
# Communications Access Methods for SAS/CONNECT® 9.4 and SAS/SHARE® 9.4

**Second Edition**

SAS® Documentation

The correct bibliographic citation for this manual is as follows: SAS Institute Inc. 2015. *Communications Access Methods for SAS/CONNECT® 9.4 and SAS/SHARE® 9.4, Second Edition*. Cary, NC: SAS Institute Inc.

**Communications Access Methods for SAS/CONNECT® 9.4 and SAS/SHARE® 9.4, Second Edition**



SAS Institute Inc., SAS Campus Drive, Cary, North Carolina 27513-2414.

July 2015

# Contents

# What's New in Communications Access Methods for SAS/CONNECT 9.4 and SAS/SHARE 9.4

## Overview

SAS/CONNECT and SAS/SHARE Communications Access Methods have the following changes and enhancements:

- new SAS/CONNECT spawner management interface that is supported on all operating environments (UNIX, Windows, and z/OS) and is accessible through SAS Management Console and PROC IOMOPERATE.
- enhanced spawner management tools, including the ability to manage and monitor the SAS/CONNECT spawner remotely using SAS Management Console and PROC IOMOPERATE.
- new SAS/CONNECT spawner start-up options.
- enhanced logging features, including new loggers and better debugging and traceback features.
- ability to associate multiple SAS/CONNECT servers with one spawner. In versions prior to SAS 9.4, only one SAS/CONNECT server could be associated with a SAS/CONNECT spawner.
- increased allowable length for IBM password phrases.

## Spawner Enhancements

### *New Spawner Management Interface*

The new SAS/CONNECT spawner interface makes it easier for administrators to monitor and manage spawner and server connections. The following is a list of new features available with the new SAS/CONNECT spawner management interface:

- tools for better management, monitoring, and troubleshooting of the SAS/CONNECT spawner and server connections.
- new spawner options and a new spawner start-up command that are consistent across all operating environments.

- ability to use SAS facility logging to get debug and trace information about what SAS/CONNECT is doing. You can now get the full logging and debugging support of the SAS Logging Facility for the SAS/CONNECT spawner operations.
- new SAS/CONNECT spawner operator port for administering the spawner using SAS Management Console or PROC IOMOPERATE.
- ability to associate multiple SAS/CONNECT servers with one spawner that listens on multiple ports.

Here are some of the actions that are available with the new SAS/CONNECT spawner in SAS Management Console and PROC IOMOPERATE:

- stop, pause, resume, quiesce, and refresh actions available for the new SAS/CONNECT spawner operator port
- connect, disconnect, stop, pause, quiesce, and refresh actions available for the SAS/CONNECT spawner
- request information about defined servers and spawned servers
- configure the SAS/CONNECT spawner to have multiple client connection ports open
- request the logical server name from the SAS/CONNECT client
- list clients that are connected to the SAS/CONNECT spawner
- display graphs that show the spawned server activity
- display a list of spawned servers and the user name and host that are associated with the spawned server
- display options and properties that are associated with the SAS/CONNECT spawner
- list the active, inactive, or terminated sessions
- display loggers, logging levels, and log events for the SAS/CONNECT spawner
- display performance counters

For more information about using PROC IOMOPERATE to manage the SAS/CONNECT spawner, see “Using PROC IOMOPERATE to Configure the SAS/CONNECT Spawner ” on page 78.

For more information about using SAS Management Console to manage the SAS/CONNECT spawner, see “Using SAS Management Console to Administer the SAS/CONNECT Spawner ” on page 74.

### New Spawner Name and Start-Up Command

The new cross-platform executable program to start the SAS/CONNECT spawner is now the same across all supported operating systems. The new spawner invocation command is **CNTSPAWN**.

This new command replaces the Windows **SPAWNER** command, and the UNIX and z/OS **SASTCPD** command.

### New or Enhanced Spawner Options

The following spawner options are new or enhanced and are supported on UNIX, Windows, and z/OS operating environments:

- You can now specify the -HELP option on the spawner invocation command to print a help message.
- The new -MGMTPORT option enables you to specify an operator port for administering the SAS/CONNECT spawner using SAS Management Console or PROC IOMOPERATE.
- The new -LOG | -LOGFILE option enables you to specify the filename to use for log message output.
- The new -SERVICEDIRECTORY option replaces the -WORKDIR option as the preferred way to specify the Windows service directory when the spawner is installed as a Windows service.
- The -UNINSTALL option replaces the -DELETE option.
- The -LOGCONFIGLOC option replaces the -LOGEVENTS option.
- The -LOGCONFIGLOC option replaces the -TIMESTAMP option.
- -SSPI authentication is no longer enabled by default on the SAS/CONNECT spawner.

Beginning in SAS 9.4, the following options are no longer supported:

- -TIMESTAMP
- -LOGEVENTS
- -SECURITY | -NOSECURITY
- -DELETE

For a complete list of SAS/CONNECT spawner options, see Chapter 7, “Managing the SAS/CONNECT Spawner,” on page 83.

### Spawner Logger Enhancements

In SAS 9.4, enhancements to the SAS/CONNECT logging facility enable you to perform the following tasks:

- get more detailed spawner log output when the -LOGCONFIGLOC option is specified. In previous releases, only the start and stop output was available.
- use the SAS Logging Facility to get debug and trace information about spawner activity.
- get information about what ports are opened.
- get counters and information set up by the spawner.
- get information about how many servers are active and who owns them.

## Other Communication Access Methods Changes

### SAS Support for IBM Password Phrase Length on z/OS

SAS supports passwords that are mixed case, and it supports the IBM standard for password phrases that have a length of up to 100 characters. For more information about

mixed-case passwords and password phrases on z/OS, see “Allowing Mixed-Case Passwords (PASSWORD Option)” and “Assigning password phrases” in *z/OS V1R12.0 Security Server RACF Security Administrator's Guide* (IBM).

# *Part 1*

# Introduction

*Chapter 1*

# Communication Access Methods



## 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. TCP/IP is supported under the UNIX, Windows, and z/OS operating environments.

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.

## About TCP/IP Internet Protocol (IP) Addressing

TCP/IP applications refer to networked computers via their fully qualified domain names (FQDN) and their IP addresses. Because IP addresses can change easily, SAS applications that contain hardcoded IP addresses are prone to maintenance problems. To avoid such problems, use of an FQDN is preferred over an IP address. The name-resolution system that is part of the TCP/IP protocol is responsible for locating the IP address that is associated with the FQDN.

SAS 9.2 introduced support for the Internet Protocol, IPv6, which is the successor to Internet Protocol, IPv4. Rather than replacing IPv4 with IPv6, SAS now supports both protocols. There will be a lengthy transition period during which the two protocols will coexist. A primary reason for the new protocol is that the limited supply of 32-bit IPv4 address spaces was being depleted. IPv6 uses a 128-bit address scheme, which provides more IP addresses than does IPv4.

Here are examples of an FQDN, an IPv6 address, and an IPv4 address:

```
d6292.us.company.com
db8::01
10.23.2.3
```

For details, see "Internet Protocol Version 6 (IPv6)" in *SAS Language Reference: Concepts*.

## Operating Environments Supported in SAS 9.4

Your SAS/CONNECT or SAS/SHARE sessions must run a supported SAS release under a supported operating environment. For information about supported SAS releases and operating environments, see http://support.sas.com/resources/sysreq/hosts.

## Finding Information in This Documentation

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

1. Find the part in this document 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 chapter in this document for the server operating environment (For example, UNIX).
2. In that chapter, find the section for the communications access method that you will use (for example, UNIX: TCP/IP).
3. In that section, find the heading for the SAS product that you will use (for example, UNIX: TCP/IP for SAS/CONNECT).

The tasks outlined in each chapter for configuring and starting the SAS/CONNECT Spawner are for manual deployment of the SAS/CONNECT Spawner. In most cases, the SAS/CONNECT Spawner is deployed using the SAS Deployment Wizard and the information is stored in the metadata server. In this case, manual setup is not required.

## Accessibility Features in SAS Products

For information about accessibility for any of the products mentioned in this book, see the documentation for that product. If you have questions or concerns about the accessibility of SAS products, send email to accessibility@sas.com.

# Part 2

# Access Methods by Operating Environment

*Chapter 2*

# UNIX: TCP/IP Access Method

# 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 the following requirements are met:

- 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.

For details about setting up and using encryption, see *Encryption in SAS*. After an encryption service is set up in your environment, set a SAS encryption option that is appropriate to the encryption service and requirements of the client or the server session.

## TCPMSGLEN Environment Variable

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 that you can specify in the SIGNON statement or as a SAS option. For details, see "TBUFSIZE= System Option" in *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. The TCP/IP access method then issues the number of send and receive messages that are necessary to complete the message transaction.

The value for TCPMSGLEN 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. If the TCPMSGLEN environment variable is not set, SAS uses the TCP stack's default size and allows autotuning if implemented by the stack.

Example:

```
-set tcpmsglen 65536
```

### *SAS/CONNECT Options Only*

TCPPORTFIRST=*port-number* (set at the server)
TCPPORTLAST=*port-number* (set at the server)
restrict 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 start-up 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 2.1 on page 16 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.

## 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 TCP/IP as the access method.

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 an encryption service 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 details about encryption options, see *Encryption in SAS* located in the Base SAS Help and Documentation.

## 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 multiprocessor computer

  *Note:* This method is most useful if your client computer is equipped with symmetric multiprocessor (SMP) hardware.

- a spawner

- a Telnet daemon

## Signing On to the Same Multiprocessor Computer

### Task List

If your client computer is equipped with SMP, and if you want to run one or more server sessions on your computer, 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;
```

You can also specify it 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 multiprocessor computer.

*Note:* PROCESS=, REMOTE=, CREMOTE=, and CONNECTREMOTE= can be used interchangeably. For details, see "CONNECTREMOTE= System Option" in *SAS/CONNECT User's Guide*.

For details about the SIGNON statement, see "SIGNON Statement and Command" in *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 multiprocessor computer.

The SASCMD option can be specified in an OPTIONS statement:

```
options sascmd="SAS-command" | "!SASCMD";
```

You can also specify it 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:* To execute additional commands before starting SAS, you can write a script that contains the SAS start-up commands that are appropriate for your operating environment. You can then specify this script as the value in the SASCMD= option.

For details, see "SASCMD= System Option" in *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 multiprocessor computer.

```
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 multiprocessor computer. 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

### Task List

1. Ensure that the spawner is running on the server.
2. Specify the server and spawner 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 6, "Introduction to the SAS/CONNECT Spawner," on page 73.

*Note:* The system administrator for the computer 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 Spawner Service

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

```
options remote=node-name[.service-name | .port-number];
```

You can also specify it directly in the SIGNON statement or command:

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

The *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 of the following:

- the short computer 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.

Here is the process for evaluating *node-name*:

1. If *node-name* is a macro variable, the value of the macro variable is passed to the operating environment's **getnameinfo()** 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 **getnameinfo()** function.

3. If **getnameinfo()** fails to resolve *node name*, an error message is returned and the sign-on fails.

   *Note:* The order in which the **getnameinfo()** 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 the *service-name* when signing on. Therefore, the value of *service-name* that was specified during spawner start-up must be identical to the value of the *service-name* that is specified in the sign-on. Alternatively, you can specify the explicit port number that is associated with *service-name* directly in the SIGNON statement.

Example 1:

In the following example, REMHOST is the name of the node on which the spawner runs, and **spawner1** is the name of the service that is defined at the client. The client service **spawner1** must be assigned to the same port that the spawner is listening on.

```
signon remhost.spawner1;
```

Example 2:

In the following example, the macro variable REMHOST is assigned to the fully qualified name of the computer 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 6, “Introduction to the SAS/CONNECT Spawner,” on page 73.

### Specifying a Sign-On Script

Sample script files are provided with SAS/CONNECT for signing on and signing off. You must specify the script filename in the SIGNON statement or command when signing on using a script file. The SIGNON statement executes the script file. By default, the script prompts for user ID and password.

Use the FILENAME statement to assign the fileref RLINK to the appropriate script file. For example:

```
filename rlink '!sasroot/misc/connect/script-name';
```

The sample scripts are installed at this location:

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

The script is based on the server that you are connecting to.*script-name* specifies the appropriate script file for the server.

The following table lists the scripts that are provided in SAS software.

***Table 2.1*** *SAS/CONNECT Sign-on Scripts for TCP/IP under UNIX*

| Server | Script Name |
|---|---|
| TSO under z/OS | **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** |
| UNIX | **tcpunix.scr** |
| Windows | **tcpwin.scr** |

### Specifying a User ID and Password

You must provide a user ID and password to sign on to a secured server if you are not using a sign-on script. For example:

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

### Signing On

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

### Task List

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 in an OPTIONS statement:

```
options remote=node-name;
```

You can also specify it 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 15.

### 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

## Overview

The tasks outlined in this chapter describe the manual setup of the SAS/CONNECT spawner. In most cases, the SAS/CONNECT spawner is deployed using the SAS Deployment Wizard and manual setup is not required. The SAS Deployment Wizard stores information about the spawner in the metadata server. For more information about the SAS 9.4 Deployment Wizard, see *SAS Deployment Wizard and SAS Deployment Manager: User's Guide.*

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

## Task List

Steps for manually deploying the SAS Connect spawner:

1. Configure the UNIX spawner service.

   *Note:* This step is required only if you plan to start the SAS/CONNECT spawner using the -SERVICE <*service-name*> option.

2. Start the UNIX spawner at 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 9, "Configuring SAS/CONNECT for Use with a Firewall," on page 103.

### Configuring the UNIX Spawner Service

If you plan to start the SAS/CONNECT spawner by specifying the *service-name* on the spawner start-up command, then you need to configure the spawner service in the TCP/IP **services** file. However, if you plan to start the SAS/CONNECT spawner by specifying the explicit *port-number* or if you use the information defined in the SAS/CONNECT server's metadata, then you do not need to perform this step.

The **services** file is a configuration file that maps port numbers to named services. This mapping allows programs to access ports by name as well as by port number.

In UNIX, the location of the services file is usually **/etc/services**.

To configure the spawner service, use a text editor to add the connection service name, the port number, and the communications protocol to the **services** file. The file uses the following format:

<*official-service-name*> <*port-number/protocol-name*> <*alias-name*>

as shown in the following example:

```
spawnport   4016/TCP   spnprt
```

The <*alias-name*> specification is optional.

For a more detailed explanation of these parameters and a sample TCP/IP services file, see "Configuring the Services File" on page 99.

### Starting the SAS/CONNECT Spawner on UNIX

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 connections to the server. After the spawner program receives a request, it starts a SAS server session.

If network security is desired, set the appropriate encryption options when starting the spawner. For more details about encryption, see *Encryption in SAS*, located in the Base SAS Help and Documentation.

### SAS/CONNECT Server Example

The following command starts the SAS/CONNECT spawner on UNIX:

```
cntspawn -service spawner -mgmtport 7555 -sascmd "/u/username/mystartup"
```

The -SERVICE option specifies that the service named **spawner** listens for incoming connections. The -MGMTPORT option specifies the port number for operator (administrative) connections. The -SASCMD option specifies the path to the **mystartup** command file, which starts SAS on the server.

## 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 computer that a client accesses 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** file, see "Configuring the Services File" on page 99.

### 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 "LIBNAME Statement" in *SAS/SHARE User's Guide* and "OPERATE Procedure" in *SAS/SHARE User's Guide* 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 an encryption service 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 encryption 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 details about encryption, see *Encryption in SAS*, located in the Base SAS Help and Documentation.

### 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. To do this, you use 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. a valid node name

*node* can be either of the following:

- valid TCP/IP node name
- IP address

For more information, see "About TCP/IP Internet Protocol (IP) Addressing" on page 4.

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 99.

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, and 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 before 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;
```

For more information about creating valid SAS names, see "Rules for Words and Names in the SAS Language" in *SAS Language Reference: Concepts*. For details about LIBNAME and PROC OPERATE, see "LIBNAME Statement" in *SAS/SHARE User's Guide* and "OPERATE Procedure" in *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 **getnameinfo()** 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 99.

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 11.

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, you must configure resources on the computer that the server runs on. in order to verify a client's identity and the user's authority to access resources. 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 the following **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 cntspawn objspawn
   chmod 4755 sasauth sasperm cntspawn 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 Client/Server Transfers

If an encryption service 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 encryption 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 details about encryption, see *Encryption in SAS*, located in the Base SAS Help and Documentation.

### 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. Here is the syntax:

`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 99.

*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 more information about creating valid SAS names, see “Rules for Words and Names in the SAS Language” in *SAS Language Reference: Concepts*. For details about PROC SERVER, see “SERVER Procedure” in *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 computer. 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.

*Chapter 3*

# Windows: TCP/IP Access Method



## SAS/SHARE Client Tasks

### *Task List*

*Note:* All SAS/CONNECT—related information in this document has been moved to the SAS/CONNECT User's Guide. For information related to using TCP/IP for SAS/CONNECT in a Windows operating environment, see "Windows Operating Environment" in *SAS/CONNECT User's Guide*.

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 99.

### 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 an encryption service is configured at the client, you can specify SAS options to encrypt data that a client transfers to a server. Here is an example:

```
options netencrypt netencryptalgorithm=ssl;
```

The NETENCRYPT option specifies that all data transfers between a client and a server will be encrypted. SSL is the encryption service that is specified in the NETENCRYPTALGORITHM= option. For details about encryption, see *Encryption in SAS*, located in the Base SAS Help and Documentation.

### 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* is specified as the fully qualified domain name (FQDN). Here is an example:

```
mktserver.acme.com
```

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 99.

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, and 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 would cause the macro variable to be resolved by the SAS parser before syntactic evaluation of the SERVER= option. The access method evaluates the node name in a two-level server name.

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;
```

For details about creating valid SAS names, see "Rules for Words and Names in the SAS Language" in *SAS Language Reference: Concepts*. For details about LIBNAME and PROC OPERATE, see "LIBNAME Statement" in *SAS/SHARE User's Guide* and "OPERATE Procedure" in *SAS/SHARE User's Guide*.

## SAS/SHARE Client Example

The following example shows the statements that are specified at a Windows client that accesses a server by using a different user context. The LIBNAME statement specifies the SAS 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

*Note:* All SAS/CONNECT—related information in this document has been moved to the SAS/CONNECT User's Guide. For information related to using TCP/IP for SAS/CONNECT in a Windows operating environment, see "Windows Operating Environment" in *SAS/CONNECT User's Guide*.

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 encryption 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 99.

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 11.

Example:

```
options tcpsec=_secure_;
```

## Assigning User Rights for a Server That Is Running Secured

If you use only 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 an encryption service 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 encryption service that is specified in the NETENCRYPTALGORITHM= option. For details about encryption, see *Encryption in SAS*, located in the Base SAS Help and Documentation.

### 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 99. *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
- SSPI

## Simulated Logon Method

### Overview of 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 in the following situations:

- 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.

### Requirements for Using Simulated Logon with SAS/CONNECT or SAS/SHARE

To authenticate user credentials (user ID and password) of SAS/CONNECT or SAS/SHARE clients, the administrator of the computers that the SAS/CONNECT client and server sessions or the SAS/SHARE client and server sessions run on must assign the appropriate rights to users.

Here are the requirements for SAS/CONNECT and SAS/SHARE:

- assignment of the “Log on as batch job” right to users in client sessions that access SAS/CONNECT server sessions.

- assignment of the “Act as part of the operating system” right to users who start SAS/SHARE servers or SAS/CONNECT spawners.

Here are the requirements for SAS/CONNECT only:

- assignment of the “Increase quotas” right to users who start a SAS/CONNECT spawner.
- assignment of the “Replace a process level token” right to users who start a SAS/CONNECT spawner.

  *Note:* Because the SAS/CONNECT spawner usually runs as a service under the LocalSystem account, these permissions are already set by default and user rights do not need to be changed.

Here are the requirements for SAS/SHARE only:

- specification of the system option TCPSEC=_SECURE_ in the server session.
- specification of the AUTHENTICATE=REQUIRED option in the PROC SERVER statement that is used to start a SAS/SHARE server session. REQUIRED is the default value.

## SSPI

### Overview of SSPI

Security Support Provider Interface (SSPI), also referred to as Integrated Windows Authentication (IWA), enables transparent authentication for connections between Windows computers. 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 computers, and the user who runs the client computer is a member of a domain that is trusted at the server computer.

For more information, see “Integrated Windows Authentication” in *SAS Intelligence Platform: Security Administration Guide*, available at: http://support.sas.com/documentation/onlinedoc/intellplatform/index.html.

### SSPI Requirement for SAS/CONNECT

In versions prior to SAS 9.4, SSPI is enabled by default. To disable it, specify -NOSSPI on the spawner command. In SAS 9.4 and later, -SSPI is not enabled by default, and you must specify -SSPI on the spawner start-up command to enable it.

If you use SAS Deployment Wizard to configure and deploy SAS, the -SSPI option is automatically added to the **`ConnectSpawner.bat`** and **`ConnectSpawner.sh`** script files. To disable it, edit the script files by adding -NOSSPI or removing -SSPI.

### SSPI Requirement for SAS/SHARE

In order to use SSPI for authentication, the SAS/SHARE server administrator must do the following:

- specify the option TCPSEC=_SECURE_
- specify the option AUTHENTICATE=REQUIRED in the PROC SERVER statement. REQUIRED is the default value.

*Chapter 4*

# z/OS: TCP/IP Access Method

# Prerequisites for Using TCP/IP under z/OS

### *Task List*

- Verify that the software requirements are met.
- Verify that the SAS SVC routine has been installed.
- 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*

Ensure that the following requirements are met:

- Base SAS software and either SAS/CONNECT or SAS/SHARE are installed on both the client and the server.
- SAS/CONNECT and SAS/SHARE also require the IBM z/OS Communications Server or any server that is functionally compatible with the IBM z/OS Communications Server.
- 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 47.

### *Installation of 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, do not repeat the step here. If you need to perform the installation, see the Configuration Guide for SAS Foundation for z/OS (PDF) on the SAS Install Center web page.

   Because SAS SVC in SAS 9.4 is backward compatible, it replaces the SAS SVC routines from previous releases. You can continue using previous releases of Base

SAS, SAS/CONNECT, and SAS/SHARE with the SAS 9.4 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 how 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 systems, as needed.

   If you have more than one z/OS system, verify that the SAS SVC is installed on all the systems that will be running SAS/CONNECT or SAS/SHARE at your site.

### *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 to 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.

### *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.

For details about setting up and using encryption services, see *Encryption in SAS*, located in the Base SAS Help and Documentation. After an encryption service is set up in your environment, you set SAS encryption options that are appropriate to the encryption 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 that you can specify in the SIGNON statement or as a SAS option. For details, see "TBUFSIZE= System Option" in *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 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. If the TCPMSGLEN environment variable is not set, SAS uses the TCP stack's default size and allows autotuning if implemented by the stack.

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 4.1 on page 40 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 sign-on 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 that was 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.

## *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 details about encryption services, see the *Encryption in SAS*, located in the Base SAS Help and Documentation.

### 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

#### Task List

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 6, "Introduction to the SAS/CONNECT Spawner," on page 73.

*Note:* The system administrator for the computer 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 in an OPTIONS statement:

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

You can also specify it 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 of the following:

- the short computer 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. For more information, see "About TCP/IP Internet Protocol (IP) Addressing" on page 4.

Here is the process for evaluating *node-name*:

1. If *node-name* is a macro variable, the value of the macro variable is passed to the operating environment's **`getnameinfo()`** 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 **`getnameinfo()`** function.
3. If **`getnameinfo()`** fails to resolve *node-name*, an error message is returned and the sign-on fails.

   *Note:* The order in which the **`getnameinfo()`** 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. **`cspawn`** is the name of the service that is defined at the client. The client service **`cspawn`** must be assigned to the same port that the spawner is listening on.

```
signon remhost.cspawn;
```

Example 2:

In the following example, the macro variable REMHOST is assigned to the fully qualified name of the computer 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 6, "Introduction to the SAS/CONNECT Spawner," on page 73.

### 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 this location:

```
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.

The following table lists the scripts that are supplied in SAS software.

***Table 4.1*** *SAS/CONNECT Sign-on Scripts for Using TCP/IP under z/OS*

| Server | Script Name |
|---|---|
| TSO under z/OS | **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 | **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, then 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 logon 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

#### Task List

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 in an OPTIONS statement:

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

You can also specify it 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 40.

#### 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

### Overview

The tasks outlined in this chapter describe the manual setup and configuration of the SAS/CONNECT Spawner. In most cases, the SAS Connect spawner is deployed using the SAS Deployment Wizard, which stores information in the metadata server, and manual setup is not required.

*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 9, "Configuring SAS/CONNECT for Use with a Firewall," on page 103.

### Task List

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

1. Verify that the SAS SVC routine has been installed. See "Installation of the SAS SVC Routine" on page 34.
2. Start the spawner program at the z/OS server. For details, see "SAS/CONNECT Spawner in z/OS " on page 86.

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 sign-on. 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 118.

### SAS/CONNECT Server Example

The following command starts the spawner on a z/OS computer. The -SERVICE option specifies the name of the port, **o390spawn**, that the spawner listens on for client requests to start the server. 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 computer.

```
cntspawn -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 computer that a client will access the server from. For details about editing the SERVICES file, see "Configuring the Services File" on page 99.

## 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. Here is the syntax for the COMAUX1= option:

```
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
```

### 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 “LIBNAME Statement” in *SAS/SHARE User's Guide* and “OPERATE Procedure” in *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 transfers between a client and a server will be encrypted. The RC4 encryption algorithm is assigned in the NETENCRYPTALGORITHM= option. For details about encryption services, see *Encryption in SAS*, located in the Base SAS Help and Documentation.

### 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 “Configuring the Services File” on page 99.

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:

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

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

For details about SAS naming rules, see *SAS Language Reference: Concepts*. For details about LIBNAME, see "LIBNAME Statement" in *SAS/SHARE User's Guide*, and for details about PROC OPERATE, see "OPERATE Procedure" in *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 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. To allow a client to connect to a server that runs under z/OS, verify that the SAS SVC routine has been installed. See "Installation of the SAS SVC Routine" on page 34.
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.

### 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 99.

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 11.

### Encrypting Data in Client/Server 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 details about encryption services, see *Encryption in SAS*, located in the Base SAS Help and Documentation.

### 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.

Here is the syntax for the COMAUX1= option:

```
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
```

### 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 46. 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 "SERVER Procedure" in *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 computer 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

### Planning for TCP/IP

Here are the primary configuration issues to consider when preparing your site for TCP/IP to run under the z/OS operating environment.

1. For the IBM IP Communications Server, configure or verify the host name configuration by adding a HOSTNAME statement in the appropriate IBM TCPIP.DATA file.

   For information about TCP/IP stacks and how to determine whether a system uses single or multiple TCP/IP stacks, see "Configuring TCP/IP Stacks" on page 48.

2. Use the IBM z/OS Name Resolver for the resolution of domain names.

3. Verify that appropriate services are configured for SAS/CONNECT or SAS/SHARE in the SERVICES file. For details, see "The Services File" on page 58.

## 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 consists 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 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 IBM z/OS IP Communications Server TCP/IP package has been installed.

  *Note:* SAS supports any vendor's TCP/IP software that is functionally compatible with the IBM z/OS IP Communications Server package.

- 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).
- The IBM z/OS Name Resolver must be active.

## Configuring 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 z/OS operating environment 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*. These examples show typical entries in the BPXPRM*nn* PARMLIB member for INET and CINET physical file systems.

- Defining a single IBM TCP/IP stack

  This 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 IBM TCP/IP stack that supports IPv4 and IPv6

  This 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 that enables IPv4 and IPv6. For details, see "About TCP/IP Internet Protocol (IP) Addressing" on page 4.

```
FILESYSTYPE TYPE(INET) ENTRYPOINT(EZBPFINI)
    SUBFILESYSTYPE NAME(TCPIP)
  TYPE(INET)
  ENTRYPOINT(EZBPFINI)
   NETWORK DOMAINNAME(AF_INET)
      DOMAINNUMBER(2)
      MAXSOCKETS(64000)
      TYPE(INET)
   NETWORK DOMAINNAME(AF_INET6)
      DOMAINNUMBER(19)
      TYPE(INET)
```

- Defining multiple IBM TCP/IP stacks

  This example shows the statements in the IBM PARMLIB BPXPRM*nn* member that define a multiple TCP/IP stack system. This example includes two IBM stacks, TCPIP and TCPIP2.

  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.

```
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)
```

For details, see "References" on page 59.

### System and Process Limits

These IBM system values are set in the PARMLIB member BPXPRM*nn* and affect the number of TCP/IP sockets that SAS can use:

MAXSOCKETS
: is a system limit that 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
: is a process limit that specifies the maximum number of file descriptors that a single process can have open concurrently, such as all open files, directories, sockets, and pipes. IBM recommends that this value be set to 64000.

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

For details about MAXSOCKETS and MAXFILEPROC, see "References" on page 59.

## TCP/IP Host Name Configuration

### IP Addresses

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

**`getnameinfo()`**
: retrieves a string that contains its host name.

**`getaddrinfo()`**
: 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

When an IBM TCP/IP stack starts, the configuration process searches for the host name in the TCPIP.DATA data set. For details, see "TCP/IP Stack Configuration Files" on page 52.

- 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 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.

   For releases prior to z/OS 1.2, the stack's host name is set to a NULL string.

   For 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 in which they appear in the data set.

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

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

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

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

  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.

  Therefore, it is critical that VMCF is running before any TCP/IP stacks are started.

- IBM System Name Considerations

  The 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 "References" on page 59.

## TCP/IP Stack Configuration Files

### About 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. Here are the configuration files:

- "IBM PROFILE.TCPIP File" on page 52
- "IBM TCPIP.DATA File" on page 53

### IBM PROFILE.TCPIP File

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
: is the 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 "References" on page 59.

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. When the PROFILE DD statement is specified, no dynamic allocation is performed.

SAS does not 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 operates.

### IBM TCPIP.DATA File

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.

DOMAINORIGIN (or DOMAIN)
specifies the name of the domain origin, which is appended to the host name to form the fully qualified domain name of the host.

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

For details about the IBM TCPIP.DATA statements, see "References" on page 59.

The IBM TCP/IP stack and the IBM Communication Server applications, including the IBM z/OS Resolver, 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. *userid*.TCPIP.DATA
6. SYS1.TCPPARMS(TCPDATA)
7. DEFAULTTCPIPDATA value ( z/OS 1.2 and later releases)
8. TCPIP.TCPIP.DATA

### Host Name Resolution for Systems That Run Multiple TCP/IP Stacks

There is usually a one-to-correspondence between a TCP/IP stack and its host name. The host name is obtained using the **`gethostname()`** function. However, for systems that run using multiple TCP/IP stacks, the identity of the stack's host name is ambiguous. Here is the process for resolving the host name for a multiple TCP/IP stack system:

- The process that calls **`gethostname()`** might be bound to a specific TCP/IP stack. The binding is referred to as TCP/IP stack affinity. If affinity to a specific TCP/IP stack has been established, the **`gethostname()`** function returns the host name for the specific stack.

  There are several methods of setting the TCP/IP stack affinity. SAS uses the UNIX System Services call, **`pfsctl(BPX1PCT)`**. For details, see "Determining SAS TCP/IP Stack Affinity" on page 53.

- Otherwise, the process returns the host name of the default TCP/IP stack. For details about the default stack, see "Configuring TCP/IP Stacks" on page 48.

### Determining SAS TCP/IP Stack Affinity

The SAS TCP/IP library uses the UNIX System Services call **`pfsctl(BPX1PCT)`** to determine whether the z/OS system is running a single TCP/IP stack environment (INET) or a multiple TCP/IP stack environment (CINET). Here is the process:

- If the **pfsctl()** call returns zero, the z/OS System is running an INET environment.
- If the **pfsctl()** call returns the number of CINET TCP/IP stacks and their names, the z/OS System is running a CINET environment.

  1. The SAS TCP/IP library searches for the SAS environment variable TCPIPMCH in the data set that is specified by the ddname TKMVSENV.

     For details about specifying the TCPIPMCH environment variable and the TKMVSENV data set, see "SAS Environment Variables" on page 54.

     - If the TCPIPMCH environment variable is set to a value of "*", the SAS TCP/IP library does not attempt to set the TCP/IP stack affinity.
     - If the TCPIPMCH environment variable is set to a valid stack name, the SAS TCP/IP library sets the TCP/IP stack affinity to this value.
     - Otherwise, the TCP/IP stack affinity is set to the first TCP/IP stack name that was returned by the previous call to **pfsctl()**.

  2. The SAS TCP/IP library resets the stack affinity by making another call to **pfsctl()** using the appropriate flags and specifying the TCP/IP stack name.

     The value that is passed to **pfsctl()** must match the value of the NAME substatement, which is included in the SUBFILESYSTYPE statement, which is defined in the CINET PFS TCP/IP stack in the IBM BPXPRM*xx* PARMLIB member.

### TKMVSENV Data Set

A SAS data set, referred to as the TKMVSENV data set file, can be used to specify SAS environment variables. If you use SAS environment variables, you must allocate the TKMVSENV DD in the JCL or CLIST that executes SAS. For example, here are the allocation statements for the data set SAS.DATA.TKMVSENV, which contains the desired environment variable information:BATCH statement:

```
//TKMVSENV DD DISP=SHR,DSN=SAS.DATA.TKMVSENV
```

*Note:* Line numbering in the TKMVSENV data set must be disabled.

TSO statement:

```
ALLOC F(TKMVSENV) DA('SAS.DATA.TKMVSENV') SHR
```

Each logical record contains an environment variable assignment in this format: **SET** *environment_variable_name=value*.

### SAS Environment Variables

Environment variables, which are specified in the TKMVSENV file using the SET command, are used to customize TCP/IP for SAS.

_BPXK_SETIBMOPT_TRANSPORT
: controls which TCP/IP stack the metadata server binds to when running in a 64–bit multi-stack environment. When you are run a 64-bit SAS Metadata Server in a multiple TCP/IP stack environment, the metadata server binds to every available stack on the system. To control which stack the metadata server binds to, use the LE environment variable, _BPXK_SETIBMOPT_TRANSPORT=, rather than the TCIPMCH environment variable, to specify the stack name and set stack affinity.

See the "TCPIPMCH" on page 55 for the equivalent TKMVSENV option for 31-bit SAS.

Example:

```
set _BPXK_SETIBMOPT_TRANSPORT="stack-name"
```

TCPIPMCH
for 31-bit SAS, specifies the IBM TCP/IP stack name to set the stack affinity for z/OS systems that are running more than one TCP/IP stack. The TCPIPMCH SAS environment variable is useful for running multiple TCP/IP packages: either multiple TCP/IP vendor packages or multiple instances of the same vendor's TCP/IP. The TCPIPMCH environment variable is used to specify the TCP/IP stack name, such as a started task. Setting this environment variable is the equivalent of the TCPIPJOBNAME and TCPIPUSERID configuration keywords within the IBM TCPIP.DATA file. If the default value for the TCP/IP stack name is not specified, the value is the first TCP/IP stack that is defined to the system.

Example:

```
set _TCPIPMCH="stack-name"
```

See the "_BPXK_SETIBMOPT_TRANSPORT" on page 54 environment variable for the equivalent TKMVSENV option for 64–bit SAS.

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 that you can specify in the SIGNON statement or as a SAS option. For details, see "TBUFSIZE= System Option" in *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 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. If the TCPMSGLEN environment variable is not set, SAS uses the TCP stack's default size and allows autotuning if implemented by the stack.

Example:

```
set TCPMSGLEN=65536
```

TCP_POLL_INTERVAL
used to ensure responsiveness of SAS spawners and servers to various conditions outside of normal request processing. When idle, servers and spawners periodically awaken to check for requests. The interval in seconds for this check is governed by the TCP_POLL_INTERVAL= environment variable. Generally the default setting of 60 seconds should be acceptable. However, if you wish to configure the interval, set it in the TKMVSENV file by specifying the TCP_POLL_INTERVAL= variable.

Example:

```
set TCP_POLL_INTERVAL=50
```

A value of zero means the server will remain idle and only awaken for request processing. An idle server might be subject to S522 (Job Wait Time-out) abend. However, a spawner defined as an MVS started task or as a UNIX System Services daemon process should not be subject to idle wait termination. NI

CONNECTWDWAIT
used to limit the possibility that a client session disconnect might orphan a runaway DMR mode session. SAS starts a 'watchdog' thread to monitor the connection. The default interval is five seconds. If a disconnect occurs, CONNECTWDWAIT will check 18 times and then terminate the DMR thread (for a default elapsed time of 90 seconds). Setting the CONNECTWDWAIT value to zero means the process will not monitor the connection.

Example:

```
set CONNECTWDWAIT=10
```

## Setting Stack Affinity With 64-Bit SAS

If you are running the 64-bit SAS Metadata Server in a multiple TCP/IP stack environment, the server binds to every available stack on the system by default. To specify the name of the TCPIP stack that the metadata server binds to, use the _BPXK_SETIBMOPT_TRANSPORT environment variable.

Set the _BPXK_SETIBMOPT_TRANSPORT variable in the ***z64-saspref ix*.TKMVSENV** file as follows:

```
set  _BPXK_SETIBMOPT_TRANSPORT="stack-name"
```

For information about setting TCPIPMCH, see "Configuring SAS to Use the IBM z/OS Name Resolver" on page 58.

For more information about environment variables in the TKMVSENV File, see "TKMVSENV File" in *SAS Companion for z/OS*.

## TCP/IP Name Resolver Configuration

### Name Resolver: Definition

A name resolver is a set of routines that acts 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 **`getnameinfo()`** and **`getaddrinfo()`**.

A name resolver must be configured for each host. Here are the locations for UNIX configuration files:

**`/etc/hosts`**
contains the local host configuration data.

**`/etc/resolve/conf`**
contains the DNS domain name and the name servers' IP addresses.

**`/etc/service`**
contains the service configuration data.

### IBM z/OS Name Resolver

Starting with z/OS V1R4, IBM introduced support for Internet Protocol Version 6 (IPv6), which is the successor to the Internet Protocol Version 4 (IPv4). In order to support IPv6, new USS BPX calls for the IBM name resolver were introduced to implement the protocol-independent resolver functions that are described in the RFC 3493 specification. Here is a list of the IBM name resolver functions that are supported in IPv6 and IPv4:

***Table 4.2*** *IBM Name Resolver Functions*

| IPv6 | IPv4 |
|---|---|
| getnameinfo(BPX1GNI) | gethostbyname() |
| getaddrinfo(BPX1GAI) | gethostbyaddr() |

When the IBM z/OS Name Resolver is started, it reads the 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 reads any name resolver directives that are located in this global TCPIP.DATA file. The IBM z/OS Name Resolver then searches for a local TCPIP.DATA file in this 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

Here are some useful IBM z/OS Name Resolver Server directives:

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*.

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

SAS uses the IBM z/OS Name Resolver by default. No configuration is necessary unless SAS is running under a multiple TCP/IP stack system (CINET). If SAS is running under a multiple TCP/IP stack system, the SAS TCP/IP library will need to set the TCP/IP stack affinity. Specify this SAS environment variable:

```
set TCPIPMCH=stack-affinity
```

TCPIPMCH is a SAS environment variable that is used to specify the name for the TCP/IP stack, which is also known as a started task. TCPIPMCH is equivalent to the TCPIPJOBNAME and TCIPPUSERID configuration keywords that are used in the IBM TCPIP.DATA file.

If a value is not specified for TCPIPMCH, the SAS TCP/IP library will use the first TCP/IP stack name that is returned by a call to the UNIX System Service, **`pfsctl()`**, which might not be the appropriate TCP/IP stack. For information, see "TCPIPMCH" on page 55.

## 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 99.

### Configuring SAS Services

A service for each SAS server session ( SAS/CONNECT or SAS/SHARE) must be defined in the SERVICES file on each computer that a SAS client session runs on.

*Note:* Some TCP/IP stacks can restrict the range of ports that can be used. For details, see "TCP/IP Stack Configuration Files" on page 52.

### The Services File Search Order

The z/OS Name Resolver searches for the SERVICES file, using this 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

## Documentation

The IBM documentation is also available from http: publib.boulder.ibm.com.

## References

Albitz, Paul and Cricket Liu. 2001. *DNS and BIND. 4th Ed*. Cambridge, MA: O'Reilly Media.

IBM 2010. *z/OS V1R11.0 Communications Server IP Configuration Guide*. 17th ed. RTP, NC: IBM.

IBM 2007. *z/OS V1R9.0 UNIX System Services Planning (GA22-7800-12)*. RTP, NC: IBM.

*Chapter 5*

# z/OS: XMS Access Method



## Prerequisites for Using XMS under z/OS

### *Task List*

- Verify that software requirements are met.

- Verify that the SAS SVC routine has been installed.
- 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 the following requirements are met:

- 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.

### Installation of 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 SAS 9.4 of SAS software, do not repeat the step here. If you need to perform the installation, see the *Configuration Guide for SAS 9.4 Foundation for z/OS* (PDF) on the SAS Install Center web page.

   Because SAS SVC in SAS 9.4 is backward compatible, it replaces the SAS SVC routines from previous releases. You can continue using previous releases of Base SAS, SAS/CONNECT, and SAS/SHARE with the SAS 9.4 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 how 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 systems, as needed.

   If you have more than one z/OS system, verify that the SAS SVC is installed on all the systems that will be running SAS/CONNECT or SAS/SHARE at your site.

### 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 69.

### SAS/CONNECT and SAS/SHARE Network Security

"Installing the SAS SVC Routine" on page 68. 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.

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 69.

Defining an inactive subsystem causes a z/OS computer 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 symmetric multiprocessor (SMP) computer.

### 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 multiprocessor computer 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 Multiprocessor Computer

### Tasks for Signing On to an SMP Computer

If your client computer is equipped with SMP, and if you want to run one or more server sessions on your computer, 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;
```

You can also specify it 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 multiprocessor computer.

*Note:* PROCESS=, REMOTE=, CREMOTE=, and CONNECTREMOTE= can be used interchangeably. For details, see "CONNECTREMOTE= System Option" in *SAS/CONNECT User's Guide*.

For details about SIGNON, see SIGNON statement.

### 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 multiprocessor computer.

The SASCMD option can be specified in an OPTIONS statement:

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

You can also specify it 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 before 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 "SASCMD= System Option" in *SAS/CONNECT User's Guide* and the SIGNON statement.

### *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 multiprocessor computer.

```
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 multiprocessor computer. 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.

Here is the syntax for the COMAUX1= option:

COMAUX1=*alternate-method*

Example:

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

### Specifying the Server

To use the XMS access method, a client and a server must run on the same computer (or node) that runs the z/OS operating environment.

In most cases, when using the XMS access method, you can specify the server using only a server ID, because the client and server sessions run on the same node. However, if you specify XMS as the primary access method and TCP as the auxiliary access method in the client session, you must use a two-level server name to accommodate the server-naming requirements of TCP.

Here is the format for the two-level node name:

SERVER=*node.server-ID*

The *server-ID* can be a maximum of eight characters in length, and it must be identical to the service name that is specified in the SERVICES file. For details, see "Configuring the Services File" on page 99. For details about creating valid SAS names, see "Rules for Words and Names in the SAS Language" in *SAS Language Reference: Concepts*.

Example 1:

```
options comamid=xms comaux1=tcp;
libname mylib '.' server=srvnode.share1;
```

In this example, if a client/server connection succeeds using the XMS access method, the node part of the server ID is ignored and only the server ID is used. However, if the client/server connection fails using XMS, but succeeds using TCP, the two-level server name is required in order for the SERVER procedure statement to succeed. For details about the COMAUX1= option, see *SAS/SHARE User's Guide*. For details about the

LIBNAME statement and the OPERATE procedure, see “LIBNAME Statement” in *SAS/SHARE User's Guide* and “OPERATE Procedure” in *SAS/SHARE User's Guide*.

Example 2:

```
options comamid=xms comaux1=tcp;
libname mylib '.' server=srvnode.__5000;
```

In this example, a port number rather than a server ID is specified. Two consecutive underscores (no spaces) precede the port number.

*Note:* A port number is valid for the TCP access method, but not for the XMS access method. If XMS is used, a port specification results in a failure. If TCP is used, a port specification is valid.

If the node name is not a valid SAS name, the node name can be assigned to a SAS macro variable. The macro variable is substituted for the node name 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 3:

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

This example shows the assignment of only the node name to a macro variable.

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

Example 4:

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

This example shows the assignment of the node name and the server ID to a macro variable.

### 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 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 SAS 6.09, do not repeat the step now.

   If you need to perform the installation, see the *Configuration Guide for SAS 9.3 Foundation for z/OS*.

   Because SAS SVC in SAS 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 SAS 6.09 SAS SVC that is installed on your computer.

2. Verify the SAS options for the SVC routine.

   You must verify that SAS for the SVC routine accurately reflects how 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 computers 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
```

### Specifying a Server Name

To use the XMS access method, a server and a client must run on the same computer 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 eight characters in length.

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

### SAS/SHARE Server Example

The following example shows the statements that you specify to start the server. A two-level server name, such as RMTNODE.SHARE1, is used in the z/OS operating environment. The XMS access method is assumed by default.

```
proc server id=rmtnode.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 69.

   *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 70.

### Steps for Installing the Load Module

*Note:* This task is required.

To use the cross-memory access method for communication between a SAS/CONNECT server and user, you must copy the module SASVXMS0 from the SAS load library data set into an authorized library. You must then rename this module SASVXMS (removing the 0). It is very important that you perform these two tasks in that order.

When SAS/CONNECT software loads the module SASVXMS, it must find that module to be marked authorized, re-entrant, and reusable, and to have been loaded from an authorized library. The version of SASVXMS that was distributed with Version 6 of SAS/SHARE software can be used ONLY with Version 6.

To install the SASVXMS0 load module, perform these tasks:

1. Copy SASVXMS0 into an authorized link list library.

   In a production environment, SAS recommends that you copy the SASVXMS0 module into an authorized link list library. Alternatively, you can install this module into the link pack area. You can use any standard utility program to copy the SASVXMS0 module from your &prefix.LIBRARY data set to your authorized library.

   Note: A user abend 984 occurs if the SASVXMS module is not installed in an authorized library or the library is in a STEPLIB concatenation where one of the libraries is not authorized

2. Rename SASVXMS0.

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

   Rename SASVXMS0 to SASVXMS. Specify the SAS 9.4 Foundation option COMAMID=XMS.

   *Note:* The XMS access method does not support communication between a Version 6 SAS session and a SAS/SHARE 9.4 server, nor does it support communication between a SAS 9.4 Foundation session and a Version 6 SAS/SHARE server. If you want to run SAS/SHARE 9.4 and Version 6 SAS/SHARE software concurrently, you MUST rename the Version 6 SASVXMS0 module to SASVXMSn and set COMAMID=XMSn in Version 6.

## Defining an Anchor Point

### Anchor Point: Definition

1. Define an inactive z/OS subsystem. The anchor point is specified by defining an inactive z/OS subsystem. Defining an inactive subsystem 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.

   Note that, although you define a subsystem to z/OS, it will never be considered active and will provide no system services because the **SSCTSSVT** field of the SSCVT will never be nonzero. You can define the inactive subsystem by adding an entry to any of the following:

   - the IEFJSSNT member of SYS1.LINKLIB
   - an IEFSSNxx member of SYS1.PARMLIB

   Consult the z/OS system initialization and tuning documentation for the details of each alternative. Regardless of which method you choose, you must include the subsystem name and you must not specify an initialization routine name. Use the name SAS0 unless it conflicts with standards or conventions at your site.

2. Set the SAS 9.4 Foundation option SUBSYSID= to specify the inactive subsystem that you defined. The name you specify for the inactive subsystem defined as the anchor point for the cross-memory access method must also be specified as the value of the SAS 9.4 Foundation option SUBSYSID=. This option is specified, typically in a SAS 9.4 Foundation configuration file, by the SAS/SHARE software consultant.

# Part 3

# Spawners and Services File

*Chapter 6*

# Introduction to the SAS/CONNECT Spawner



## Spawner Definition

A SAS spawner is a program that starts a SAS session on the server on behalf of a connecting client. The SAS/CONNECT spawner runs on the SAS/CONNECT server, listens for requests, and opens a connection to the server on behalf of the client. Signing on to the SAS/CONNECT spawner is an alternative to signing on to a server by using a Telnet daemon. The SAS/CONNECT spawner can listen for requests on multiple ports that are defined in metadata and on a single port that is defined on the spawner invocation command.

Spawner invocation options enable you to start and manage the SAS/CONNECT spawner. These invocation options consist of options specific to the SAS/CONNECT spawner as well as Base SAS system options. For more information about these options, see “General Spawner Options” on page 88.

# 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 during sign-on. Without encryption, the user ID and password would pass through the network as clear, readable text, which presents a security risk.

To encrypt all data that flows through the network after sign-on (such as data being processed by remote submits and data transfers), you must use a security service. For details about security services that are supported in SAS 9.4, see *Encryption in SAS*.

### 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 computer's security and economizes resources. For details, see Chapter 9, “Configuring SAS/CONNECT for Use with a Firewall,” on page 103.

### Eliminates the Need for a Sign-On Script

The primary purpose of a sign-on script is to do the following:

- 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.

# Using SAS Management Console to Administer the SAS/CONNECT Spawner

### Overview

SAS Management Console is the primary administrative user interface for administering SAS servers in the SAS Intelligence Platform. SAS Management Console includes a variety of plug-ins that are used to create and maintain various resources available in the SAS Intelligence Platform. The Server Manager plug-in is used to manage the SAS/CONNECT spawner and SAS/CONNECT server.

The next few sections offer a brief introduction to the spawner management interface in SAS Management Console. Complete documentation for SAS Management Console and the Server Manager plug-in can be found in the following SAS Intelligence Platform documents:

- *SAS Intelligence Platform: Overview*

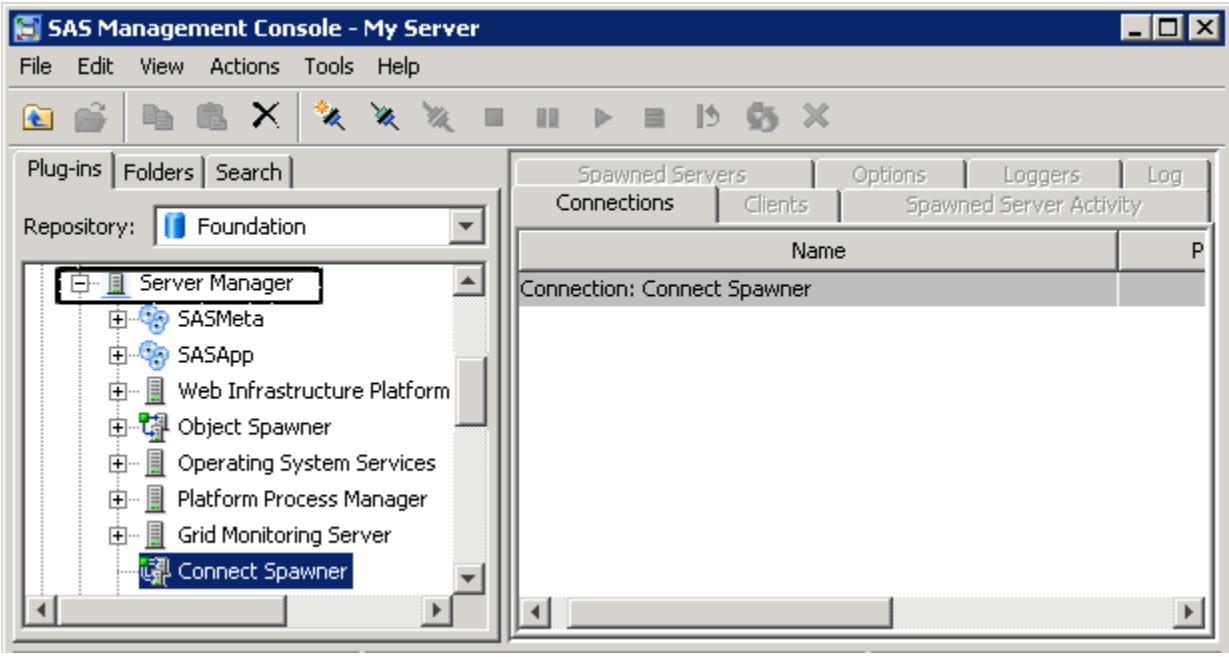
- *SAS Intelligence Platform: System Administration Guide*
- *SAS Management Console: Guide to Users and Permissions*
- *SAS Intelligence Platform: Application Server Administration Guide*

These documents can be found on the SAS Intelligence Platform Product Documentation page at http://support.sas.com/documentation/onlinedoc/intellplatform/.

## Using the Server Manager Plug-in to Access the SAS/CONNECT Spawner

The SAS/CONNECT spawner is located in the SAS Management Console - Server Manager plug-in. You can access the SAS/CONNECT spawner by expanding the Server Manager node and selecting the SAS/CONNECT Spawner, as shown in Figure 6.1.

***Figure 6.1*** *SAS/CONNECT Spawner in the Server Manager Plug-in of the SAS Management Console*



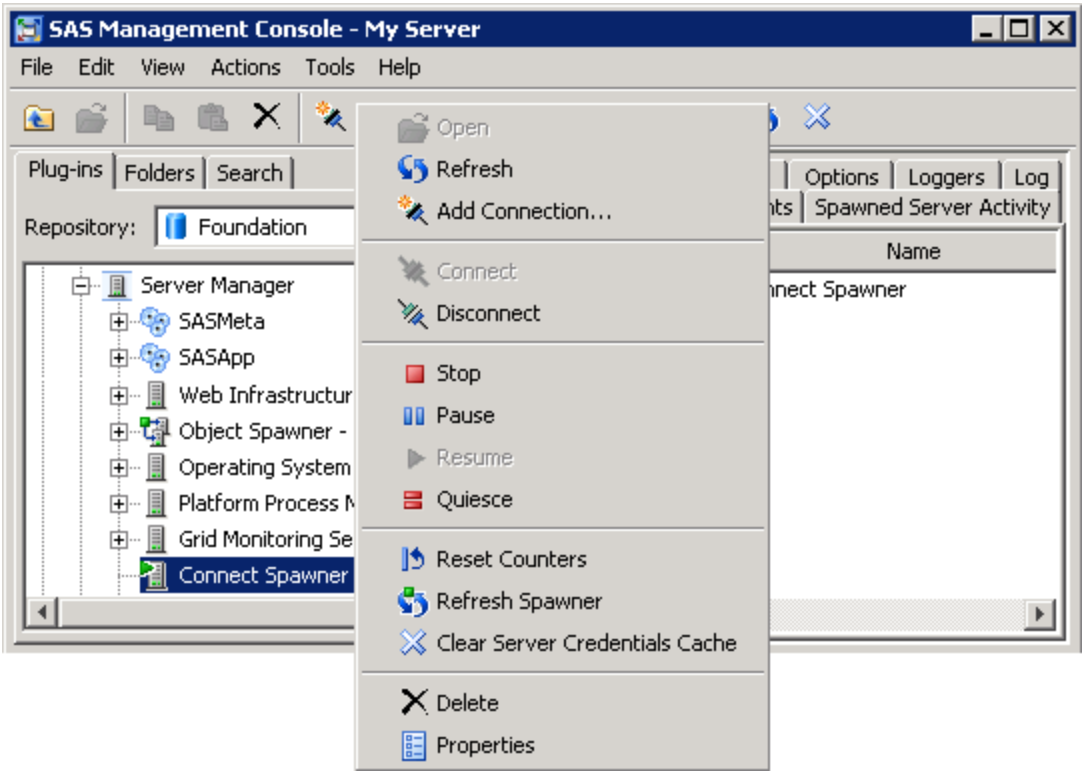

You can use SAS Management Console to perform the following spawner operations, as shown in Figure 6.2 on page 76.

- add a connection to the spawner
- connect to or disconnect from the spawner
- stop, pause, resume, or quiesce the spawner
- reset the counters
- refresh the spawner
- clear the spawner credentials cache
- delete the spawner
- view the properties of the spawner

Until you connect to the spawner, most of the tasks listed in the pop-up menu (for example, stop, pause, and quiesce the spawner) are disabled. After you have connected,

these tasks are enabled. Figure 6.2 on page 76 shows these tasks enabled in the pop-up menu after connecting to the spawner.

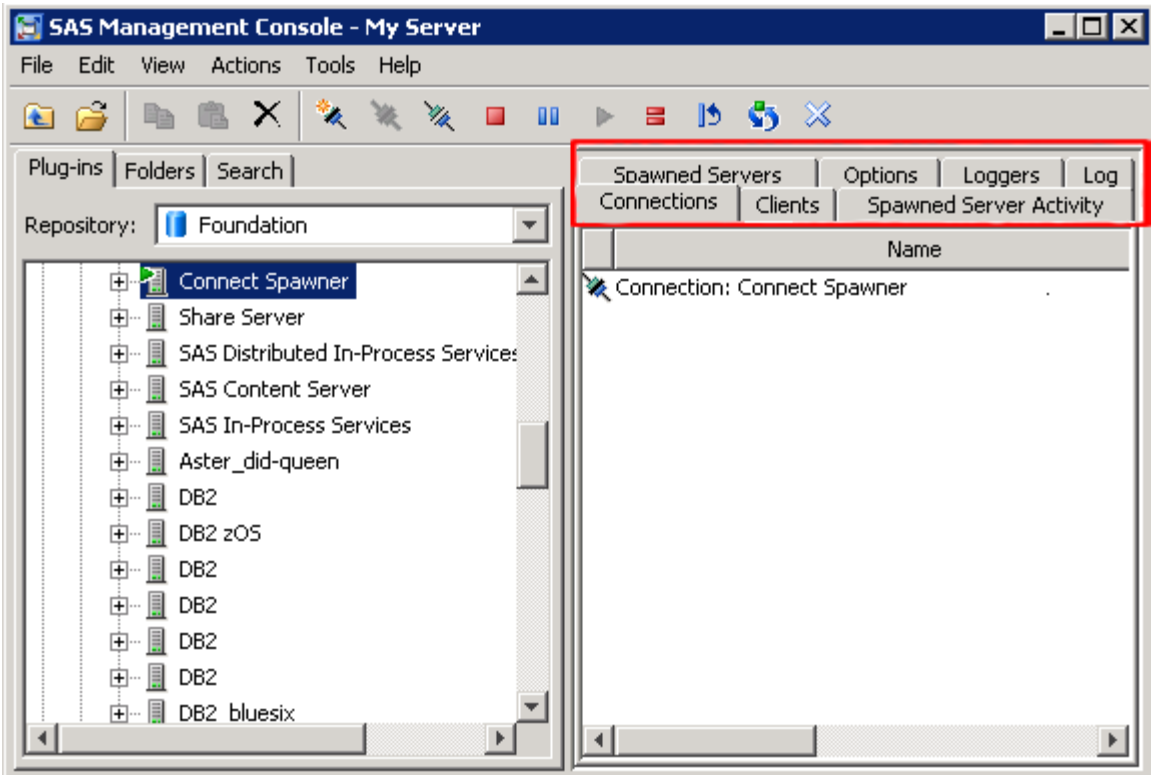
***Figure 6.2*** *Tasks Available for Operating the SAS/CONNECT Spawner, Once Connected*




Once you are connected to the spawner, tabs that contain information about the spawner become active in the right pane of the Server Manager window. Many of the fields in these tabs not only display information about the spawner, but they also enable you to update and configure spawner and server settings.

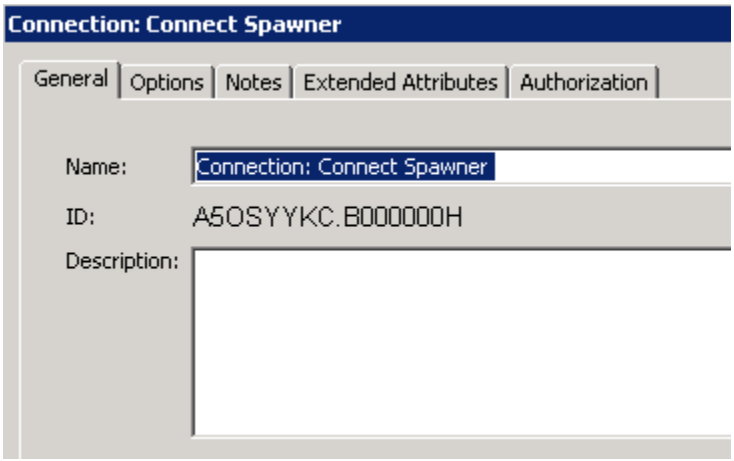
- Connections Tab — displays a list of the active, inactive, or terminated sessions
- Clients Tab — displays a list of clients that are or have been connected to the currently selected spawner
- Spawned Server Activity Tab — displays graphs showing the spawned server activity
- Spawned Servers Tab — displays a list of spawned servers and the user name and host associated with the spawned server
- Options Tab — displays the options and properties associated with the spawner
- Loggers and Log Tabs — enables you to view and update information about the loggers, log events, and associated logging levels

***Figure 6.3*** *Tabs for Monitoring the SAS/CONNECT Spawner*




In addition, when you right-click on the spawner in the Connections Tab in the right pane of the Server Manager Console, Spawned Servers Tab, the Connection Properties window is displayed, as shown in Figure 6.4. This window contains a series of tabs that enable you to view and update authentication properties, the spawner name, the spawner description, extended attributes, and authorization properties. The Options Tab enables you to change the operator port and specify advanced encryption options, such as authentication type and authentication domain.

***Figure 6.4*** *SAS/CONNECT Spawner: Server Manager Connection Properties Tabs*




For information about using SAS/CONNECT spawner invocation options, see "Options for Starting and Managing the SAS/CONNECT Spawner" on page 88.

## Using PROC IOMOPERATE to Configure the SAS/CONNECT Spawner

PROC IOMOPERATE, which administers SAS servers that support the SAS IOM infrastructure, supports the management of the SAS/CONNECT Spawner. Here are a few of the PROC IOMOPERATE statements that are available for managing the SAS/CONNECT spawner:

- LIST DEFINED SERVERS
- LIST SPAWNED SERVERS
- REFRESH CONFIG
- STOP SPAWNED

The syntax for connecting to the SAS/CONNECT spawner using PROC IOMOPERATE is as follows:

**PROC IOMOPERATE**;
CONNECT <*connect-options*>;
<*optional-argument(s)*>;
QUIT;

For documentation and syntax for the IOMOPERATE procedure, see *SAS Intelligence Platform: Application Server Administration Guide* on the SAS Intelligence Platform Product Documentation page.

## Operating Environment Support for Spawners

SAS 9.4 supports TCP/IP spawners under these operating environments:

- "SAS/CONNECT Spawner in UNIX" on page 84
- "SAS/CONNECT Spawner in Windows" on page 85
- "SAS/CONNECT Spawner in z/OS " on page 86

For a list of all available spawner invocation options, see "Options for Starting and Managing the SAS/CONNECT Spawner" on page 88.

## Client Connection Using a Spawner

- The spawner service can be configured in the client's SERVICES file, but this is not a requirement. The spawner can use a port number or have the port information (service or port) defined in metadata. You can explicitly specify the port on the command line and in the SIGNON statement. For information about configuring the SERVICES file, see "Configuring the Services File" on page 99.
- 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. Windows users can use SSPI if it is enabled. See "SSPI" on page 31 for information about SSPI.

- If you do not use a script file, use the following syntax to connect to the spawner:

```
options remote=spawner-ID;
signon user=user-ID password=password;
```

In the UNIX and Windows environments, the TCP/IP access method is used by default. The spawner connects the client to the spawner named SPAWN that runs on the UNIX node RMTHOST. If there is no 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 sign-on. The SIGNON statement makes the connection and starts a SAS session on the server.

# Spawner Connection Examples

## List of Examples

The following examples show how to manually configure the spawner. Most spawners now are deployed using the SAS Deployment Wizard and manual configuration is not required. Information used by the spawner is contained in metadata.



## Scripted Sign-on to a UNIX Spawner

### Server

From the UNIX node that the server runs on, specify the following command to start the spawner.

```
cntspawn -service spawn1 -mgmtport 5030
```

The -MGMTPORT option specifies the operator port to be used for administrative purposes. The -SERVICE option specifies the name of the service, **spawn1**, that listens for incoming server requests. The -service option can specify a defined TCP/IP service or a numeric port value. What is used when the spawner is started determines what will be used by the client. In the following example the-SERVICE option is used to specify the numeric port value of the service during spawner start-up:

```
cntspawn -service 5020 -mgmtport 5030
```

A user can then sign on using the same port-number in the SIGNON statement:

```
filename rlink '!sasroot\connect\saslink\tcpunx.scr';
signon rmthost.5020;
```

The -SERVICE option values used to start the spawner determine what will be used by the client to sign on.

As in the first example, the -MGMTPORT option specifies the operator port to be used for administrative purposes.

### Client

At a Windows client, the statements shown in Example Code 6.1 on page 80 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.

For more information, see "About TCP/IP Internet Protocol (IP) Addressing" on page 4. 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.

***Example Code 6.1*** *Scripted Sign-on to a UNIX Spawner: Client*

```
options comamid=tcp;
filename rlink '!sasroot\connect\saslink\tcpunx.scr';
signon rmthost.spawner;
```

## Scriptless Sign-on to a Windows Spawner That Runs as a Service

### Server

The following command installs the spawner service on a Windows computer:

```
cntspawn -install
```

For this example, note the following:

- The spawner is being installed as a Windows service, but since the -SERVICE option is not used to specify the port number or name, the spawner will listen on the default Telnet port (23) and be named SAS Connect Spawner by default.
- Because a sign-on script is not being used and the -SASCMD= option is not specified, the spawner will look for the SAS executable in the SAS installation directory.
- Since the -MGMTPORT is not specified, the operator port will default to 7541.

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 ConnectSpawner.bat script file command
- the NET START command
- the services applet
- a reboot of the computer

### 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 computer or a macro variable that contains the IP address of the Windows computer that the spawner runs on. The user ID and password to the server are 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_;
```

For Windows users, if SSPI has been enabled, then you do not need to specify the user ID and password in the SIGNON statement. See "SSPI" on page 31 for more information about SSPI.

*Note:* The password is displayed as Xs in the SAS log.

## Encrypted Sign-on to a z/OS Spawner

### Server

The following z/OS command starts 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.

Here is an example:

```
//SPAWNER  PROC PROG=CNTSPAWN,
//         SERVICE='spawner',
//         MGMTPORT=7541,
//         PARMFILE='SAS.SPAWNER.PARMS'
//*
//SPAWNER  EXEC PGM&PROG,REGION=40M,
//         PARM='-service &SERVICE -MGMTPORT &MGMTPORT=<//DDN:PARMS'
//*
//STEPLIB   DD DISP=SHR
//     DSN=<customer.high.level.pfx>.LIBRARY
//          DD DISP=SHR,
//     DSN=<customer.high.level.pfx>.LIBE
// 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
```

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 enables a client to sign on to a server multiple times using the same user ID and password.

The -MGMTPORT option specifies port 7451 as the port for operator connections.

### Client

In the following example, the client specifies a 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. You 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 7*

# Managing the SAS/CONNECT Spawner



## Managing the SAS/CONNECT Spawner

### *Overview*

Spawner invocation options consist of options that are specific to the spawner and several SAS system options that you can use to run and configure the spawner from the command line.

The SAS/CONNECT spawner and the command to start it have been unified into a single executable program with one set of options for all platforms (UNIX, Windows, and z/OS). These options can be logically grouped into the following categories:

- General spawner options
- Windows service options

- Open systems-specific options

The syntax for starting the SAS/CONNECT spawner is as follows:

**CNTSPAWN** <*options*>

The following sections provide platform-specific information for starting and managing the SAS/CONNECT spawner on a UNIX, Windows, and z/OS server.

# SAS/CONNECT Spawner in UNIX

## Location of the Spawner Executable in UNIX

The SAS/CONNECT spawner executable file, **cntspawn.exe**, is located in the directory in which SAS is installed at your site or on your computer. Throughout this document, this folder is referred to as the ***SAS-installation-directory***.

In UNIX the spawner executable is installed by default in the following location:

```
SAS-installation-directory/utilities/bin/cntspawn.exe
```

## Managing SAS/CONNECT Spawner in UNIX

### Spawner Security

If connecting to a UNIX server using 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 Spawner in UNIX

To start the SAS/CONNECT spawner in UNIX, specify the spawner invocation command as shown here:

```
cntspawn <options>
```

Example:

```
cntspawn -mgmtport 7555 -sascmd "/u/username/mystartup"
```

For a list of other available spawner invocation options, see "Options for Starting and Managing the SAS/CONNECT Spawner" on page 88.

If you are running the SAS/CONNECT spawner as a TCP/IP service and have configured the spawner service in the TCP/IP services file, then you can start the spawner using the -SERVICE option as shown in the following example:

```
cntspawn -service service-name -<options>
```

Example:

```
cntspawn -service mySpawner -mgmtport 7555 -sascmd "/u/username/mystartup"
```

For information about configuring the spawner service, see "Configuring the UNIX Spawner Service" on page 18.

### Stopping the Spawner in UNIX

To end the spawner, type CTRL-C to kill the process.

# SAS/CONNECT Spawner in Windows

## Location of the Spawner Executable in Windows

The SAS/CONNECT spawner executable file, **cntspawn.exe**, is located in the directory in which SAS is installed at your site or on your computer. Throughout this document, this folder is referred to as the ***SAS-installation-directory***.

In Windows the spawner executable is installed by default in the following location:

*SAS-installation-directory*\SASFoundation\9.4\cntspawn.exe

Here is an example of the spawner location in a typical SAS Foundation installation on Windows:

```
C:\Program Files\SASHome\SASFoundation\9.4\cntspawn.exe
```

## Managing the SAS/CONNECT Spawner in Windows

### User Rights in Windows

By default, when the SAS/CONNECT spawner is installed as a Windows service, it runs under the LocalSystem User ID, which has all the required User Rights for running the SAS/CONNECT spawner.

If you do not install the SAS/CONNECT spawner as a Windows service (that is, if you run it from your system prompt), the Windows User ID that is used to start the SAS/CONNECT spawner must be the local Administrator of the machine and must have the following User Rights:

- act as part of the operating system
- bypass traverse checking (the default is everyone)
- increase quotas
- replace a process level token
- log on locally (the default is everyone)

The Windows user ID specified at sign-on needs only the User Right “log on as a batch job.”

### Basic Tasks

1. Install the spawner as a Windows service.
2. Start the spawner in Windows.
3. Stop the spawner in Windows.
4. Uninstall the spawner as a Windows service.

### Installing the SAS/CONNECT Spawner as a Service in Windows

To install the SAS/CONNECT spawner on a Windows server, specify **cntspawn -install** from the Windows command line, using the full pathname to the executable in quotes, as shown here:

```
"SAS-installation-directory\SASFoundation\9.4\cntspawn.exe" -install
```

After running this command, Windows assigns the default name, **SAS Connect Spawner**, to the spawner service. You can assign a different name to the spawner service by using the -SERVICENAME option with the -INSTALL option on the CNTSPAWN command.

For more information about using the -SERVICENAME option, see -SERVICENAME in "Options for Starting and Managing the SAS/CONNECT Spawner" on page 88.

### Starting and Stopping the SAS/CONNECT Spawner in Windows

Once the spawner is installed, it can be started or stopped using any one of the following methods:

- specify the NET START or NET STOP command with the spawner service name:

```
NET START "SAS Connect Spawner"
NET STOP "SAS Connect Spawner"
```

- use the Windows Services Manager plug-in to start or stop the spawner service.
- type CTRL-C or double-click in the upper left corner of the window in which the spawner is running.

### Uninstalling the SAS/CONNECT Spawner in Windows

You can uninstall the SAS/CONNECT spawner by specifying the -UNINSTALL option with the CNTSPAWN command. Specify the full pathname to the CNTSPAWN executable in quotes, as shown here:

```
"SAS-installation-directory\SASFoundation\9.4\cntspawn.exe" -uninstall
```

If you used the -SERVICENAME option with the -INSTALL option when you installed the spawner to explicitly name the spawner service, then you must use the -SERVICENAME option with the -UNINSTALL option to uninstall it. Use quotes around the full pathname and around the spawner service name.

```
"SAS-installation-directory\SASFoundation\9.4\cntspawn.exe" -uninstall
-servicename "mySpawner"
```

For more information about using the -UNINSTALL option, see -UNINSTALL in "Options for Starting and Managing the SAS/CONNECT Spawner" on page 88.

## SAS/CONNECT Spawner in z/OS

### Setting Up the SAS/CONNECT Spawner in z/OS

The SAS/CONNECT spawner runs as a z/OS started task and uses z/OS UNIX System Services (USS) to start each user's SAS/CONNECT session. The spawner listens for requests from clients and passes them to the start-up command that is associated with the service and port in which there is activity. The start-up command starts a SAS/CONNECT server session.

The basic steps for starting the spawner and SAS server session in z/OS are as follows:

- Configure SAS TCP/IP. See "Prerequisites for Using TCP/IP under z/OS" on page 34.

- Create the spawner started task. See "Starting the Spawner in z/OS" on page 87 and z/OS Security Server (RACF) Command Language Reference from IBM.
- Create a Shell script that defines a SAS start-up command. For details, see "Starting SAS Using a Shell Script" on page 87.

### Location of the Spawner in z/OS

The SAS/CONNECT spawner module is CNTSPAWN and it is located in the SAS load module library. This library might be installed in LPA, LINKLIST, or allocated to the STEPLIB DD in the Spawner Started Task JCL. On z/OS systems, SAS recommends operating servers as started tasks. Therefore, to start the SAS/CONNECT spawner, you should create a started task procedure in the system PROCLIB library. A sample spawner started task procedure can be found in **'&prefix.BAMISC(SPNCCNTL)'**.

### Spawner User ID on z/OS

The z/OS system considers the SAS/CONNECT spawner a daemon process. Therefore, 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 SAS load module library must be program-controlled, but the spawner user ID does not require READ access to the BPX.DAEMON profile. The spawner no longer requires APF authorization.

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

### Starting the Spawner in z/OS

To start the z/OS spawner, use the following command:

```
START <started-task-name>
```

This command initiates the started task. The spawner started task requires a PARMS file. The PARMS file contains any spawner invocation options that you need to specify when starting the spawner on z/OS. A sample PARMS file can be found in **'&prefix.BAMISC(SPNCPARM)'**. Implementing SAS TSO Support

See Configuration Guide for SAS® 9.4 Foundation for z/OS for more information about the **'&prefix.BAMISC'**. library.

For a list of spawner invocation options, see "Options for Starting and Managing the SAS/CONNECT Spawner" on page 88.

### Starting SAS Using a Shell Script

The spawner uses UNIX System Services (USS) to invoke a USS file system shell script that starts SAS.

Example:

```
-sascmd "/usr/local/bin/spawnsas.sh -nosasuser -noterminal"
```

This command assumes that a shell script named **spawnsas.sh** is installed in **/usr/local/bin**. The command specifies the -NOSASUSER SAS system option . The

-NOTERMINAL option prevents the display of a dialog box in the server session. In addition, the two double quotation marks 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 sample UNIX shell script parses the command and interprets environment variables to build a TSO command to start SAS. This command is executed using either the USS **/bin/tso** command or the USS **/bin/tsocmd** command.

You must change the values of **&prefix** to the high-level qualifier of your CList library that contains the TSO command to start SAS. The BPX environment variables are specified to improve the start-up performance of a spawned SAS session.

A sample USS script file can be found in **'&prefix.BAMISC(SPNCSHEL)'**.

*Note:* The /bin/tso command mentioned above is used by default in the UNIX shell script. If you need to run authorized commands in SAS 9.3 and later releases, use the /bin/tsocmd command instead. See Usage Note 54530 for information about setting the /bin/tsocmd command in the UNIX shell script.

### Ending the Spawner on z/OS

To stop the spawner, do one of the following:

- use CTRL-C to kill the process
- if the spawner was started as a started task, specify **STOP** *<started-task-name>*

```
STOP  <started-task-name>
```

## Options for Starting and Managing the SAS/CONNECT Spawner

### Introduction

Spawner invocation options consist of SAS/CONNECT spawner options and SAS system options that you can use to run and configure the spawner from the command line. You can use these commands when you invoke the spawner using the CNTSPAWN command in Windows or UNIX, or in a z/OS PARMS file.

Spawner invocation options can be logically grouped into these categories:

- "General Spawner Options"
- "Windows Options"
- "UNIX and z/OS Options"

### General Spawner Options

Use the following general options with the CNTSPAWN spawner start-up command:

**-DEBUG**
turns on debug level output.

**-ENCRYPTFIPS**
specifies that the spawner uses FIPS compliant encryption to protect communications. When -ENCRYPTFIPS is specified, only SSL and AES encryption algorithms are valid.

**See** "ENCRYPTFIPS System Option" in *Encryption in SAS*

**Example** The following example enables SSL and AES encryption.

```
"SAS-installation-directory\SASFoundation\
    9.4\cntspawn.exe" -encryptfips
```

**-HELP**
specifies to print the Help message.

**-LOG | -LOGFILE <*filename*>**
specifies the filename to use for spawner log output if you are not using the -LOGCONFIGLOC option. The -LOG option should not be used with the -LOGCONFIGLOC option. If both options are specified, the -LOGCONFIGLOC option takes precedence.

You can specify the -DEBUG or -TRACE options with the -LOG <*filename*> option to have detailed spawner log messages sent to a log file.

**Example** The following example uses the ConnectSpawner.sh script on UNIX to start the SAS/CONNECT spawner and specifies that debug-level log messages are sent to the *unxspawner.log* file.

```
ConnectSpawner.sh  -start -debug -log unxspawner.log
```

**-LOGCONFIGLOC <*filename*>**
enables the SAS logging facility for SAS servers and names the location of the configuration file that is used by the SAS logging facility to create spawner log output. The configuration file is an XML file that specifies and configures loggers and appenders for the SAS/CONNECT spawner. The SAS Deployment Wizard automatically creates an initial logging configuration file for the spawner named **`logconfig.xml`** that you can modify as needed to adjust your logging configuration. The file is located in the *sas-installation-directory*`/Lev-`*n*`/ConnectSpawner/` directory on UNIX and the *sas-installation-directory*`\Lev-`*n*`\ConnectSpawner\` directory on Windows. The file contains the pattern layout for the messages that are generated and automatically directed to an output device, such as a console or a log file. Relevant log data for the Windows spawner might include the date and time, the log level, the thread ID, and the logger.

See "Sample SAS/CONNECT Spawner logconfig.xml File" on page 95 for an example of a spawner log configuration file in the UNIX environment.

The file-specification that defines the location of the XML configuration file must be a valid filename or a path and filename for your operating environment. If the path contains spaces, enclose the file-specification in quotation marks.

**Note** If -LOGCONFIGLOC is specified, spawner messages are routed by default to the App.Connect.Spawner logger.

**See** For information about using the -LOGCONFIGLOC option in the SAS Logging Facility, see "LOGCONFIGLOC= System Option" in *SAS Logging: Configuration and Programming Reference*.

**Example** The following spawner start-up command invokes the SAS Logging Facility and specifies the name and location of the logging configuration file, *winspawnerlog.xml*.

```
"SAS-installation-directory\SASFoundation\
   9.4\cntspawn.exe" -logconfigloc winspawnerlog.xml
```

**Example** "Sample SAS/CONNECT Spawner logconfig.xml File" on page 95

**-MGMTPORT**
enables you to specify the service name or port number that will listen for operator connections. Operator connections are connections made through the operator port. The operator port is a unique port number that is used for administrative purposes.

**Range** 1- 65535

**-METAENCRYPTALG *algorithm* | *NONE***
specifies the type of encryption algorithm to use when communicating with the metadata server. The following algorithms can be used: RC2, RC4, TripleDES, SAS Proprietary, and AES.

**-METAENCRYPTLEVEL *<level>***
specifies the level of encryption when communicating with the metadata server.

**-METAPASS**
specifies the password of user who is to connect to metadata server.

**-METAPORT *<port>***
specifies the port to connect to on metadata server.

**-METASERVER *<host>***
specifies the name or IP address of the metadata server.

**-METAUSER *<user-id>***
specifies the user ID of the user who is to connect to metadata server.

**-NETENCRKEY *<keysize>***
specifies the number of bits in data encryption keys.

**-NETENCRYPT *<algorithm>***
specifies that network encryption is required.

**-NETENCRYPTALGORITHM *<algorithm>***
specifies the name of encryption algorithm.

**-NOCLEARTEXT**
prevents sign-ons from clients that do not support user ID and password encryption. This option prevents clients that are running older releases (prior to SAS 6.09E and SAS 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 plaintext 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 sign-on from clients that use scripts, and allows sign-on 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 or to commands defined in metadata. Specifying -NOSCRIPT restricts clients from specifying additional options in SAS start-up commands or

script files. When -NOSCRIPT is specified, either -SASCMD must also be specified or logical Connect Servers must be defined in metadata.

*Note:* If a scriptless server defined in metadata does not have a valid spawner SASCMD value, the logical server will be ignored.

**-SASCMD | -CMD <*command*>**

**Windows**

specifies the SAS command or a command file that invokes SAS when a client attempts to connect to a server using the port defined by the -SERVICE command. The -SERVICE option specifies an alternate port that the spawner uses to listen for incoming requests for connection.

In Windows, you can use either a batch file, which is signified by the .bat extension, or a command file, which is signified by the .cmd extension. Here is an example of a batch file:

```
cd !sasroot
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.

**UNIX**

specifies the SAS command or a command file 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.

Here is a sample UNIX command file:

```
#!/bin/ksh
#----------------------------------
# mystartup
#----------------------------------
. ~/.profile
sas -noterminal -nosyntaxcheck $*
#-----------------------------
```

*Note:* The $* positional parameter enables you to specify additional SAS options when you invoke SAS.

**z/OS**

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 sign-on 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.

For an example of starting the spawner in z/OS, see "Encrypted Sign-on to a z/OS Spawner" on page 81.

Use the -SASCMD option to do the following

- 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 sign on using the SAS/CONNECT spawner:

```
-DMR -COMAMID<access-method> -NOSPLASH -ICON -NOTERMINAL
```

**-SASDAEMONSERVICE *service-name***
specifies the service name or port number that the SAS/CONNECT server uses to listen for SAS child process connections. When socket inheritance is enabled, the SAS client and the SAS/CONNECT server communicate via this port. If you use a service, its name must be configured in the SERVICES file on the computer that the SAS/CONNECT server session runs on.

**-SASSPAWNERCN *<name>***
specifies the name of the spawner definition to retrieve from the SAS Metadata Server.

If the -SASSPAWNERCN option is specified, you must either specify the -XMLCONFIGFILE option or you must specify the -METASERVER, -METAPORT, -METAUSER, and -METAPASS options. The -XMLCONFIGFILE option specifies the filename to use to get SAS Metadata Server access information. This file configures how the SAS/CONNECT Spawner will connect to the SAS Metadata Server to retrieve its configuration information.

For details about generating a SAS/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.

**-SERVICE *<port-number | service-name>***
specifies the service name or port number to use to listen for client connections.

The -SERVICE option values that are used to start the spawner determine what will be used by the client to sign on.

In the following example, the spawner is started by specifying the *port-number* as the value of the -SERVICE option during spawner start-up:

```
"SAS-installation-directory\SASFoundation\9.4\cntspawn.exe" -service 5020
```

The client can then sign on by specifying the explicit *port-number* in the SIGNON statement:

```
signon node-name.5020 -mgmtport 5030
```

**Note** If the -SERVICE option is not specified, the spawner will listen on the Telnet port (23).

**-SHELL**
specifies that the started SAS/CONNECT servers will allow X commands.

Without specifying the -SHELL option to the spawner, X command processing is disabled, by default.

**-SSLCLIENTAUTH**
specifies that the server requires client authentication for SSL connections.

**-SSLCRLCHECK**
specifies that the server check CRL for revoked digital certificates for SSL.

**-SSPI | -NOSSPI**
identifies support for the Security Support Provider Interface for single sign-on connections to the spawner. If the client and the server run under Windows and if the client does not supply a user ID and password to the server, SSPI (Security Support Provider Interface) is used to perform client authentication. SSPI authentication is

disabled by default. To enable SSPI authentication, you must specify -SSPI in the spawner start-up command. In versions prior to 9.4, SSPI was enabled by default.

**Default** -NOSSPI

**-TRACE | VERBOSE**
turns on trace level output.

**-XMLCONFIGFILE "*fully-qualified-path*"**
specifies the filename to use to get SAS Metadata Server access information. A path that includes one or more spaces must be enclosed in quotation marks.

If -XMLCONFIGFILE is used, -SASSPAWNERCN must also be used.

**Alias** -OMRCONFIGFILE

## *Windows Options*

Use the following service options to create, modify, and remove SAS/CONNECT spawner service definitions in the Windows operating environment:

**-INSTALL <-INSTALLDEPENDENCIES *service-name*>**
**<-SERVICEDESCRIPTION *description*> <-SERVICEDIRECTORY *directory-name*> -SERVICENAME *service-name*> <-SERVICEPASS *password*>**
**<-SERVICEUSER *user-ID*>**
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 Connect Spawner
```

You can install each instance of the spawner by using the following command:

```
C:\SAS>cntspawn -install
```

When you install a spawner without specifying -NAME, it is installed as "SAS Connect Spawner." 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.

If you try to install a second spawner without specifying -NAME, the attempt will fail and you will get an error.

This option can be abbreviated as -I.

**-INSTALLDEPENDENCIES *service-name–1<, service-name–2><, ...>***
specifies the Windows service that must be started before the spawner service starts. The *service-name* value is the name of the dependent service that is displayed in the Microsoft Windows Services snap-in (services.msc).

**Alias** -IDEP

**-SERVICEDESCRIPTION *"service-description"***
specifies the description that you assign to the spawner that is installed and started as a Windows service, when you also specify the -INSTALL option.. The -SERVDESC option is valid only when installing the spawner using the -INSTALL option on the CNTSPAWN command. The description can be viewed with the services applet in Windows. A specified spawner description cannot exceed 256 characters and must be enclosed in quotation marks if it contains one or more spaces.

The following example shows how to use the -INSTALL, -NAME, and -SERVDESC options on the CNTSPAWN command to install a spawner named

"SAS spawner 5" and specify a description, which will be displayed in the Services Control Manager Window:

```
cntspawn -install -servicename "SAS spawner 5" -servdesc
 "A SAS process that listens for requests to spawn SAS/Connect
servers"
```

**Alias** -SERVDESC

**-SERVICEDIRECTORY** *directory*
specifies the directory in which to run the Windows service, when you also specify the -INSTALL option.

**Alias** -SERVDIR

**-SERVICENAME** *"service-name"*
specifies the name that you assign to the spawner that is installed, or uninstalled, and started as a service in the Windows operating environment. A specified name overrides the default name that is automatically assigned when the -INSTALL option is used.

When you install a spawner without specifying -SERVICENAME, it is installed as "SAS Connect Spawner". If you try to install a second spawner without specifying -name, the attempt will fail and you will get an error.

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:

```
cntspawn -install -servicename "Doug's spawner"
```

The following example shows how to uninstall an explicitly named Windows spawner by using the -UNINSTALL command:

```
cntspawn -uninstall -servicename "Doug's spawner"
```

**Alias** -NAME

**-SERVICEPASS** *password*
specifies the password for the user account that spawner will run under as a service. For details about SSL, see *Encryption in SAS*, located in the Base SAS Help and Documentation.

**Alias** -SERVPASS

**-SERVICEUSER=***user-ID*
specifies a user name that the Windows service will run under, when you also specify the -INSTALL option. This option can be abbreviated as -SU.

**Alias** -SU, -SERVUSER

**-SSLCERTISS <***issuer***>**
specifies the name of the issuer of the digital certificate that SSL should use.

**-SSLCERTSERIAL <***serial***>**
specifies the serial number of the digital certificate that SSL should use.

**-SSLCERTSUBJ *<subject>***
specifies the subject name of the digital certificate that SSL should use.

**-UNINSTALL**
instructs the spawner to uninstall as a Windows service, which was previously installed and started by using the -INSTALL option.

If you used the -SERVICENAME option with the -INSTALL option to install a spawner, you can use the -SERVICENAME option with the -UNINSTALL option to identify the spawner to be removed.

```
cntspawn -uninstall -servicename "Doug's spawner"
```

**Alias** -DEINSTALL or -DI

## UNIX and z/OS Options

**-SSLCALISTLOC *<filename>***
specifies the name of the file that contains the list of trusted certificate authorities.

**-SSLCERTLOC *<filname>***
specifies the name of the file that contains the public certificate to use for SSL.

**-SSLCRLLOC *<filename>***
specifies the location of CRL file.

**-SSLPKCS12LOC *<filename>***
specifies the name of the file that contains PKCS12 information to use for SSL.

**-SSLPKCS12PASS *<password>***
specifies the password to use to decrypt the PKCS12 information.

**-SSLPVTKEYLO *<filename>***
specifies the name of the file that contains the public certificate's private key to use for SSL.

**-SSLPVTKEYPASS *<password>***
specifies the password to use to decrypt the private key, if the private key is encrypted.

For more information about using encryption options in SAS see "SAS System Options for Encryption" in *Encryption in SAS*.

For example programs, see "Encryption Technologies: Examples" in *Encryption in SAS*.

## Sample SAS/CONNECT Spawner logconfig.xml File

Here is an example of a spawner log configuration file (logconfig.xml) in the UNIX environment:

```
<?xml version="1.0" encoding="UTF-8"?>
<logging:configuration xmlns:logging="http://www.sas.com/xml/logging/1.0/">

   <!-- Rolling log file with default rollover of midnight -->

   <appender class="RollingFileAppender" name="TimeBasedRollingFile">
        <param name="Append" value="false"/>
        <param name="Unique" value="true"/>
        <param name="ImmediateFlush" value="true"/>
        <rollingPolicy class="TimeBasedRollingPolicy">
```

```
            <param name="FileNamePattern" value="/<SAS-configuration-directory>
             /Lev<n/ConnectSpawner/Logs/
             ConnectSpawner_%d_%S{hostname}_%S{pid}.log"/>
        </rollingPolicy>
        <layout>
            <param name="HeaderPattern" value="Host: '%S{hostname}',
             OS: '%S{os_family}', Release: '%S{os_release}',
             Command:'%S{startup_cmd}'"/>
            <param name="ConversionPattern" value="%d %-5p [%t] :%u - %m"/>
        </layout>
</appender>

<!-- Unix System Facility Appender, writes to unix system log -->
<appender class="UNXFacilityAppender" name="UnixSysLog">
      <filter class="RepeatMatchFilter">
         <param name="AcceptOnMatch" value="false"/>
      </filter>
      <layout>
          <param name="ConversionPattern" value="%-5p [%t] :%u - %m"/>
      </layout>
</appender>

<!-- Administration message logger -->
<logger name="Admin">
    <level value="Info"/>
    <appender-ref ref="UnixSysLog"/>
</logger>

<!-- Application message logger -->
<logger name="App">
    <level value="Trace"/>
</logger>

<!-- Audit message logger -->
<logger name="Audit">
    <level value="Info"/>
</logger>

<!-- IOM protocol message logger -->
<logger name="IOM">
    <level value="Info"/>
</logger>

<!-- Logging Facility logger -->
<logger name="Logging">
     <level value="Error"/>
     <appender-ref ref="UnixSysLog"/>
</logger>

<!-- Root logger -->
<root>
   <level value="Error"/>
   <appender-ref ref="TimeBasedRollingFile"/>
      <!-- Caution: Do NOT edit, modify or remove the following statement. -->
      <appender-ref ref="IOMServer"/>
</root>
```

```
   <!-- Caution: Do NOT edit or modify the configuration information below. -->
   <!-- Settings are enabled for the internal server execution environment. -->

   <!-- IOM Server Appender -->
   <appender class="IOMServerAppender" name="IOMServer">
      <param name="MaxEntries" value="10000"/>
      <layout>
         <param name="ConversionPattern" value="%d %-5p [%t] %u - %m"/>
      </layout>
   </appender>

   <!-- Event Appender -->
   <appender class="IOMEventAppender" name="Events">
      <param name="Scope" value="server"/>
         <param name="Threshold" value="Debug"/>
      <layout>
         <param name="ConversionPattern" value="%d %-5p [%t] %u - %m"/>
      </layout>
   </appender>

   <!-- Server Administration Message Logger -->
   <logger name="Perf.ARM.IOM.ConnectSpawner.ServerAdministration">
      <level value="Debug"/>
      <appender-ref ref="Events"/>
   </logger>

</logging:configuration>
```

*Chapter 8*

# TCP/IP SERVICES File



## Configuring the Services File

### *Services That Require an Entry in the Services File*

Here are typical examples of port services that require configuration in the **`services`** file:

- Telnet service
- spawner port
- SAS/SHARE server ID or port
- firewall computer port
- 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`**. The **`services`** file on Windows is typically located in the **`C:\Windows\System32\drivers\etc`** directory.

### *Example Services File*

Here is an example excerpt from a **`services`** file.

*<official-service-name> <port-number/protocol-name> <alias-name>*

```
# The form for each entry is:
#
# <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
# A blank line goes here.
```

*Note:* You must enter a blank line (press the ENTER key) at the end of the **services** file. If a blank line is not at the end of the file, the final line in the file is not detected. For example, if you run a SAS script that contains the name of the configured SAS/SHARE**sea**, this error message is displayed:

```
Cannot find TCP service 'sea'
```

## Explanation of Fields

Here is an explanation of each field:

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 "Rules for Words and Names in the SAS Language" in *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 can refer 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.

# Part 4

# Firewalls, Scripts, and Error Messages

*Chapter 9*

# Configuring SAS/CONNECT for Use with a Firewall



## Firewall Concepts

*firewall*
is a controlled gateway between two networks. A firewall limits external client connections to a set of restricted ports on one or more computers that are inside the firewall.

Web servers and other network applications can also use firewalls to limit access to servers. SAS/CONNECT permits TCP/IP connections between clients outside a firewall to a spawner that runs on a SAS/CONNECT server inside a firewall.

*socket inheritance*
enables 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.

Before this innovation, 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 improves the security of a firewall configuration and simplifies administration of a firewall configuration.

## Requirements for Using a Firewall

- The external clients and the servers within the firewall must be running SAS 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 predefined server port.
- A spawner must be running on a server inside the firewall. For complete details about the spawner program, see Chapter 6, "Introduction to the SAS/CONNECT Spawner," on page 73.

# Firewall Configurations

## *Overview of Firewall Configurations*

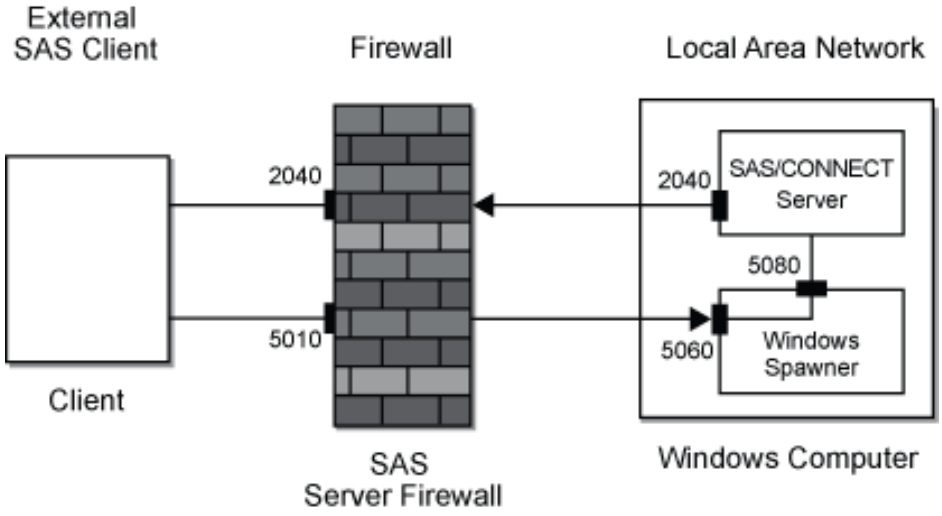
The supported firewall configurations are distinguished by these characteristics:

- A range of restricted ports is available for client/server connections across a firewall.
- A single port is available for all client/server connections across a firewall.

## *Setting Up a Firewall Configuration That Uses Restricted Ports*

The example configuration includes an external SAS client, a firewall, and a SAS/CONNECT server session and a spawner program that run on the local area network. Each external client connects to the server using a range of restricted ports.

***Figure 9.1*** *Firewall Configuration That Uses Restricted Ports*




Here are the steps for setting up a firewall configuration:

1. At each external SAS client, the user must configure the firewall port, 5010, in its services file.

```
fireport              5010/tcp            # Firewall computer port
```

FIREPORT is a defined service in the client's services file that is associated with port 5010. FIREPORT is the single port through which all external SAS clients will access SAS servers in the internal network.

2. The administrator of the firewall server must configure these ports:

   - the restricted ports that are used by the external SAS clients and a mapping to the equivalent port numbers on the SAS/CONNECT server
   - the firewall port, 5010, and a mapping to 5010 on the SAS/CONNECT server or another port number on the SAS/CONNECT server

   *Note:* Restricted ports are implemented using the TCPPORTFIRST= and TCPPORTLAST= system options that are specified in the SAS start-up file. (See step 4.)

   For example, if the external SAS clients use restricted ports 2040 through 2044, the administrator of the firewall server must configure those ports on the firewall server. Also, the administrator must map those ports to the same port numbers on the SAS/CONNECT server.

   Specific details about configuring and mapping ports on the firewall server vary according to the specific firewall software that is used.

3. The administrator of the SAS/CONNECT server must configure these ports in its services file:

   - the port that is used by the external SAS client to communicate with the spawner
   - the ports that are used by the spawner to communicate with the SAS/CONNECT server

   Here is an example of these entries in the services file:

   ```
   spawnport          5060/tcp       # Port for external SAS client to spawner
   servport           5080/tcp       # Port for spawner and SAS/CONNECT server
   ```

   SPAWNPORT is a defined service in the services file that is associated with port 5060. SERVPORT is associated with port 5080.

4. The administrator of the SAS/CONNECT server must configure one or more restricted ports in the SAS start-up file that executes when the spawner starts the SAS/CONNECT session.

   ```
   sas.exe -tcpportfirst 2040 -tcpportlast 2040 %*
   ```

   SAS is started and the restricted port is 2040. In this example, all communications between external SAS clients and the SAS/CONNECT server are restricted to port 2040.

   A range of ports could be specified by increasing the values assigned to the TCPPORTFIRST= and TCPPORTLAST= system options.

5. The administrator of the SAS/CONNECT server must start the spawner using a command that disables socket inheritance:

   ```
   cntspawn -noinheritance -service spawnport -sasdaemonservice servport
   -sascmd mysas.cmd
   ```

   *Note:* The command to start the SAS/CONNECT spawner is **`CNTSPAWN`**.

   The restricted port that is used by the SAS client and the SAS/CONNECT server is specified in the **`mysas.cmd`** script via the TCPPORTFIRST= and TCPPORTLAST= system options.

   Here is an explanation of the spawner command:

***Table 9.1*** *Explanation of Spawner Command*

| Command | Description |
|---|---|
| **cntspawn** | Starts the spawner. |
| **-noinheritance** | Specifies that sockets cannot be inherited. |
| **-service spawnport** | Specifies the service or its port, 5060, at which the spawner listens for requests from SAS clients to connect to a SAS/CONNECT server. |
| **-sasdaemonservice servport** | Specifies the service or port, 5080, through which the spawner relays the SAS client's request to connect to the SAS/CONNECT server. |
| **-sascmd mysas.cmd** | Specifies the script that starts the SAS/CONNECT session. The script might contain SAS options that restrict ports. |

For details about spawner options, see "Overview" on page 83.

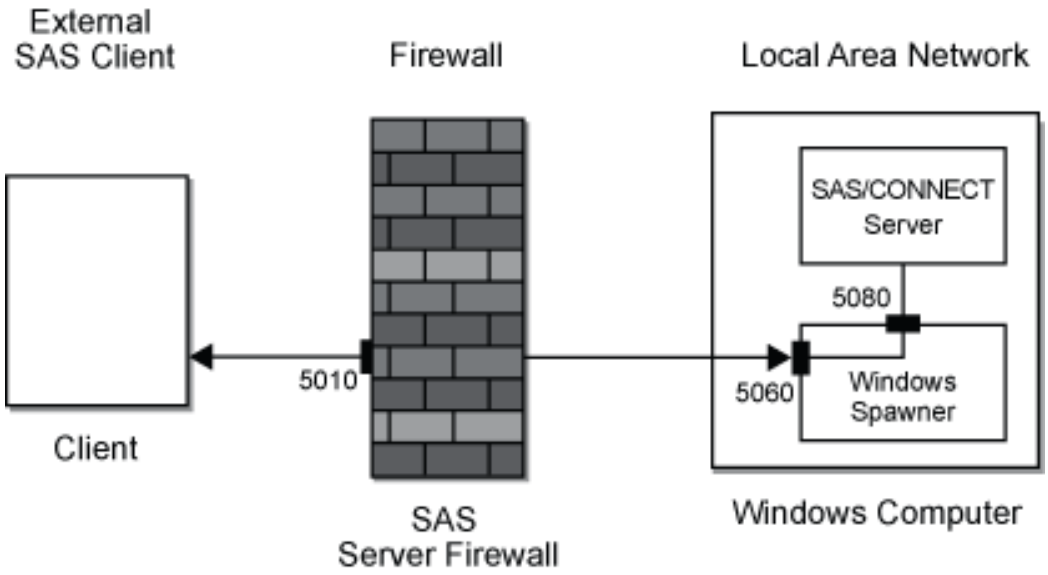
6. To test the configuration, start a SAS session on a computer that is outside the firewall and sign on to the server that is inside the firewall. Here is an example:

```
options comamid=tcp;
signon firewall.fireport username="myuser" password="mypass";
```

## *Setting Up a Firewall Configuration That Uses a Single Port*

The example configuration includes an external SAS client, a firewall, and a SAS/CONNECT server session and a spawner program that run on the local area network. Each external client connects to the server using a single port, which is enabled by socket inheritance.

***Figure 9.2*** *Firewall Configuration That Uses a Single Port*




Here are the steps for setting up a firewall configuration:

1. At each external SAS client, the user must configure the firewall port, 5010, in its services file.

   ```
   fireport                5010/tcp             # Firewall computer port
   ```

   FIREPORT is a defined service in the TCP/IP services file that is associated with port 5010. FIREPORT is the single port through which all external SAS clients will access SAS servers in the internal network.

   *Note:* The firewall server does not necessarily have to run SAS software.

2. The administrator of the firewall server must configure the firewall port, 5010, and map it to another port number on the SAS/CONNECT server.

   Specific details about configuring and mapping ports on the firewall server vary according to the specific firewall software that is used.

3. The administrator of the SAS/CONNECT server must configure these ports in its services file:

   - the port that is used by the external SAS client to communicate with the spawner
   - the ports that are used by the spawner to communicate with the SAS/CONNECT server

   Here is an example of these entries in the services file:

   ```
   spawnport          5060/tcp       # Port for external SAS client to spawner
   servport           5080/tcp       # Port for spawner and SAS/CONNECT server
   ```

   SPAWNPORT is a defined service in the services file that is associated with port 5060. SERVPORT is associated with port 5080.

4. The administrator of the SAS/CONNECT server starts the spawner:

   ```
   cntspawn -service spawnport -sasdaemonservice servport
   -sascmd mysas.cmd
   ```

   *Note:* The command to start the SAS/CONNECT spawner is **CNTSPAWN**.

   Here is an explanation of the spawner command:

***Table 9.2*** *Explanation of Spawner Command*

| Command | Description |
|---|---|
| **cntspawn** | Starts the spawner. |
| **-service spawnport** | Specifies the service or its port, 5060, at which the spawner listens for requests from SAS clients to connect to a SAS/CONNECT server. |
| **-sasdaemonservice servport** | Specifies the service or port, 5080, through which the spawner relays the SAS client's request to connect to the SAS/CONNECT server. |
| **-sascmd mysas.cmd** | Specifies the script that starts the SAS/CONNECT session. |

For details about spawner options, see Chapter 6, “Introduction to the SAS/CONNECT Spawner,” on page 73.

5. To test the configuration, start a SAS session on a computer that is outside the firewall and sign on to the server that is inside the firewall. Here is an example:

```
options comamid=tcp;
signon firewall.fireport username="myuser" password="mypass";
```

*Chapter 10*

# Sign-On Scripts



## Script Rules

### *Syntax*

For details about writing scripts, see the topic about "SAS/CONNECT Sign-on Script Files" in *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, ensure 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

*n* can be any number, including decimal fractions. SECOND is an alias for SECONDS. Here are some examples of valid time specifications:

- 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 enter 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 the following:

```
type "sas options ('dmr comamid=-" enter;
type "tcp')" enter;
```

*Note:* Do not insert spaces around the hyphen.

## Sample Scripts

### The Scripts


- "TCPUNIX.SCR Script" on page 111
- "TCPWIN.SCR Script" on page 115
- "TCPMVS.SCR Script" on page 118

- "TCPTSO9.SCR Script" on page 121


### 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) 2007 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'; */


1  log "NOTE: Script file 'tcpunix.scr' entered.";

    if not tcp then goto notcp;
2  if signoff then goto signoff;

     /* --------------- TCP/IP SIGNON --------------------------------*/

3  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)      --*/
/*-----------------------------------------------------------------*/

4  input 'Userid?';
     type LF;
5  waitfor 'Password', 30 seconds : nolog;
     input nodisplay 'Password?';
     type LF;

unx_log:
6  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.     */
   /* NOSYNTAXCHECK prevents remote side from going into       */
   /* syntax checking mode when a syntax error is encountered.   */
7  type 'sas -dmr -comamid tcp -noterminal -nosyntaxcheck' LF;
8  waitfor 'SESSION ESTABLISHED', 90 seconds : nosas;
9  log 'NOTE: SAS/CONNECT conversation established.';
   stop;
10 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 -nosyntaxcheck       */
      /*#---------------------------------------------------------*/
      /*                                                         */
      /* 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;               */
11 type 'sas -dmr -comamid tcp -noterminal ';
     type '-nosyntaxcheck' 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.                 */
12 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 -------------------------------*/

13 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 quoted 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 server by using the TCP/IP access method.

```
/* trace on; */
/* echo  on; */
/*--------------------------------------------------------------------*/
/*--            Copyright (C) 2007 by SAS Institute Inc., Cary NC  --*/
/*--                                                               --*/
/*-- name:      tcpwin.scr                                         --*/
/*--                                                               --*/
/*-- purpose:   SAS/CONNECT SIGNON/SIGNOFF script for connecting   --*/
/*--            to a Windows host by                               --*/
/*--            using the TCP/IP access method.                    --*/
/*--                                                               --*/
/*-- notes:   1. You must have the spawner program executing on    --*/
/*--              the remote Windows 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           --*/
/*--             environment is "sas".                                --*/
/*--             If this is incorrect for your site, change the       --*/
/*--             contents of the line that contains:                  --*/
/*--             type 'sas ...                                        --*/
/*--                                                                  --*/
/*-- support:   SAS Institute staff                                   --*/
/*--                                                                  --*/
/*------------------------------------------------------------------*/
1 log "NOTE: Script file 'tcpwin.scr' entered.";

    if not tcp then goto notcp;
2 if signoff then goto signoff;

    /* --------------- TCP/IP SIGNON ---------------------------------*/

3 waitfor 'Username:'
          , 'Hello>'               :  ready
          , 'access denied'        :  nouser
          , 120 seconds            :  noprompt
          ;
4 input 'Userid?';
    type LF;
5 waitfor 'Password:' , 120 seconds: nolog;
    input nodisplay 'Password?';
    type LF;
6 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.     */
    /* NOSYNTAXCHECK prevents remote side from going into syntax */
    /* checking mode when a syntax error is encountered.          */
7 type 'sas -dmr -comamid tcp -noterminal -nosyntaxcheck' LF;
8 waitfor 'SESSION ESTABLISHED', 120 seconds : nosas;
9 log 'NOTE: SAS/CONNECT conversation established.';
    stop;


/*---------------- TCP/IP SIGNOFF ---------------------------------*/
10 signoff:
    log 'NOTE: SAS/CONNECT conversation terminated.';
    stop;

/*--------------- SUBROUTINES ---------------------------------*/
```

```
/*--------------- ERROR ROUTINES
-------------------------------*/
11
  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 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 10.
3. The WAITFOR statement awaits the login prompt from the server and branches to various error handles if the 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 from a z/OS server with TSO. The TCP/IP access method is used.

```
/*-------------------------------------------------------------------*/
/*--            Copyright (C) 2007 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                               --*/
/*--                                                               --*/
/*-------------------------------------------------------------------*/

1   log "NOTE: Script file 'tcpmvs.scr' entered.";

  if not tcp then goto notcp;
```

```
2  if signoff then goto signoff;

/* ----------------------- TCP SIGNON ----------------------------*/

   /* make sure you are running the IBM TCP/IP */
3   waitfor 'ENTER USERID'                : tsologon,
          120 seconds                    : noinit;

/*-------------------------- TSO LOGON -----------------------------*/

tsologon:
4   input 'Userid?';
   type LF;
5   waitfor 'ENTER PASSWORD', 60 seconds : nolog;

tsopass:
   input nodisplay 'Password?';
   type LF;

tsodone:
6 waitfor 'SESSION ESTABLISHED',
          'PASSWORD INVALID'            : tsopass,
          'ENTER NEW PASSWORD'          : tsonewp,
          'CURRENTLY LOGGED ON'         : dup_log,
          'NOT VALID'                   : nouser,
          120 seconds                   : notso;
   waitfor 1 second;

7  log 'NOTE: SAS/CONNECT conversation established.';
   stop;

tsonewp:
8   input nodisplay 'New Password?';
   type LF;

9  waitfor 'VERIFY NEW PASSWORD',
          120 seconds                   : notso;

   input nodisplay 'Verify New Password';
   type LF;

   goto tsodone;

/*---------------------------- SIGNOFF -----------------------------*/
10 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 ------------------------*/

11 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 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. 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 from 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) 2007 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                                --*/
/*--                                                               --*/
/*-------------------------------------------------------------------*/

1   log "NOTE: Script file 'tcptso9.scr' entered.";

   if not tcp then goto notcp;
2  if signoff then goto signoff;

/* ----------------------- TCP SIGNON ------------------------------*/

   /* make sure you are running the IBM TCP/IP or the z/OS spawner   */

3  waitfor 'Username:'                  : spnlogon,
          'ENTER USERID'                : tsologon,
          120 seconds                   : noinit;

/*------------------------ SPAWNER LOGON --------------------------*/

spnlogon:
4  input 'Userid?';
   type LF;

5  waitfor 'Password', 120 seconds : spnfail;
   input nodisplay 'Password?';
   type LF;

spndone:
6  waitfor 'Hello>',
          'Userid'                      : spnlogon,
          'Password expired'            : spnnewp,
          120 seconds                   : spnfail;

7  type "/usr/local/bin/spawnsas.sh nosasuser opt(''dmr comamid=tcp'')" LF;
```

```
8 waitfor 'SESSION ESTABLISHED', 120 seconds : spnfail;

9 log 'NOTE: SAS/CONNECT conversation established.';
  stop;

spnnewp:
10  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:
11  input 'Userid?';
  type LF;
12  waitfor 'ENTER PASSWORD', 60 seconds : nolog;
  input nodisplay 'Password?';
  type LF;

tsodone:
13  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.';
  /* NOTERMINAL suppresses prompts from      */
  /* remote SAS session.  NOSYNTAXCHECK prevents remote side from        */
  /* going into syntax checking mode when a syntax error is encountered. */
14  type "sas o('dmr,comamid=TCP,noterminal,nosyntaxcheck')" LF;
15  waitfor 'SESSION ESTABLISHED', 120 seconds : nosas;

16  log 'NOTE: SAS/CONNECT conversation established.';
  stop;

tsonewp:
17  input nodisplay 'New Password?';
  type LF;

  waitfor 'VERIFY NEW PASSWORD',
          120 seconds                    : notso;
```

```
   input nodisplay 'Verify New Password';
   type LF;

   goto tsodone;

/*---------------------------- SIGNOFF-----------------------------*/
18 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-------------------------------------------------*/

19 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--------------------------------*/
20 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 “Starting SAS Using a Shell Script” on page 87.
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 the 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 sign-on.
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.

*Chapter 11*

# Error Messages



## 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 sign-on for the following reasons:

- 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 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 application. Contact on-site SAS support personnel for assistance.

```
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.

# 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 sign-on for the following reasons:

- 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 application. Contact on-site SAS support personnel for assistance.

**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.

# 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 sign-on for the following reasons:

- 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 application. 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 44 for information about two-level server names.

# 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 for UNIX Environments*
- *SAS Companion for Windows*
- *SAS Companion for z/OS*
- *Encryption in SAS*
- *Configuration Guide for SAS Foundation for Microsoft Windows for x64*
- *Configuration Guide for SAS Foundation for Microsoft Windows*
- *Configuration Guide for SAS Foundation for z/OS*
- *Configuration Guide for SAS Foundation for UNIX Environments*

For a complete list of SAS publications, go to sas.com/store/books. If you have questions about which titles you need, please contact a SAS Representative:

SAS Books
SAS Campus Drive
Cary, NC 27513-2414
Phone: 1-800-727-0025
Fax: 1-919-677-4444
Email: sasbook@sas.com
Web address: sas.com/store/books

# Glossary

**access method**
*See* communications access method.

**authentication**
*See* client authentication.

**autoexec file**
a file that contains SAS statements that are executed automatically when SAS is invoked. 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. *See also* fileref.

**client authentication (authentication)**
the process of verifying the identity of a person or process for security purposes.

**client session**
a SAS session that is running on a client computer. A client session accepts SAS statements and passes those that are submitted to the server for processing. The client session manages the output and messages from both the client session and the server session.

**command file**
a file that contains operating system commands to be executed in sequence.

**communications access method (access method)**
an interface between SAS and the network protocol or interface that is used to connect two operating environments. Depending on the operating environments, SAS/SHARE and SAS/CONNECT use either the TCP/IP or XMS communications access method. *See also* TCP/IP.

**control program**
a low-level software interface, such as SAS/CONNECT software, between communications hardware and applications programs. A control program works in conjunction with an adapter.

**Cross-Memory Services (XMS)**
a cross-task communication interface that is part of z/OS. XMS is used by programs that run within a single z/OS operating environment. XMS is also the name of the SAS communications access method that uses XMS for client/server communication.

**data set**
*See* SAS data set.

**descriptor information**
information about the contents and attributes of a SAS data set. For example, the descriptor information includes the data types and lengths of the variables, as well as which engine was used to create the data. SAS creates and maintains descriptor information within every SAS data set.

**DNS**
*See* Domain Name System.

**domain name resolution (name resolution)**
in a TCP/IP network, the process of converting a server name to an IP address. *See also* Domain Name System.

**domain name resolver (name resolver)**
in a TCP/IP network, client software that uses one or more domain name servers to convert a server name to an IP address or vice versa. *See also* domain name server, domain name resolution.

**domain name server (name server)**
an Internet server program that converts domain names to IP addresses. *See also* Domain Name System.

**Domain Name System (DNS)**
a distributed database system on the Internet that maps domain names to IP addresses. The Domain Name System also provides information about which TCP/IP services are available to the server host, the location of the domain name servers in the network, and other information about server hosts and networks. *See also* domain name server, TCP/IP.

**encryption**
the conversion of data by the use of algorithms or other means into an unintelligible form in order to secure data (for example, passwords) in transmission and in storage.

**external file**
a file that is created and maintained by a host operating system or by another vendor's software application. An external file can read both data and stored SAS statements.

**file reference**
*See* fileref.

**fileref (file reference)**
a name that is temporarily assigned to an external file or to an aggregate storage location such as a directory or a folder. The fileref identifies the file or the storage location to SAS. *See also* libref.

**firewall**
a set of related programs that protect the resources of a private network from users from other networks. A firewall can also control which outside resources the internal users are able to access. *See also* socket inheritance, port, SAS/CONNECT spawner.

**Integrated Object Model server (IOM server)**
a SAS object server that is launched in order to fulfill client requests for IOM services.

**Internet Protocol Version 4 (IPv4)**
a protocol that specifies the format for network addresses for all computers that are connected to the Internet. This protocol, which is the predecessor of Internet Protocol Version 6, uses dot-decimal notation to represent 32-bit address spaces. An example of an Internet Protocol Version 4 address is 10.23.2.3. *See also* IP address.

**Internet Protocol Version 6 (IPv6)**
a protocol that specifies the format for network addresses for all computers that are connected to the Internet. This protocol, which is the successor of Internet Protocol Version 4, uses hexadecimal notation to represent 128-bit address spaces. The format can consist of up to eight groups of four hexadecimal characters, delimited by colons, as in FE80:0000:0000:0000:0202:B3FF:FE1E:8329. As an alternative, a group of consecutive zeros could be replaced with two colons, as in FE80::0202:B3FF:FE1E:8329. *See also* IP address.

**IOM server**
*See* Integrated Object Model server.

**IP address**
a unique network address that is assigned to each computer that is connected to the Internet. The IP address can be specified in either of two formats: Internet Protocol Version 4 (IPv4) or Internet Protocol Version 6 (IPv6). *See also* Internet Protocol Version 4.

**IPv4**
*See* Internet Protocol Version 4.

**IPv6**
*See* Internet Protocol Version 6.

**library reference**
*See* libref.

**libref (library reference)**
a SAS name that is associated with the location of a SAS library. For example, in the name MYLIB.MYFILE, MYLIB is the libref, and MYFILE is a file in the SAS library. *See also* SAS library.

**name resolution**
*See* domain name resolution.

**name resolver**
*See* domain name resolver.

**name server**
*See* domain 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.

**port**
in a network that uses the TCP/IP protocol, an endpoint of a logical connection between a client and a server. Each port is represented by a unique number.

**Remote Library Services (RLS)**
a feature of SAS/SHARE and SAS/CONNECT software that enables you to read, write, and update remote data as if it were stored on the client. RLS can be used to access SAS data sets on computers that have different architectures. RLS also provides read-only access to some types of SAS catalog entries on computers that have different architectures.

**return code**
a numeric value that indicates whether a request was successful. A return code can also indicate a specific error or warning.

**RLS**
*See* Remote Library Services.

**SAS data file**
a type of SAS data set that contains data values as well as descriptor information that is associated with the data. The descriptor information includes information such as the data types and lengths of the variables, as well as the name of the engine that was used to create the data. *See also* SAS data set.

**SAS data set (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. *See also* descriptor information.

**SAS library**
one or more files that are defined, recognized, and accessible by SAS, and that are referenced and stored as a unit. Each file is a member of the library.

**SAS system option (system option)**
a type of SAS language element that is applied to any of a number of operations during a SAS session. System options can control SAS session initialization, SAS interactions with hardware and software, and input and output processing of SAS files.

**SAS/CONNECT client**
a SAS session that receives services, data, or other resources from a specified server. The server can run on the same computer as the client or on a different computer (across a network).

**SAS/CONNECT server**
a SAS session that delivers services, data, or other resources to a requesting client. The server can run on the same computer as the client, or on a networked computer.

**SAS/CONNECT spawner (spawner)**
a program that runs on a remote computer and that listens for SAS/CONNECT client requests for connection to the remote computer. When the spawner program receives a request, it invokes a SAS session on the remote computer.

**SAS/SHARE client**
a SAS/SHARE session that acts as a client. The user who runs a SAS/SHARE client accesses data on a SAS/SHARE server through Remote Library Services (RLS). *See also* server, SAS/SHARE server, Remote Library Services.

**SAS/SHARE server**
the result of an execution of the SERVER procedure, which is part of SAS/SHARE software. A server runs in a separate SAS session that services users' SAS sessions by controlling and executing input and output requests to one or more SAS libraries. *See also* server, SAS/SHARE client.

**SASProprietary algorithm**
a fixed encoding algorithm that is included with Base SAS software. The SASProprietary algorithm requires no additional SAS product licenses. It provides a medium level of security.

**script**
an external file that contains SAS script statements. The script file is stored on a client and provides instructions for establishing and terminating a SAS/CONNECT session. Script files are executed by the SIGNON and SIGNOFF commands. *See also* external file.

**script statement**
a special kind of SAS statement that was developed for use in scripts for SAS/CONNECT software. Script statements are used only in scripts.

**Secure Sockets Layer (SSL)**
an encryption protocol for securely communicating across the Internet. SSL uses encryption algorithms RC2, RC4, DES, TripleDES, and AES.

**Security Support Provider Interface (SSPI)**
a built-in security provider for Microsoft Windows computers. In a network, SSPI transfers user context information from a user's client computer to the server. This enables users who are members of a trusted domain to be authenticated automatically.

**server**
software that provides either resources or services to requesting clients, possibly over a network.

**services file**
a file that contains a list of service names and the TCP/IP ports that are mapped to those services. The services file is stored on both the SAS client and the SAS server. The UNIX services file is located in /etc/services. A service can be specified for any of the following: a SAS/CONNECT spawner, a SAS/SHARE server, an MP CONNECT pipe, and a firewall server. *See also* port, firewall, SAS/SHARE 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.

**SMP**
*See* symmetric multiprocessing.

**socket**
the endpoint of a connection in a TCP/IP network. A socket is the combination of a TCP port and an IP address. By analogy, a socket is like a telephone to which a telephone number has been assigned. The TCP port is like a telephone number, and the IP address is like the location of the telephone. *See also* port, services file, socket inheritance, IP address.

**socket inheritance**
the mechanism by which a SAS/CONNECT server that is running a spawner uses a single firewall socket (or port) for SAS/CONNECT server-to-client communications. Socket inheritance increases the security of private networks by limiting the number of ports that are used for connections through a firewall. *See also* SAS/CONNECT spawner, port, firewall.

**spawner**
*See* SAS/CONNECT spawner.

**SSL**
*See* Secure Sockets Layer.

**SSPI**
*See* Security Support Provider Interface.

**symmetric multiprocessing (SMP)**
a type of hardware and software architecture that can improve the speed of I/O and processing. An SMP machine has multiple CPUs and a thread-enabled operating system. An SMP machine is usually configured with multiple controllers and with multiple disk drives per controller.

**system option**
*See* SAS system option.

**TCP/IP**
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 appropriate 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.

**TLS**
*See* Transport Layer Security.

**Transport Layer Security (TLS)**
the successor to Secure Sockets Layer (SSL), a cryptographic protocol that is designed to provide communication security over the Internet. TLS uses asymmetric cryptography for authentication and confidentiality of the key exchange, symmetric encryption for data/message confidentiality, and message authentication codes for message integrity. Several versions of the protocols are in widespread use in applications such as web browsing, electronic mail, Internet faxing, instant messaging and voice-over-IP (VoIP). *See also* Secure Sockets Layer.

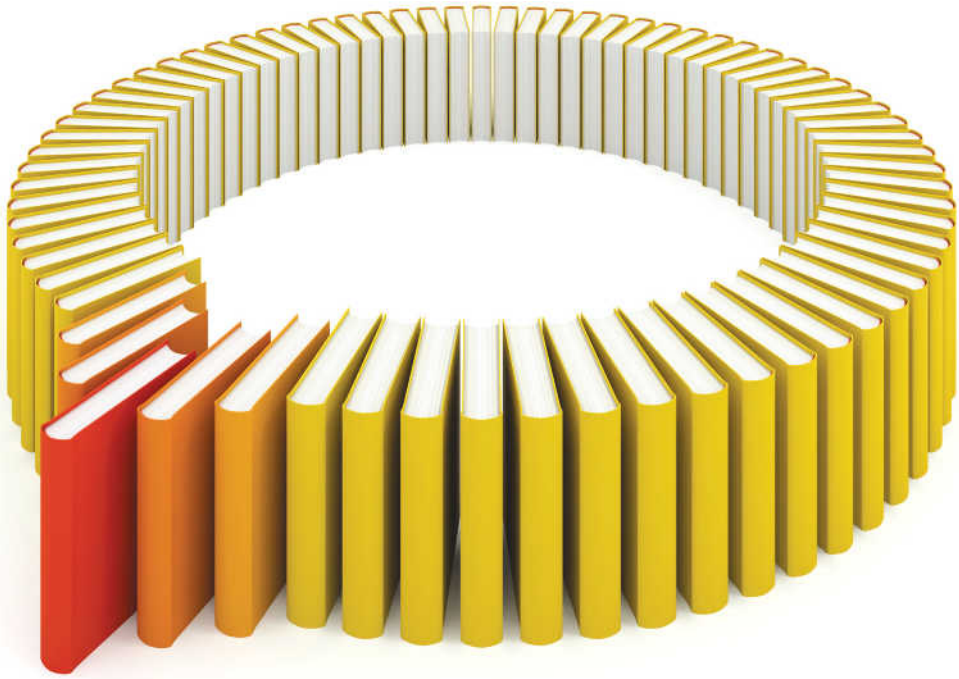
**user context**
a set of information about the user who is associated with an active session. The user context contains information such as the user's identity and profile. *See also* Security Support Provider Interface.

**user right**
any of a set of privileges that are assigned to each user of a client computer and to a server computer in a Windows domain. Setting the appropriate user rights on the server computer enables users to connect to a secure server.

**XMS**
*See* Cross-Memory Services.

# Index

## T



# Gain Greater Insight into Your SAS® Software with SAS Books.

Discover all that you need on your journey to knowledge and empowerment.

support.sas.com/bookstore
*for additional books and resources.*