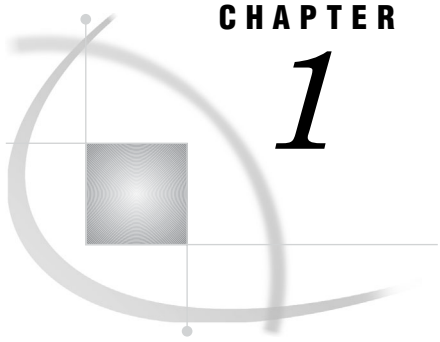


CHAPTER

1

Getting Started with SAS in UNIX Environments

Starting SAS Sessions in UNIX Environments

Invoking SAS

A SAS session is invoked using a link in the **!SASROOT** directory. Your UNIX administrator can add this link to the list of commands for your operating environment.

Ask your system administrator for the command that invokes SAS at your site. At many sites, the command to invoke SAS is **sas**, but a different command might have been defined during the SAS installation process at your site. This documentation assumes that SAS is invoked by the **sas** command.

Note: Before you start your SAS session, review the different techniques for interrupting and terminating your SAS session (see "Exiting or Interrupting Your SAS Session in UNIX Environments "on page 25). Also, if you cannot stop your SAS session, contact your system administrator. △

SAS Invocation Scripts

SAS is invoked by scripts that are located in the **!SASROOT/bin** directory. A SAS invocation script is created for each language that is installed. The invocation scripts are named using the language codes of the installed language. For example, **sas_en** invokes the English version of SAS. All languages are installed in all locations.

For more information about setting up SAS, refer to the installation documentation for the UNIX environment.

SAS Configuration Files

SAS creates a separate configuration file for each language that is installed. The language-specific configuration files have the form **!SASROOT/nls/<language>/sasv9.cfg** for each language. An additional configuration file that is language independent is **!SASROOT/sasv9.cfg**. This master configuration file in **!SASROOT** is used by all languages in addition to the language-specific files in **!SASROOT/nls/<language>/**. You can modify these configuration files to meet your needs. For information about how to customize SAS configuration files, see "Customizing Your SAS Session by Using Configuration and Autoexec Files" on page 20.

Regenerating SAS Invocation Scripts

The SAS invocation scripts that exist in **!SASROOT/bin** should not be modified. SAS Setup enables you to regenerate the default SAS invocation scripts that are located in **!SASROOT/bin**. To regenerate the invocation scripts, perform the following steps:

1. Run **SAS Setup** from **!SASROOT/sassetup**. Make sure that you have the appropriate privilege to update files in **SASROOT**.
2. Select **Run Setup Utilities** from the **SAS Setup Primary Menu**.
3. Select **Perform SAS Software Configuration**.
4. Select **Recreate the SAS Invocation Scripts**.

Syntax of the SAS Command

The general form of the SAS command is as follows:

sas <*-option1…-option-n*> <*filename*>

You can use these arguments with the SAS command:

-option1 ... -option-n
specifies SAS system options to configure your session or X command line options. See Chapter 18, "System Options under UNIX," on page 339 and "X Command Line Options" on page 13 for more information. If you omit any options (either on the command line or in the configuration file), the SAS (or site-specific) default options are in effect.

filename
specifies the name of the file containing the SAS program to be executed. Specifying a filename on the SAS command invokes a batch SAS session. Omit the filename to begin an interactive session.

If the file is not in the current directory, specify its full pathname.

Example: Invoke an Interactive SAS Session

To invoke an interactive SAS session, without specifying any SAS system options, enter

```
sas
```

The execution mode will depend on your default settings. For more information, see "Selecting a Method of Running SAS in UNIX Environments" on page 7.

To specify the NODATE and LINESIZE system options, you could enter

```
sas -nodate -linesize 80
```

What If SAS Does Not Start?

If SAS does not start, the SAS log might contain error messages that explain the failure. However, error messages that SAS issues before the SAS log is initialized are written to the SAS console log.

Under UNIX, the STDOUT fileref specifies the location of the console log.

Running SAS in a Foreground or Background Process

UNIX is a multitasking operating system, so you can run multiple processes at the same time. For example, you can have one process running in the foreground and three in the background.

A *foreground process* executes while you wait for the prompt; that is, you cannot execute additional commands while the current command is being executed. After you enter a command, the shell starts a process to execute the command. After the system executes the command, the shell displays the prompt and you can enter additional commands. The following is an example of SAS executing as a foreground process:

```
sas
```

A *background process* executes independently of the shell. After you enter a command, the shell starts a process to execute the command, and then issues the system prompt. You can enter other commands or start other background processes without waiting for your initial command to execute. The following is an example of the command that is used to execute a background process:

```
sas&
```

Note: Both the C shell and the Korn shell include commands that enable you to move jobs among three possible states: running in the foreground, running in the background, and suspended. △

Selecting a Method of Running SAS in UNIX Environments

You can run SAS in the following modes:

- SAS windowing environment
- interactive line mode
- batch mode

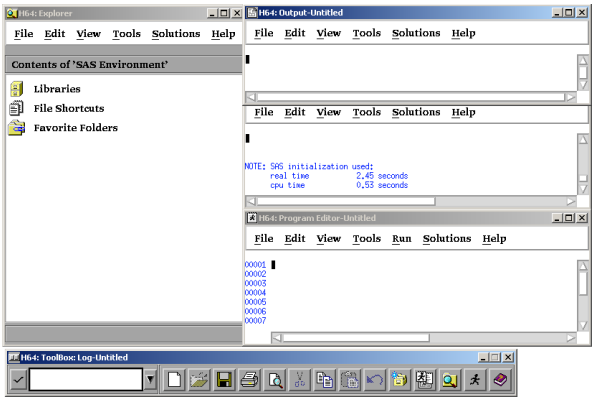
Ask your UNIX system administrator which interface or mode of operation is the default at your site.

SAS Windowing Environment in UNIX Environments

Introduction to the SAS Windowing Environment

You interact with SAS through windows using the keyboard, mouse, menus, and icons. The windowing environment includes, but is not limited to, the Explorer, Program Editor, Output, Log, and Results windows. The following display shows the Explorer, Output, Log, and Program Editor windows. The ToolBox window is also displayed.

Display 1.1 Windows in the SAS Windowing Environment



Your SAS session might default to the windowing environment interface. If you want to use the windowing environment, you can start your SAS session as a foreground process, or as a background process by adding an ampersand (&) to your SAS command line. See "Running SAS in a Foreground or Background Process" on page 6 for an example of these SAS commands.

For more information about using the windowing environment, see Chapter 7, "Working in the SAS Windowing Environment," on page 137.

Note: If you are not using an X display, then you can invoke SAS in interactive line mode by using the NODMS system option. For more information, see "Interactive Line Mode in UNIX Environments" on page 9. △

What Is the Explorer Window?

Explorer is a windowing environment for managing basic SAS software tasks such as viewing and managing data sets, libraries, members, applications, and output. The SAS Explorer is a central access point from which you can do the following:

- manipulate SAS data through a graphical interface
- access the Program Editor, Output, and Log windows (as well as other windows)
- view the results of SAS procedure output in the Results window
- import files into SAS

What Are the Program Editor, Output, and Log Windows?

The Program Editor, Output, and Log windows enable you to edit and execute SAS programs and display output. For more information about these windows, see the online SAS Help and Documentation.

Invoking SAS in the Windowing Environment

You can use the following commands to specify which windows open when a SAS session starts.

- You can open the Program Editor, Output, and Log windows by specifying the DMS system option:

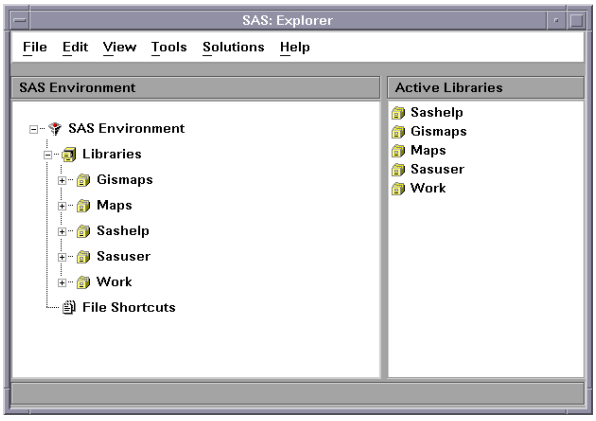
```
sas -dms
```

- You can open the Program Editor, Output, Log, and Results windows, as well as the Explorer window, by specifying the DMSEXP system option:

```
sas -dmsexp
```

- You can open only the Explorer window by specifying the EXPLORER system option:

```
sas -explorer
```



The default specification for invoking SAS is **`sas -dms -dmsexp`**. This command displays the Program Editor, Output, Log, and Results windows as well as the Explorer window. If you invoke SAS without the **`-dmsexp`** option, the Explorer window does not display.

SAS also opens a toolbox from which you can open additional SAS windows. For more information about the toolbox, see Chapter 7, "Working in the SAS Windowing Environment," on page 137.

Exiting SAS in the Windowing Environment

To end your SAS session, enter the BYE or ENDSAS command in the command window, or select **File ▶ Exit** from the drop-down menu of the SAS session that you want to end.

Interactive Line Mode in UNIX Environments

Introduction to Interactive Line Mode

If you are not using an X display, you can invoke SAS in interactive line mode by using the NODMS system option.

You enter SAS statements line by line in response to prompts issued by SAS. SAS reads the source statements from the terminal as you enter them. DATA and PROC steps execute when one of the following occurs:

- a RUN, QUIT, or DATALINES statement is entered
- another DATA or PROC statement is entered
- the ENDSAS statement is entered

To use interactive line mode, you must run SAS in the foreground.

Invoking SAS in Interactive Line Mode

To start an interactive line mode session, invoke SAS with the NODMS or NODMSEXP system option:

```
sas -nodms
sas -nodmsexp
```

By default, SAS log and procedure output (if any) appear on your display as each step executes.

You can also invoke SAS in interactive line mode and pass parameters to it:

```
sas -sysparm 'A B C' progparm.sas
```

The value **A B C** is assigned to the SYSPARM macro variable, which can be read by the program **progparm.sas**.

After you invoke SAS, the **1?** prompt appears, and you can begin entering SAS statements. After you enter each statement, a line number prompt appears.

Exiting SAS in Interactive Line Mode

You can end the session by pressing the EOF key (usually CTRL+D; see "Using Control Keys" on page 25) or by issuing the ENDSAS statement:

```
endsas;
```

The session ends after all SAS statements have executed.

Batch Mode in UNIX Environments

Introduction to Running SAS in Batch Mode

To run SAS in batch mode, you specify your SAS program name in the SAS invocation command. You can run batch mode in the foreground, in the background by specifying an ampersand at the end of the SAS command, or submit your application to the batch queue by using the **batch**, **at**, **nohup**, or **cron** UNIX commands. (For more information, refer to the UNIX man pages for the **batch**, **at**, **nohup**, or **cron** commands.) If you start your application with one of these UNIX commands and you log off of your system, then your application will complete execution. If your application contains statements that start an interactive procedure such as FSEDIT, then you need to run your batch application in the foreground.

Invoking SAS in Batch Mode

To invoke SAS in batch mode, you must specify a filename in the SAS command. For example, if **weekly.sas** is the file that contains the SAS statements to be executed, and you want to specify the NODATE and LINESIZE system options, you would enter the following command:

```
sas weekly.sas -nodate -linesize 90
```

The command would run the program in the foreground. If you want to run the program in the background, add the ampersand to the end of the command:

```
sas weekly.sas -nodate -linesize 90 &
```

SAS creates a .log file and a .lst file in the current directory that contains the log and procedure output.

Submitting a Program to the Batch Queue

To submit your program to the batch queue, you can use the **batch**, **at**, **nohup**, or **cron** commands. For example, you could submit **weekly.sas** from your shell prompt as follows:

```
$ at 2am
sas weekly.sas
<control-D>
warning: commands will be executed using /usr/bin/sh
job 8400.a at Wed Jun 11 02:00:00 2008
$
```

If you create a file that contains the SAS command (for example, **cmdfile.sh**) that is necessary to run your program, then you can enter the following command at your shell prompt:

```
at 2am < cmdfile.sh
```

SAS sends the output to a file that has the same name as the program. The output file has an extension of .lst. The log file writes to a file with an extension of .log. Both of these files are written to your current directory. Refer to the UNIX man pages for these commands for more information on submitting jobs to the batch queue. For more

information about routing output, see Chapter 4, "Printing and Routing Output," on page 89.

If you submit a file in batch mode, then a line that is greater than 256 bytes will be truncated. An explicit message about this truncation is written to the SAS log.

Note: If your program contains statements that start an interactive procedure such as the FSEDIT procedure, you will need to run your program as a foreground process. △

Writing Data from an External File Using UNIX Pipes

You can use a UNIX pipe to write data from an external file to a SAS program. For example, suppose that your data resides in the external file **mydata** and your SAS program **myprog.sas** includes this statement:

```
infile stdin;
```

Issue this command to have **myprog.sas** read data from the external file **mydata**:

```
cat mydata | sas myprog.sas
```

For information about using external files, see Chapter 3, "Using External Files and Devices," on page 67. See "File Descriptors in the Bourne and Korn Shells" on page 78 for another way to have a SAS program read data from an external file.

Running SAS on a Remote Host in UNIX Environments

Introduction to Running SAS on a Remote Host

When you invoke SAS in an interactive mode, you can run SAS on your local host, or you can run SAS on a remote host and interact with the session through an X server running on your workstation. The server provides the display services that are needed for the X Window System.

Most of the time, the server name is derived from the computer's name. For example, if your computer is named **green**, the name of the server is **green:0.0**. In most cases, the X server will already be running when you log in. If you need to start your server manually, consult the documentation that is provided with your X Window System software.

To run SAS on a remote host, you must tell SAS which display to use by either setting the DISPLAY environment variable or specifying the **-display** X command line option.

Steps for Running SAS on a Remote Host

To run SAS on a remote host, follow these steps:

1 Make sure that the clients running on the remote host have permission to connect to your server. With most X servers, authorization is controlled by using an .Xauthority file that is located in the user's home directory. Additionally, the **xhost** command can be used to circumvent authority. To use the **xhost** client to permit all remote hosts to connect to your server, enter the following command at the system prompt on the system that is running your X server:

```
xhost +
```

If your system does not control access with the **xhost** client, consult your system documentation for information on allowing remote access.

For information about editing and displaying authorization information, see the UNIX man page for xauth.

2 Log in to the remote system, or use a remote shell.

3 Identify your server as the target display for X clients that are run on the remote host. You can identify your server in one of two ways:

a Set the DISPLAY environment variable. In the Bourne and Korn shells, you can set the DISPLAY variable as follows:

```
DISPLAY=green:0.0
export DISPLAY
```

In the Korn shell, you can combine these two commands:

```
export DISPLAY=green:0.0
```

In the C shell, you must use the UNIX **setenv** command:

```
setenv DISPLAY green:0.0
```

The DISPLAY variable will be used by all X clients on the system.

Note: To determine the shell for your current system, type **ps** at the UNIX command prompt or check the value of the SHELL environment variable. △

b Use the DISPLAY system option. For example:

```
sas -display green:0.0
```

If you have trouble establishing a connection, you can try using an IP address instead of a display name, for example:

```
-display 10.22.1.1:0.0
```

Note: This option is a command line option for the X Window System, not for SAS. Specifying this option in a SAS configuration file or in the SASV9_OPTIONS environment variable might cause problems when you are running other interfaces. △

Preventing SAS from Attempting to Connect to the X Server

To prevent SAS from attempting to connect to the X server, unset the DISPLAY environment variable and use the **-noterminal** SAS option on the command line. The **-noterminal** option specifies that you do not want to display the SAS session. You must specify this option to generate a graph in batch mode. You must also specify this option when you use PROC IMPORT and PROC EXPORT. For more information, see "Running SAS/GRAPH Programs" in *SAS/GRAPH: Reference*.

Troubleshooting Connection Problems

If SAS cannot establish a connection to your display, it prints a message that indicates the nature of the problem and then terminates. An example of a message that you might receive is the following:

```
ERROR:  The connection to the X display server could not be made.
        Verify that the X display name is correct, and that you have
        access authorization. See the online Help for more information
```

```
about connecting to an X display server.
```

Make sure that you have brought up the SAS session correctly. You might need to use the **xhost** client (enter **xhost +**) or some other method to change display permissions. You can also specify the NODMS system option when you invoke SAS to bring your session up in line mode.

If you are unable to invoke SAS, try running another application such as **xclock**. If you cannot run the application, you should contact your UNIX system administrator for assistance.

X Command Line Options

How to Specify X Window System Options

When you invoke some X clients, such as SAS, you can use command line options that are passed to the X Window System. In general, you should specify X Window System options after SAS options on the command line.

Supported X Command Line Options

The following list describes the X command line options that are available when you invoke a SAS session from the command prompt.

-display *host*:*server*.*screen*
specifies the name or IP address of the terminal on which you want to display the SAS session. For example, if your display node is **wizard** whose IP address is 10.22.1.1:0.0, you might enter

```
-display wizard:0.0
```

or

```
-display 10.22.1.1:0.0
```

-name *instance-name*
reads the resources in your SAS resource file that begin with *instance-name*. For example, **-name MYSAS** reads the resources that begin with **MYSAS**, such as

```
MYSAS.dmsfont: Cour14
MYSAS.defaultToolbox: True
```

-title *string*
specifies a title for your SAS session window. Titles can contain up to 64 characters. Window titles are displayed in the case in which they are entered, which can be lower case, mixed case, or upper case. To use multiple words in the title, enclose the words in single or double quotation marks. For example, **-title MYSAS** produces **MYSAS:Explorer** in the title bar of the Explorer window.

-xrm *string*
specifies a resource to override any defaults. For example, the following resource turns off the Confirm dialog box when you exit SAS:

```
-xrm 'SAS.confirmSASExit: False'
```

Unsupported X Command Line Options

SAS does not support the following X command line options because their functionality is not applicable to SAS or is provided by SAS resources. Refer to "Overview of X Resources" on page 163 for more information on SAS resources.

-geometry
Window geometry is specified by the **SAS.windowHeight**, **SAS.windowWidth**, **SAS.maxWindowHeight**, and **SAS.maxWindowWidth** resources.

-background, -bg
These options are ignored.

-bordercolor, -bd
These options are ignored. Refer to "Defining Colors and Attributes for Window Elements (CPARMS)" on page 197 for a description of specifying the color of window borders.

-borderwidth, -bw
These options are ignored. The width of window borders is set by SAS.

-foreground, -fg
These options are ignored.

-font, -fn
SAS fonts are specified by the **SAS.DMSFont**, **SAS.DMSboldFont**, and **SAS.DMSfontPattern** resources.

-iconic
This option is ignored.

-reverse, -rv, +rv
These options are ignored. Refer to "Defining Colors and Attributes for Window Elements (CPARMS)" on page 197 for a description of specifying reverse video.

-selectionTimeout
Timeout length is specified by the **SAS.selectTimeout** resource.

-synchronous, +synchronous
The XSYNC command controls the X synchronization.

-xn1language
This option is ignored.

Executing Operating System Commands from Your SAS Session

Deciding Whether to Run an Asynchronous or Synchronous Task

You can execute UNIX commands from your SAS session either asynchronously or synchronously. When you run a command as an *asynchronous* task, the command executes independently of all other tasks that are currently running. To run a

command asynchronously, you must use the SYSTASK statement. See "SYSTASK Statement" on page 332 for information about executing commands asynchronously.

When you execute one or more UNIX commands *synchronously*, then you must wait for those commands to finish executing before you can continue working in your SAS session. You can use the CALL SYSTEM routine, %SYSEXEC macro program statement, X statement, and X command to execute UNIX commands synchronously. The CALL SYSTEM routine can be executed with a DATA step. The %SYSEXEC macro statement can be used inside macro definitions, and the X statement can be used outside of DATA steps and macro definitions. You can enter the X command on any SAS command line. See "CALL SYSTEM Routine" on page 259 and "Macro Statements in UNIX Environments" on page 285 for more information.

Executing a Single UNIX Command

To execute only one UNIX command, you can enter the X command, X statement, CALL SYSTEM routine, or %SYSEXEC macro statement as follows:

X *command*

X *command*;

CALL SYSTEM ('*command*');

%SYSEXEC *command*;

Note: When you use the %SYSEXEC macro statement, if the UNIX command you specify includes a semicolon, you must enclose the UNIX command in a macro quoting function. Refer to *SAS Macro Language: Reference* for more information about quoting functions. △

Example 1: Executing a UNIX Command by Using the X Statement

You can use the X statement to execute the **ls** UNIX command (in a child shell) as follows:

```
x ls -l;
```

Example 2: Executing a UNIX Command by Using the CALL SYSTEM Routine

Inside a DATA step, you can use the CALL SYSTEM routine to execute a **cd** command, which will change the current directory of your SAS session:

```
data _null_;
call system ('cd /users/smith/report');
run;
```

The search for any relative (partial) filenames during the SAS session will now begin in the **/users/smith/report** directory. When you end the session, your current directory will be the directory in which you started your SAS session.

For more information about the CALL SYSTEM routine, see "CALL SYSTEM Routine" on page 259.

How SAS Processes a Single UNIX Command

When you specify only one command, SAS checks to see whether the command is **cd**, **pwd**, **setenv**, or **umask** and, if so, executes the SAS equivalent of these commands. The SAS **cd** and **pwd** commands are equivalent to their Bourne shell counterparts. The SAS

setenv command is equivalent to its C shell namesake. The SAS **umask** command is equivalent to the numeric mode of the **umask** command supported by the Bourne, Korn, and C shells. These four commands are built into SAS because they affect the environment of the current SAS session. When executed by SAS software, they affect only the SAS environment and the environment of any shell programs started by the SAS session. They do not affect the environment of the shell program that began your SAS session.

If the command is not **cd**, **pwd**, or **setenv**, SAS starts a shell in which it executes the command that you specified. The shell that is used depends on the SHELL environment variable. If the command is **umask**, but you do not specify a *mask*, then SAS passes the command to the shell in which the current SAS session was started. For more information about the **umask** command, see "Changing the File Permissions for Your SAS Session" on page 16.

Executing Several UNIX Commands

You can also use the X command, X statement, CALL SYSTEM routine, and %SYSEXEC macro statement to execute several UNIX commands:

X '*command-1*;*...command-n*'

X '*command-1*;*...command-n*';

CALL SYSTEM ('*command-1*;*...command-n*');

%SYSEXEC *quoting-function*(*command-1*;*...command-n*);

Separate each UNIX command with a semicolon (;).

Note: When you use the %SYSEXEC macro statement to execute several UNIX commands, because the list of commands uses semicolons as separators, you must enclose the string of UNIX commands in a macro quoting function. Refer to *SAS Macro Language: Reference* for more information on quoting functions. △

Example: Executing Several Commands Using the %SYSEXEC Macro

The following code defines and executes a macro called **pwdls** that executes the **pwd** and **ls -l** UNIX commands:

```
%macro pwdls;
%sysexec %str(pwd;ls -l);
%mend pwdls;
%pwdls;
```

This example uses **%str** as the macro quoting function.

How SAS Processes Several UNIX Commands

When you specify more than one UNIX command (that is, a list of commands separated by semicolons), SAS passes the entire list to the shell and does not check for the **cd**, **pwd**, **setenv**, or **umask** commands, as it does when a command is specified by itself (without semicolons).

For more information about how SAS processes the **cd**, **pwd**, **setenv**, or **umask** commands, see "How SAS Processes a Single UNIX Command" on page 15.

Changing the File Permissions for Your SAS Session

At invocation, a SAS session inherits the file permissions from the parent shell. Any file that you create will inherit these permissions. If you want to change or remove file

permissions from within SAS, issue the following command in the X statement:**umask**. The **umask** command applies a new "mask" to a file, that is, it sets new file permissions for any new file that you create. In this way, the **umask** command can provide file security by restricting access to new files and directories for the current process.

The default value for **umask** is 000, but you can set different values for **umask** in your .kshrc and .cshrc files. These values affect all child processes that are executed in the shell. Any subsequent file that you create during the current SAS session will inherit the permissions that you specified. The permissions of a file created under a given mask are calculated in octal representation.

Note: The value of a mask can be either numeric or symbolic. For more information about this command, see the UNIX man page for **umask**. △

Executing X Statements in Batch Mode

If you run your SAS program in batch mode and if your operating system supports job control, the program will be suspended when an X statement within the program needs input from the terminal.

If you run your SAS program from the batch queue by submitting it with the **at** or **batch** commands, SAS processes any X statements as follows:

- If the X statement does not specify a command, SAS ignores the statement.
- If any UNIX command in the X statement attempts to get input, it receives an end-of-file (standard input is set to **/dev/null**).
- If any UNIX command in the X statement writes to standard output or standard error, the output is mailed to you unless it was previously redirected.

Customizing Your SAS Registry Files

SAS registry files store information about the SAS session. The SAS registry is the central storage area for configuration data for SAS. The following list identifies some of the data that is stored in the registry:

- the libraries and file shortcuts that SAS assigns at start–up. These shortcuts could include secure information, such as your password.
- the printers that are defined for use and their print setup.
- configuration data for various SAS products.

The Sasuser registry file (called regstry.sas7bitm) contains your user defaults. These registry entries can be customized by using the SAS Registry Editor or by using PROC REGISTRY. For more information, see "The SAS Registry" in *SAS Language Reference: Concepts*.

CAUTION:
For experienced users only. Registry customization is generally performed by experienced SAS users and system administrators. △

Customizing Your SAS Session by Using System Options

Ways to Customize Your SAS Session

You can customize your SAS environment in several ways. One way is through the use of SAS system options. For information about other ways to customize a SAS session, see Chapter 8, "Customizing the SAS Windowing Environment," on page 161.

Ways to Specify a SAS System Option

SAS options can be specified in one or more ways:

- in a configuration file
- in the SASV9_OPTIONS environment variable
- in the SAS command
- in an OPTIONS statement (either in a SAS program or an autoexec file)
- in the System Options window

Table 18.1 on page 342 shows where each SAS system option can be specified.

Any options that do not affect the initialization of SAS, such as CENTER and NOCENTER, can be specified and changed at any time.

Some options can be specified only in a configuration file, in the SASV9_OPTIONS variable, or in the SAS command. These options determine how SAS initializes its interfaces with the operating system and the hardware; they are often called configuration options. After you start a SAS session, these options cannot be changed. Usually, configuration files specify options that you would not change very often. In those cases when you need to change an option just for one job, specify the change in the SAS command.

Overriding the Default Value for a System Option

The default values for SAS system options will be appropriate for many of your SAS programs. However, you can override a default setting using one or more of the following methods:

configuration file
: Modify your current configuration file (see "Order of Precedence for Processing SAS Configuration Files" on page 22) or create a new configuration file. Specify SAS system options in the file by preceding each with a hyphen. For ON/OFF options, just list the keyword corresponding to the appropriate setting. For options that accept values, list the keyword identifying the option followed by the option value. All SAS system options can appear in a configuration file.

  For example, a configuration file might contain these option specifications:

```
-nocenter
-verbose
-linesize 64
```

SASV9_OPTIONS environment variable
: Specify SAS system options in the SASV9_OPTIONS environment variable before you invoke SAS. See "Defining Environment Variables in UNIX Environments" on page 23.

Settings that you specify in the SASV9_OPTIONS environment variable affect SAS sessions that are started when the variable is defined.

For example, in the Korn shell, you would use:

```
export SASV9_OPTIONS='-fullstimer -nodate'
```

SAS command
Specify SAS system options in the SAS command. Precede each option with a hyphen:

```
sas -option1 -option2...
```

For ON/OFF options, list the keyword corresponding to the appropriate setting. For options that accept values, list the keyword that identifies the option, followed by the option value. For example,

```
sas -nodate -work mywork
```

Settings that you specify in the SAS command last for the duration of the SAS session or, for those options that can be changed within the session, until you change them. All options can be specified in the SAS command.

OPTIONS statement within a SAS session
Specify SAS system options in an OPTIONS statement at any point within a SAS session. The options are set for the duration of the SAS session or until you change them. When you specify an option in the OPTIONS statement, do not precede its name with a hyphen (-). If the option has an argument, use = after the option name.

For example,

```
options nodate linesize=72;
options editcmd='/usr/bin/xterm -e vi';
```

Refer to *SAS Language Reference: Dictionary* for more information about the OPTIONS statement. Not all options can be specified in the OPTIONS statement. To find out about a specific option, look up its name in Table 18.1 on page 342.

OPTIONS statement in an autoexec file
Specify SAS system options in an OPTIONS statement in an autoexec file. For example, your autoexec file could contain the following statements:

```
options nodate pagesize=80;
filename rpt '/users/myid/data/report';
```

System Options window
Change the SAS system options from within the System Options window.

In general, use quotation marks to enclose filenames and pathnames specified in the OPTIONS statement or the System Options window. Do not use quotation marks otherwise. Any exceptions are discussed under the individual option. You can use the abbreviations listed in Table 2.6 on page 51 to shorten the filenames and pathnames you specify.

How SAS Processes System Options Set in One Place

If the same option is set more than once within the SAS command, a configuration file, or the SASV9_OPTIONS environment variable, only the last setting is used; the others are ignored. For example, the DMS option is ignored in the following SAS command:

```
sas -dms -nodms
```

The DMS option is also ignored in the following configuration file:

```
-dms
-linesize 80
-nodms
```

By default, if you specify the HELPLOC, MAPS, MSG, SAMPLOC, SASAUTOS, or SASHELP system options more than one time, the last value that is specified is the value that SAS uses. If you want to add additional pathnames to the pathnames already specified by one of these options, you must use the APPEND or INSERT system options. For more information, see "APPEND System Option" on page 355 and "INSERT System Option" on page 377.

How SAS Processes System Options Set in Multiple Places

When the same option is set in more than one place, the most recent specification is used. The following places are listed in order of precedence. For example, a setting made in the System Options window or OPTIONS statement will override any other setting, but if you set a system option using the SASV9_OPTIONS environment variable, then this option will override only the setting for the same system option in your configuration file.

Order of Precedence for Processing System Options

The precedence for processing system options is as follows:

1. System Options window or OPTIONS statement (from a SAS session or job).
2. autoexec file that contains an OPTIONS statement (after SAS initializes).
3. SAS command.
4. SASV9_OPTIONS environment variable.
5. configuration files (before SAS initializes). For more information, see "Order of Precedence for Processing SAS Configuration Files" on page 22.

For example, if a configuration file specifies NOSTIMER, you can override the setting in the SAS command by specifying –FULLSTIMER.

By default, if you specify the HELPLOC, MAPS, MSG, SAMPLOC, SASAUTOS, or SASHELP system option more than one time, the last value that is specified is the value that SAS uses. If you want to add additional pathnames to the pathnames already specified by one of these options, you must use the APPEND or INSERT system options to add the new pathname. See "APPEND System Option" on page 355 and "INSERT System Option" on page 377 for more information.

Customizing Your SAS Session by Using Configuration and Autoexec Files

Customizing Your SAS Session

You can customize your SAS environment in several ways. To customize your SAS environment at the point of invocation, you can use configuration and autoexec files. For information about how to customize a SAS session using the windowing environment, see Chapter 8, "Customizing the SAS Windowing Environment," on page 161.

Introduction to Configuration and Autoexec Files

You can customize your SAS session by defining configuration and autoexec files. You can use these files to specify system options and to execute SAS statements automatically whenever you start a SAS session. (SAS system options control many aspects of your SAS session, including output destinations, the efficiency of program execution, and the attributes of SAS files and libraries. Refer to *SAS Language Reference: Dictionary* for a complete description of SAS system options.)

The configuration file (for SAS 9.2) is typically named sasv9.cfg, and the autoexec file is named autoexec.sas. These files typically reside in the directory where SAS was installed. By default, this directory is the **!SASROOT** directory.

You can have customized configuration and autoexec files in your user home directory. If you do, then SAS will use the customizations specified in these files when you start a SAS session. For more information about the order of precedence SAS uses when processing configuration files, see "Order of Precedence for Processing SAS Configuration Files" on page 22.

SAS system options can be restricted by a UNIX system administrator, so that once they are set by the administrator, they cannot be changed by a user. A system option can be restricted globally, by group, and by user. For more information, see the configuration guide for the UNIX environment on **support.sas.com**, and see in *SAS Language Reference: Dictionary*.

Differences between Configuration and Autoexec Files

The differences between configuration files and autoexec files are as follows:

- Configuration files can contain only SAS system option settings, while autoexec files can contain any valid SAS statement. For example, you might want to create an autoexec file that includes an OPTIONS statement to change the default values of various system options and LIBNAME and FILENAME statements for the SAS libraries and external files that you use most often.
- Configuration files are processed *before* SAS initializes, while autoexec files are processed immediately *after* SAS initializes but before it processes any source statements. An OPTIONS statement in an autoexec file is equivalent to submitting an OPTIONS statement as the first statement of your SAS session.

Creating a Configuration File

To create a configuration file, follow these steps:

1. Use a text editor to write the SAS system options into a UNIX file. Save the file as either sasv9.cfg or .sasv9.cfg. (See "Order of Precedence for Processing SAS Configuration Files" on page 22 for more information.)
2. Specify one or more system options on each line. Use the same syntax that you would use for specifying system options with the SAS command, but do not include the SAS command itself. For example, a configuration file might contain the following lines:

```
-nocenter
-verbose
-linesize 64
-work /users/myid/tmp
```

3. Save and close the configuration file.

Order of Precedence for Processing SAS Configuration Files

SAS is shipped with a default configuration file in the **!SASROOT** directory. Your on-site SAS personnel can edit this configuration file so that it contains whichever options are appropriate to your site.

You can also create one or more of your own configuration files. SAS reads option settings from each of these files in the following order:*

1. sasv9.cfg in the **!SASROOT** directory. (See Appendix 1, "The !SASROOT Directory," on page 421.)
2. sasv9_local.cfg in the **!SASROOT** directory. (See Appendix 1, "The !SASROOT Directory," on page 421.)
3. .sasv9.cfg in your home directory. (Notice the leading period.)
4. sasv9.cfg in your home directory.
5. sasv9.cfg in your current directory.
6. any restricted configuration files. Restricted configuration files contain system options that are set by the site administrator and cannot be changed by the user. Options can be restricted globally, by group, or by user. For more information about restricted configuration files, see the configuration guide for the UNIX environment.

For each system option, SAS uses the last setting it encounters; any other settings are ignored. For example, if the WORKPERMS system option is specified in sasv9.cfg in the **!SASROOT** directory and in sasv9.cfg in your current directory, SAS will use the value specified in sasv9.cfg in your current directory.

Specifying a Configuration File for SAS to Use

When you specify a configuration file for SAS to use, you bypass the search of the configuration files listed in "Order of Precedence for Processing SAS Configuration Files" on page 22.

Note: SAS still processes any restricted configuration files that exist. The settings in these files take precedence over the settings in the configuration file that you specify. △

To specify a configuration file, complete one of the following steps:

- specify a configuration file with the CONFIG system option in the SAS command:

```
sas -config filename
```

- specify a configuration file in the SASV9_OPTIONS environment variable. See "Defining Environment Variables in UNIX Environments" on page 23. For example, in the Korn shell, you would use:

```
export SASV9_OPTIONS='-config filename'
```

- define the environment variable SASV9_CONFIG. See "Defining Environment Variables in UNIX Environments" on page 23. For example, in the Korn shell, you would use:

```
export SASV9_CONFIG=filename
```

filename is the name of a file that contains SAS system options.

* For future releases of SAS, the names of these files will change accordingly.

If you have specified a configuration file in the SASV9_OPTIONS or SASV9_CONFIG environment variables, you can prevent SAS from using that file by specifying NOCONFIG in the SAS command.

If SAS cannot find SASV9_OPTIONS, the following message is written to the SAS log:

```
ERROR: Cannot open [/fullpath/filename]: No such
       file or directory.
```

Defining Environment Variables in UNIX Environments

What Is a UNIX Environment Variable?

UNIX *environment variables* are variables that apply to both the current shell and to any subshells it creates (for example, when you send a job to the background or execute a script). If you change the value of an environment variable, the change is passed forward to subsequent shells but not backward to the parent shell.

In a SAS session, you can use the SASV9_OPTIONS environment variable to specify system options and the SASV9_CONFIG environment variable to specify a configuration file. You can also use environment variables as filerefs and librefs in various statements and commands. Filerefs and librefs consist of uppercase letters, digits, and the underscore character in environment variable names. Other characters are not recognized by SAS.

Note: A SAS/ACCESS product initializes the environment variables it needs when loading. Any changes that you make to an environment variable after initialization will not be recognized. For more information, see the documentation for your SAS/ACCESS product. △

How to Define an Environment Variable for Your Shell

The way in which you define an environment variable depends on the shell that you are running. (To determine which shell you are running, type **ps** at the command prompt or **echo $SHELL** to see the current value of the SHELL environment variable.)

Bourne and Korn Shells

In the Bourne shell and in the Korn shell, use the **export** command to export one or more variables to the environment. For example, these commands make the value of the variable **scname** available to all subsequent shell scripts:

```
$ scname=phonelist
$ export scname
```

In the Korn shell, you can combine these commands into one command:

```
$ export scname=phonelist
```

If you change the value of **scname**, then the new value affects both the shell variable and the environment variable. If you do not export a variable, only the shell script in which you define has access to its value.

C Shell

In the C shell (csh and tcsh), you set (define and export) environment variables with the **setenv** (set environment) command. For example, this command is equivalent to the commands shown previously:

```
% setenv scname phonelist
```

Displaying the Value of an Environment Variable

To display the values of individual environment variables, use the **echo** command and parameter substitution. An example is: **echo $SHELL** which returns the current value of the SHELL environment variable. Use the **env** (or **printenv**) command to display all environment variables and their current values.

Determining the Completion Status of a SAS Job in UNIX Environments

The exit status for the completion of a SAS job is returned in **$STATUS** for the C shell, and in **$?** for the Bourne and Korn shells. A value of 0 indicates normal termination. You can affect the exit status code by using the ABORT statement. The ABORT statement takes an optional integer argument, *n*, which can range from 0 to 255.

Note: Return codes of 0–6 and return codes greater than 977 are reserved for use by SAS. △

The following table summarizes the values of the exit status code.

Table 1.1 Exit Status Code Values

Condition	Exit Status Code
All steps terminated normally	0
SAS System issued warnings	1
SAS System issued errors	2
User issued ABORT statement	3
User issued ABORT RETURN statement	4
User issued ABORT ABEND statement	5
SAS could not initialize because of a severe error	6
User issued ABORT RETURN *n* statement	*n*
User issued ABORT ABEND *n* statement	*n*

If you specify the ERRORABEND SAS system option on the command line, and the job has errors, the exit status code is set to 5.

UNIX exit status codes are in the range 0-255. Numbers greater than 255 might not print what you expect because the code is interpreted as a signed byte.

Exiting or Interrupting Your SAS Session in UNIX Environments

Methods for Exiting SAS

Use one of the following methods to exit a SAS session:

- Select **File ▶ Exit** if you are using SAS in the windowing environment.
- Use **endsas;**.
- Enter **BYE** in the ToolBox if you are using SAS in the windowing environment.
- Use CTRL+D if this control key sequence is your EOF command and if you are using SAS in interactive line mode.

Methods for Interrupting or Terminating SAS

In addition to the methods for exiting SAS, SAS provides methods for interrupting or terminating a SAS session. SAS does not recommend that you use these methods until you have tried to exit SAS by one of the methods listed in "Methods for Exiting SAS" on page 25.

You can interrupt or terminate SAS in the following ways:

- Press the interrupt or quit control key.
- Use the SAS Session Manager.
- Enter the UNIX **kill** command. Use this command when all other methods of exiting SAS have failed.

  Using the UNIX **kill** command on a SAS process that is running might corrupt data sets that are open for write or update access.

Using Control Keys

Control keys enable you to interrupt or terminate your session by pressing the interrupt or quit key sequence. However, control keys can be used only when your SAS program is running in interactive line mode or in batch mode in the foreground. You cannot use control keys to stop a background job.

Note: You cannot use control keys to stop a batch job that has been submitted with the **batch**, **at**, **nohup**, or **cron** command. △

Because control keys vary from system to system, issue the UNIX **stty** command to determine which key sends which signal. The **stty** command varies considerably among UNIX operating environments, so check the UNIX man page for **stty** before using the command. Usually, one of these forms of the command will print all of the current terminal settings:

```
stty
stty -a
stty everything
```

The output should contain lines similar to these:

```
intr = ^C; quit = ^\; erase = ^H;
kill = ^U; eof = ^D; eol = ^@
```

The caret (^) represents the CTRL key. In this example, CTRL+C is the interrupt key and CTRL+\ is the quit key.

Pressing the quit key is equivalent to specifying the **-SIGTERM** option on the **kill** command.

Using the SAS Session Manager

If you invoke SAS in the windowing environment, you can use the SAS Session Manager to interrupt or terminate your SAS session. The SAS Session Manager is automatically minimized when you start SAS. To interrupt or terminate your SAS session, open the SAS Session Manager window and click **Interrupt** or **Terminate**.

If asynchronous SAS/CONNECT tasks are running when you terminate a SAS session, these tasks are terminated and no warning message is displayed.

Note: Clicking **Interrupt** is equivalent to specifying the **-SIGINT** option on the **kill** command. Clicking **Terminate** is equivalent to specifying the **-SIGTERM** option on the **kill** command. △

For more information about the SAS Session Manager, see "The SAS Session Manager (motifxsassm) in UNIX" on page 141.

Using the UNIX kill Command

Note: Use the **kill** command only after you have tried all other methods to exit your SAS session. △

The **kill** command sends an interrupt or terminate signal to SAS, depending on which signal you specify. You can use the **kill** command to interrupt or terminate a SAS session running in any mode. The **kill** command cannot be issued from within a SAS session. You must issue it from another terminal or from another window (if your terminal permits it).

The format of the **kill** command is:

kill <-*signal-name*> *pid*

To send the interrupt signal, specify **-SIGINT**. To send the terminate signal, specify **-SIGTERM**. Use the **ps** command and its options to determine the process identification number (*pid*) of the SAS session that you want to interrupt or terminate.

The results of using the **ps** command differ in different operating environments. See the UNIX man page for your operating environment for specific information about the **ps** command and its options. Adding options helps to determine which process you want to kill if you have more than one SAS process running. Also, servers (metadata, OLAP, and so on) leave a process identification number in their start-up directories. You can use this number with the **kill** command.

The following table lists some of the important **kill** signals.

Table 1.2 Description of Important kill Signals

Signal	Option	Description
0	SIGNULL	Checks access to process identifier
1	SIGHUP	Causes SAS to terminate
2	SIGINT	Causes SAS to interrupt the session
3	SIGQUIT	Causes SAS to terminate and generates a core file

Signal	Option	Description
9	SIGKILL	Causes a forced termination of the SAS session
15	SIGTERM	Causes SAS to terminate

For more information, see the UNIX man pages for the **ps** and **kill** commands.

Messages in the SAS Console Log

If SAS encounters an error or warning condition when the SAS log is not available, then any messages that SAS issues are written to the SAS console log. Normally, the SAS log is unavailable only early in SAS initialization and late in SAS termination.

If you are using the -STDIO option, the log is displayed in stderr, and the listing is displayed in STDOUT.

Ending a Process That Is Running as a SAS Server

If you need to end a process running as a SAS server, use one of the following methods:

- If you are using the SAS Metadata Server, use the SAS Management Console to end a process.
- If you are using another SAS server, use the UNIX scripts that shipped with the servers to stop the process. You can also use these scripts to start (or restart) a server, as well as determine whether the server is already running. For more information about these scripts, contact your site administrator.

  Note: If the server does not respond to the UNIX script, then you can use the **kill** command to end the server process. For more information, see "Using the UNIX kill Command" on page 26. △

Ending a SAS Process on a Relational Database

How to Interrupt a SAS Process

CAUTION:

When you interrupt a SAS process, you might terminate the current query. If you are using the current query to create a new data set, then the data set is still created even if the query is terminated. If you are using the current query to overwrite a data set, the data set is not overwritten if the query is terminated. In most cases you do not receive a warning that the query did not complete. △

The method that you use to interrupt a SAS process depends on how you invoke SAS.

- If you are running SAS in interactive line mode or in batch mode using a foreground process, then you can use either of the following methods to interrupt SAS:
  - Press the control key sequence that is set to interrupt in the shell that invoked SAS. In most cases, this control key sequence is CTRL+C. See the man page for the **stty** command to determine the appropriate key sequence for your environment.

- Use the **-SIGINT** option in the **kill** command. For more information, see "Using the UNIX kill Command" on page 26.

- If you are running the SAS windowing environment in the foreground, then click **Interrupt** in the SAS Session Manager window.

  Note: You can access the SAS Session Manager by invoking SAS with the -dms or -dmsexp option. Select SAS: Session Management from the menu. △

- If you are running SAS in batch mode, then you must click **Interrupt** in the SAS Session Manager window. You cannot use a control key sequence to interrupt the SAS process.

The interrupt signal is sent to the host supervisor. The supervisor determines which DATA steps or procedures are running, and gives you options to interrupt these DATA steps or procedures. The actions that you can take appear in the interrupt menu. After you select an action, the host supervisor performs the operation you selected. Because the options in the interrupt menu are dependent on what is currently executing, you might see a different interrupt menu for the following:

- each SAS procedure. The only option that procedures can act on is the Halt Datastep/Proc option.
- a DATA step. The options available when SAS is processing a DATA step are different from when SAS is processing a procedure.
- each SAS application. For example, SAS webAF has a different interrupt menu than the one for PROC SQL.

Note: Depending on the relational database, the interrupt signal might be handled differently. The interrupt signal is not handled until a safe point in the code is reached that allows the interrupt handler to be run safely. △

Example: Interrupt Menu for PROC SQL

The following is an example of the interrupt menu that you might see if you issue an interrupt signal while SAS is processing a PROC SQL statement:

```
Select:
      1. Cancel Submitted Statements
      2. Halt Datastep/Proc: SQL
      C. Cancel the dialog
      T. Terminate the SAS System
```

The following table explains each of these options:

Table 1.3 Description of Interrupt Menu Options for PROC SQL

Option	Description	What This Option Does
1	Cancel Submitted Statements	Selecting this option will end the current DATA step or procedure and the underlying DBMS process. Outstanding source code that is waiting to execute will be flushed from the system. In interactive mode, you will return to the command prompt.
2	Halt Datastep/Proc: SQL	If you select this option and SAS is currently executing an SQL procedure, then the following menu appears: **Press:** **C to continue** **Q to cancel the current query** **S to cancel the submitted statements** **X to exit SQL procedure** **?** □ If you select **C**, then the menu will disappear, but because the current query ended when you interrupted the SAS process, SAS will not return to the current query. Instead, SAS will begin processing the next line of code. □ If you select **Q**, then SAS cancels the current query even if it is on a relational database. SAS continues processing the next statement. □ If you select **S**, then all of the PROC SQL statements that you submitted are canceled. □ If you select **X**, SAS exits the current SQL procedure and starts processing the next statement in the submit block.
C	Cancel the dialog	Selecting this option returns you to normal processing; however, the current query might have been interrupted. If you are running a long query and the control is on the DBMS server, then selecting **C** will end the current query. If you are running a short query and SAS has the control, then selecting **C** will cause the interrupt menu to disappear and the current query will continue. To determine whether the query was interrupted while reading or writing out the DBMS data, use PROC PRINT to view the partially created DBMS table or SAS data set.
T	Terminate SAS	Selecting this option ends your SAS session as well as the current query.

How to Terminate a SAS Process

The method that you use to terminate a SAS process depends on how you invoke SAS.

- If you are running the SAS windowing environment in the foreground, then click **Terminate** in the SAS Session Manager window.
- If you are running an interactive SAS process in the background, then you must click **Terminate** in the SAS Session Manager window. You cannot use a control key sequence to terminate the SAS process.

If you click **Terminate** in the SAS Session Manager, then a dialog box appears confirming that you want to end the session. If you click **OK**, then both the SAS session and the current query are terminated. If you click **Cancel**, then you are returned to the SAS session.

What Happens When You Interrupt a SAS Process and the Underlying DBMS Process

CAUTION:

Interrupting a SAS process and the underlying DBMS process might kill all jobs that are running on your DBMS. Interrupting your SAS and DBMS processes should be an exception. Extensive care should be taken when you construct your queries.

If SAS sends SQL to an RDBMS, there is no way to interrupt the SQL statements because SAS no longer has control of them. The statements are running in the RDBMS. △

Note: In this section, SAS process refers to a series of events. It is not the process on the operating system. When you interrupt or terminate a SAS process, the process on the operating system might still be running. △

When you interrupt or terminate a query on a server, the following processes stop:

- processing of current extractions. For example, suppose you forgot to include a WHERE clause in your SQL query and are now extracting 1 billion rows into SAS. Issuing an interrupt stops the SAS process and the extract step in the DBMS.
- processing of queries that are in progress on the server. For example, you have a very complex extract query that runs for a long time before producing a result. Issuing an interrupt stops the SAS and DBMS processes. As a result, the complex query running on your DBMS server is interrupted and terminated.
- update, delete, and insert processing. For example, you are updating, deleting, or inserting many rows in your DBMS. An interrupt stops the SAS and DBMS processes.