgorithm=rc2;
```

The NETENCRYPT option specifies that all data transactions between a client and a server will be encrypted. The RC2 encryption algorithm is specified. For general information about network security, see "SAS/CONNECT and SAS/SHARE Network Security" on page 76.

Specifying the Server

If the client and server sessions are running on different network nodes, you must include the TCP/IP node in the server ID in the LIBNAME or the PROC OPERATE statement by using a two-level name as follows:

```
SERVER=node.server
```

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

node must be a valid SAS node name. If the server and the client sessions are running on the same node, you can 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 and use the DCL symbol for *node* in the two-level server name.
The access method evaluates the node name, using this order of precedence:

- □ SAS macro variable
- □ DCL symbol
- □ valid node name.

server can be either a *server-ID* or a *port*.

The *server-ID* must be identical to the service name that is specified in the SERVICES file. For details, see "Configuring the SERVICES File" on page 145.

Example 1:

A SAS macro variable is used to contain the name of a server node.

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

Example 2:

The DCL symbol SRVNODE is assigned to the fully qualified node name and is then used in the two-level node name.

```
$srvnode:==mktserve.acme.com
libname mylib server=srvnode.server;
```

Example 3:

A *port* is the unique number that is associated with the service that is used for passing data to and receiving data from the server.

Precede the port number with two consecutive underscores.

Note: Do *not* space after the first underscore or the second underscore. △

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

Example 4:

You can also include the server's port number or service by using a macro variable that specifies the port or service.

```
%let server2=host.name.com 5000;
libname servf1 'pathname' server=server2;
```

or

```
%let server2=12.34.56.78 5000;
libname servf1 'pathname' server=server2;
```

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. △

For details about creating valid SAS names, see *SAS Language Reference: Concepts*. For details about the LIBNAME and PROC OPERATE statements, see *SAS/SHARE User's Guide*.

SAS/SHARE Client Example

The following example shows the statements that are specified at an OpenVMS Alpha client to access a server by using the TCP/IP access method. The LIBNAME statement specifies the data library that is accessed through the server. The value _PROMPT_ in the USER= option specifies that the client must provide a valid user ID and password to access the server. The SERVER= option specifies the two-level server name RMTNODE.SHARE1.

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

SAS/SHARE Server Tasks

Task List

1 Configure the SAS/SHARE server service.
2 Specify SAS options and network security (optional).
   - If the server is to run secured, set the TCPSEC= option to require client authentication.
   - Specify security service options to encrypt client/server data transfers.
3 Specify TCP/IP as the communications access method.
4 Specify the server.

Configuring the Server Service

Each server must be defined as a service in the SERVICES file on each node that each client will access. The SERVICES file is located in the directory where the TCP/IP software is installed. Find out the correct location of the TCP/IP software in your operating environment. For details about editing the SERVICES file, see "Configuring the SERVICES File" on page 145.

Example:

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

Setting the TCPSEC Option to Require Client Authentication

To authenticate clients that attempt to access the server, you must specify the value _SECURE_ in the TCPSEC= option to require that clients provide a user ID and a password that are valid on the server. For details about the TCPSEC= option, see "SAS/SHARE Options Only" on page 59.

Example:
TCPSEC:==_SECURE_

Encrypting Data in Server/Client Transfers

If network security is configured at the server, you can specify SAS options to encrypt data that a server transfers to a client. For example:

```
options netencrypt netencryptalgorithm=sasproprietary;
```

The NETENCRYPT option specifies that all data transfers between a server and a client will be encrypted. SASPROPRIETARY is the encryption algorithm. For general information about network security, see "SAS/CONNECT and SAS/SHARE Network Security" on page 58.

Specifying TCP/IP as the Communications Access Method

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

Example:

```
/comamid=tcp
```

The COMAMID= option specifies the communications access method. TCP specifies the TCP/IP access method.

You can specify the COMAMID option in an OPTIONS statement, in a configuration file, or in a SAS start-up command.

Specifying the Server

You must specify the name of the server in the SERVER= option in the PROC SERVER statement. The syntax follows:

```
SERVER=server-ID
```

server-ID can be either a *server-ID* or a *port* number. The value for *server-ID* corresponds to the service that was configured in the SERVICES file. For details, see "Configuring the SERVICES File" on page 145. *port* is the unique number that is associated with the service that is used for transferring data between a client and a server.

Precede the port number with two consecutive underscores.

Note: Do *not* space after the first underscore or the second underscore. △

Note: Specifying a server by using a port number is *not* supported for ODBC clients. △

Examples:

```
proc server server=apex;
proc server server=__5000;
```

For details about creating valid SAS names, see *SAS Language Reference: Concepts*. For details about PROC SERVER, see the SERVER procedure in the *SAS/SHARE User's Guide*

SAS/SHARE Server Example

The following example shows the statements that you specify at an OpenVMS Alpha machine to start a server. The value _SECURE_ that is specified in the TCPSEC option requires clients to provide a user ID and a password that are valid on the server. The NETENCRYPT option specifies that encryption is required, and the NETENCRYPTALGORITHM option specifies that the encryption algorithm is SASPROPRIETARY. The COMAMID= option specifies the TCP/IP access method. The PROC SERVER statement specifies the server SHARE1.

```
%let tcpsec=_secure_;
options netencrypt;
options netencryptalgorithm=sasproprietary;
options comamid=tcp;
proc server id=share1;
run;
```

PART 4

UNIX Operating Environments

Chapter **5**. . . . . **UNIX: TCP/IP Access Method** *75*

CHAPTER

5

UNIX: TCP/IP Access Method

Prerequisites for Using TCP/IP under UNIX **76**
Task List **76**
Software Requirements **76**
SAS/CONNECT and SAS/SHARE Network Security **76**
SAS/CONNECT Options Only **77**
SAS/SHARE Options Only **78**
SAS/CONNECT Client Tasks **78**
Task List **78**
Specifying TCP/IP as the Communications Access Method **78**
Encrypting Data in Client/Server Transfers **79**
Choosing a Method to Use to Sign On **79**
Signing On to the Same Multi-Processor Machine **79**
Specifying the Server Session **79**
Starting SAS Using the SASCMD Option **80**
Signing On to the Server Session **80**
Signing On Using a Spawner **80**
Ensuring That the Spawner Is Running on the Server **81**
Specifying the Server and the Spawner Service **81**
Specifying a Sign-On Script or a User ID and Password **82**
Specifying a Sign-On Script **82**
Specifying a User ID and Password **82**
Signing On Using the Spawner **83**
Signing On Using a Telnet Daemon **83**
Specifying the Server **83**
Specifying a Sign-On Script File **83**
Signing On to the Server Session **83**
SAS/CONNECT Server Tasks **84**
Task List **84**
Configuring the UNIX Spawner Service **84**
Starting the UNIX Spawner **84**
SAS/CONNECT Server Example **84**
SAS/SHARE Client Tasks **84**
Task List **84**
Configuring the Server Service **85**
Specifying TCP/IP as the Communications Access Method **85**
Accessing a Secured Server **85**
Encrypting Data in Client/Server Transfers **85**
Specifying the Server **86**
SAS/SHARE Client Example **87**
SAS/SHARE Server Tasks **87**
Task List **87**

Configuring the Server Service **87**
Setting the TCPSEC Option to Require Client Authentication **87**
Configuring User Access Authority **88**
Configuring the Authentication Program **88**
Configuring the Permission Program **88**
Encrypting Data in Server/Client Transfers **88**
Specifying TCP/IP as the Communications Access Method **89**
Specifying the Server **89**
SAS/SHARE Server Example **89**

Prerequisites for Using TCP/IP under UNIX

Task List

- □ Verify that software requirements are met.
- □ If using network security, set the appropriate SAS options.
- □ Set the appropriate options for SAS/CONNECT and SAS/SHARE.

Software Requirements

Ensure that

- □ Base SAS and either SAS/CONNECT or SAS/SHARE are installed on both the client and the server.
- □ Any TCP/IP package that comes with the operating environment has been installed.

SAS/CONNECT and SAS/SHARE Network Security

Encryption is the process of transforming plaintext into a less readable form (called ciphertext) by using a mathematical process. The ciphertext is translated back to plaintext for anyone who can supply the appropriate key, which is necessary for decrypting (or unlocking) the ciphertext.

SAS/CONNECT and SAS/SHARE support the following network security services in the UNIX operating environment.

SASproprietary
: a fixed encoding algorithm that is included with Base SAS software and is available in all SAS supported operating environments. It requires no additional SAS product licenses.

SAS/SECURE
: an add-on product that uses the encryption algorithms RC2, RC4, DES, and tripleDES.

SSL
: SSL is an abbreviation for Secure Sockets Layer, which is a protocol that provides network security and privacy. Developed by Netscape Communications, SSL uses encryption algorithms that include RC2, RC4, DES, tripleDES, and MD5.

For complete details about setting up and using network security, see the *SAS/CONNECT User's Guide*. After network security is set up in your environment, you set a SAS encryption option that is appropriate to the network security service and to the requirements of the client or the server session.

SAS/CONNECT Options Only

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 32768) that you can specify in the SIGNON statement or as a SAS option. For details, see the TBUFSIZE= system option in the *SAS/CONNECT User's Guide*.

  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 value for TCPMSGLEN (default=32768) must be set at both the client and server. If the values that are set for TCPMSGLEN at the client and at the server are different, the smaller value of the two is used during the SAS/CONNECT session.

  Example:

```
-set tcpmsglen 65536
```

TCPPORTFIRST=*port-number* (set at the server)
TCPPORTLAST=*port-number* (set at the server)
: restricts the range of TCP/IP ports through which clients can connect to a server.

  Within the range of 0 through 32767, assign a beginning value to TCPPORTFIRST and an ending value to TCPPORTLAST. To restrict the range of ports to only one port, set the values for TCPPORTFIRST and TCPPORTLAST to the same number. Consult with your network administrator for advice about setting these values.

  At the server, you can set TCPPORTFIRST and TCPPORTLAST in a SAS startup command or in the SAS configuration file.

  In the following example, the server is restricted to the TCP/IP ports 4020 through 4050:

```
-tcpportfirst 4020;
-tcpportlast 4050;
```

TCPTN3270 (set at the client)
: supports connections to z/OS servers that use the full-screen 3270 Telnet protocol. The script file TCPTSO32.SCR is provided. See Table 5.1 on page 82 for a complete list of sign-on scripts.

You can set the TCPTN3270 option only in the SAS configuration file. If you do not set this option, the TCP/IP access method uses the Telnet line-mode protocol by default.

Example:

```
-set TCPTN3270 1
```

SAS/SHARE Options Only

TCPSEC=_SECURE_ | _NONE_ (set at the server)
: specifies whether the TCP/IP access method verifies user access authority before allowing clients to access the server. The TCPSEC option must be set at the server before the server session is started. The default is _NONE_.

SECURE
: requires that the TCP/IP access method verify the authority of clients that attempt to access the server. Each client must supply a user ID and a password that are valid at the server.

NONE
: specifies that the TCP/IP access method does not verify the authority of SAS/SHARE clients that attempt to access the server.

Examples:

```
%let TCPSEC=_secure_;
%let TCPSEC=_none_;
```

SAS/CONNECT Client Tasks

Task List

1 Specify TCP/IP as the communications access method.
2 Specify encryption of client/server data transfers (optional).
3 Sign on to the server.

Note: SAS/CONNECT enables TCP/IP connections from clients outside a firewall to spawners that run on servers inside a firewall. For details, see Chapter 14, "Configuring SAS/CONNECT for Use with a Firewall," on page 149. △

Specifying TCP/IP as the Communications Access Method

TCP/IP is the default communications access method for all operating environments, except z/OS. Therefore, you do not have to explicitly specify the default.

If you choose to specify TCP/IP to connect to a server, you can use the COMAMID= option in an OPTIONS statement.

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

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*. Alternatively, you can set this option in a SAS start-up command or in a SAS configuration file.
Example:

```
options comamid=tcp;
```

Encrypting Data in Client/Server Transfers

If network security is available and is configured at the client, you can specify SAS options to encrypt all data that is transferred between a client and a server. In the following example, the NETENCRYPTALGORITHM= option specifies the SSL algorithm.

```
options netencryptalgorithm=ssl;
```

For complete details about network security options, see the *SAS/CONNECT User's Guide*.

Choosing a Method to Use to Sign On

Based on your operating environment, you can use one of the following methods to sign on.

- the same multi-processor machine

  Note: This method is most useful if your client machine is equipped with SMP (Symmetric Multi-Processor) hardware. △

- a spawner
- a Telnet daemon.

Signing On to the Same Multi-Processor Machine

If your client machine is equipped with SMP (Symmetric Multi-Processors), and if you want to run one or more server sessions on your machine, perform these tasks:

1. Specify the server session.
2. Specify the SASCMD command to start SAS.
3. Sign on to the server session.

Specifying the Server Session

You can specify the server session in an OPTIONS statement:

```
OPTIONS PROCESS=session-ID;
```

or in the SIGNON statement or command:

```
SIGNON session-ID;
```

session-ID must be a valid SAS name that is 1 to 8 characters in length, and is the name that you assign to the server session on the same multi-processor machine.

Note: PROCESS=, REMOTE=, CREMOTE=, and CONNECTREMOTE= can be used interchangeably. For details, see the CONNECTREMOTE= system option in the *SAS/CONNECT User's Guide*. △

For details about SIGNON=, see the SIGNON statement in the *SAS/CONNECT User's Guide*.

Starting SAS Using the SASCMD Option

Use the SASCMD option to specify the SAS command and any additional options that you want to use to start SAS in a server session on the same multi-processor machine.
The SASCMD option can be specified either in an OPTIONS statement:

```
OPTIONS SASCMD="SAS-command" | "!SASCMD";
```

or directly in the SIGNON statement or command:

```
SIGNON name SASCMD="SAS-command" | "!SASCMD";
```

The -DMR option is automatically appended to the command. If *!SASCMD* is specified, SAS/CONNECT starts SAS on the server by using the same command that was used to start SAS for the current (parent) session.

Note: In order to execute additional commands prior to starting SAS, you might write a script that contains the SAS start-up commands that are appropriate for the operating environment. Specify this script as the value in the SASCMD= option. △

For details, see the SASCMD= system option and the SIGNON statement in the *SAS/CONNECT User's Guide*.

Signing On to the Server Session

Example 1:
In the following example, TCP is the access method, SAS1 is the name of the server session, and SAS_START is the command that starts SAS on the same multi-processor machine.

```
options comamid=tcp;
signon sas1 sascmd='sas_start';
```

Example 2:
In the following example, the values for the COMAMID=, SASCMD=, and PROCESS= options are set in the OPTIONS statements. The SASCMD= option identifies the command that starts SAS. The PROCESS= option identifies the server session on the same multi-processor machine. Because the SASCMD= and the PROCESS= options are defined, only a simple SIGNON statement is needed.

```
options comamid=tcp sascmd="sas_start";
options process=sas1;
signon;
```

Signing On Using a Spawner

1 Ensure that the spawner is running on the server.
2 Specify the server and an optional service.
3 Specify the sign-on script (if you are signing on using a script),
or specify a user ID and password (if you are signing on without a script).
4 Sign on to the server using a spawner.

Ensuring That the Spawner Is Running on the Server

Before you can access the spawner, the spawner program must be running on the server. For information about the spawner that you are connecting to, see Chapter 7, "SAS/CONNECT Spawners," on page 113.

Note: The system administrator for the machine that the spawner runs on must start the spawner. The spawner program on the server cannot be started by the client. △

Specifying the Server and the Spawner Service

The name of the server can be specified either in an OPTIONS statement:

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

or directly in the SIGNON statement or command:

```
SIGNON node-name[.service-name | .port-number];
```

node-name is based on the server that you are connecting to. *node-name* must be a valid SAS name that is 1 to 8 characters in length and is either:

- □ the short machine name of the server that you are connecting to. This name must be defined in the **/etc/hosts** file in the client operating environment or in your Domain Name Server (DNS).
- □ a macro variable that contains either the IP address or the name of the server that you are connecting to.

The process for evaluating *node-name* follows:

1. If *node-name* is a macro variable, the value of the macro variable is passed to the operating environment's GETHOSTBYNAME function.
2. If *node-name* is not a macro variable or the value of the macro variable does not produce a valid value, *node-name* is passed to the GETHOSTBYNAME function.
3. If GETHOSTBYNAME fails to resolve *node name*, an error message is returned and the signon fails.

   Note: The order in which the GETHOSTBYNAME function calls the DNS or searches the HOSTS file to resolve *node-name* varies based on the operating environment implementation. △

You specify *service-name* when connecting to a server that runs a spawner program that is listening on a port other than the Telnet port. If the spawner was started by using the **-service** spawner option, you must specify an explicit *service-name*. The value of *service-name* and the value of the **-service** spawner option must be identical. Alternatively, you can specify the explicit port number that is associated with *service-name*.
Example 1:
In the following example, REMHOST is the name of the node on which the spawner runs, and PORT1 is the name of the service that is defined at the client. The client service PORT1 must be assigned to the same port that the spawner is listening on.

```
signon remhost.port1;
```

Example 2:
In the following example, the macro variable REMHOST is assigned to the fully-qualified name of the machine on which the server runs. This server has a spawner running that is listening on port 5050. The server session that is specified in the SIGNON statement uses the node name REMHOST and the service name 5050, which is the explicit port value.

```
%let remhost=pc.rem.us.com;
signon remhost.5050;
```

You can also assign a specific port number by including the port number in the definition of the macro variable, for example,

```
%let remhost=pc.rem.us.com 5050;
signon remhost;
```

Specifying a Sign-On Script or a User ID and Password

You can use a sign-on script to sign on to the spawner, or you can sign on to a spawner without a script. If you do not use a sign-on script and if the spawner is running secured, you must supply a user ID and password to sign on to the spawner.

Note: If you connect to a spawner, you can sign on by using a script unless the spawner is started using the -NOSCRIPT option. If the -NOSCRIPT option is set, you cannot use a script. If there is no script, you do not assign the fileref RLINK in a FILENAME statement. For information about the spawner that you are connecting to, see Chapter 7, "SAS/CONNECT Spawners," on page 113. △

Specifying a Sign-On Script

If you are signing on by using a script, you must specify the script that you want to use. The script file is executed by the SIGNON statement or command. By default, the script prompts for user ID and password.

To use one of the sample script files that are provided with SAS/CONNECT for signing on and signing off, assign the fileref RLINK to the appropriate script file. The script is based on the server that you are connecting to. The sample scripts are installed at

```
!sasroot/misc/connect
```

To specify a script, use the FILENAME statement. For example,

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

Script-name specifies the appropriate script file for the server.

Table 5.1 on page 82 lists the scripts that are provided in SAS software:

Table 5.1 SAS/CONNECT Sign-on Scripts for TCP/IP under UNIX

Server	Script Name
TSO under OS/390	**tcptso.scr**
TSO under z/OS, SAS 9 or later	**tcptso9.scr**
z/OS (without TSO)	**tcpmvs.scr**
z/OS (using full-screen 3270 Telnet protocol)	**tcptso32.scr**
OpenVMS Alpha	**tcpvms.scr**
UNIX	**tcpunix.scr**
Windows	**tcpwin.scr**

Specifying a User ID and Password

If you are signing on to the spawner without using a script and the spawner is running secured, you must submit the SIGNON statement and provide a user ID and a password in order to log on to the server. For example,

```
SIGNON USER=user-ID | _PROMPT_ [ PASSWORD=password | _PROMPT_ ];
```

Signing On Using the Spawner

In the following example, a client connects to a UNIX server by using a spawner without a script. In the SIGNON statement, RMTHOST.SPAWNER specifies the node RMTHOST and the service SPAWNER. This server specification presumes that a spawner is running on the node RMTHOST, and that the spawner was started using the service SPAWNER. Specifying USER=_PROMPT_ causes a dialog box to appear so that a user ID and a password can be provided.

Example:

```
options comamid=tcp;
signon rmthost.spawner user=_prompt_;
```

Signing On Using a Telnet Daemon

1 Specify the server.
2 Specify a sign-on script.
3 Sign on to the server session.

Specifying the Server

The name of the server can be specified either in an OPTIONS statement:

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

or directly in the SIGNON statement or command:

```
SIGNON node-name;
```

Specifying a Sign-On Script File

If you are signing on by using a script, you must specify the script that you want to use. The script file is executed by the SIGNON statement or command. By default, the script prompts for user ID and password. For details, see “Specifying a Sign-On Script” on page 82.

Signing On to the Server Session

In the following example, you specify the statements at a UNIX client to use the TCP/IP access method to connect to a z/OS server. The FILENAME statement identifies the script file that you use to sign on to a server. The script file contains a prompt for a user ID and a password that are valid on the server. The COMAMID= option specifies the TCP/IP communications access method for connecting to the server RMTNODE, which is specified in the REMOTE= option.

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

SAS/CONNECT Server Tasks

Task List

1 Configure the UNIX spawner service.
2 Start the UNIX spawner at the server.

Note: If the UNIX spawner is not being used, there are no server tasks. △

Configuring the UNIX Spawner Service

To enable clients to connect to a UNIX server by using the UNIX spawner, configure the spawner service in the **`/etc/services`** file at the server. For details, see Chapter 13, "TCP/IP SERVICES File," on page 145.

Starting the UNIX Spawner

You must start the UNIX spawner on a UNIX server to enable clients to connect to it. The spawner program resides on a server and listens for SAS/CONNECT client requests for connection to the server. After the spawner program receives a request, it starts a server session. For details about starting the UNIX spawner, see Chapter 10, "UNIX Spawner," on page 127.

If network security has been configured at the server, set the appropriate encryption options when starting the spawner.

SAS/CONNECT Server Example

The following command starts the UNIX spawner. The -SERVICE option specifies the service SPAWNER that listens for incoming connections. The -SASCMD option specifies the path to the MYSTARTUP file, which starts the SAS session on the server.

```
sastcpd -service spawner -sascmd "/u/username/mystartup"
```

SAS/SHARE Client Tasks

Task List

1 Configure the server service.
2 Specify TCP/IP as the communications access method.
3 Access a secured server.
4 Specify encryption of client/server data transfers (optional).
5 Specify the server.

Configuring the Server Service

Each server must be defined as a service in the **/etc/services** file on each machine that a client will access the server from. This file is usually located in the directory that the TCP/IP software is installed in. For details about editing the **/etc/services**, see "Configuring the SERVICES File" on page 145.

Specifying TCP/IP as the Communications Access Method

TCP/IP is the default communications access method in the UNIX operating environment. You can omit specifying the access method in the COMAMID= option and the TCP/IP access method is assumed, by default.

If you choose to specify TCP/IP to connect to a server, you can use the COMAMID= option in an OPTIONS statement.

```
options comamid=tcp;
```

The COMAMID= option specifies the communications access method. TCP specifies the TCP/IP access method.

Alternatively, you can specify the COMAMID= option in a configuration file or in a SAS start-up command.

Accessing a Secured Server

Requiring clients to supply a valid user ID and password when attempting to access a server enforces server security. The values for a user ID and a password are provided in the USER= and PASSWORD= options in the LIBNAME statement and the PROC OPERATE statement. For details about supplying a user ID and a password, see the LIBNAME statement and the OPERATE procedure in the *SAS/SHARE User's Guide*.

Example:

```
libname sasdata 'edc/prog2/sasdata' server=rmtnode.share user=_prompt_ ;
```

The value _PROMPT_ requires the client to provide a user ID and password when a client attempts to access the server.

Encrypting Data in Client/Server Transfers

If network security is configured at the client, you can specify SAS options to encrypt data that a client transfers to a server. For example:

```
options netencrypt netencryptalgorithm=ssl;
options sslcalistloc="/users/johndoe/certificates/cacerts.pem";
```

The NETENCRYPT option specifies that all data transfers between a client and a server will be encrypted. SSL is the network security service that is specified in the NETENCRYPTALGORITHM= option. The SSLCALISTLOC= option specifies the name of a file that contains a list of CA certificates that are to be trusted. For general information about security services, see "SAS/CONNECT and SAS/SHARE Network Security" on page 76.

Specifying the Server

If the client and server sessions are running on different network nodes, you must include the TCP/IP node in the server ID in the LIBNAME or the PROC OPERATE statement by using a two-level server name as follows:

```
SERVER=node.server
```

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

1 a SAS macro variable
2 an environment variable
3 valid node name.

node can be either of the following:

□ valid TCP/IP node name
□ IP address

If the server and the client sessions are running on the same node, you can omit the node name.

server can be either of the following:

□ *server-ID*
□ *port*

The *server-ID* must be identical to the service name that is specified in the **/etc/services** file. For details, see "Configuring the SERVICES File" on page 145.

Example 1:

A *port* is the unique number that is associated with the service that is used for passing data to and receiving data from the server.

Precede the port number with two consecutive underscores.

Note: Do *not* space after the first underscore or the second underscore. △

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

Example 2:

If the TCP/IP node name is not a valid eight-character 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.

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

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

Example 3:

You might assign the node name and the server ID to a macro variable.

```
%let srvnode=mktserver.acme.com 5000;
libname sales server=srvnode;
```

or

```
%let srvnode=12.34.56.78 5000;
libname sales server=srvnode;
```

For details about creating valid SAS names, see *SAS Language Reference: Concepts*. For details about LIBNAME and PROC OPERATE, see the LIBNAME statement and the OPERATE procedure in the *SAS/SHARE User's Guide*.

SAS/SHARE Client Example

The following example shows the statements that are specified at a UNIX client to access a server by using the TCP/IP access method. The LIBNAME statement specifies the SAS data library that is accessed through the server. The value _PROMPT_ in the USER= option specifies that the client must provide a valid user ID and password to access the server. The SERVER= option specifies the two-level server name RMTNODE.SHARE1.

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

SAS/SHARE Server Tasks

Task List

1 Configure the SAS/SHARE server service.
2 Specify SAS options and security programs and services (optional).
   - ☐ If the server is to run secured, set the TCPSEC= option to require client authentication.
   - ☐ Configure the authorization of users on servers.
   - ☐ Configure the Authentication program.
   - ☐ Configure the Permission program.
   - ☐ Specify options to encrypt client/server data transfers.

3 Specify TCP/IP as the communications access method.
4 Specify the server.

Configuring the Server Service

Each server must be defined as a service in the **/etc/services** file on each node that a client will access. For details about editing the **/etc/services** file, see “Configuring the SERVICES File” on page 145.
Example:

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

Setting the TCPSEC Option to Require Client Authentication

To authenticate connecting clients, you must specify the value _SECURE_ in the TCPSEC= option to require that clients provide a user ID and a password that are valid on the server. For details about the TCPSEC= option, see “SAS/SHARE Options Only” on page 78.
Example:

```
options TCPSEC=_secure_;
```

Configuring User Access Authority

If SAS was installed from the root account, you can assume that the following task has already been performed. If SAS was not installed from the root account, in order to verify a client's identity and the user's authority to access resources, you must configure resources on the machine that the server runs on. You can provide security on the server by using one of the following.

1 From the root account, to access the SAS Setup Primary menu, issue the following command at a shell prompt (where ***!SASROOT*** is the directory in which SAS was installed).

```
!SASROOT/sassetup
```

From the SAS Setup Primary menu, select
`Run Setup Utilities -> Perform SAS System Configuration -> Configure User Authorization`

2 Alternatively, issue the following commands at a UNIX shell prompt:

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

Configuring the Authentication Program

To configure the Authentication program, **!sasroot/utilities/bin/sasauth** must be owned by root, and the "Set-user-id" mode bit must be set for the file (**chmod 4755 !sasroot/utilities/bin/sasauth**). The built-in Authentication program **sasauth** is started automatically when a client accesses a server that is secured. This program verifies the user ID and password that allows a client to access the server.

Configuring the Permission Program

To configure the Permission program, ***!sasroot*/utilities/bin/sasperm** must be owned by root, and the "Set-user-id" mode bit is set for the file (**chmod 4755 *!sasroot*/utilities/bin/sasperm**).

When given a validated user ID, the server automatically runs the default program **sasperm**. The **sasperm** program verifies that the requesting user has access authority to the file or to the directory that is specified. **sasperm** validates:

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

Encrypting Data in Server/Client Transfers

If network security is configured at the server, you can specify SAS options to encrypt data that a server transfers to a client. For example:

```
options netencrypt netencryptalgorithm=ssl;
options sslcalistloc="/users/johndoe/certificates/cacerts.pem";
```

The NETENCRYPT option specifies that all data transfers between a server and a client will be encrypted. SSL is the network security service that is specified in the NETENCRYPTALGORITHM= option. The SSLCALISTLOC= option specifies the name of a file that contains a list of CA certificates that are to be trusted. For general information about network security, see "SAS/CONNECT and SAS/SHARE Network Security" on page 76.

Specifying TCP/IP as the Communications Access Method

You must specify the TCP/IP communications access method at the server before a client can access it. Use the COMAMID= option in an OPTIONS statement.

Example:

```
options comamid=tcp;
```

The COMAMID= option specifies the communications access method. TCP specifies the TCP/IP access method.

Alternatively, you can specify the COMAMID= option in a SAS start-up command or in a SAS configuration file.

Specifying the Server

You must specify the name of the server in the SERVER= option in the PROC SERVER statement. The syntax follows:

```
SERVER=server-ID
```

server-ID can be either a *server-ID* or a *port* number. The value for *server-ID* corresponds to the service that was configured in the **/etc/services** file. For details, see "Configuring the SERVICES File" on page 145. *port* is the unique number that is associated with the service that is used for transferring data between a client and a server.

Precede the port number with two consecutive underscores.

Note: Do *not* space after the first underscore or the second underscore. △

Note: Specifying a server by using a port number is *not* supported for ODBC clients. △

Examples:

```
proc server server=apex;
proc server server=__5000;
```

For details about creating valid SAS names, see *SAS Language Reference: Concepts*. For details about PROC SERVER, see the SERVER procedure in the *SAS/SHARE User's Guide*.

SAS/SHARE Server Example

The following example shows commands that you specify in the server configuration file on a UNIX machine. The value _SECURE_ that is specified in the TCPSEC option requires clients to provide a user ID and a password that are valid on the server.

```
-set TCPSEC _secure_

options comamid=tcp;
```

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

The COMAMID= option specifies the TCP/IP access method. The PROC SERVER statement specifies the server SHARE1.

PART 5

Windows Operating Environments

Chapter ***6***. . . . . **Windows: TCP/IP Access Method** *93*

CHAPTER 6
Windows: TCP/IP Access Method

Prerequisites for Using TCP/IP under Windows **94**
Task List **94**
Software Requirements **94**
Contexts for User IDs **94**
User Context: Definition **94**
Accessing a Secured Server Using Your Own Context **95**
Accessing a Server Using a Different Context **95**
SAS/CONNECT and SAS/SHARE Server Security **95**
SAS/CONNECT and SAS/SHARE Network Security **95**
SAS/CONNECT Options Only **96**
SAS/SHARE Options Only **97**
SAS/CONNECT Client Tasks **97**
Task List **97**
Specifying TCP/IP as the Communications Access Method **98**
Encrypting Data in Client/Server Transfers **98**
Choosing a Method to Use to Sign On **98**
Signing On to the Same Multi-Processor Machine **98**
Specifying the Server Session **98**
Starting SAS Using the SASCMD Option **99**
Signing On to the Server Session **99**
Signing On Using a Spawner **100**
Ensuring That the Spawner Is Running on the Server **100**
Specifying the Server and the Spawner Service **100**
Specifying a Sign-On Script or a User ID and Password **101**
Specifying a Sign-On Script **101**
Specifying a User ID and Password **102**
Signing On Using the Spawner **102**
Signing On Using a Telnet Daemon **102**
Specifying the Server **102**
Specifying a Sign-On Script File **102**
Signing On to the Server Session **102**
SAS/CONNECT Server Tasks **103**
Task List **103**
Configuring the Windows Spawner Service **103**
Assigning User Rights for a Server That Is Running Secured **103**
Encrypting Data in Server/Client Transfers **103**
Starting the Windows Spawner **104**
SAS/CONNECT Server Example **104**
SAS/SHARE Client Tasks **104**
Task List **104**
Configuring the Server Service **105**

Specifying TCP/IP as the Communications Access Method **105**
Encrypting Data in Client/Server Transfers **105**
Specifying the Server **105**
SAS/SHARE Client Example **106**
SAS/SHARE Server Tasks **107**
Task List **107**
Configuring the Server Service **107**
Setting the TCPSEC Option to Require Client Authentication **107**
Assigning User Rights for a Server That Is Running Secured **107**
Encrypting Data in Server/Client Transfers **108**
Specifying TCP/IP as the Communications Access Method **108**
Specifying the Server **108**
SAS/SHARE Server Example **108**
Data Security for SAS/CONNECT or SAS/SHARE Servers **109**
Client Authentication **109**
Simulated Logon Method **109**
Requirements for Using Simulated Logon with SAS/CONNECT or SAS/SHARE **109**
SSPI **110**
SSPI Requirements for SAS/CONNECT **110**
SSPI Requirements for SAS/SHARE **110**

Prerequisites for Using TCP/IP under Windows

Task List

System Administrator or User

- □ Verify that software requirements are met.
- □ If running the SAS/CONNECT or SAS/SHARE server secured, you must understand user contexts and know the two methods for authenticating clients.
- □ If using network security, set the appropriate SAS options.
- □ Set the appropriate SAS/CONNECT and SAS/SHARE options.

Software Requirements

Ensure that

- □ Base SAS and either SAS/CONNECT or SAS/SHARE are installed on both the client and the server.
- □ The Microsoft TCP/IP System Driver that is provided with the Windows operating environment is installed and configured.

Contexts for User IDs

User Context: Definition

User context is the identifying credentials of the client who is attempting to access a secured server. Identifying credentials include the user ID, password, and file access permissions. Users can specify their own user context or a different user context when accessing a server.

Users specify their own user contexts when logging on to a server by using their user IDs and passwords to access files that they have permission to access.

Users can specify different user contexts when logging on to a server by using someone else's user ID and password. Supplying someone else's user ID and password gives permission to users to access files that they might otherwise be denied access to. A system administrator's user ID and password is an example of a different user context that might be specified. Such a context does not belong to the user but can be granted to the user for access to specific files.

Accessing a Secured Server Using Your Own Context

To access a secured server by using your own user context, specify your user ID and password.

Note: If SSPI (Security Support Provider Interface) is available, you do *not* need to specify a user ID and password. For details, see "SSPI" on page 110. △

Accessing a Server Using a Different Context

To access a server by using a different context, specify the appropriate user ID and password.

Note: If SSPI is available, you must specify the user ID explicitly in a sign-on script or as an option in the SIGNON statement for SAS/CONNECT or in the LIBNAME statement for SAS/SHARE. For details, see "SSPI" on page 110. △

SAS/CONNECT and SAS/SHARE Server Security

Security for a SAS/CONNECT or a SAS/SHARE server's resources can be enforced only by authenticating the identity of the user who runs the client session that is accessing the server session.

Two methods are available for authenticating a client's identity:

- □ Simulated logon (introduced in Version 6)
- □ Microsoft SSPI (introduced in Version 8).

For complete details about server security, see "Data Security for SAS/CONNECT or SAS/SHARE Servers" on page 109.

SAS/CONNECT and SAS/SHARE Network Security

Encryption is the process of transforming plaintext into a less readable form (called ciphertext) by using a mathematical process. The ciphertext is translated back to plaintext for anyone who can supply the appropriate key, which is necessary for decrypting (or unlocking) the ciphertext.

SAS/CONNECT and SAS/SHARE support the following network security services in the Windows operating environment:

SASproprietary
: a fixed encoding algorithm that is included with Base SAS software and is available in all SAS supported operating environments. It requires no additional SAS product licenses.

SAS/SECURE
: an add-on product that uses the encryption algorithms RC2, RC4, DES, and tripleDES.

SSL
: is an abbreviation for Secure Sockets Layer, which is a protocol that provides network security and privacy. Developed by Netscape Communications, SSL uses encryption algorithms that include RC2, RC4, DES, tripleDES, and MD5.

For complete details about setting up and using network security, see the *SAS/CONNECT User's Guide*. After network security is set up in your environment, you set SAS encryption options that are appropriate to the network security service and to the requirements of the client or the server session.

SAS/CONNECT Options Only

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 32768) that you can specify in the SIGNON statement or as a SAS option. For details, see the TBUFSIZE= system option in the *SAS/CONNECT User's Guide*.

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 value for TCPMSGLEN (default=16384) must be set at both the client and the server. If the values that are set for TCPMSGLEN at the client and at the server are different, the smaller value of the two is used during the SAS/CONNECT session.

Example:

```
-set tcpmsglen 8192
```

TCPPORTFIRST=*port-number*(set at the server)
TCPPORTLAST=*port-number*(set at the server)

restricts the range of TCP/IP ports that clients can use to remotely access servers.

Within the range of 0 through 32767, assign a beginning value to TCPPORTFIRST and an ending value to TCPPORTLAST. To restrict the range of ports to only one port, set the values for TCPPORTFIRST and TCPPORTLAST to the same number. Consult with your network administrator for advice about these settings.

At the server, you can set TCPPORTFIRST and TCPPORTLAST in a SAS start-up command or in the configuration file.

In the following example, the server is restricted to the TCP/IP ports 4020 through 4050:

```
options tcpportfirst=4020;
options tcpportlast=4050;
```

TCPTN3270 (set at the client)

TCPTN3270 is an environment variable that supports connections to z/OS servers that use the full-screen 3270 Telnet protocol. The script file TCPTSO32.SCR is provided. See Table 6.1 on page 101 for a complete list of sign-on scripts.

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

Examples:

```
-set tcptn3270 1

options set=tcptn3270 1;
```

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

SAS/SHARE Options Only

AUTHSERVER *domain-or-server*
: specifies the location of the database that contains the user ID and password pairs that are used for validation.

You can specify the AUTHSERVER option in an OPTIONS statement in a SAS session or in an AUTOEXEC file, in a SAS configuration file, in a SAS start-up command, or as a SAS macro variable.

You can also specify a single domain in the form *domain\user ID* when you provide your user ID to the Windows environment.

Example:

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

The domain name **apex** identifies the location of the user ID and password database. The user ID **bass** and the password **time2go** will be verified in the **apex** user ID and password database.

TCPSEC=_SECURE_ | _NONE_ (set at the server)
: specifies whether the TCP/IP access method verifies user access authority before allowing clients to access the server. The TCPSEC option must be set at the server before the server session is started. The default is _NONE_.

SECURE
: requires that the TCP/IP access method verify the authority of clients that attempt to access the server. Each client must supply a user ID and a password that are valid at the server.

NONE
: specifies that the TCP/IP access method does NOT authenticate SAS/SHARE clients that attempt to access the server.

Examples:

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

SAS/CONNECT Client Tasks

Task List

1 Specify TCP/IP as the communications access method.

2 Specify encryption of client/server data transfers (optional).

3 Sign on to the server.

Note: SAS/CONNECT permits TCP/IP connections between clients outside a firewall to spawners that run on hosts inside a firewall. For details, see Chapter 14, "Configuring SAS/CONNECT for Use with a Firewall," on page 149. △

Specifying TCP/IP as the Communications Access Method

TCP/IP is the default communications access method for all operating environments, except z/OS. Therefore, you do not have to explicitly specify the default.

If you choose to specify TCP/IP to connect to a server, you can use the COMAMID= option in an OPTIONS statement.

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

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*. Alternatively, you can set this option in a SAS start-up command or in a SAS configuration file.

Example:

```
options comamid=tcp;
```

Encrypting Data in Client/Server Transfers

If network security is available and is configured at the client, you can specify SAS options to encrypt all data that is transferred between a client and a server. In the following example, the NETENCRYPTALGORITHM= option specifies the RC2 encryption algorithm.

```
options netencryptalgorithm=rc2;
```

For complete details about network security options, see the *SAS/CONNECT User's Guide*.

Choosing a Method to Use to Sign On

- the same multi-processor machine

  Note: This method is most useful if your client machine is equipped with SMP (Symmetric Multi-Processor) hardware. △

- a spawner
- a Telnet daemon.

Signing On to the Same Multi-Processor Machine

If your client machine is equipped with SMP (Symmetric Multi-Processors), and if you want to run one or more server sessions on your machine, perform these tasks:

1. Specify the server session.
2. Specify the SASCMD command to start SAS.
3. Sign on to the server session.

Specifying the Server Session

You can specify the server session in an OPTIONS statement:

```
OPTIONS PROCESS=session-ID;
```

or in the SIGNON statement or command:

```
SIGNON session-ID;
```

session-ID must be a valid SAS name that is 1 to 8 characters in length, and is the name that you assign to the server session on the same multi-processor machine.

Note: PROCESS=, REMOTE=, CREMOTE=, and CONNECTREMOTE= can be used interchangeably. For details, see the CONNECTREMOTE= system option in the *SAS/CONNECT User's Guide*. △

For details about SIGNON=, see the SIGNON statement in the *SAS/CONNECT User's Guide*.

Starting SAS Using the SASCMD Option

Use the SASCMD option to specify the SAS command and any additional options that you want to use to start SAS in the server session on the same multi-processor machine.
The SASCMD option can be specified either in an OPTIONS statement:

```
OPTIONS SASCMD="SAS-command" | "!SASCMD";
```

or directly in the SIGNON statement or command:

```
SIGNON name SASCMD="SAS-command" | "!SASCMD";
```

The -DMR option is automatically appended to the command. If *!SASCMD* is specified, SAS/CONNECT starts SAS on the server by using the same command that was used to start SAS for the current (parent) session.

Note: In order to execute additional commands prior to starting SAS , you might write a script that contains the SAS start-up commands that are appropriate for the operating environment. Specify this script as the value in the SASCMD= option. △

For details, see the SASCMD= system option and the SIGNON statement in the *SAS/CONNECT User's Guide*.

Signing On to the Server Session

Example 1:
In the following example, TCP is the access method, SAS1 is the name of the server session, and SAS_START is the command that starts SAS on the same multi-processor machine.

```
options comamid=tcp;
signon sas1 sascmd='sas_start';
```

Example 2:
In the following example, the values for the COMAMID=, SASCMD=, and PROCESS= options are set in OPTIONS statements. The SASCMD= option identifies the command that starts SAS. The PROCESS= option identifies the server session on the same multi-processor machine. Because the SASCMD= and the PROCESS= options are defined, only a simple SIGNON statement is needed.

```
options comamid=tcp sascmd="sas_start";
options process=sas1;
signon;
```

Signing On Using a Spawner

1 Ensure that the spawner is running on the server.
2 Specify the server and an optional service.
3 Specify the sign-on script (if you are signing on using a script), or specify a user ID and password (if you are signing on without a script).

   Note: If the SSPI is available or the server is not running secured, you do not have to specify a user ID and password. For details, see "SSPI" on page 110. △

4 Sign on to the server using a spawner.

Ensuring That the Spawner Is Running on the Server

Before you can access the spawner, the spawner program must be running on the server. For information about the spawner that you are connecting to, see Chapter 7, "SAS/CONNECT Spawners," on page 113.

Note: The system administrator for the machine that the spawner runs on must start the spawner. The spawner program on the server cannot be started by the client. △

Specifying the Server and the Spawner Service

The name of the server can be specified either in an OPTIONS statement:

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

or directly in the SIGNON statement or command:

```
SIGNON node-name[.service-name | .port-number];
```

node-name is based on the server that you are connecting to. *node-name* must be a valid SAS name that is 1 to 8 characters in length and is either:

- □ the short machine name of the server that you are connecting to. This name must be defined in the HOSTS file in the client operating environment or in your Domain Name Server (DNS).
- □ a macro variable that contains either the IP address or the name of the server that you are connecting to.

You specify *service-name* when connecting to a server that runs a spawner program that is listening on a port other than the Telnet port. If the spawner was started by using the -SERVICE spawner option, you must specify an explicit *service-name*. The value of *service-name* and the value of the -SERVICE spawner option must be identical. Alternatively, you can specify the explicit port number that is associated with *service-name*.

Example 1:

In the following example, REMHOST is the name of the node that the spawner runs on, and PORT1 is the name of the service that is defined at the client. The client service PORT1 must be assigned to the same port that the spawner is listening on.

```
signon remhost.port1;
```

Example 2:

In the following example, the macro variable REMHOST is assigned to the fully-qualified name of the machine that the server runs on. This server has a spawner running that is listening on port 5050. The server session that is specified in the

SIGNON statement uses the node name REMHOST and the service-name 5050, which is the explicit port value.

```
%let remhost=pc.rem.us.com;
signon remhost.5050;
```

You can also assign a specific port number by including the port number in the definition of the macro variable, for example,

```
%let remhost=pc.rem.us.com 5050;
signon remhost;
```

Specifying a Sign-On Script or a User ID and Password

You can use a sign-on script to sign on to the spawner, or you can sign on to a spawner without a script. If you sign on to a secured spawner without a script, you must supply a user ID and password *unless* SSPI is available. For details, see “SSPI” on page 110.

Note: If you connect to a spawner, you can sign on by using a script unless the spawner is started by using the -NOSCRIPT option. If the -NOSCRIPT option is set, you cannot use a script. If there is no script, you do not assign the fileref RLINK in a FILENAME statement. For information about the spawner that you are connecting to, see Chapter 7, “SAS/CONNECT Spawners,” on page 113. △

Specifying a Sign-On Script

If you are signing on by using a script, you must specify the script that you want to use. The script file is executed by the SIGNON statement or command. By default, the script prompts for user ID and password.

To use one of the sample script files that are provided with SAS/CONNECT for signing on and signing off, assign the fileref RLINK to the appropriate script file. The script is based on the server that you are connecting to. The sample scripts are installed at

```
!sasroot\CONNECT\SASLINK
```

To specify a script, use the FILENAME statement. For example,

```
FILENAME RLINK '!sasroot\connect\saslink\script-name';
```

script–name specifies the appropriate script file for the server.

Table 6.1 on page 101 lists the scripts that are provided in SAS software:

Table 6.1 SAS/CONNECT Sign-on Scripts for TCP/IP under Windows

Server	Script Name
TSO under OS/390	**tcptso.scr**
TSO under z/OS, SAS 9 or later	**tcptso9.scr**
z/OS (without TSO)	**tcpmvs.scr**
z/OS (using full-screen 3270 Telnet protocol)	**tcptso32.scr**
OpenVMS Alpha	**tcpvms.scr**

Server	Script Name
UNIX	**tcpunix.scr**
Windows	**tcpwin.scr**

Specifying a User ID and Password

If SSPI is available, you can submit the SIGNON statement without a user ID and password. If SSPI is not available and you are signing on to a secured spawner without using a script, you must provide a user ID and password in order to log on. For example,

```
SIGNON USER=user-ID | _PROMPT_ [ PASSWORD=password | _PROMPT_ ];
```

Signing On Using the Spawner

To start SAS, sign on to the server using the spawner.

In the following example, a Windows client connects to a Windows server by using a spawner without a script file. In the SIGNON statement, RMTHOST.SPAWNER specifies the node RMTHOST and the service SPAWNER. This server specification presumes that a spawner is running on the node RMTHOST, and that the spawner was started using the service SPAWNER. Because SSPI is used, the client does not set the USER= and PASSWORD= options.

Example:

```
options comamid=tcp;

signon rmthost.spawner;
```

Signing On Using a Telnet Daemon

1 Specify the server.
2 Specify a sign-on script file.
3 Sign on to the server session.

Specifying the Server

The name of the server can be specified either in an OPTIONS statement:

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

or directly in the SIGNON statement or command:

```
SIGNON node-name;
```

Specifying a Sign-On Script File

If you are signing on by using a script, you must specify the script that you want to use. The script file is executed by the SIGNON statement or command. By default, the script prompts for user ID and password. For details, see “Specifying a Sign-On Script” on page 101.

Signing On to the Server Session

In the following example, you specify the statements at a Windows client to use the TCP/IP access method to connect to a z/OS server. The FILENAME statement identifies the script file that you use to sign on to a server. The script file contains a prompt for a user ID and a password that are valid on the server. The COMAMID= option specifies the TCP/IP communications access method for connecting to the server RMTNODE, which is specified in the REMOTE= option.

```
filename rlink '!sasroot\CONNECT\SASLINK\tcptso.scr';
options comamid=tcp remote=rmtnode;
signon;
```

SAS/CONNECT Server Tasks

Task List

1 Configure the Windows spawner service.

2 Configure user rights and security services (optional).
  - □ Assign user rights if the server is to run secured.
  - □ Specify security service options to encrypt client/server data transfers.

3 Start the Windows spawner at the server.

Note: If the Windows spawner is not being used, there are no server tasks. △

Configuring the Windows Spawner Service

To enable clients to connect to a Windows server by using the Windows spawner, configure the spawner service in the **SERVICES** file at the server. For details, see Chapter 13, "TCP/IP SERVICES File," on page 145.

Assigning User Rights for a Server That Is Running Secured

If you only use SSPI for authentication, setting user rights is not necessary.

If you use the simulated logon method of authentication, the following user rights must be set at the server machine:

- □ "Act as part of the operating system" for the user who runs the spawner
- □ "Increase quotas" for the user who runs the spawner
- □ "Replace process level tokens" for the user who runs the spawner
- □ "Log on as batch job" for all clients who need to connect to the server.

For details about the simulated logon and SSPI methods of authentication, see "Data Security for SAS/CONNECT or SAS/SHARE Servers" on page 109.

Encrypting Data in Server/Client Transfers

If network security is configured at the server, you can specify SAS options to encrypt data that a server transfers to a client. For example:

```
options netencrypt netencryptalgorithm=ssl;
```

The NETENCRYPT option specifies that all data transfers between a server and a client will be encrypted. SSL is the network security service that is specified in the NETENCRYPTALGORITHM= option. For general information about network security, see "SAS/CONNECT and SAS/SHARE Network Security" on page 95.

Starting the Windows Spawner

You must start the Windows spawner on a Windows server to enable clients to connect to it. The spawner program resides on a server and listens for SAS/CONNECT client requests for connection to the server. After the spawner program receives a request, it starts a server session. For details about starting the Windows spawner, see Chapter 11, "Windows Spawner," on page 131.

Specifying the -SECURITY option in the Windows spawner start-up command requires authentication of connecting clients.

If network security has been configured at the server, set the appropriate encryption options when starting the spawner.

SAS/CONNECT Server Example

Setting these options on the command line restricts access to ports 5020 through 5050.

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

The following example shows the spawner start-up command. The TCP/IP access method is specified. The -FILE option executes the MYSAS.CMD file, which starts a SAS session.

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

For details about the contents of a command file and how to run the Windows spawner, see Chapter 11, "Windows Spawner," on page 131. Options that are specified during spawner start-up override options that are specified in a server configuration file.

SAS/SHARE Client Tasks

Task List

1 Configure the server service.

2 Specify TCP/IP as the communications access method.

3 Access a secured server.

4 Specify encryption of client/server data transfers (optional).

5 Specify the server name.

Configuring the Server Service

Each server must be defined as a service in the SERVICES file on each machine that a client will access the server from. The SERVICES file is usually located in the directory where the TCP/IP software is installed. For details about editing the SERVICES file, see "Configuring the SERVICES File" on page 145.

Specifying TCP/IP as the Communications Access Method

TCP/IP is the default communications access method that is used in the Windows operating environment. You can omit specifying the access method in the COMAMID= option and the TCP/IP access method is assumed, by default.

If you choose to specify TCP/IP to connect to a server, you can use the COMAMID= option in an OPTIONS statement.

```
options comamid=tcp;
```

The COMAMID= option specifies the communications access method. TCP specifies the TCP/IP access method.

Alternatively, you can specify the COMAMID= option in a configuration file or in a SAS start-up command.

Encrypting Data in Client/Server Transfers

If network security is configured at the client, you can specify SAS options to encrypt data that a client transfers to a server. An example follows:

```
options netencrypt netencryptalgorithm=ssl;
```

The NETENCRYPT option specifies that all data transfers between a client and a server will be encrypted. SSL is the network security service that is specified in the NETENCRYPTALGORITHM= option. For general information about network security, see "SAS/CONNECT and SAS/SHARE Network Security" on page 95.

Specifying the Server

If the client and server sessions are running on different network nodes, you must include the TCP/IP node in the server ID in the LIBNAME and PROC OPERATE statements by using a two-level server name as follows:

SERVER=*node.server*

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

1. SAS macro variable
2. environment variable
3. acceptable node name.

node can be either of the following:

- valid TCP/IP node name
- IP address.

If the server and the client sessions are running on the same node, you can omit the node name.

server can be either of the following:

- □ *server-ID*
- □ *port*.

The *server-ID* must be identical to the service name that is specified in the SERVICES file. For details, see “Configuring the SERVICES File” on page 145.

Example 1:

A *port* is the unique number that is associated with the service that is used for passing data to and receiving data from the server.

Precede the port number with two consecutive underscores.

Note: Do *not* space after the first underscore or the second underscore. △

Note: Specifying a server by using a port number is *not* supported for ODBC clients. △

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

Example 2:

If the TCP/IP node name is not a valid eight-character 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.

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. △

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

Example 3:

You might assign the node name and the server ID to a macro variable.

```
%let srvnode=mktserver.acme.com 5000;
libname sales server=srvnode;
```

or

```
%let srvnode=12.34.56.78 5000;
libname sales server=srvnode;
```

For details about creating valid SAS names, see *SAS Language Reference: Concepts*. For details about LIBNAME and PROC OPERATE, see the LIBNAME statement and the OPERATE procedure in the *SAS/SHARE User’s Guide*.

SAS/SHARE Client Example

The following example shows the statements that are specified at a Windows client who accesses a server by using a different user context. The LIBNAME statement specifies the SAS data library that is accessed through the server, which is specified by the two-level server name RMTNODE.SHARE1.

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

SAS/SHARE Server Tasks

Task List

1 Configure the SAS/SHARE server.

2 Configure SAS options and security programs and services (optional).

- □ If the server is to run secured, specify the TCPSEC= option to require client authentication.
- □ Assign user rights for a server that is to run secured.
- □ Specify security service options to encrypt client/server data transfers.

3 Specify TCP/IP as the communications access method.

4 Specify the server.

Configuring the Server Service

Each server must be defined as a service in the SERVICES file on each node that a client will access. The SERVICES file is located in the directory where the TCP/IP software is installed. For details about editing the SERVICES file, see "Configuring the SERVICES File" on page 145.

Example:

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

Setting the TCPSEC Option to Require Client Authentication

To authenticate connecting clients, you must specify the value _SECURE_ in the TCPSEC= option to require that clients provide a user ID and a password that are valid on the server. For details about the TCPSEC= option, see "SAS/SHARE Options Only" on page 97.

Example:

```
options tcpsec=_secure_;
```

Assigning User Rights for a Server That Is Running Secured

If you only use SSPI for authentication, setting user rights is not necessary.

If you use the simulated logon method of authentication, the following user rights must be set at the server machine:

- □ "Act as part of the operating system" for the server administrator
- □ "Increase quotas" for the server administrator
- □ "Replace process level tokens" for the server administrator
- □ "Log on as batch job" for all clients who need to access to the server.

Encrypting Data in Server/Client Transfers

If network security is configured at the server, you can specify SAS options to encrypt data that a server transfers to a client. For example:

```
options netencrypt netencryptalgorithm=ssl;
```

The NETENCRYPT option specifies that all data transfers between a server and a client will be encrypted. SSL is the network security service that is specified in the NETENCRYPTALGORITHM= option. For general information about network security, see "SAS/CONNECT and SAS/SHARE Network Security" on page 95.

Specifying TCP/IP as the Communications Access Method

TCP/IP is the default communications access method on Windows. You can omit specifying the access method in the COMAMID= option, and the TCP/IP communications access method is assumed, by default. If you choose to specify TCP/IP to connect to a server, you can use the COMAMID= option in an OPTIONS statement.

```
options comamid=tcp;
```

Alternatively, you can specify the COMAMID option in a SAS configuration file or in a SAS start-up command.

Specifying the Server

You must specify the name of the server in the SERVER= option in the PROC SERVER statement.

```
SERVER=server
```

server can be either a *server-ID* or a *port* number. The value for *server-ID* corresponds to the service that was configured in the SERVICES file. For details, see "Configuring the SERVICES File" on page 145. *port* is the unique number that is associated with the service that is used for transferring data between a client and a server.
Precede the port number with two consecutive underscores.

Note: Do not space after the first underscore or the second underscore. △

Examples:

```
proc server server=apex;
proc server server=_ _5000;
```

For details about SAS naming rules, see *SAS Language Reference: Concepts*. For details about the PROC SERVER statement, see the *SAS/SHARE User's Guide*.

SAS/SHARE Server Example

The following example shows the statements that you specify in a SAS session at the machine where the server runs. The TCP/IP access method is specified and the server SHARE1 is started on the Windows machine.

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

Data Security for SAS/CONNECT or SAS/SHARE Servers

Client Authentication

Authentication is the act of verifying the identity of the user who is attempting to access a machine, that is, the machine that either the client session or the server session runs on. Authentication is performed so that a machine can use the identity information to make decisions about the user's authority to access protected resources. Under Windows, the user ID, password, and access permissions make up a user context.

Resources on a SAS/CONNECT or a SAS/SHARE server are considered to be protected when both of the following conditions are met:

- □ The server requires that the client provide its identity.
- □ The client presents an identity that is successfully authenticated.

After the client's identity is authenticated, the client is given the appropriate permissions to access the server's resources.

Under Windows, two methods are available for authenticating a client's identity:

- □ Simulated logon (introduced in Version 6)
- □ SSPI (introduced in Version 8).

Simulated Logon Method

The simulated logon method is the most commonly used method of authentication and is available in all SAS supported operating environments. In a simulated logon, the client provides a user ID and password that are checked by the server.

You use a simulated logon when:

- □ the client or the server (or both) does not run on a Windows machine
- □ the user who runs the client machine is not a "trusted" user at the server machine
- □ the user who runs the client machine wants to log on by using a different user context.

For details about user context, see "Contexts for User IDs" on page 94.

Requirements for Using Simulated Logon with SAS/CONNECT or SAS/SHARE

In order to authenticate the credentials (user ID and password) of a user who runs a client session, the server administrator of the SAS/CONNECT or the SAS/SHARE server must either set user rights for others or possess user rights.

SAS/CONNECT and SAS/SHARE Requirements

- □ set the "Log on as batch job" user right at the server machine for the user who runs the client session
- □ possess the "Act as part of the operating system" user right on the server machine

SAS/CONNECT Only Requirements

- □ possess the "Increase quotas" user right on the server machine
- □ possess the "Replace a process level token" user right on the server machine
- □ set the -SECURITY option at spawner invocation.

SAS/SHARE Only Requirements

- □ set the option TCPSEC=_SECURE_
- □ set the option AUTHENTICATE=REQUIRED in the PROC SERVER statement. REQUIRED is the default value.

SSPI

SSPI (Security Support Provider Interface) enables transparent authentication for connections between Windows machines. Users that are members of a "trusted" domain are authenticated automatically, and user context information is transferred to the server.

Windows attempts to use SSPI for authentication whenever a user ID is not explicitly supplied.

SSPI is available *only* when the client and the server sessions both run on Windows machines, *and* the user who runs the client machine is a member of a domain that is "trusted" at the server machine.

SSPI Requirements for SAS/CONNECT

In order to use SSPI for authentication, the SAS/CONNECT server administrator must:

- □ set the -SECURITY option at spawner invocation.

SSPI Requirements for SAS/SHARE

In order to use SSPI for authentication, the SAS/SHARE server administrator must:

- □ specify the option TCPSEC=_SECURE_
- □ specify the option AUTHENTICATE=REQUIRED in the PROC SERVER statement. REQUIRED is the default value.

PART 6

Spawners and Files

Chapter **7**. . . . . **SAS/CONNECT Spawners** *113*

Chapter **8**. . . . . **OpenVMS Alpha Spawner** *119*

Chapter **9**. . . . . **z/OS Spawner** *123*

Chapter **10**. . . . . **UNIX Spawner** *127*

Chapter **11**. . . . . **Windows Spawner** *131*

Chapter **12**. . . . . **Encryption Options** *137*

Chapter **13**. . . . . **TCP/IP SERVICES File** *145*

CHAPTER 7 SAS/CONNECT Spawners

Spawner Definition **113**
Benefits of Using a Spawner to Sign On to a Server **114**
  Protects Client's User ID and Password **114**
  Controls Client Access to the Server in a Firewall Configuration **114**
  Eliminates the Need for a Sign-On Script **114**
Support for Spawners by Operating Environment **114**
Client Connection to a Spawner **114**
Spawner Connection Examples **115**
  Scripted Signon to a UNIX Spawner **115**
    Server **115**
    Client **115**
  Scriptless Signon to a Windows Spawner That Runs as a Service **116**
    Server **116**
    Client **116**
  Scripted Signon to an OpenVMS Spawner **116**
    Server **116**
    Client **116**
  Encrypted Signon to a z/OS Spawner **117**
    Server **117**
    Client **118**

Spawner Definition

A *spawner* is a program that starts a SAS session on the server on behalf of the connecting client. Signing on to a SAS/CONNECT spawner that runs on a server is an alternative to signing on to a server by using a Telnet daemon. A spawner is assigned to a single port on the server. The port listens for requests for connection to the server.

Note: Do not confuse a SAS/CONNECT spawner with the SAS Integration Technologies object spawner. Visit the **`support.sas.com/rnd/itech/library/index.html`** Web site for information about the planning and administration of a SAS Integration Technologies object spawner. △

Benefits of Using a Spawner to Sign On to a Server

Protects Client's User ID and Password

By default, the spawner encrypts the client's user ID and password that are sent to the spawner during signon. Without encryption, the user ID and password would pass through the network as clear, readable text, which presents a security risk.

Note: Release 6.09E and subsequent releases and Release 6.11 TS040 and subsequent releases support user ID and password encryption. Prior releases do not support user ID and password encryption. △

To encrypt all data that flows through the network after signon (such as for remote submits and data transfers), you must use a security service. For details about security services that are supported in SAS 9.1, see the *SAS/CONNECT User's Guide*.

Controls Client Access to the Server in a Firewall Configuration

A spawner can be used to control the number of ports that clients outside a firewall can use to access a server that is inside the firewall. Controlled client access facilitates a machine's security and economizes resources. For details, see Chapter 14, "Configuring SAS/CONNECT for Use with a Firewall," on page 149.

Eliminates the Need for a Sign-On Script

The primary purpose of a sign-on script is to

- send the user ID and password to the server
- supply the SAS command for starting the SAS session on the server.

Because the user ID and password can be directly specified as options in the SIGNON statement (or command), and the spawner controls the start-up of a SAS session on the server, the need for a sign-on script is eliminated.

Support for Spawners by Operating Environment

SAS 9 supports only the TCP/IP access method in the following operating environments:

- OpenVMS Alpha
- UNIX
- Windows
- z/OS.

Client Connection to a Spawner

1 The spawner service must be configured in the client's SERVICES file. Verify that the spawner is configured in the SERVICES file. For details, see Chapter 13, "TCP/IP SERVICES File," on page 145.

2 If you use a script when connecting to a spawner, script file processing passes the user ID and password to the server.

However, if you do *not* use a script file, you can deliver the user ID and password to the server by specifying values for the USERID= and PASSWORD= options in the SIGNON statement.

3 If you do *not* use a script file, use the following syntax to connect to the spawner:

```
options comamid=tcp remote=spawner-ID;
signon user=user-ID password=password;
```

Example:

```
options comamid=tcp remote=rmthost.spawn;
signon user=slim password=_prompt_;
```

The COMAMID option specifies the TCP/IP access method, which is used to connect the client to the spawner SPAWN that runs on the UNIX node RMTHOST. In the absence of a script file, the user ID and password are specified as options in the SIGNON statement. The value _PROMPT_ for PASSWORD causes SAS to prompt for a password at signon. The SIGNON statement makes the connection and starts a SAS session on the server.

Spawner Connection Examples

Scripted Signon to a UNIX Spawner

Server

From the UNIX node that the server will run on, use the following command to start the spawner.

```
sastcpd -service spawner -sascmd /u/username/mystartup
```

The -SERVICE option specifies the name of the service SPAWNER that listens for incoming connections. The -SASCMD option specifies the path to the MYSTARTUP file, which starts the SAS session on the server. For a description of the -SASCMD option and an example of the content of the MYSTARTUP executable file, see Chapter 10, "UNIX Spawner," on page 127.

Client

At a Windows client, the following statements are entered to sign on to the UNIX node RMTHOST by using the TCP/IP access method. A script file that is assigned to the RLINK fileref prompts the user at the client for the user ID and the password that are needed to log on to the UNIX server. The server name (in this example, RMTHOST) must be either the name of the UNIX node or a macro variable that contains the IP address or the name of the UNIX node that runs the spawner. The SIGNON statement contains the ID of the server session, which is specified as a two-level name: the node name and the service name. A two-level name is needed when signing on to a UNIX node that runs a spawner.

```
options comamid=tcp;
filename rlink '!sasroot\connect\saslink\tcpunx.scr';
signon rmthost.spawner;
```

Scriptless Signon to a Windows Spawner That Runs as a Service

Server

The following command installs the spawner service on a Windows machine:

```
C:\SAS> spawner -install -authserver ntdomain
```

In this example, the -INSTALL option installs the spawner as a service on a Windows machine. The -AUTHSERVER option specifies NTDOMAIN as the database to be used for performing user authentication of the user ID and password for connecting clients.

After the service is installed, it must be started before it can be used. You can start the service using either of the following:

- the NET START command
- the services applet
- rebooting the machine.

Client

From any client, the following statements connect to the spawner program by using the TCP/IP access method. The SIGNON statement specifies the ID of the server session REMNODE. This ID must be the name of the Windows machine or a macro variable that contains the IP address of the Windows machine that the spawner runs on. Because a script file is not used, the user ID and password to the server must be specified as options in the SIGNON statement. The value _PROMPT_ in the SIGNON statement causes SAS to prompt for the password.

```
options comamid=tcp;
signon remnode user=joeblack password=_prompt_;
```

Note: The password is displayed as Xs in the SAS log. △

Scripted Signon to an OpenVMS Spawner

Server

The following command starts the spawner VMSSPAWN on an OpenVMS Alpha machine. The absence of the -SASCMD option in the spawner start-up command implies that the client will use a script file to specify the SAS command that starts SAS on the OpenVMS Alpha machine.

```
sastcpd -service vmsspawn
```

Client

At a UNIX client, the following statements specify the script file TCPVMS.SCR, which makes a connection to the spawner MONARCH.VMSSPAWN. The machine name (in this example, MONARCH) must be either the name of the OpenVMS Alpha node or a macro variable that contains the IP address or name of the OpenVMS Alpha node that the spawner runs on. In this example, an OPTIONS statement specifies the ID of the server session as a two-level name, which represents the node name and the service name. A two-level name is needed when signing on to an OpenVMS Alpha machine that runs a spawner.

```
options comamid=tcp;
options remote=monarch.vmsspawn;
filename rlink "!sasroot/misc/connect/tcpvms.scr";
signon;
```

Encrypted Signon to a z/OS Spawner

Server

The following z/OS command starts the spawner on the z/OS server.

```
START SPAWNER
```

This command activates the started task procedure. SPAWNER is the name of the service that is defined in the started task procedure.

An example follows:

```
//SPAWNER  PROC PROG=SASTCPD,
//         SERVICE='spawner',
//         PARMFILE='SAS.SPAWNER.PARMS'
//*
//SPAWNER  EXEC PGM&PROG,REGION=40M,
//         PARM='-service &SERVICE =<//DDN:PARMS'
//*
//STEPLIB  DD   DISP=SHR,DSN=SAS.AUTH.LOAD
//PARMS    DD   DISP=SHR,DSN=&PARMFILE,FREE=CLOSE
//SYSPRINT DD   SYSOUT=*
//SYSTERM  DD   SYSOUT=*
//SYSUDUMP DD   SYSOUT=*
```

PARMFILE contains the options that start a spawner. For example,

```
-netencryptalgorithm rc2
-sascmd "/usr/local/bin/spawnsas.sh nosasuser opt(''dmr noterminal comamid=tcp'')"
```

The -NETENCRYPTALGORITHM option specifies that the spawner is started using the RC2 encryption algorithm. The -SASCMD option specifies a UNIX System Services shell script that starts SAS. This command assumes that a shell script named **`spawnsas.sh`** is installed in **`/usr/local/bin`**. The command specifies the SAS CLIST option NOSASUSER, which specifies that a user's SASUSER file should not be allocated. NOSASUSER allows a client to sign on to a server multiple times using the same user ID and password. The parenthesis enclose the SAS system options DMR and COMAMID=TCP. NOTERMINAL prevents the display of requestor windows in the server session.

Client

In the following example, the client specifies user ID and password encryption by setting the RC2 encryption algorithm. In this example, the two-level name, which represents the node name and the service name, specifies the ID of the server session in the SIGNON statement. A two-level name is needed when signing on to a z/OS operating environment that runs a spawner. The user must supply a valid user ID and password as values for the USER= and PASSWORD= options in the SIGNON statement.

```
options netencryptalgorithm=rc2;
signon rmthost.spawner user=joeblack password=born2run;
```

CHAPTER

8

OpenVMS Alpha Spawner

OpenVMS Alpha Spawner Requirements **119**
  Location of the OpenVMS Alpha Spawner **119**
  Spawner Privileges **119**
Starting the OpenVMS Alpha Spawner **120**
Ending the OpenVMS Spawner **121**

OpenVMS Alpha Spawner Requirements

Location of the OpenVMS Alpha Spawner

The OpenVMS Alpha spawner program is stored on the server's node in the executables directory. An alias can be defined that points to the appropriate directory and executable for the spawner program by using the following DCL command:

```
SASTCPD:==SAS$LIBRARY:SASTCPD.EXE
```

Included in the installation of Base SAS are sample DCL files that demonstrate how to start the daemon as a detached process. The samples are located in SAS$ROOT:[MISC.BASE]. Make a back-up copy of these files before you make any modifications to them.

SASTCPD_STARTUP.COM
: executes SASTCPD.COM as a detached process.

SASTCPD_SETUP.COM
: defines the necessary process-level logicals and symbols that are required to run SASTCPD as a detached process. This file is generated during the installation of Base SAS.

SASTCPD_TEMPLATE.COM
: performs setup, which might be needed to enable the client process to execute.

Spawner Privileges

In order for the spawner to start the SAS process, the spawner must have the SYSPRV privilege.

Starting the OpenVMS Alpha Spawner

The syntax for the command to start the OpenVMS Alpha spawner follows:

SASTCPD <*options*>

options can be any of the following.

-HELP
prints a list of valid parameters.

-NOCLEARTEXT
prevents signons from clients that do not support user ID and password encryption. This option prevents clients that are running “older” releases (prior to 6.09E and 6.11 TS040, which do not support user ID and password encryption) from signing on to the spawner program. However, the default permits both encrypted and clear-text user IDs and passwords.

-NOINHERITANCE
disables socket inheritance. Socket inheritance allows SAS/CONNECT servers to use the socket connection that is established between the SAS/CONNECT client and the spawner. Socket inheritance saves resources and is easier to configure when clients connect to a server that is within a firewall. Socket inheritance is enabled by default.

-NOSCRIPT
prevents signon from clients that use scripts, and allows signon only from clients that do not use scripts.

This option requires that the client specify a user ID and a password during signon. For details, see the SIGNON statement in the *SAS/CONNECT User’s Guide*.

-NOSCRIPT can be useful if you want to limit SAS start-up commands to the use of the -SASCMD option. Specifying -NOSCRIPT restricts clients from specifying additional options in SAS start-up commands or script files. If -NOSCRIPT is used, -SASCMD must also be used.

-OMRCONFIGFILE *fully-qualified-path*
specifies a fully-qualified path to the configuration file in XML format that contains the information necessary to connect to a SAS Metadata Server. For details about creating the configuration file, see the Base SAS Help for the Metadata Server Connections window.

If -OMRCONFIGFILE is used, -SASSPAWNERCN must also be used.

-SASCMD “*command*”
specifies the SAS command or a command file that is specific to the OpenVMS Alpha operating environment that starts a SAS session when you sign on without a script. If the client does not specify a script file at sign on, the -SASCMD option must be specified when starting the spawner. For example:

```
sastcpd -service sasjob -sascmd "@mystartup.com"
```

The MYSTARTUP file contains these lines, which starts SAS in an OpenVMS Alpha environment:

```
$!
$! mystartup
$!
sas /DMR/NOTERMINAL/NOSYNTAXCHECK/COMAMID=TCP
$ exit
```

-SASSPAWNERCN "*CONNECT-spawner-object-name*"
: specifies the name of the CONNECT spawner object in the SAS Metadata Repository. A name that includes one or more spaces must be enclosed in quotation marks. For details about generating a CONNECT spawner definition, see the help for the SAS/CONNECT Spawner server type in the Server Manager of SAS Management Console.

  If -SASSPAWNERCN is used, -OMRCONFIGFILE must also be used.

-SERVICE *service-name*
: specifies the name of the service that the spawner uses to listen for incoming requests. This value is identical to the *service* value in the REMOTE= option that the user specifies at the client prior to sign on. Because there is no default, you must specify this value.

  The service name must also be defined in the SERVICES file at both the client and the server. For details, see Chapter 13, "TCP/IP SERVICES File," on page 145.

-SHELL
: enables the SAS session that is invoked by the spawner to create a shell, which is required in order for the server's SAS session to execute commands.

encryption-options
: The encryption options that you can set depend on which security service is used. For details, see Chapter 12, "Encryption Options," on page 137.

For an example of how to start a spawner, see "Scripted Signon to an OpenVMS Spawner" on page 116.

Ending the OpenVMS Spawner

To end the spawner, enter the interrupt signal (which, usually, is CTRL-C). If the OpenVMS Alpha spawner is running in the background, kill its process.

CHAPTER 9

z/OS Spawner

z/OS Spawner Requirements **123**
Location of the z/OS Spawner **123**
z/OS Spawner User ID Requirement **123**
Assigning a User ID to the Started Task **123**
z/OS Version Level Requirement **124**
Starting the z/OS Spawner **124**
Defining the Shell Script for Starting SAS **125**
Ending the z/OS Spawner **126**

z/OS Spawner Requirements

Location of the z/OS Spawner

The z/OS spawner program, SASTCPD, is located in the SAS load library. If the BPX.DAEMON RACF profile is enabled and RACF Program Control is active, then this library, as well as the SAS Transient Library, must be RACF program controlled.

z/OS Spawner User ID Requirement

If the BPX.DAEMON profile in the RACF FACILITY class and RACF Program Control are active, then the user ID for the spawner started task does not require superuser (uid=0) authority.

The spawner user ID does not require READ access to the BPX.DAEMON profile. This is a change from prior releases of the spawner.

The spawner no longer requires APF authorization.

Assigning a User ID to the Started Task

To assign a user ID to the started task, do either of the following:

- Add the started task to the RACF Started Procedures Table ICHRIN03.
- Define a profile for the started task in the RACF class STARTED.

For details, see *z/OS Security Server (RACF) Command Language Reference* from IBM.

z/OS Version Level Requirement

The spawner requires z/OS or OS/390 Version 2 Release 10.

Starting the z/OS Spawner

In order to start the z/OS spawner, you must specify options in the PARMS file. You can use any of the following options.

-HELP
: prints a list of valid options.

-NOCLEARTEXT
: prevents signons from clients that do not support user ID and password encryption. This option prevents clients that are running “older” releases (prior to 6.09E and 6.11 TS040, which do not support user ID and password encryption) from signing on to the spawner program. However, the default permits both encrypted and clear-text user IDs and passwords.

-NOINHERITANCE
: disables socket inheritance. Socket inheritance allows SAS/CONNECT servers to use the socket connection that is established between the SAS/CONNECT client and the spawner. Socket inheritance saves resources and is easier to configure when clients connect to a server that is within a firewall. Socket inheritance is enabled by default.

-NOSCRIPT
: prevents signon from clients that use scripts, and allows signon only from clients that do not use scripts.

  This option requires that the client specify a user ID and a password during signon. For details, see the SIGNON statement in the *SAS/CONNECT User’s Guide*.

  -NOSCRIPT can be useful if you want to limit SAS start-up commands to the use of the -SASCMD option. Specifying -NOSCRIPT restricts clients from specifying additional options in SAS start-up commands or script files. If -NOSCRIPT is used, -SASCMD must also be used.

-OMRCONFIGFILE *fully-qualified-path*
: specifies a fully-qualified path to the configuration file in XML format that contains the information necessary to connect to a SAS Metadata Server. Use UNIX file-naming conventions for specifying the path. For details about creating the configuration file, see the Base SAS Help for the Metadata Server Connections window.

  If -OMRCONFIGFILE is used, -SASSPAWNERCN must also be used.

-SASCMD “*command*”
: specifies a UNIX System Services (USS) shell script for starting a SAS session. You must use -SASCMD and a shell script if you do not specify a signon script in the client session using an RLINK fileref.

  The script interprets the command arguments and environment variables and builds a TSO command that invokes a SAS session. For an example of a SAS start-up shell script, see “Defining the Shell Script for Starting SAS” on page 125.

-SASSPAWNERCN “*CONNECT-spawner-object-name*”
: specifies the name of the CONNECT spawner object to use in the SAS Metadata Repository. A name that includes one or more spaces must be enclosed in

quotation marks. For details about generating a CONNECT spawner definition for the SAS Metadata Server, see the help for SAS/CONNECT Spawner server type in the Server Manager of SAS Management Console.

If -SASSPAWNERCN is used, -OMRCONFIGFILE must also be used

-SERVICE *service-name* | *port-number*

specifies the name or TCP/IP port number of the service that the z/OS spawner uses to listen for incoming requests. This value is identical to the service value in the REMOTE= option that the user specifies at the client prior to signon. Because there is no default, you must specify this value.

Service names and ports must be configured in the SERVICES file on both the client and server. Choose a unique port number that is in the range of 1 to 65535. For details about the SERVICES files, see Chapter 13, "TCP/IP SERVICES File," on page 145.

encryption-options

The encryption options that you can set depend on which security service is used. For details, see Chapter 12, "Encryption Options," on page 137.

For an example of how to start the spawner, see "Encrypted Signon to a z/OS Spawner" on page 117.

Defining the Shell Script for Starting SAS

The spawner invokes a UNIX System Services (USS) shell script that can be specified in either a SAS/CONNECT signon script or the -SASCMD spawner option.

Example:

```
-sascmd "/usr/local/bin/spawnsas.sh nosasuser opt(''dmr noterminal comamid=tcp'')"
```

This command assumes that a shell script named **`spawnsas.sh`** is installed in **`/usr/local/bin`**. The command specifies the SAS CLIST option NOSASUSER, and the SAS system options DMR and COMAMID=TCP. NOTERMINAL prevents the display of requestor windows in the server session. The parenthesis enclose the SAS options. In addition, the two single quotes around the SAS options are required.

The shell script interprets the parameters that are received from the spawner and builds a TSO command that starts a SAS session.

The following shell script parses a command and interprets environment variables to build a TSO command to start SAS. This command is executed using the USS **`/bin/tso`** command. In this example, you *must* change the values of &prefix to the high-level qualifier of your CLIST library that contains the TSO command to start SAS.

Example Code 9.1 Shell Script to Invoke SAS

```
#!/bin/sh

#
#  Initialize SAS start-up command...
#

cmd="/bin/tso -t EX '&prefix.CLIST(SAS)' '"

#
#  Construct CLIST parameters from command arguments
#
```

```
for arg in "$@"; do
   cmd="$cmd$arg "
done

#
#  Construct CLIST parameters from environment variables
#

if [ -n "$INHERIT" ] ; then
   inherit="INHERIT($INHERIT)"
fi
if [ -n "$NETENCRALG" ] ; then
   netencralg="NETENCRALG($NETENCRALG)"
fi
if [ -n "$SASDAEMONPORT" ] ; then
   sasdaemonport="SASDAEMONPORT($SASDAEMONPORT)"
fi
if [ -n "$SASCLIENTPORT" ] ; then
   sasclientport="SASCLIENTPORT($SASCLIENTPORT)"
fi
# if [ -n "$TCPDFILE" ] ; then
#    tcpdebug="TCPDEBUG(62)"
# fi

cmd="$cmd $sasdaemonport $sasclientport $inherit $netencralg'"

#
#  Set additional environment variables...
#  SYSPROC specifies the data set containing the SAS CLIST
#

export SYSPROC=&prefix.CLIST
export STEPLIB=

#
#  Start SAS
#
exec $cmd
```

Ending the z/OS Spawner

To stop the spawner, enter the following system command:

```
STOP SPAWNER
```

CHAPTER
10

UNIX Spawner

UNIX Spawner Requirements **127**
Location of the UNIX Spawner **127**
Starting a UNIX Spawner **127**
Starting the UNIX Spawner **127**
Ending the UNIX Spawner **129**

UNIX Spawner Requirements

Location of the UNIX Spawner

The UNIX spawner is located in the directory !*sasroot*/**`utilities/bin`**.

Starting a UNIX Spawner

If connecting to a UNIX server via a spawner, SAS/CONNECT uses the default authentication mechanism to verify the user ID and to verify that the password is correct for the specified user ID.

Starting the UNIX Spawner

The syntax for the command to start the UNIX spawner follows:

sastcpd <*options*>

options can be any of the following.

-nocleartext
prevents signons from clients that do not support user ID and password encryption. This option prevents clients that are running “older” releases (prior to 6.09E and 6.11 TS040, which do not support user ID and password encryption) from signing on to the spawner program. However, the default permits both encrypted and clear-text user IDs and passwords.

-noinheritance
disables socket inheritance. Socket inheritance allows SAS/CONNECT servers to use the socket connection that is established between the SAS/CONNECT client

and the spawner. Socket inheritance saves resources and is easier to configure when clients connect to a server that is within a firewall. Socket inheritance is enabled by default.

-noscript
: prevents signon from clients that use scripts, and allows signon only from clients that do not use scripts.

  This option requires that the client specify a user ID and a password during signon. For details, see the SIGNON statement in the *SAS/CONNECT User's Guide*.

  -noscript is useful if you want to limit SAS start-up commands to the use of the -sascmd option. Specifying -noscript restricts clients from specifying additional options in SAS start-up commands or script files. If -noscript is used, -sascmd must also be used.

-omrconfigfile *fully-qualified-path*
: specifies a fully-qualified path to the configuration file in XML format that contains the information necessary to connect to a SAS Metadata Server. For details about creating the configuration file, see the Base SAS Help for the Metadata Server Connections window.

  If -omrconfigfile is used, -sasspawnercn must also be used.

-sascmd "*command*"
: specifies the SAS command or a command file that is specific to the UNIX operating environment that starts a SAS session when you signon without a script. If the client does not specify a script file at sign on, the -sascmd option must be specified when starting the spawner.

  An example of a script for starting SAS for UNIX follows.

```
#!/bin/ksh
#---------------------------------
# mystartup
#---------------------------------
. ~/.profile
sas -dmr -noterminal -nosyntaxcheck -device grlink -comamid tcp
#-----------------------------
```

-sasspawnercn "*CONNECT-spawner-object-name*"
: specifies the name of the CONNECT spawner object in the SAS Metadata Repository. A name that includes one or more spaces must be enclosed in quotation marks. For details about generating a CONNECT spawner definition for the SAS Metadata Server, see the help for the SAS/CONNECT Spawner server type in the Server Manager of SAS Management Console.

  If -sasspawnercn is used, -omrconfigfile must also be used.

-service *service-name*
: specifies the name of the service that the UNIX spawner program uses to listen for incoming requests. This value is identical to the *service* value in the REMOTE= option that the user specifies at the client at sign on. Because there is no default, you must specify this value.

  The service name must also be defined in the /etc/services file at both the client and the server. For details, see Chapter 13, "TCP/IP SERVICES File," on page 145.

-shell
: enables the SAS session that is invoked by the spawner program to create a shell, which is required in order for the server to execute commands or to run X commands.

encryption-options
The encryption options that you can set depend on which security service is used. For details, see Chapter 12, “Encryption Options,” on page 137.

For an example of how to start the UNIX spawner, see “Scripted Signon to a UNIX Spawner” on page 115.

Ending the UNIX Spawner

To end the spawner, enter the interrupt signal (which, usually, is CTRL-C). If the UNIX spawner is running in the background, kill its process.

CHAPTER 11 Windows Spawner

Windows Spawner Requirements **131**
Location of the Windows Spawner **131**
Windows Security **131**
Starting the Windows Spawner **131**
Ending the Windows Spawner **135**

Windows Spawner Requirements

Location of the Windows Spawner

The default location of the Windows spawner in the Windows operating environment is !sasroot.

Windows Security

CAUTION:
To run a spawner secured by specifying the -SECURITY option, the person who runs the spawner must have the following user rights in the Windows domain:

act as part of the operating environment

bypass traverse checking (the default is everyone)

increase quotas

replace a process level token

log on locally (the default is everyone)

The client that is connecting to the spawner at signon needs only the Windows domain user right “log on as a batch job.” △

Starting the Windows Spawner

The syntax for the command to start the Windows spawner follows.

SPAWNER <*options*>

options can be any of the following:

-AUTHSERVER *domain-or-server*
: specifies the location of the user authentication database. You can specify the name of either a Windows domain or a Windows server where the database resides.

Instead of specifying a single domain in the -AUTHSERVER option, you can bypass this option and specify the domain name in the form *domain\user-ID* when you provide your user ID to the Windows environment. For example,

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

The domain name APEX identifies the location of the user authentication database. The user ID BASS and the password TIME2GO are verified against the user ID-and-password database of the specified domain.

-DELETE
: calls the Service Control Manager to remove the SAS Job Spawner Windows service, which was previously installed and started by using the -INSTALL option. If multiple instances of the spawner service are running, you can specify which instance to delete by using the -INSTANCE option. You do not need to specify the -INSTANCE option to reference the first instance of the spawner.

If you used the -NAME option with the -INSTALL option to install a spawner, you can use the -NAME option with the -DELETE option to identify the spawner to be deleted.

```
spawner -delete -name "Glenn's spawner"
```

-HELP
: prints a list of valid parameters.

-INSTALL
: causes an instance of a spawner to be installed as a Windows service. Each spawner instance is assigned a name, by default, in the following form:

```
SAS Job Spawner #xx
```

xx can range from 2 to 99. For example, if three instances of the spawner are installed, they are named, by default:

- SAS Job Spawner
- SAS Job Spawner #2
- SAS Job Spawner #3.

After the spawner is installed, unless the -NOSECURITY option is specified, the spawner will run secured.

You can install each instance of the spawner by using the following command:

```
C:\SAS> SPAWNER -install
```

Instead of accepting a default name for a spawner service, you can assign a specific name to a spawner service by using the -NAME option.

You can start the spawner service by using either the NET START command or the services applet, or by rebooting the machine that the spawner runs on.

-INSTALLDEPENDENCIES "*service- name*"
: specifies the Windows service that must be started before the spawner service starts.

-INSTANCE *xx*
: identifies the instance of the spawner service to be deleted by using the -DELETE option, where *xx* represents values from 2 to 99. If only the first instance of a spawner service (named SAS Job Spawner) is running and you want to delete it,

you do not need to specify the instance. If you want to delete any instance other than the first one that was installed, you can either specify the instance or use -NAME to specify the spawner.

For example, if three instances of the spawner are running and you want to delete only the second instance, use the following command:

```
spawner -delete  -instance 2
```

To delete a specific spawner that was installed by using the -NAME option, use the following command:

```
spawner -delete -name "Glenn's spawner"
```

-NAME *spawner-service-name*

specifies the name that you assign to the spawner that is installed and started as a service. A specified name overrides the default name that is automatically assigned when the -INSTALL option is used.

A specified spawner name cannot exceed 80 alphanumeric characters. A name string that includes one or more spaces must be enclosed in quotation marks.

The following example shows how to install an explicitly named spawner as a service:

```
spawner -install -name "Glenn's spawner"
```

The following example shows how to start an explicitly named Windows spawner by using the NET START command:

```
net start "Glenn's spawner"
```

-NOCLEARTEXT

prevents signons from clients that do not support user ID and password encryption. This option prevents clients that are running "older" releases (prior to 6.09E and 6.11 TS040, which do not support user ID and password encryption) from signing on to the spawner program. However, the default permits both encrypted and clear-text user IDs and passwords.

-NOINHERITANCE

disables socket inheritance. Socket inheritance allows SAS/CONNECT servers to use the socket connection that is established between the SAS/CONNECT client and the spawner. Socket inheritance saves resources and is easier to configure when clients connect to a server that is within a firewall. Socket inheritance is enabled by default.

-NOSCRIPT

prevents signon from clients that use scripts, and allows signon only from clients that do not use scripts.

-NOSCRIPT can be useful if you want to limit SAS start-up commands to the use of the -SASCMD option. Specifying -NOSCRIPT restricts clients from specifying additional options in SAS start-up commands or script files. If -NOSCRIPT is used, -SASCMD must also be used.

-OMRCONFIGFILE "*fully-qualified-path*"

specifies a fully-qualified path to the configuration file in XML format that contains the information necessary to connect to a SAS Metadata Server. A path that includes one or more spaces must be enclosed in quotation marks. For details about creating the configuration file, see the Base SAS Help for the Metadata Server Connections window.

If -OMRCONFIGFILE is used, -SASSPAWNERCN must also be used.

-SASCMD

specifies the SAS command or a command file that invokes SAS when a client attempts to connect to a server.

Use the -SASCMD option in order to

- □ invoke SAS from a directory that is not the default location
- □ specify different SAS start-up command options
- □ execute other statements before invoking SAS.

The following options are supplied by default when you invoke SAS:

```
-DMR -COMAMID access-method -NOLOGO -ICON -NOTERMINAL
```

An alternate file that can be invoked is a batch file, which is signified by the .BAT extension. An example of a batch file follows.

```
cd \sas
sas.exe %*
```

The first line changes to the directory where the SAS executable is stored. The second line starts SAS. Add options as needed at this SAS start-up command.

-SASSPAWNERCN *"CONNECT-spawner-object-name"*

specifies the name of the CONNECT spawner object to use in the SAS Metadata Repository. A name that includes one or more spaces must be enclosed in quotation marks. For details about generating a CONNECT spawner definition for the SAS Metadata Server, see the help for the SAS/CONNECT Spawner server type in the Server Manager of SAS Management Console.

If -SASSPAWNERCN is used, -OMRCONFIGFILE must also be used.

-SECURITY | -NOSECURITY

-SECURITY specifies that clients supply their own unique user IDs and passwords in order to connect to a spawner. -NOSECURITY specifies that all server sessions will be started by using a common user ID and password.

If the spawner is installed as a service (-INSTALL is specified), security is assumed by default.

The person who installs the spawner must have the following user rights in the Windows domain:

act as part of the operating environment

bypass traverse checking (the default is everyone)

increase quotas

replace a process level token

log on locally (the default is everyone).

All users who connect to the spawner must have the Windows domain user right "log on as a batch job".

-SERVICE *port-number* | *service-name*

specifies an alternate port that the spawner uses to listen for incoming requests for connection. The default is the Telnet port.

SERVUSER=*user-ID*
SERVPASS=*password*

are used if the spawner is installed as a service (-INSTALL is specified). However, if the spawner is installed as a service and the SSL encryption service is used (-NETENCRYPTALGORITHM=SSL is specified), SERVUSER= and SERVERPASS= must be specified. For details about SSL, see Chapter 12, "Encryption Options," on page 137.

In order to obtain a digital certificate from a certificate store, you must specify SERVUSER= and SERVPASS=, which define the user ID and password to be used to start the spawner service.

Specify both the SERVUSER= and the SERVPASS= options.

USERID=*user-ID*
PASSWORD=*password*

You must specify USERID= and PASSWORD= if the spawner is installed as a service (-INSTALL is specified) and the spawner explicitly runs unsecured (-NOSECURITY is specified).

Because the spawner is running unsecured, clients do not have their own identities authenticated. Instead, all clients that connect to a spawner will use a common user ID and password.

Specify both the USERID= and PASSWORD= options.

encryption-options

The encryption options that you can set depend on which security service is used. For details, see Chapter 12, "Encryption Options," on page 137.

For an example of starting the Windows spawner as a service, see "Scriptless Signon to a Windows Spawner That Runs as a Service" on page 116.

Ending the Windows Spawner

To end the spawner that runs in a DOS window, type CTRL-C or double-click in the upper-left corner of the window that runs the spawner.

If the Windows Spawner runs as a service, you can end the spawner by using one of the following:

- the Windows Services panel
- Type **`net stop "sas job spawner"`**
- the command **`spawner -delete`**.

CHAPTER 12 Encryption Options

Security Services **137**
Security Service Requirements **138**
  TCP/IP Is the Only Access Method Supported **138**
  Operating Environments Supported **138**
Encryption Options **138**
  Overview **138**
SAS/SECURE Options **139**
SAS Proprietary Options **141**
SSL Options **141**

Security Services

To encrypt client/server data transfers across a network, you can use any of the following security services.

SASproprietary
: a fixed encoding algorithm is included with Base SAS software and is available in all supported operating environments. It requires no additional SAS product licenses. The SAS proprietary algorithm is strong enough to protect your data from casual viewing. However, a determined hacker can break this encoding.

SAS/SECURE
: an add-on product that provides additional encryption algorithms besides the SAS proprietary algorithm. The encryption algorithms that are supported are: RC2, RC4, DES, and TripleDES. SAS/SECURE software requires a license and must be installed on each machine that runs a SAS/CONNECT server or a SAS/SHARE server that will use the encryption algorithms.

SSL
: SSL is an abbreviation for Secure Sockets Layer, which is a protocol that provides network security and privacy. Developed by Netscape Communications, SSL uses encryption algorithms that include RC2, RC4, DES, tripleDES, and MD5. Not limited to providing only encryption services, SSL can also perform client and server authentication, and it uses message authentication codes. SSL is supported by both Netscape Navigator and Internet Explorer. Many Web sites use the protocol to protect confidential user information, such as credit card numbers. By convention, URLs that require an SSL connection begin with https: instead of http:. The SSL protocol is application independent and allows protocols such as HTTP, FTP, and Telnet to be transparently layered above it. SSL is optimized for HTTP.

Security Service Requirements

TCP/IP Is the Only Access Method Supported

TCP/IP is the only communications access method that can be used to connect a client to a server via a spawner.

Operating Environments Supported

The following table shows which security services support which operating environments.

Table 12.1 Security Services Supported by Operating Environment

Operating Environments	Security Service		
	SASProprietary	SAS/SECURE	SSL
OpenVMS Alpha	X		
z/OS	X	X	
UNIX	X	X*	X
Windows	X	X	X

* SAS/SECURE supports the following UNIX operating environments only:

- Compaq Tru64 UNIX
- HP UX on Itanium 64-bit platform
- HP UX on a 64-bit platform
- Linux for Intel Architecture on 32-bit platform
- Solaris on a 64-bit platform

Encryption Options

Overview

The encryption options that you can specify depend on which security service is used. The following table lists the options and indicates which service supports the option.

Table 12.2 Encryption Options

Encryption Options	Security Services			
	SAS Proprietary	SAS/ SECURE	SSL	
			UNIX	Windows
-NETENCRYPT	X	X	X	X
-NETENCRYPTALGORITHM	X	X	X	X
-NETENCRYPTKEYLEN		X		

Encryption Options	Security Services			
	SAS Proprietary	SAS/ SECURE	SSL	
			UNIX	Windows
-SSLCLIENTAUTH			X	X
-SSLCRLCHECK			X	X
-SSLCALISTLOC			X	
-SSLCERTLOC			X	
-SSLCRLLOC			X	
-SSLPVTKEYLOC			X	
-SSLPVTKEYPASS			X	
-SSLCERTISS				X
-SSLCERTSERIAL				X
-SSLCERTSUBJ				X

SAS/SECURE Options

The following options can be used with SAS/SECURE.

-NETENCRYPT | -NONETENCRYPT
-NETENCRYPT specifies that encryption *is* required. -NONETENCRYPT specifies that encryption is *not* required, but is optional. If -NETENCRYPT is *not* specified, or if neither -NETENCRYPT nor -NONETENCRYPT is explicitly specified, -NONETENCRYPT is the default.

If the -NETENCRYPTALGORITHM option is set and if both the client and the server are capable of encryption, the default for this option is that encryption is used. If encryption algorithms are specified but either the client or the server is incapable of encryption, then encryption is not performed.

There is an interaction between -NETENCRYPT and -NETENCRYPTALGORITHM. For details, see information about -NETENCRYPTALGORITHM.

Encryption might *not* be supported at the client or at the server if

- A release of SAS (prior to Version 8) that does not support encryption is being run.
- Your site (the client or the machine where the spawner is running) does not have security software installed.
- You specified incompatible encryption algorithms in SAS sessions on the client and the server.
- You do not have a cryptographic service provider installed in your Windows operating environment.
- You did not specify SASPROPRIETARY, which is the only algorithm that is supported in the OpenVMS Alpha operating environment.

-NETENCRYPTALGORITHM RC2 | RC4 | DES | TripleDES
Set this option at the server to specify one or more encryption algorithms to use in a SAS/CONNECT session. Setting this option at the client is optional. The client and the server must share an encryption algorithm in common. There is an

interaction between either NETENCRYPT or NONETENCRYPT and NETENCRYPTALGORITHM. Possible sign-on interaction results follow.

Table 12.3 Sign-on Interaction Results between Options

Server Settings	Client Settings	Sign-On Results
-NONETENCRYPT -NETENCRYPTALGORITHM=*algorithm*	No settings	If the client is capable of encryption, an encrypted signon occurs. Otherwise, the signon occurs without encryption.
-NETENCRYPT -NETENCRYPTALGORITHM=*algorithm*	No settings	If the client is capable of encryption, an encrypted signon occurs. Otherwise, signon fails.
No settings	-NONETENCRYPT -NETENCRYPTALGORITHM=*algorithm*	Signon (without encryption) occurs
No settings	-NETENCRYPT -NETENCRYPTALGORITM=*algorithm*	Signon fails
-NETENCRYPT or -NONETENCRYPT -NETENCRYPTALGORITHM=*algorithm-a*	-NETENCRYPTALGORITHM=*algorithm-b*	Regardless of whether -NETENCRYPT or -NONETENCRYPT is specified, signon fails

If you specify multiple encryption algorithms, repeat the -NETENCRYPTALGORITHM option for each algorithm name.

-NETENCRYPTKEYLEN *n*

Set this option in the SAS session on either the client or the server. It specifies the key length to be used by the encryption algorithm.

Valid values for this option are

128	specifies 128-bit RC2 and RC4 algorithms.
40	specifies 40-bit RC2 and RC4 algorithms.
0	no value is set. This is the default.

If you require extra security, set the -NETENCRYPTKEYLEN option to 128. If you prefer to save CPU, set the -NETENCRYPTKEYLEN option to 40.

By default, if you try to connect a machine that is capable of only a 40-bit key length to a machine that is capable of both a 40-bit and a 128-bit key length, the connection is made using the lesser key length. If both machines are capable of 128-bit key lengths, a 128-bit key length is used.

SAS Proprietary Options

The following options apply to SASproprietary.

-NETENCRYPT | -NONETENCRYPT

-NETENCRYPT specifies that encryption *is* required. -NONETENCRYPT specifies that encryption is *not* required, but is optional. If -NETENCRYPT is *not* specified, or if neither -NETENCRYPT nor -NONETENCRYPT is explicitly specified, -NONETENCRYPT is the default.

If the -NETENCRYPTALGORITHM option is set and if both the client and the server are capable of encryption, the default for this option is that encryption is used. If encryption algorithms are specified but either the client or the server is incapable of encryption, then encryption is not performed.

There is an interaction between -NETENCRYPT and -NETENCRYPTALGORITHM. For details, see information about -NETENCRYPTALGORITHM.

Encryption might *not* be supported at the client or at the server if

- □ A release of SAS (prior to Version 8) that does not support encryption is being run.
- □ Your site (the client or the machine where the spawner is running) does not have security software installed.
- □ You specified incompatible encryption algorithms in SAS sessions on the client and the server.
- □ You do not have a cryptographic service provider installed in your Windows operating environment.
- □ You did not specify SASPROPRIETARY, which is the only algorithm that is supported in the OpenVMS Alpha operating environment.

-NETENCRYPTALGORITHM SASPROPRIETARY

Set this option at the server to specify one or more encryption algorithms to use in a SAS/CONNECT session. Setting this option at the client is optional. However, the client and the server must share an encryption algorithm in common. There is an interaction between either -NETENCRYPT or -NONETENCRYPT and -NETENCRYPTALGORITHM. See Table 12.2 on page 138 for possible sign-on interaction results.

SSL Options

The following options apply to SSL.

Note: SSL must be installed and configured before the SSL options can be used. For details, see SSL setup in the *SAS/CONNECT User's Guide*. △

-NETENCRYPT | -NONETENCRYPT

-NETENCRYPT specifies that encryption *is* required. -NONETENCRYPT specifies that encryption is *not* required, but is optional. If -NETENCRYPT is *not* specified, or if neither -NETENCRYPT nor -NONETENCRYPT is explicitly specified, -NONETENCRYPT is the default.

If the -NETENCRYPTALGORITHM option is set and if both the client and the server are capable of encryption, the default for this option is that encryption is used. If encryption algorithms are specified but either the client or the server is incapable of encryption, then encryption is not performed.

There is an interaction between -NETENCRYPT and -NETENCRYPTALGORITHM. For details, see information about -NETENCRYPTALGORITHM.

Encryption might *not* be supported at the client or at the server if

- □ A release of SAS (prior to Version 8) that does not support encryption is being run.
- □ Your site (the client or the machine where the spawner is running) does not have security software installed.
- □ You specified incompatible encryption algorithms in SAS sessions on the client and the server.
- □ You do not have a cryptographic service provider installed in your Windows operating environment.
- □ You did not specify SASPROPRIETARY, which is the only algorithm that is supported in the OpenVMS Alpha operating environment.

-NETENCRYPTALGORITHM SSL

Set this option at the server to specify one or more encryption algorithms to use in a SAS/CONNECT session. Setting this option at the client is optional. However, the client and the server must share an encryption algorithm in common. There is an interaction between either -NETENCRYPT or -NONETENCRYPT and -NETENCRYPTALGORITHM. See Table 12.2 on page 138 for possible sign-on interaction results.

-SSLCLIENTAUTH | -NOSSLCLIENTAUTH

Use this option to specify whether the server should require SSL to provide client authentication. Server authentication is always performed, but this option enables a user to control client authentication. This option is meaningful only when used on a server.

Valid only if using the SSL security service, this option is valid in UNIX and Windows operating environments.

-SSLCRLCHECK | -NOSSLCRLCHECK

Use this option to specify whether Certificate Revocation Lists (CRLs) are checked when a digital certificate is validated.

Valid only if using the SSL security service, this option is valid in UNIX and Windows operating environments.

-SSLCALISTLOC *file-path*

specifies the location of a file that contains the digital certificates for the trusted certificate authorities (CA). The value must be a name of a file that contains a list of CA digital certificates that are to be trusted. This option is required at the client. This option is required at the server only if -SSLCLIENTAUTH is also enabled.

Valid only if using the SSL security service, this option is valid in UNIX operating environments only.

-SSLCERTLOC *file-path*

specifies the name of a file that contains a digital certificate. This option is required on a server. This option is required on a client only when client authentication is being performed.

Valid only if using the SSL security service, this option is valid in UNIX operating environments only.

-SSLCRLLOC *file-path*

specifies the location of a Certificate Revocation List (CRL). Use this option only when the -SSLCRLCHECK option is enabled.

Valid only if using the SSL security service, this option is valid in UNIX operating environments only.

-SSLPVTKEYLOC *file-path*

specifies the name of the file that contains the private key that corresponds to the digital certificate that was specified by -SSLCERTLOC. Use this option only when the -SSLCERTLOC option is specified.

Valid only if using the SSL security service, this option is used in UNIX operating environments only.

-SSLPVTKEYPASS *password*

specifies the password that SSL should use to decrypt the private key that is stored in the file that is specified by -SSLPVTKEYLOC.

This option is needed only when the private key is encrypted.

Valid only if using the SSL security service, this option is used in UNIX operating environments only.

-SSLCERTISS *issuer-of-certificate*

specifies the name of the issuer of the digital certificate that SSL should use. This option is used with -SSLCERTSERIAL to uniquely identify a digital certificate from the Microsoft certificate store.

Valid only if using the SSL security service, this option is valid in Windows operating environments only.

-SSLCERTSERIAL *serial-number*

specifies the serial number of the digital certificate that SSL should use. This option is used with -SSLCERTISS to uniquely identify a digital certificate from the Microsoft certificate store.

Valid only if using the SSL security service, this option is used in Windows operating environments only.

-SSLCERTSUBJ *subject-name*

specifies the subject name in the digital certificate that can be used to search for the certificate in the Microsoft certificate store.

Valid only if using the SSL security service, this option is used in Windows operating environments only.

CHAPTER 13 TCP/IP SERVICES File

Configuring the SERVICES File **145**
Services That Require an Entry in the SERVICES File **145**
Example SERVICES File **145**
Explanation of Fields **146**

Configuring the SERVICES File

Services That Require an Entry in the SERVICES File

You must have an entry in the SERVICES file for each of the following services, as needed:

- □ Telnet service
- □ Spawner port
- □ SAS/SHARE server ID or port
- □ Firewall machine ports
- □ Dedicated TCP/IP port service that is used for MP CONNECT piping.

Note: If you have access to a UNIX operating environment, see the **services** (4) manual page for more information about this file. △

The location of the SERVICES file depends on the operating environment. For example, the UNIX services file is located at **/etc/services**.

Example SERVICES File

Here is an example excerpt from a SERVICES file.

```
# The form for each entry is:
# <official service name>  <port number/protocol name>  <alias name>
# <comments>
#
# Port Services

telnet             23/tcp               # Telnet service
spawnport          4016/tcp             # UNIX spawner port
mktserve           4017/tcp             # Server for Marketing & Sales
```

```
server1           5011/tcp            # SAS/SHARE server 1
sassrv2           5012/tcp            # SAS/SHARE server 2
firewall          5010/tcp            # Firewall machine port
pipe1             5020/tcp            # MP Connect pipe
sea               5021/tcp    biscuit # SAS/SHARE server 3
```

Explanation of Fields

An explanation of each field follows.

official service name
: specifies the name of the service. Service names must meet the criteria for a valid SAS name. (For details about SAS naming rules, see *SAS Language Reference: Concepts*.) For example, you can create a service named SPAWNER for the UNIX spawner program. You will need the Telnet service when signing on to any server that does not use a PC or a UNIX spawner program.

  You will also use the service name as the value for the REMOTE= option or in the SIGNON statement to perform a server sign on.

port number
: is a unique number that is associated with the service name. Each reference to that service in other node SERVICES files must match the service's port number exactly. Port numbers 0 through 1023 are reserved for system use. Port numbers that are greater than 1023 are available for user-created services.

protocol name
: identifies the protocol. **`udp`** and **`tcp`** are examples of protocol names.

alias name
: is an optional synonym for the service. Alias names can be application- or user-dependent. For example, one application can refer to the server as **`sea`** and another application refers to the same server as **`biscuit`**.

  Note: Each client and server must configure the alias in its SERVICES file before the alias can be successfully used. For example, **`sea`** and **`biscuit`** must be configured in the SERVICES file of each client and server that will use the alias. △

comments
: describe the service.

P A R T 7

Configuring SAS/CONNECT for Use with Firewalls

Chapter 14 . . . . **Configuring SAS/CONNECT for Use with a Firewall** *149*

CHAPTER 14

Configuring SAS/CONNECT for Use with a Firewall

Definitions **149**
Firewall **149**
Socket Inheritance **149**
Requirements for Using a Firewall **149**
Firewall Configuration Example **150**
Firewall Configuration **150**
External Windows Client Connecting through a Firewall **150**
External UNIX Client Connecting through a Firewall **151**

Definitions

Firewall

A *firewall* is a controlled gateway between two networks. Firewalls are used to provide a secure connection between an internal network and the Internet.

Web servers and other network applications can also use firewalls to limit access to sensitive data. SAS/CONNECT permits TCP/IP connections between clients outside a firewall to spawners that run on servers inside a firewall via socket inheritance.

Socket Inheritance

Socket inheritance allows the server session to inherit the socket that the spawner uses to communicate with the client session. The socket is then used for subsequent communications between the client and the server session. Socket inheritance is significant because a single port can be used for starting an unlimited number of server sessions.

Prior to this innovation, the spawner monopolized an entire port, listening for connections, and a separate port was opened for each client that connected to a server by using a spawner. Socket inheritance limits the number of ports that are used for connections through a firewall, which makes the firewall configuration more secure.

Requirements for Using a Firewall

- The external clients and the servers within the firewall must be running SAS Release 6.12 TS065 or later.

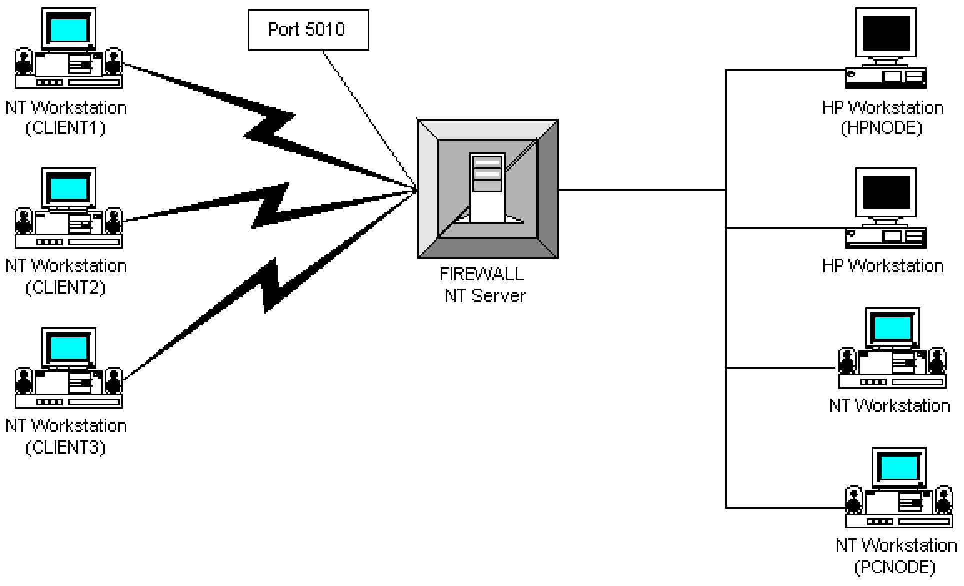
- □ The TCP/IP communications access method must be used for establishing a network connection between clients and servers.
- □ Firewall software must be installed on the server that maintains the separation between the internal network and the Internet.
- □ A port must be defined on the firewall server to be used as a gateway between external clients and the internal network. The firewall software must route the firewall server port to the pre-defined server port.
- □ A spawner must be running on a server inside the firewall. For complete details about the spawner program, see Chapter 7, "SAS/CONNECT Spawners," on page 113.
- □ Each port number can be configured into the respective **`services`** file.

Firewall Configuration Example

Firewall Configuration

The following example configuration includes three Windows NT client workstations that are external to the organization, a firewall that runs on a Windows NT server, and an internal local area network (LAN) that contains multiple Windows NT and UNIX workstations.

Figure 14.1 Firewall Configuration Example



External Windows Client Connecting through a Firewall

Port 5010 on the Windows NT server that runs the firewall software has been defined as the single port through which all external clients gain access to servers on the internal network. On the firewall machine, port 5010 must be routed to port 5080 on the Windows NT server PCNODE that runs a spawner on the internal LAN.

```
spawner -inheritance -service 5080
options comamid=tcp;
signon firewall.5010
```

The spawner is initialized to listen on port 5080 on PCNODE. An external client uses TCP/IP to sign on to a server that is located inside the firewall by using port 5010 on the Windows NT server FIREWALL. The firewall software routes traffic from port 5010 to port 5080 on PCNODE.

External UNIX Client Connecting through a Firewall

Port 5010 on the UNIX server that runs the firewall software has been defined as the single port through which all external clients gain access to servers on the internal network. On the firewall machine, port 5010 must be routed to port 5080 on the UNIX operating environment HPNODE that runs a spawner on the internal LAN.

```
sastcpd -inheritance -service 5080
options comamid=tcp;
signon firewall.5010
```

The spawner is initialized to listen on port 5080 on HPNODE. An external client uses TCP/IP to sign on to a server that is located inside the firewall by using port 5010 on the UNIX server FIREWALL. The firewall software routes traffic from port 5010 to port 5080 on HPNODE.

P A R T 8

SAS/CONNECT Scripts

Chapter ***15*** . . . . **Sign-On Scripts** *155*

CHAPTER

15

Sign-On Scripts

Script Rules **155**
  Syntax **155**
  Specifying Time **156**
  Using the WAITFOR and TYPE Statements **156**
Sample Scripts **156**
  TCPUNIX.SCR Script **156**
  TCPWIN.SCR Script **161**
  TCPMVS.SCR Script **164**
  TCPTSO9.SCR Script **167**

Script Rules

Syntax

For details about writing scripts, see Sign-on script files in the *SAS/CONNECT User's Guide*.

- Like other SAS statements, all script statements must end with a semicolon(;).
- Script statements have a free format, which means that there are no spacing or indention requirements. A statement can be contained within a single line or it can span several lines. Statement keywords are not case sensitive.
- Enclose case-sensitive text strings in quotation marks. For example, if your script defines a text string in a WAITFOR statement, be sure that the uppercase and lowercase characters in the text string exactly match the text string from the server.
- You can use either single or double quotation marks to quote a string, such as a server command, in a script statement. The rules that you use to embed quotation marks in a SAS statement and to embed quotation marks in a script statement are the same.
- Any script statement can include a label specification. The label must be a valid SAS name, with a maximum of eight characters. The first character must be an alphabetic character or an underscore. A label must be followed immediately by a colon (:) and it can be defined only once in the script.

Specifying Time

Some script statements specify time in seconds, as follows:

n SECONDS

where *n* can be any number, including decimal fractions. SECOND is an alias for SECONDS. Examples of valid time specifications follow:

0 SECONDS

0.25 SECONDS

1 SECOND

3.14 SECONDS

Using the WAITFOR and TYPE Statements

When writing a script or modifying an existing script, pay special attention to the WAITFOR and the TYPE statements. To ensure that the script recognizes the expected prompt during each stage of sign on, specify the exact sequence of prompts and responses for the server.

You might test the sequence by experimenting at the server by going through the process that you want to capture in the WAITFOR and the TYPE statements. For each display at the server, choose a word from that display for the WAITFOR statement. Capture in a TYPE statement the information that you type in response to a display. Be sure to note all carriage returns or other special keys.

For example, if TSO is the server and you need to use a TYPE statement in a sign-on script whose length is greater than 80 characters, divide the TYPE statement into two or more TYPE statements.

To divide the TYPE statement, insert a hyphen (-) at the division point. The remote TSO machine interprets the hyphen as the continuation of the TYPE statement from the previous line.

For example, consider the following TYPE statement:

```
type "sas options ('dmr comamid=tcp')" enter;
```

To divide the statements, change it to:

```
type "sas options ('dmr comamid=-" enter;
type "tcp')" enter;
```

Note: Do not insert spaces around the hyphen. △

Sample Scripts

TCPUNIX.SCR Script

The following script connects a client to a UNIX server by using the TCP/IP access method.

```
/* trace on; */
/* echo  on; */
/*------------------------------------------------------------------*/
```

```
/*--           Copyright (C) 1996 by SAS Institute Inc., Cary NC  --*/
/*--                                                              --*/
/*-- name:      tcpunix.scr                                       --*/
/*--                                                              --*/
/*-- purpose:   SAS/CONNECT SIGNON/SIGNOFF script for connecting  --*/
/*--            to any UNIX host by means of the TCP/IP access    --*/
/*--            method                                            --*/
/*--                                                              --*/
/*-- notes:   1. This script might need modifications that account --*/
/*--             for the local flavor of your UNIX environment.   --*/
/*--             The logon process should mimic the tasks that    --*/
/*--             you perform when "telnet"-ing to the same        --*/
/*--             UNIX host. If you are connecting to a spawner    --*/
/*--             that is running in your UNIX environment, this   --*/
/*--             script should need few or no modifications.      --*/
/*--                                                              --*/
/*--          2. You must have specified OPTIONS COMAMID=TCP      --*/
/*--             in the local SAS session before using the SIGNON  --*/
/*--             statement.                                       --*/
/*--                                                              --*/
/*-- assumes: 1. The command to execute SAS in your remote (UNIX)  --*/
/*--             environment is "sas". If this is incorrect       --*/
/*--             for your site, change the contents of the line   --*/
/*--             that contains:                                   --*/
/*--             type 'sas ...                                    --*/
/*--                                                              --*/
/*-- support:   SAS Institute staff                               --*/
/*--                                                              --*/
/*----------------------------------------------------------------*/


   /*--------------------------------------------------------------*/
   /*-- if you are connecting to DEC ULTRIX, and the remote       --*/
   /*-- machine does not run the DECnet connection/gateway        --*/
   /*-- software, logins by means of SAS/CONNECT will appear to   --*/
   /*-- hang. This is due to the ULTRIX "/etc/telnetd" server     --*/
   /*-- treating a DONT ECHO request for both input and output    --*/
   /*-- streams.                                                  --*/
   /*--                                                           --*/
   /*-- The DEBUG statement causes the SAS TCP/IP access method   --*/
   /*-- not to reply to the ECHO request, keeping the DEC telnetd  --*/
   /*-- server happy.                                             --*/
   /*--                                                           --*/
   /*-- Uncomment the DEBUG statement, if the logon appears to hang--*/
   /*--------------------------------------------------------------*/
   /* debug '00001000'; */

   /*--------------------------------------------------------------*/
   /*-- If you are connecting to INTEL ABI, you need to uncomment  --*/
   /*-- the following DEBUG statement.  This DEBUG statement      --*/
   /*-- allows SAS/CONNECT to set the terminal type to TTY during  --*/
   /*-- the Telnet negotiations that take place during SIGNON.    --*/
   /*--------------------------------------------------------------*/
   /* debug '00004000'; */
```

```
❶ log "NOTE: Script file ’tcpunix.scr’ entered.";

    if not tcp then goto notcp;
❷ if signoff then goto signoff;

     /* --------------- TCP/IP SIGNON ---------------------------------*/

❸ waitfor ’login:’
           , ’Username:’
           , ’Scripted signon not allowed’ : noscript
           , 120 seconds: noinit;

/*----------------UNIX LOGON---------------------------------------*/
/*-- for some reason, it needs an LF to turn the line around     --*/
/*--  after the login name has been typed. (CR will not do)      --*/
/*----------------------------------------------------------------*/

❹ input ’Userid?’;
     type LF;
❺ waitfor ’Password’, 30 seconds : nolog;
     input nodisplay ’Password?’;
     type LF;

unx_log:
❻ waitfor ’Hello>’               : unxspawn /*- UNIX spawner prompt-*/
           , ’$’                /*-- a common prompt character       --*/
           , ’>’                /*-- another common prompt character --*/
           , ’%’                /*-- another common prompt character --*/
           , ’}’                /*-- another common prompt character --*/
           , ’Login incorrect’      : nouser
           , ’Enter terminal type’  : unx_term
           , ’TERM’                 : unx_term
           , 30 seconds             : timeout
           ;


   log ’NOTE: Logged onto UNIX... Starting remote SAS now.’;
   /* NOTERMINAL suppresses prompts from remote SAS session.     */
   /* NO\$SYNTAXCHECK prevents remote side from going into       */
   /* syntax checking mode when a syntax error is encountered.   */
❼ type ’sas -dmr -comamid tcp -device grlink -noterminal -no\$syntaxcheck’ LF;
❽ waitfor ’SESSION ESTABLISHED’, 90 seconds : nosas;

❾ log ’NOTE: SAS/CONNECT conversation established.’;
   stop;

❿ unxspawn:
      /* The UNIX spawner executes only a single UNIX command     */
      /* after the client logs on.  In the TYPE statement below,  */
      /* you can specify a SAS command line. You can also specify */
      /* a UNIX shell script that issues the SAS command line in  */
      /* addition to any other commands to be executed prior to   */
      /* SAS invocation.  The following is a sample start-up      */
      /* file:                                                     */
      /*#---------------------------------------------------------*/
      /*# sas_startup                                              */
      /*#---------------------------------------------------------*/
      /*#!/bin/ksh                                                 */
      /*. ~/.profile                                               */
      /*sas -dmr -noterminal -no\$syntaxcheck -device grlink      */
      /*#---------------------------------------------------------*/
      /*                                                           */
      /* If you choose to use a "startup" file, change the TYPE   */
      /* statement below to something similar to the following:   */
      /* type '/usr/local/whatever/sas_startup' LF;                */

⓫ type 'sas -dmr -comamid tcp -device grlink -noterminal ';
     type '-no\$syntaxcheck' LF;

     waitfor 'SESSION ESTABLISHED', 90 seconds : nosas;
     stop;

/*---------------- TCP/IP SIGNOFF --------------------------------------*/
signoff:
    /* If you have established your connection to UNIX by using */
    /* a UNIX spawner, you should delete or comment the         */
    /* following WAITFOR and TYPE statements.  They are not     */
    /* necessary for signing off a UNIX spawner and will        */
    /* result in slower performance of SIGNOFF.                 */
⓬ waitfor '$'
           , '>'               /*-- another common prompt character --*/
           , '%'               /*-- another common prompt character --*/
           , '}'               /*-- another common prompt character --*/
           , 30 seconds
           ;

   type    'logout' LF;
   log 'NOTE: SAS/CONNECT conversation terminated.';
   stop;

/*--------------- SUBROUTINES -----------------------------------*/

unx_term:
/*---------------------------------------------------------------*/
/*-- Some UNIX platforms want the terminal type,              --*/
/*-- so tell them this is the most basic of terminals.        --*/
/*---------------------------------------------------------------*/
   type 'tty' LF;
   goto unx_log;


/*--------------- ERROR ROUTINES --------------------------------*/


⓭ timeout:
     log 'ERROR: Timeout waiting for remote session response.';
```

```
      abort;

      nouser:
      log 'ERROR: Unrecognized userid or password.';
      abort;

   notcp:
      log 'ERROR: Incorrect communications access method.';
      log 'NOTE: You must set "OPTIONS COMAMID=TCP;" before using this';
      log '      script file.';
      abort;

   noinit:
      log 'ERROR: Did not understand remote session banner.';

   nolog:
      log 'ERROR: Did not receive userid or password prompt.';
      abort;

   nosas:
     log 'ERROR: Did not get SAS software start-up messages.';
      abort;

   noscript:
      /* This is the result of trying to sign on with a script file */
      /* to a UNIX spawner that has been invoked with the -NOSCRIPT */
      /* option.  You need to clear any script file reference and   */
      /* then re-execute SIGNON.                                    */
      log 'ERROR: Scripted signons are not allowed.';
      log 'NOTE:  Clear any script file reference and retry SIGNON.';
      abort;
```

1 The LOG statement sends the enquoted message to the log file or to the LOG window of the SAS session at the client. Although it is unnecessary to include LOG statements in your script file, the LOG statements keep the user informed about the progress of the connection.

2 The IF/THEN statement can detect whether the script was called by the SIGNON statement or the SIGNOFF statement. When you are signing off, the IF/THEN statement directs script processing to the statement labeled SIGNOFF. See step 12.

3 The WAITFOR statement awaits the login prompt from the server. If the statement does not receive the prompt within 120 seconds, it directs script processing to branch to the statement labeled NOINIT.

4 The INPUT statement displays a window with the text **`Userid?`** to allow the user to enter a server logon user ID. The TYPE statement sends a line feed to the server to enter the user ID to the server.

5 The WAITFOR statement waits for the password prompt from the server and branches to the NOLOG label if it is not received within 30 seconds. The INPUT statement that follows the WAITFOR statement displays a window in which the user enters a password.

6 The WAITFOR statement waits for one of several common UNIX prompts and branches to various error handles if a prompt is not displayed. For a connection to the UNIX spawner, the string "Hello >" is received and the control branches to the

unxspawn label in step 10. Verify that the WAITFOR statement in the script looks for the correct prompt for your site.

7 The TYPE statement invokes SAS on the server. The -DMR option is necessary to invoke a special processing mode for SAS/CONNECT. The -COMAMID option specifies the access method that is used to make the connection.

8 The message **SESSION ESTABLISHED** is displayed when a SAS session is started on the server by using the options -DMR and -COMAMID TCP. The WAITFOR statement awaits the display of the message **SESSION ESTABLISHED** to be issued by the server. If the **SESSION ESTABLISHED** response is received within 90 seconds, processing continues with the next LOG statement. If the **SESSION ESTABLISHED** response does not occur within 90 seconds, the script assumes that the remote SAS session has not started, and processing branches to the statement labeled NOSAS.

9 After the connection has been successfully established, the user must stop the rest of the script from processing. Without this STOP statement, processing continues through the remaining statements in the script.

10 This section of code is executed when you connect to a remote UNIX spawner.

11 The TYPE statement invokes SAS on the server. The -DMR option is necessary to invoke a special processing mode for SAS/CONNECT. The -COMAMID option specifies the access method that is used to make the connection.

12 This section of code is executed when the script is invoked to terminate the link. The IF statement (see step 2) sends processing to this section of the script when the script is invoked by a SIGNOFF statement. This section logs the user off the server after the user executes **LOGOFF**. Before it stops the link, the script issues a LOG statement to notify the user that the link is terminated.

13 These statements are processed only if the prompts expected in the previous steps are not received. This section of the script issues messages to the SAS log at the client and then abnormally ends the script processing as well as the SIGNON.

TCPWIN.SCR Script

The following script signs a client on and off a Windows NT or a Windows 95 server by using the TCP/IP access method.

```
/* trace on; */
/* echo  on; */
/*------------------------------------------------------------------*/
/*--            Copyright (C) 1996 by SAS Institute Inc., Cary NC  --*/
/*--                                                               --*/
/*-- name:      tcpwin.scr                                         --*/
/*--                                                               --*/
/*-- purpose:   SAS/CONNECT SIGNON/SIGNOFF script for connecting   --*/
/*--            to either a Windows 95 or a Windows NT host by     --*/
/*--            using the TCP/IP access method.                    --*/
/*--                                                               --*/
/*-- notes:   1. You must have the spawner program executing on    --*/
/*--             the remote Windows 95 or Windows NT workstation   --*/
/*--             in order for the local session to be able to      --*/
/*--             establish the connection.  If the spawner is      --*/
/*--             running on the remote node, you will receive a    --*/
/*--             message that tells you that the connection has    --*/
/*--             been refused.                                     --*/
/*--                                                               --*/
/*--          2. You must have specified OPTIONS COMAMID=TCP       --*/
```

```
/*--             in the local SAS session before using the SIGNON  --*/
/*--             command.                                          --*/
/*--                                                               --*/
/*-- assumes: 1. The command to execute SAS in your remote         --*/
/*--             (Windows 95 or Windows NT) environment is "sas".  --*/
/*--             If this is incorrect for your site, change the    --*/
/*--             contents of the line that contains:               --*/
/*--             type 'sas ...                                     --*/
/*--                                                               --*/
/*-- support:   SAS Institute staff                                --*/
/*--                                                               --*/
/*-----------------------------------------------------------------*/

❶ log "NOTE: Script file 'tcpwin.scr' entered.";

     if not tcp then goto notcp;
❷ if signoff then goto signoff;

     /* --------------- TCP/IP SIGNON ---------------------------------*/

❸ waitfor 'Username:'
         , 'Hello>'                 :  ready
         , 'access denied'          :  nouser
         , 120 seconds              :  noprompt
         ;
❹ input 'Userid?';
     type LF;
❺ waitfor 'Password:' , 120 seconds: nolog;
     input nodisplay 'Password?';
     type LF;

❻ waitfor 'Hello>'
         , 'access denied'          :  nouser
         , 120 seconds              :  timeout
         ;

  ready:
     log 'NOTE: Logged onto Windows... Starting remote SAS now.';
     /* NOTERMINAL suppresses prompts from remote SAS session.     */
     /* NO$SYNTAXCHECK prevents remote side from going into syntax */
     /* checking mode when a syntax error is encountered.          */
❼ type 'sas -dmr -comamid tcp -device grlink -noterminal -no$syntaxcheck' LF;
❽ waitfor 'SESSION ESTABLISHED', 120 seconds : nosas;

❾ log 'NOTE: SAS/CONNECT conversation established.';
     stop;


/*---------------- TCP/IP SIGNOFF -----------------------------------*/

❿ signoff:
     log 'NOTE: SAS/CONNECT conversation terminated.';
     stop;
```

```
/*--------------- SUBROUTINES -----------------------------------*/


/*--------------- ERROR ROUTINES --------------------------------*/
⓫
  notcp:
     log 'ERROR: Incorrect communications access method.';
     log 'NOTE: You must set "OPTIONS COMAMID=TCP;" before using this';
     log '      script file.';
     abort;

  noprompt:
     log 'ERROR: Did not receive userid prompt.';
     log 'NOTE:  Ensure spawner process is running on remote node.';
     abort;

  nolog:
     log 'ERROR: Did not receive password prompt.';
     abort;



  nouser:
     log 'ERROR: Unrecognized userid or password.';
     abort;

  nosas:
     log 'ERROR: Did not get SAS software startup messages.';
     abort;

  timeout:
     log 'ERROR: Timeout waiting for remote session response.';
     abort;
```

1 The LOG statement sends the enquoted message to the log file or to the LOG window of the SAS session at the client. Although it is not necessary to include LOG statements in your script file, the LOG statements keep the user informed about the progress of the connection.

2 The IF/THEN statement detects whether the script was called by the SIGNON statement or by the SIGNOFF statement. When you sign off, the IF/THEN statement directs script processing to the statement that is labeled SIGNOFF. See step 10.

3 The WAITFOR statement awaits the login prompt from the server and branches to various error handles if this prompt is not displayed.

4 The INPUT statement displays a window with the text **`Userid?`** to allow the user to enter a server logon user ID. The TYPE statement sends a line feed to the server to enter the user ID to the server.

5 The WAITFOR statement awaits the password prompt from the server and branches to the NOLOG label if it is not received within 120 seconds. The INPUT statement that follows the WAITFOR statement displays a window in which the user enters a password.

6 The WAITFOR statement awaits the "Hello > " prompt that it expects to see from the Windows spawner. If the statement does not receive the prompt within 120 seconds, it directs script processing to branch to the statement that is labeled TIMEOUT.

7 The TYPE statement invokes SAS on the server. The -DMR option is necessary to invoke a special processing mode for SAS/CONNECT. The -COMAMID option specifies the access method that is used to make the connection.

8 The message **`SESSION ESTABLISHED`** is displayed when a SAS session is started on the server by using the -DMR and -COMAMID TCP options. The WAITFOR statement awaits the display of the message **`SESSION ESTABLISHED`** to be issued by the server. If the **`SESSION ESTABLISHED`** response is received within 120 seconds, processing continues with the next LOG statement. If the **`SESSION ESTABLISHED`** response does not occur within 120 seconds, the script assumes that the remote SAS session has not started and processing branches to the statement labeled NOSAS.

9 After the connection has been successfully established, the user must stop the rest of the script from processing. Without this STOP statement, processing continues through the remaining statements in the script.

10 This section of code is executed when the script is invoked to terminate the link. The IF statement (see step 2) sends processing to this section of the script when the script is invoked by a SIGNOFF statement. Before it stops the link, the script issues a LOG statement to notify the user that the link is terminated.

11 These statements are processed only if the prompts expected in the previous steps are not received. This section of the script issues messages to the SAS log at the client and then abnormally ends the script processing as well as the SIGNON.

TCPMVS.SCR Script

The following script enables a client to sign on and to sign off a z/OS server with TSO. The TCP/IP access method is used.

```
/*----------------------------------------------------------------*/
/*--           Copyright (C) 1990 by SAS Institute Inc., Cary NC  --*/
/*--                                                              --*/
/*-- name:      tcpmvs.scr                                        --*/
/*--                                                              --*/
/*-- purpose:   SAS/CONNECT SIGNON/SIGNOFF script for connecting   --*/
/*--            to a z/OS host via the TCP/IP access method       --*/
/*--                                                              --*/
/*-- notes:   1. This script might need modifications that account --*/
/*--             for the local flavor of your z/OS environment.    --*/
/*--             The logon procedure should mimic the tasks that   --*/
/*--             you perform when "telnet"-ing to the same         --*/
/*--             z/OS host through TSO.                            --*/
/*--                                                              --*/
/*--          2. You must have specified OPTIONS COMAMID=TCP       --*/
/*--             in the local SAS session before using the signon  --*/
/*--             command.                                          --*/
/*--                                                              --*/
/*--          3. This script supports one flavor of connection:    --*/
/*--             through a TSO session whose logon procedure       --*/
/*--             invokes SAS directly rather than the TSO TMP.     --*/
/*--                                                              --*/
/*-- support:   SAS Institute staff                               --*/
/*--                                                              --*/
```

```
/*----------------------------------------------------------------------*/

❶  log "NOTE: Script file 'tcpmvs.scr' entered.";

   if not tcp then goto notcp;
❷  if signoff then goto signoff;

/* ------------------------ TCP SIGNON ------------------------------*/

   /* make sure you are running the IBM TCP/IP */
❸  waitfor 'ENTER USERID'                : tsologon,
           120 seconds                  : noinit;

/*------------------------- TSO LOGON ------------------------------*/

tsologon:
❹  input 'Userid?';
   type LF;
❺  waitfor 'ENTER PASSWORD', 60 seconds : nolog;

tsopass:
   input nodisplay 'Password?';
   type LF;

tsodone:
❻ waitfor 'SESSION ESTABLISHED',
           'PASSWORD INVALID'           : tsopass,
           'ENTER NEW PASSWORD'         : tsonewp,
           'CURRENTLY LOGGED ON'        : dup_log,
           'NOT VALID'                  : nouser,
           120 seconds                  : notso;
   waitfor 1 second;

❼  log 'NOTE: SAS/CONNECT conversation established.';
   stop;

tsonewp:
❽  input nodisplay 'New Password?';
   type LF;

❾ waitfor 'VERIFY NEW PASSWORD',
           120 seconds                  : notso;

   input nodisplay 'Verify New Password';
   type LF;

   goto tsodone;

/*--------------------------- SIGNOFF ------------------------------*/
❿ signoff:
   type 'logoff' LF;
   waitfor 'LOGGED OFF'                 : logoff,
           20 seconds;
   log 'WARNING: Did not get messages confirming logoff.';
   abort;

logoff:
   log 'NOTE: SAS/CONNECT conversation terminated.';
   stop;

/*---------------------- TSO ERROR ROUTINES -----------------------*/

⓫ nouser:
   log 'ERROR: Unrecognized userid.';
   abort;

notcp:
   log 'ERROR: Incorrect communications access method.';
   log 'NOTE: You must set "OPTIONS COMAMID=TCP;" before using this';
   log '      script file.';
   abort;

noinit:
   log 'ERROR: Did not understand remote session banner.';
   abort;

nolog:
   log 'ERROR: Did not get userid or password prompt.';
   abort;

notso:
   log 'ERROR: Did not get TSO startup messages after logon.';
   abort;

dup_log:
   log 'ERROR: User is already logged onto TSO.';
   abort;
```

1 The LOG statement sends the enquoted message to the log file or to the LOG window of the SAS session at the client. Although it is not necessary to include LOG statements in your script file, the LOG statements keep the user informed about the progress of the connection.

2 The IF/THEN statement detects whether the script was called by the SIGNON statement. When you are signing off, the IF/THEN statement directs script processing to the statement labeled SIGNOFF. See step 10.

3 The WAITFOR statement awaits the login prompt from the server. If the statement does not receive the prompt within 120 seconds, it directs script processing to branch to the statement labeled NOINIT.

4 The INPUT statement displays a window with the text **`Userid?`** to allow the user to enter a server logon user ID. The TYPE statement sends a line feed to the server to enter the user ID to the server.

5 The WAITFOR statement waits for the password prompt from the server and branches to the NOLOG label if it is not received within 60 seconds. The INPUT statement that follows the WAITFOR statement displays a window for the user to enter a password.

6 The message **`SESSION ESTABLISHED`** is displayed when a SAS session is started on the server. The WAITFOR statement awaits the display of the message **`SESSION`**

ESTABLISHED to be issued by the server. If the **SESSION ESTABLISHED** response is received within 120 seconds, processing continues with the next LOG statement. If the **SESSION ESTABLISHED** response does not occur within 120 seconds, the script assumes that the remote SAS session has not started and processing branches to the statement labeled NOTSO.

7 After the connection has been successfully established, the user must stop the rest of the script from processing. Without this STOP statement, processing continues through the remaining statements in the script.

8 This section prompts for a new password if the password has expired. The INPUT statement displays a window with the text **New Password?** to allow the user to enter a password. The TYPE statement sends a line feed to the server to enter the password to the server.

9 The WAITFOR statement waits for the prompt to verify the new password from the server and branches to the NOTSO label if it is not received within 120 seconds. The INPUT statement that follows the WAITFOR statement displays a window in which the user re-enters the new password for verification.

10 This section of code is executed when the script is invoked to terminate the link. The IF statement (see step 2) sends processing to this section of the script when the script is invoked by a SIGNOFF statement. This section awaits a server prompt before displaying **LOGOFF**, which logs the user off the server. Before it stops the link, the script issues a LOG statement to notify the user that the link is terminated.

11 These statements are processed only if the prompts expected in the previous steps are not received. This section of the script issues messages to the SAS log at the client and then abnormally ends the script processing as well as the SIGNON.

TCPTSO9.SCR Script

The following script enables a client to sign on and to sign off a z/OS server with TSO or to a z/OS spawner. The TCP/IP access method is used.

```
/* trace on; */
/* echo on;  */
/*------------------------------------------------------------------*/
/*--           Copyright (C) 2003 by SAS Institute Inc., Cary NC  --*/
/*--                                                              --*/
/*-- name:     tcptso9.scr                                        --*/
/*--                                                              --*/
/*-- purpose:  SAS/CONNECT SIGNON/SIGNOFF script for connecting   --*/
/*--           to a z/OS host running SAS 9 or later via the      --*/
/*--           TCP/IP access method.                              --*/
/*--                                                              --*/
/*-- notes:   1. This script might need modifications that account --*/
/*--             for the local flavor of your z/OS environment.   --*/
/*--             The logon procedure should mimic the tasks that  --*/
/*--             you perform when "telnet"-ing to the same        --*/
/*--             z/OS host, either to TSO or to the z/OS          --*/
/*--             spawner.                                         --*/
/*--                                                              --*/
/*--          2. You must have specified OPTIONS COMAMID=TCP      --*/
/*--             in the local SAS session before using the SIGNON --*/
/*--             command.                                         --*/
/*--                                                              --*/
```

```
/*--          3. This script supports two flavors of connection:   --*/
/*--             through a TSO session whose logon procedure       --*/
/*--             invokes the TSO TMP or through the z/OS           --*/
/*--             spawner.                                          --*/
/*--                                                               --*/
/*--          4. If you use the spawner to start the SAS session,  --*/
/*--             in the signoff portion of the script, comment the --*/
/*--             LOGOFF command, which is only needed to complete  --*/
/*--             TSO session termination and is not necessary for  --*/
/*--             a spawned session.                                --*/
/*--                                                               --*/
/*-- assumes: 1. The shell script to execute SAS in your remote    --*/
/*--             z/OS environment is:                              --*/
/*--                "/usr/local/bin/spawnsas.sh"                   --*/
/*--             If you are using a different shell script or have --*/
/*--             your shell script stored in a different location, --*/
/*--             change the contents of the type statement that    --*/
/*--             specifies this shell script:                      --*/
/*--                type "/usr/local/bin/spawnsas.sh ..." LF;      --*/
/*--                                                               --*/
/*--          2. The command to execute SAS in your remote         --*/
/*--             (MVS/TSO) environment is "sas". If this is        --*/
/*--             incorrect for your site, change the contents of   --*/
/*--             the line for connection through TSO that          --*/
/*--             contains:                                         --*/
/*--                type "sas ..." lf;                             --*/
/*--                                                               --*/
/*-- support:   SAS Institute staff                                --*/
/*--                                                               --*/
/*-----------------------------------------------------------------*/

❶  log "NOTE: Script file 'tcptso9.scr' entered.";

   if not tcp then goto notcp;
❷  if signoff then goto signoff;

/* ------------------------ TCP SIGNON ------------------------------*/

   /* make sure you are running the IBM TCP/IP or the z/OS spawner   */
❸  waitfor 'Username:'                   : spnlogon,
          'ENTER USERID'                 : tsologon,
          120 seconds                    : noinit;

/*------------------------- SPAWNER LOGON ---------------------------*/

spnlogon:
❹  input 'Userid?';
   type LF;

❺  waitfor 'Password', 120 seconds : spnfail;
   input nodisplay 'Password?';
   type LF;

spndone:
```

```
❻  waitfor 'Hello>',
          'Userid'                       : spnlogon,
          'Password expired'             : spnnewp,
          120 seconds                    : spnfail;

❼  type "/usr/local/bin/spawnsas.sh nosasuser opt(''dmr comamid=tcp'')" LF;

❽  waitfor 'SESSION ESTABLISHED', 120 seconds : spnfail;

❾ log 'NOTE: SAS/CONNECT conversation established.';
   stop;

spnnewp:
❿  input nodisplay 'New Password?';
   type LF;

   waitfor 'Verify new password', 120 seconds : spnfail;

   input nodisplay 'Verify New Password';
   type LF;

   goto spndone;

spnfail:
   log 'ERROR: Invalid SPAWNER prompt message received.';
   abort;

/*-------------------------- TSO LOGON ------------------------------*/

tsologon:
⓫  input 'Userid?';
   type LF;
⓬  waitfor 'ENTER PASSWORD', 60 seconds : nolog;
   input nodisplay 'Password?';
   type LF;

tsodone:
⓭  waitfor 'READY',
          'CURRENTLY LOGGED ON'          : dup_log,
          'NOT VALID'                    : nouser,
          'PASSWORD INVALID'             : nopass,
          'ENTER NEW PASSWORD'           : tsonewp,
          'RECONNECT SUCCESS'            : recon,
          120 seconds                    : notso;
   waitfor 1 second;

strt_sas:
   log 'NOTE: Logged on to TSO.... Starting remote SAS now.';
   /* The value for the TCPIPPRF option, which locates TCP config         */
   /* data sets, is site specific. The need for this option depends       */
   /* on your TCP configuration. NOTERMINAL suppresses prompts from      */
   /* remote SAS session.  NOSYNTAXCHECK prevents remote side from        */
   /* going into syntax checking mode when a syntax error is encountered. */
⓮  type "sas o('dmr,comamid=TCP,noterminal,nosyntaxcheck')" LF;
```

```
⓯ waitfor 'SESSION ESTABLISHED', 120 seconds : nosas;

⓰  log 'NOTE: SAS/CONNECT conversation established.';
   stop;

tsonewp:
⓱  input nodisplay 'New Password?';
   type LF;

   waitfor 'VERIFY NEW PASSWORD',
           120 seconds                    : notso;

   input nodisplay 'Verify New Password';
   type LF;

   goto tsodone;

/*---------------------------- SIGNOFF ------------------------------*/
⓲ signoff:
/* --------- for the spawner, comment the following section ---------*/
   waitfor 'READY', 20 seconds: noterm;
   type 'logoff' LF;
   waitfor 'LOGGED OFF'                   : logoff,
           20 seconds;

   log 'WARNING: Did not get messages confirming logoff.';
   abort;

logoff:
/*---------- for the spawner, comment the previous section ----------*/

   log 'NOTE: SAS/CONNECT conversation terminated.';
   stop;

/*----- SUBROUTINES -------------------------------------------------*/

⓳ recon:
   log 'NOTE: Reconnected to previous session.  Old SAS session lost.';
   type LF;
   waitfor 'READY'       : strt_sas,
          120 seconds;
   log 'NOTE: Reconnected to a Running Session, but no READY prompt';
   abort;


/*--------------- ERROR ROUTINES --------------------------------*/

⓴ nouser:
   log 'ERROR: Unrecognized userid.';
   abort;

nopass:
   log 'ERROR: Invalid password.';
   abort;
```

```
notcp:
   log 'ERROR: Incorrect communications access method.';
   log 'NOTE: You must set "OPTIONS COMAMID=TCP;" before using this';
   log '      script file.';
   abort;

noinit:
   log 'ERROR: Did not understand remote session banner.';
   abort;

nolog:
   log 'ERROR: Did not get userid or password prompt.';
   abort;

notso:
   log 'ERROR: Did not get TSO startup messages after logon.';
   abort;

nosas:
   log 'ERROR: Did not get SAS software startup messages.';
   abort;

dup_log:
   log 'ERROR: User is already logged onto TSO.';
   abort;
noterm:
   log 'ERROR: Did not get READY prompt; remote session still logged on.';
   abort;
```

1 The LOG statement sends the quoted message to the log file or to the LOG window of the SAS session at the client. Although it is not necessary to include LOG statements in your script file, the LOG statements keep the user informed about the progress of the connection.

2 The IF/THEN statement detects whether the script was called by the SIGNON statement or by the SIGNOFF statement. When you sign off, the IF/THEN statement directs script processing to the statement that is labeled SIGNOFF. See step 18.

3 The WAITFOR statement awaits the login prompt from the server. If the statement does not receive the prompt within 120 seconds, it directs script processing to branch to the statement labeled NOINT.

4 The INPUT statement displays a window containing a prompt for a user ID so that the user can enter a server logon user ID. The TYPE statement sends a line feed to the server to enter the user ID to the server. This section is entered when the SAS/CONNECT spawner is encountered.

5 The WAITFOR statement waits for the password prompt from the server and branches to the SPNFAIL label if it is not received within 120 seconds. The INPUT statement that follows the WAITFOR statement displays a window for the user to enter a password in.

6 The WAITFOR statement awaits the Hello prompt that it expects to receive from the spawner. If WAITFOR does not receive the prompt, it branches to various condition handlers.

7 The TYPE statement calls a shell script to start SAS, and it passes the options that are needed for the SAS/CONNECT session. For a sample shell script, see “Defining the Shell Script for Starting SAS” on page 125.

8 The message **SESSION ESTABLISHED** is displayed when a SAS session is started on the server by using the DMR and COMAMID=TCP options. The WAITFOR statement awaits the display of the message **SESSION ESTABLISHED** that is issued by the server. If the **SESSION ESTABLISHED** response is received within 120 seconds, processing continues with the next LOG statement. If the response is not received in the specified time period, the script assumes that the remote SAS session has not started and processing branches to the statement labeled SPNFAIL.

9 After the connection has been successfully established, the user must stop the rest of the script from processing. Without this STOP statement, processing continues through the remaining statements in the script. Prior to the STOP, a message is output to the log, which informs the user that the connection has been established.

10 This section prompts for a new password if the password has expired.

11 The INPUT statement displays a window that contains a prompt for a user ID so that the user can enter a server logon user ID. The TYPE statement sends a line feed to the server to enter the user ID to the server. This section is entered when a TSO login is encountered.

12 The WAITFOR statement awaits the password prompt from the server and branches to the NOLOG label if it is not received within 60 seconds. The INPUT statement that follows the WAITFOR statement displays a window for the user to enter a password.

13 The WAITFOR statement awaits the READY prompt after successful TSO logon. It branches to various condition handlers if this prompt is not received.

14 The TYPE statement issues the command to start SAS through the TSO session, and passes options that are needed for the SAS/CONNECT session.

15 The message **SESSION ESTABLISHED** is displayed when a SAS session is started on the server using the DMR and COMAMID=TCP options. The WAITFOR statement awaits the display of the message **SESSION ESTABLISHED** to be issued by the server. If the **SESSION ESTABLISHED** response is received within 120 seconds, processing continues with the next LOG statement. If the response is not received within the time limit, the script assumes that the remote SAS session has not started and processing branches to the statement labeled NOSAS.

16 After the connection has been successfully established, the user must stop the rest of the script from processing. Without this STOP statement, processing continues through the remaining statements in the script. Prior to the STOP, a message is output to the log, which informs the user that the connection has been established.

17 This section prompts for a new password if the password has expired.

18 This section of code is executed when the script to terminate the link is invoked. The IF statement (see step 2) sends processing to this section of the script when the script is invoked by a SIGNOFF statement. This section awaits a server prompt before displaying LOGOFF, which logs the user off the server. Before it terminates the link, the script issues a LOG statement to notify the user that the link is terminated.

Note: If the session has been established through the z/OS spawner, the WAITFOR and TYPE statements should be deleted or commented out. They are necessary only for signing off a TSO connection. △

19 This section handles the case where SIGNON reconnects the user to a SAS session that is still running on the server. It sends the script back to the section that starts SAS through a TSO signon.

20 These statements are processed only if the prompts expected in the previous steps are not received. This section of the script issues messages to the SAS log at the client and then abnormally ends the script processing as well as the SIGNON.

PART 9

Error Messages

Chapter ***16*** . . . . **z/OS Error Messages** *177*

Chapter ***17*** . . . . **OpenVMS Alpha Error Messages** *179*

Chapter ***18*** . . . . **UNIX Error Messages** *181*

Chapter ***19*** . . . . **Windows Error Messages** *183*

CHAPTER 16
z/OS Error Messages

z/OS: TCP/IP Access Method **177**
SAS/CONNECT Error Messages under z/OS **177**
SAS/SHARE Error Messages under z/OS **177**

z/OS: TCP/IP Access Method

SAS/CONNECT Error Messages under z/OS

For TCP/IP, if SAS/CONNECT is unable to connect to the TCP/IP port, the following system message appears:

```
connection refused
```

The connection might fail at SIGNON because

- The remote side is not listening.
- The packet sequence is out of order, which can indicate that the routers are not working properly.
- The maximum number of connections has been reached.
- There is a flow problem, which indicates that too many packets are being sent to the remote side at the same time.

Under z/OS, use the NETSTAT utility to show active sockets and to show who is waiting for a socket.

SAS/SHARE Error Messages under z/OS

```
No TCP service <server-id> on this host
```

The service that is specified in the SERVERID= option is not one of the SAS/SHARE TCP/IP services that are defined in the TCP services file.

```
Cannot locate TCP host <host name>
```

The TCP/IP software is probably not running on the server's node.

```
Cannot bind TCP socket. System message is 'address already in use '
```

Another server with the same name is already running on this node, or another TCP/IP application is using the predefined port numbers that the TCP/IP access method is trying to use. If another server of the same name is running, choose one of the other defined server names. If there is no other server running that has the same name, there might be a conflict with another software package. Please contact your system administrator.

```
Cannot connect to TCP socket. System message is 'connection refused'
```

The server that is specified by the SERVER= option cannot be located on the specified node.

```
Cannot locate TCP host <node>
```

The node that is specified in a two-level name is not known to the TCP/IP software, or the TCP/IP software is not running on the user's node. See "Specifying the Server" on page 21 for information about two-level server names.

CHAPTER 17 OpenVMS Alpha Error Messages

OpenVMS Alpha: TCP/IP Access Method **179**
SAS/CONNECT and SAS/SHARE Error Messages under OpenVMS Alpha **179**
SAS/SHARE Error Messages under OpenVMS Alpha **179**

OpenVMS Alpha: TCP/IP Access Method

SAS/CONNECT and SAS/SHARE Error Messages under OpenVMS Alpha

If SAS/CONNECT or SAS/SHARE is unable to connect to the TCP/IP port, the following system message appears:

```
connection refused
```

The connection might fail at SIGNON because

□ The remote side is not listening.

□ The maximum number of connections has been reached.

SAS/SHARE Error Messages under OpenVMS Alpha

Failure to enter a server's name, a port, and an alias in the SERVICES file results in an error when you try to access the server.

The following partial SAS log shows the error message that the user receives if the server service is omitted from the SERVICES file.

```
36   options comamid=tcp fullstimer;
37   %let tcpsec=_prompt_;
38   libname sasdata 'edc.prog2.sasdata'
             server=sdcmvs.sharsrvx;
You cannot connect to server SDCMVS.SHARSRVX because
ERROR:  No TCP service 'sharsrvx' on this host.
ERROR:  Error in the LIBNAME or FILENAME statement.
```

```
No TCP service <server-id> on this host.
```

The service that is specified in the SERVERID= option is not one of the SAS/SHARE TCP/IP service names that are defined in the TCP services file.

```
Cannot locate TCP host <node>
```

The TCP/IP software is probably not running on the server's node.

```
Cannot bind TCP socket. System message is 'address already in use'
```

Another server that has the same name is already running on this node, or another TCP/IP application is using the predefined port numbers that the TCP/IP access method is trying to use. If another server that has the same name is running, choose one of the other defined server names. If there is no other server running that has the same name, there might be a conflict with another software package. Contact your system administrator.

CHAPTER

18 UNIX Error Messages

UNIX: TCP/IP Access Method **181**
SAS/CONNECT Error Messages under UNIX **181**
SAS/SHARE Error Messages under UNIX **181**

UNIX: TCP/IP Access Method

SAS/CONNECT Error Messages under UNIX

For TCP/IP, if SAS/CONNECT is unable to connect to the TCP/IP port, the following system message appears:

```
connection refused
```

The connection might fail at SIGNON because

- □ The remote side is not listening.
- □ The maximum number of connections has been reached.

SAS/SHARE Error Messages under UNIX

The TCP/IP access method that is used by SAS/SHARE sometimes issues generalized messages to identify problems. This section describes some of the most frequently encountered messages.

```
No TCP service <server-id> on this host
```

The service that is specified in the SERVERID= option is not one of the services that is defined in the TCP **`services`** file.

```
Cannot bind TCP socket.  System message is 'address already in use'
```

Another server that has the same name is already running on this node, or another TCP/IP application is using the predefined port numbers that the TCP/IP access method is trying to use. If another server of the same name is running, choose one of the other predefined server names. If there is no other server running that has the same name, there might be a conflict with another software package. Please contact your SAS support consultant.

```
Cannot connect to TCP socket.  System message is 'connection refused'
```

The server that is specified by the SERVER= option cannot be located on the specified node.

```
Cannot locate TCP host 'node'
```

The node that is specified in a two-level node name is not known to the TCP/IP software.

CHAPTER 19 Windows Error Messages

Windows: TCP/IP Access Method **183**
SAS/CONNECT Error Messages under Windows **183**
SAS/SHARE Error Messages under Windows **183**

Windows: TCP/IP Access Method

SAS/CONNECT Error Messages under Windows

For TCP/IP, if SAS/CONNECT is unable to connect to the TCP/IP port, the following system message appears:

```
connection refused
```

The connection might fail at SIGNON because

- □ The remote side is not listening.
- □ The maximum number of connections has been reached.

SAS/SHARE Error Messages under Windows

The TCP/IP access method used by SAS/SHARE sometimes issues generalized messages to identify problems. This section describes some of the most frequently encountered messages.

```
ERROR: Communication request rejected by partner:
security verification failure
```

An unauthorized client tried to connect to a secure server.

`No TCP service` *`server-id`* `on this host.`

The service that is specified in the SERVERID= option is not one of the SAS/SHARE TCP/IP service names that are defined in the TCP/IP services file.

`Cannot locate TCP host '`*`node`*`'.`

The TCP/IP software is probably not running on the server's node. The node that was specified in a two-level name is not known to the TCP/IP software, or the TCP/IP software is not running on the user's node.

```
Cannot bind TCP socket.
System message is 'address already in use'.
```

Another server that has the same name is already running on this node, or another TCP/IP application is using the predefined port numbers that the TCP/IP access method is trying to use. If another server that has the same name is running, choose one of the other predefined server names. If there is no other server running that has the same name, there might be a conflict with another software package. Contact the SAS installation representative at your site for assistance in resolving the problem.

```
Cannot connect to TCP socket.
System message is 'connection refused'.
```

The server that was specified by the SERVER= option cannot be located on the specified node.

PART 10

Appendix

Appendix 1 . . . . **Recommended Reading** *187*

APPENDIX 1 Recommended Reading

Recommended Reading **187**

Recommended Reading

Here is the recommended reading list for this title:

- *SAS/CONNECT User's Guide*
- *SAS/SHARE User's Guide*
- *Moving and Accessing SAS Files*
- SAS Companion that is specific to your operating environment

For a complete list of SAS publications, see the current *SAS Publishing Catalog*. To order the most current publications or to receive a free copy of the catalog, contact a SAS representative at

SAS Publishing Sales
SAS Campus Drive
Cary, NC 27513
Telephone: (800) 727-3228*
Fax: (919) 677-8166
E-mail: **sasbook@sas.com**
Web address: **support.sas.com/pubs**
* For other SAS Institute business, call (919) 677-8000.

Glossary

autoexec file
a file that contains SAS statements that are executed automatically when you start SAS. The autoexec file can be used to specify some of the SAS system options, as well as to assign librefs and filerefs to data sources that are used frequently.

authentication
the act of verifying the identity of the user who is attempting to access the machine that either the client session or the server session runs on. Authentication is performed so that a machine can use the identity to make decisions about the user's permissions to access protected resources. In Windows, the user ID, password, and access permissions make up a user context. See also user context.

client
the side of the client/server relationship that requests the delivery of a particular type of information. The client side is often considered the workstation or personal computer from which the request is made. Implicit in the client/server relationship is the network. See also server, SAS/CONNECT client, SAS/SHARE client.

client authentication
See authentication.

client session
See client, SAS/CONNECT client, SAS/SHARE client.

command file
a file that contains operating environment commands to be executed in sequence.

communications access method
the method that a client uses to communicate with a server. You can use the COMAMID= system option to specify the communications access method.

data set
See SAS data set.

fileref
a name that is temporarily assigned to an external file or to an aggregate storage location such as a directory or folder. The fileref identifies the file or the storage location to SAS. See also libref.

Job Control Language (JCL)
a language that is used in the z/OS operating environment to communicate information about a job to the operating environment, including the data sets, time, and memory that the job needs.

libref
a name that is temporarily associated with a SAS data library. For example, in the name SASUSER.ACCOUNTS, the name SASUSER is the libref. You can assign a libref by using a LIBNAME statement or an operating environment command.

name resolution
in TCP/IP, the process of mapping a server name to an address. The domain name system provides a facility for naming servers in which programs use remote name servers to resolve server names into IP addresses. See also name server.

name server
in TCP/IP, the server program that supplies name-to-address translation, mapping from server names to IP addresses. The server program often runs on a dedicated processor, and the operating environment itself is referred to as the name server. See also name resolution.

name resolver
in TCP/IP, the client software that uses one or more name servers to translate a server name. See also name resolution, name server.

operating environment
a computer, or a logical partition of a computer, and the resources (such as an operating system and other software and hardware) that are available to the computer or partition.

return code
a numeric value that indicates whether a request was successful. A return code can also indicate a specific error or warning.

SAS client
a SAS session that requests access to remote data by means of a SAS server. See also SAS server.

SAS/CONNECT client
a SAS/CONNECT session that acts as a client. The user that runs a SAS/CONNECT client requests services from a SAS/CONNECT server that can run on a remote single-processor machine or on a local or remote multi-processor machine. The services that are supported are: Remote Library Services, which enables access to SAS files; Compute Services, which exploits fast processing resources; and Data Transfer Services, which allows the upload or download of selected data for processing. See also client, server, SAS/CONNECT server.

SAS/CONNECT server
a SAS/CONNECT session that acts as a server. The SAS/CONNECT server runs a SAS session on a machine that receives requests for services from a SAS/CONNECT client. A server can run on a remote single-processor machine or on a local or remote multi-processor machine. The services that are supported are: Remote Library Services, which enables access to SAS files; Compute Services, which exploits fast processing resources; and Data Transfer Services, which allows the upload or download of selected data for processing. See also client, server, SAS/CONNECT client.

SAS data file
a SAS data set that contains data values as well as descriptor information.

SAS data library
a collection of one or more SAS files that are recognized by SAS and that are referenced and stored as a unit. Each file is a member of the library.

SAS data set
a file whose contents are in one of the native SAS file formats. There are two types of SAS data sets: SAS data files and SAS data views. SAS data files contain data values in addition to descriptor information that is associated with the data. SAS data views contain only the descriptor information plus other information that is required for retrieving data values from other SAS data sets or from files whose contents are in other software vendors' file formats. See also SAS data file, SAS data view.

SAS data view
a type of SAS data set that retrieves data values from other files. A SAS data view contains only descriptor information such as the data types and lengths of the variables (columns), plus other information that is required for retrieving data values from other SAS data sets or from files that are stored in other software vendors' file formats. See also SAS data set.

SASproprietary
a fixed encoding algorithm that is included with Base SAS software and is available in all supported operating environments. It requires no additional SAS product licenses. The SAS proprietary algorithm is strong enough to protect your data from casual viewing, but a determined hacker can break this encoding.

SAS/SECURE
an add-on product that provides additional encryption algorithms beyond the SAS proprietary algorithm. The RC2, RC4, DES, and TripleDES security algorithms are supported. SAS/SECURE software requires a license and must be installed on each machine that runs a SAS/CONNECT or a SAS/SHARE server that will use the encryption algorithms.

SAS/SHARE client
a SAS/SHARE session that acts as a client. The user that runs a SAS/SHARE client accesses data on a SAS/SHARE server through Remote Library Services (RLS). See also client, server, SAS/SHARE server.

SAS/SHARE server
the result of an execution of the SERVER procedure. The SERVER procedure is part of SAS/SHARE software. A server runs in a separate SAS execution that services users' SAS sessions by controlling and executing input and output requests to one or more SAS data libraries. See also client, server, SAS/SHARE client.

SAS spawner
a program that runs on a remote host and listens for client requests for connection to the remote host. When the spawner program receives a request, it invokes a SAS session on the remote host.

script
an external file that contains SAS script statements. The script is stored on a client and provides the instructions for establishing and terminating a SAS/CONNECT link. Scripts are executed by the SIGNON and SIGNOFF commands. See also script statement.

script statement
a special kind of SAS statement developed for use in scripts for SAS/CONNECT software. Script statements are not used in any kind of SAS program except a script.

Secure Sockets Layer (SSL)
a protocol that was developed by Netscape for transmitting private documents across the Internet. SSL uses a private key to encrypt data that is transmitted between a Web browser and a server.

server
the side of the client/server relationship that receives requests from a "client" and responds to those requests by delivering a particular type of information. Access to computing resources and files (SAS files and external files) might be requested by a client. Implicit in the client/server relationship is the network. "Server" sometimes refers to the machine that the server runs on. See also client, SAS/CONNECT server, SAS/SHARE server.

server session
See server.

simulated logon
a commonly used method of client authentication that is available in all operating environments. In a simulated logon, the client provides a user ID and password that are checked by the server.

spawner
See SAS spawner.

SSPI (Security Support Provider Interface)
Microsoft's built-in security provider for Windows machines. SSPI enables transparent authentication for connections between Windows machines. Users that are members of a "trusted" domain are authenticated automatically, and user context information is transferred to the server.

TCP/IP (Transmission Control Protocol/Internet Protocol)
an abbreviation for a pair of networking protocols. Transmission Control Protocol (TCP) is a standard protocol for transferring information on local area networks such as Ethernets. TCP ensures that process-to-process information is delivered in the proper order. Internet Protocol (IP) is a protocol for managing connections between operating environments. IP routes information through the network to a particular operating environment and fragments and reassembles information in transfers.

user context
the identifying credentials of the client who is attempting to access a secured server. Identifying credentials include the user ID, password, and file access permissions. Users can use their own user context or a different user context when accessing a server. A different user context (such as for a system administrator) does not belong to the user but can be granted to the user for access to specific files.

user rights
a set of privileges that is assigned to each user of a client machine and a server machine in a Windows domain. Setting the appropriate user rights at the server machine permits users to connect to a secured server. User rights are configured in the Microsoft Windows User Manager panel of the Administrative Tools program.

XMS (Cross-Memory Services)
an interface that is part of the z/OS operating environment and is used by programs that run within a single z/OS operating environment.

Index

A

anchor points
  XMS under z/OS 53
Authentication program
  TCP/IP under UNIX 88
AUTHSERVER option
  TCP/IP under Windows 97

C

client authentication
  TCP/IP under OpenVMS Alpha 70
  TCP/IP under UNIX 87
  TCP/IP under Windows 107, 109
  TCP/IP under z/OS 23
communications access methods
  definition 3
  supported by operating environment 3
  supported by SAS/CONNECT and SAS/SHARE 3
  supported in SAS 9.1 4

D

data security
  TCP/IP under Windows 109

E

encryption 137
  security services options 138
  TCP/IP under OpenVMS Alpha 60, 68, 70
  TCP/IP under UNIX 79, 88
  TCP/IP under Windows 98, 103, 105, 108
  TCP/IP under z/OS 15, 21, 23
  XMS under z/OS 48
error messages
  TCP/IP under OpenVMS Alpha 179
  TCP/IP under UNIX 181
  TCP/IP under Windows 183
  TCP/IP under z/OS 177

F

firewalls 149
  configuration example 150
  external UNIX client 151
  external Windows client 150
  requirements for 149
  spawners and 114

L

load module
  XMS under z/OS 53
logon procedure
  TCP/IP under z/OS 19

M

multi-processor machines
  TCP/IP under UNIX 79
  TCP/IP under Windows 98
  XMS under z/OS 48

N

name resolution 11
name resolver 11
name server 11
network security
  TCP/IP under OpenVMS Alpha 58
  TCP/IP under UNIX 76
  TCP/IP under Windows 95
  TCP/IP under z/OS 11
  XMS under z/OS 46

O

OpenVMS Alpha
  See TCP/IP access method, OpenVMS Alpha
OpenVMS Alpha spawner 119
  ending 121
  location of 119
  privileges 119
  scripted signon to 116
  starting 120
OpenVMS Alpha spawner service
  configuring 66
  starting 67
OpenVMS spawner program
  encryption 139, 141

P

passwords
  spawners and 114
Permission program
  TCP/IP under UNIX 88

R

resources
  XMS under z/OS 46

S

SAS, starting
  TCP/IP under OpenVMS Alpha 61
  TCP/IP under UNIX 80
  TCP/IP under Windows 99
  XMS under z/OS 48
  z/OS spawner 125
SAS/CONNECT
  communications access methods supported 3
SAS/SECURE 137
  security services options 139
  TCP/IP under Windows 96
  TCP/IP under z/OS 12
SAS/SHARE
  communications access methods supported 3
SAS SVC routine
  installing 51
  TCP/IP under z/OS 23
SASCMD option
  TCP/IP under OpenVMS Alpha 61
  TCP/IP under UNIX 80
  TCP/IP under Windows 99
  XMS under z/OS 48
SASproprietary encoding algorithm 137
  security services options 141
  TCP/IP under Windows 95
  TCP/IP under z/OS 12

SASTCPD command
OpenVMS Alpha spawner 120
scoket inheritance 149
SECPROFILE= option
TCP/IP under z/OS 14
secured servers
TCP/IP under OpenVMS Alpha 68
TCP/IP under UNIX 85
TCP/IP under Windows 103, 107
TCP/IP under z/OS 20
security services 137
access method supported 138
encryption options 138
operating environments supported 138
SAS/SECURE options 139
SASproprietary options 141
SSL options 141
server name
XMS under z/OS 52
server security
TCP/IP under Windows 95
server service
TCP/IP under OpenVMS Alpha 67, 70
TCP/IP under UNIX 85, 87
TCP/IP under Windows 105, 107
TCP/IP under z/OS 20, 23
server sessions
TCP/IP under OpenVMS Alpha 61, 62
TCP/IP under UNIX 79, 80
TCP/IP under Windows 98
XMS under z/OS 48, 49
servers
TCP/IP under OpenVMS Alpha 68, 71
TCP/IP under UNIX 86, 89
TCP/IP under Windows 105, 108
TCP/IP under z/OS 21, 24
XMS under z/OS 50
SERVICES file
configuring 145
shell scripts
z/OS spawner 125
sign-on scripts
spawners and 114
specifying time 156
syntax 155
TCP/IP under OpenVMS Alpha 64
TCP/IP under UNIX 82, 156
TCP/IP under Windows 101, 161
TCP/IP under z/OS 17, 164, 167
TCPMVS.SCR script 164
TCPTS09.SCR script 167
TCPUNIX.SCR script 156
TCPWIN.SCR script 161
TYPE statement and 156
WAITFOR statement and 156
signing on
TCP/IP under OpenVMS Alpha 61
TCP/IP under UNIX 79
TCP/IP under Windows 98
TCP/IP under z/OS 15
XMS under z/OS 48
simulated logon method
TCP/IP under Windows 109
SMP machines
TCP/IP under OpenVMS Alpha 61
TCP/IP under UNIX 79
TCP/IP under Windows 98
XMS under z/OS 48
software requirements
TCP/IP under OpenVMS Alpha 58
TCP/IP under UNIX 76
TCP/IP under Windows 94
TCP/IP under z/OS 11
XMS under z/OS 46
spawners 113
See also OpenVMS Alpha spawner
See also UNIX spawner
See also Windows spawner
See also z/OS spawner
client connection to 114
connection examples 115
firewalls and 114
passwords and 114
sign-on scripts and 114
support by operating environment 114
TCP/IP under OpenVMS alpha 66
TCP/IP under OpenVMS Alpha 62, 67
TCP/IP under UNIX 80, 84
TCP/IP under Windows 100, 103, 104
TCP/IP under z/OS 15
user IDs and 114
SSL (Secure Sockets Layer) 137
security services options 141
TCP/IP under Windows 96
SSPI (Security Support Provider Interface)
TCP/IP under Windows 110
SUBSYSID= option
XMS under z/OS 47
subsystem identifier
XMS under z/OS 47
system configuration
XMS under z/OS 52

T

TCP/IP access method 3
security services 138
TCP/IP access method, OpenVMS Alpha 58
accessing secured server 68
client authentication 70
client example 69
client tasks (SAS/CONNECT) 60
client tasks (SAS/SHARE) 67
configuring server service 67, 70
configuring spawner service 66
encryption 60, 68, 70
error messages 179
network security 58
options (SAS/CONNECT) 59
options (SAS/SHARE) 59
server example 67, 71
server tasks (SAS/CONNECT) 66
server tasks (SAS/SHARE) 70
signing on to same SMP machine 61
signing on with spawner 62
signing on with Telnet daemon 65
signon method 61
software requirements 58
specifying 60, 67, 70
specifying server 68, 71
starting spawner service 67
TCP/IP access method, UNIX 76
accessing secured servers 85
client authentication 87
client example 87
client tasks (SAS/CONNECT) 78
client tasks (SAS/SHARE) 84
configuring authentication program 88
configuring permission program 88
configuring server service 85, 87
configuring spawner service 84
configuring user access authority 88
encryption 79, 85, 88
error messages 181
network security 76
options (SAS/CONNECT) 77
server example 84, 89
server tasks (SAS/CONNECT) 84
server tasks (SAS/SHARE) 87
sign-on scripts 156
signing on to same SMP machine 79
signing on with spawner 80
signing on with Telnet daemon 83
signon method 79
software requirements 76
specifying 78, 85, 89
specifying server 86, 89
starting spawner 84
TCP/IP access method, Windows 94
client authentication 107, 109
client example 106
client tasks (SAS/CONNECT) 97
client tasks (SAS/SHARE) 104
configuring server service 105, 107
configuring spawner service 103
data security 109
encryption 98, 103, 105, 108
error messages 183
network security 95
options (SAS/CONNECT) 96
options (SAS/SHARE) 97
server example 104, 108
server security 95
server tasks (SAS/CONNECT) 103
server tasks (SAS/SHARE) 107
sign-on scripts 161
signing on to same SMP machine 98
signing on with spawner 100
signing on with Telnet daemon 102
signon method 98
simulated logon method 109
software requirements 94
specifying 98, 105, 108
specifying server 105, 108
SSPI 110
starting the spawner 104
user context 94
user rights for secured server 103, 107
TCP/IP access method, z/OS 10
accessing secured servers 20
client authentication 23
client example 22
client tasks (SAS/CONNECT) 14
client tasks (SAS/SHARE) 20
configuring server service 20, 23
encryption 15, 21, 23
error messages 177
installing logon procedure 19
installing SAS SVC routine 23
network security 11

options (SAS/CONNECT) 12, 13
options (SAS/SHARE) 12, 13
server example 19, 25
server tasks (SAS/CONNECT) 18
server tasks (SAS/SHARE) 22
sign-on scripts 164, 167
signing on with spawner 15
signing on with Telnet daemon 18
signon method 15
software requirements 11
specifying 15, 20, 24
specifying server 21, 24
terminology 11
TCPIPMCH= option
TCP/IP under z/OS 12
TCPIPPRF= option
TCP/IP under z/OS 12
TCPMSGLEN option
TCP/IP under OpenVMS Alpha 59
TCP/IP under UNIX 77
TCP/IP under Windows 96
TCP/IP under z/OS 13
TCPMVS.SCR script 164
TCPPORTFIRST= option
TCP/IP under OpenVMS Alpha 59
TCP/IP under UNIX 77
TCP/IP under Windows 96
TCP/IP under z/OS 13
TCPPORTLAST= option
TCP/IP under OpenVMS Alpha 59
TCP/IP under Windows 96
TCP/IP under z/OS 13
TCPSEC= option
TCP/IP under OpenVMS Alpha 59, 70
TCP/IP under UNIX 78, 87
TCP/IP under Windows 97, 107
TCP/IP under z/OS 13, 23
TCPTN3270 option
TCP/IP under OpenVMS Alpha 59
TCP/IP under UNIX 77
TCP/IP under Windows 96
TCP/IP under z/OS 13
TCPTS09.SCR script 167
TCPUNIX.SCR script 156
TCPWIN.SCR script 161
Telnet daemon for signing on
TCP/IP under OpenVMS Alpha 65
TCP/IP under UNIX 83
TCP/IP under Windows 102
TCP/IP under z/OS 18
time
specifying for sign-on scripts 156
TYPE statement
sign-on scripts and 156

U

UNIX
See TCP/IP access method, UNIX
UNIX spawner 127
ending 129
location of 127
scripted signon to 115
starting 127
UNIX spawner service
configuring 84
starting 84
user access authority
TCP/IP under UNIX 88
user context
TCP/IP under Windows 94
user IDs
spawners and 114
z/OS spawner 123
user rights
TCP/IP under Windows 103, 107

W

WAITFOR statement
sign-on scripts and 156
Windows
See TCP/IP access method, Windows
Windows spawner 131
ending 135
location of 131
security 131
starting 104, 131
Windows spawner service
configuring 103
scriptless signon to 116

X

XMS access method 3
XMS access method, z/OS 46
client example 50
client tasks (SAS/CONNECT) 47
client tasks (SAS/SHARE) 49
defining resources 46
encryption 48
installation tasks 52
installing load module 53
installing SAS SVC routine 51
network security 46
server example 52
server name 52
server tasks (SAS/SHARE) 51
signing on to multi-processor machine 48
software requirements 46
specifying 47, 49, 51
specifying server 50
SUBSYSID= option 47
system configuration 52

Z

z/OS
See TCP/IP access method, z/OS
See XMS access method, z/OS
z/OS spawner 123
assigning user ID to started task 123
encrypted signon to 117
ending 126
location of 123
shell script for starting SAS 125
starting 124
user ID requirement 123
version level requirement 124

Your Turn

If you have comments or suggestions about *Communications Access Methods for SAS/CONNECT 9.1 and SAS/SHARE 9.1*, please send them to us on a photocopy of this page or send us electronic mail.

For comments about this book, please return the photocopy to

SAS Publishing
SAS Campus Drive
Cary, NC 27513
email: **`yourturn@sas.com`**

For suggestions about the software, please return the photocopy to

SAS Institute Inc.
Technical Support Division
SAS Campus Drive
Cary, NC 27513
email: **`suggest@sas.com`**