SAS Publishing

# SAS® 9.1 Companion for Windows

*The Power to Know*®

The correct bibliographic citation for this manual is as follows: SAS Institute Inc. 2004. *SAS® 9.1 Companion for Windows*. Cary, NC: SAS Institute Inc.

**SAS® 9.1 Companion for Windows**

ISBN 1-59047-200-4

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

1st printing, January 2004

# Contents

*What's New* **ix**
Overview **ix**
Basic Operation of SAS **ix**
Accessibility **x**
Enhanced Editor **xi**
Graphical User Interface **xi**
SAS Language Elements **xii**

**PART 1 Running SAS under Windows 1**

**Chapter 1 △ Getting Started 3**
SAS: Exploiting the Power of Windows **4**
Starting SAS **6**
Files Used by SAS **12**
Submitting SAS Code **23**
Interrupting Your SAS Session **25**
Running Windows or MS-DOS Commands from within SAS **25**
Terminating a SAS Process **28**
Ending Your SAS Session **29**

**Chapter 2 △ Interacting with SAS under Windows 31**
Overview of the SAS Interface **33**
Working within Your SAS Session **35**
Customizing Your SAS Session **56**
Accessing Online Help and Documentation **72**
Accessibility Features in SAS under Windows **74**

**Chapter 3 △ Using the SAS Editors 81**
Using the Enhanced Editor **82**
Using the Program Editor **108**

**Chapter 4 △ Using SAS Files 117**
Introduction to SAS Files **118**
Multiple Engine Architecture **122**
Using Data Libraries **125**
Accessing SAS Files from Multiple SAS Sessions **135**
Using SAS Files from Other Versions with SAS 9.1 for Windows **136**
Using SAS 9.1 Files with Previous Releases **140**
Using Remote Host SAS Files in SAS 9.1 **140**
Reading BMDP, OSIRIS and SPSS Files **140**
Transferring SAS Files between Operating Environments **143**
Accessing Database Files with SAS/ACCESS Software **143**

Using the SAS ODBC Driver to Access SAS Data from Other Applications **143**

**Chapter 5 △ Using External Files 145**
About External Files **146**
Referencing External Files **146**
Accessing External Files with SAS Statements **156**
Accessing External Files with SAS Commands **158**
Advanced External I/O Techniques **160**
Reading Data from the Communications Port **162**

**Chapter 6 △ Managing SAS Output 165**
Printing **166**
Routing Procedure Output to a Web Browser **179**
Routing Procedure Output and the SAS Log to a File **182**
Producing Graphics **184**

**Chapter 7 △ Performance Considerations 195**
Hardware Considerations **195**
Windows Features That Optimize Performance **198**
SAS Features That Optimize Performance **205**
Network Performance Considerations **206**
Advanced Performance Tuning Methods **206**

**PART *2* Using SAS with Other Windows Applications 211**

**Chapter 8 △ Using Lotus Notes to Distribute SAS Data 213**
Introduction to Using Lotus Notes with SAS **213**
Populating a Lotus Notes Database Using the DATA Step and SCL Code **214**

**Chapter 9 △ Using Windows System Tools with SAS 221**
Introduction to Using Windows System Tools with SAS **221**
Event Viewer Application Log **222**
Performance Tools **226**
Starting SAS as a Windows Service **231**

**Chapter 10 △ Using OLE in SAS/AF Software 241**
About OLE **242**
SAS/AF Catalog Compatibility **242**
Inserting an OLE Object in a FRAME Entry **242**
Editing an OLE Object within a FRAME Entry **245**
Invoking OLE Verbs **247**
Using Linked OLE Objects **247**
Converting OLE Objects **249**
Automating OLE Objects and Applications **250**
Using OLE Custom Controls (OCXs) in Your SAS/AF Application **257**

**Chapter 11 △ Controlling SAS from Another Application Using OLE 265**

Introduction to Automating SAS **265**
Creating an Instance of SAS **265**
Getting Feedback from the SAS Session **266**
Examples of Automating SAS with OLE **267**
Methods and Properties for Use with a SAS OLE Automation Object **268**

**Chapter 12 △ Using Dynamic Data Exchange 273**
Overview of Dynamic Data Exchange (DDE) **273**
DDE Syntax within SAS **273**
Referencing the DDE External File **274**
DDE Examples **276**

**Chapter 13 △ Using Unnamed and Named Pipes 283**
Overview of Pipes **283**
Using Unnamed Pipes **284**
Using Named Pipes **286**

**Chapter 14 △ Accessing External DLLs from SAS 295**
Overview of Dynamic Link Libraries in SAS **295**
The SASCBTBL Attribute Table **296**
Special Considerations When Using External DLLs **301**
Examples **310**

**Chapter 15 △ Considerations for SAS/AF Programmers 315**
Controlling the Appearance and Behavior of SAS **315**
Controlling the Main SAS Window **315**
Accessing External DLLs from SAS **318**
Designing, Saving, and Loading Custom Toolbar Controls **319**
Invoking SAS/AF Applications Automatically **319**
Associating Your Own Logo and Icons with Your SAS/AF Application **319**
Incorporating Electronic Mail into Your SAS/AF Application **320**

**PART *3* Features of the SAS Language for Windows 321**

**Chapter 16 △ Commands under Windows 323**
SAS Commands under Windows **324**

**Chapter 17 △ Formats under Windows 375**
SAS Formats under Windows **375**
Writing Binary Data **375**
Converting User-Written Formats from Earlier Releases to SAS 9.1 **376**

**Chapter 18 △ Functions and CALL Routines under Windows 385**
SAS Functions and Call Routines under Windows **385**

**Chapter 19 △ Informats under Windows 413**
SAS Informats under Windows **413**

Reading Binary Data 413
Converting User-Written Informats from Earlier Releases to SAS 9.1 414

**Chapter 20 △ Procedures under Windows 423**
SAS Procedures under Windows 423

**Chapter 21 △ Statements under Windows 441**
SAS Statements under Windows 441

**Chapter 22 △ System Options under Windows 465**
SAS System Options under Windows 467
Displaying SAS System Option Settings 468
Changing SAS System Option Settings 468
Processing System Options That Are Set in Several Places 470
Summary of System Options for Windows 470

**Chapter 23 △ Length and Precision of Variables 579**
Length and Precision of Variables under Windows 579
Numeric Variables 579
Character Variables 580

**Chapter 24 △ Macro Facility under Windows 581**
SAS Macro Facility under Windows 581
Automatic Macro Variables 581
Macro Statements 583
Macro Functions 583
Autocall Libraries 584

**PART 4 Appendices 585**

**Appendix 1 △ SCL Methods for Automating OLE Objects 587**
Summary of OLE Class Methods 587

**Appendix 2 △ Error Messages for SAS under Windows 597**
Overview of SAS Error Messages 597
Return Codes and Completion Status 597
Accessing Files 598
Using SAS Features 599
Using OLE 600
Using Networks 601
Resolving Internal Errors 601
Resolving Operating System and Windows Error Messages 602
Initialization and Termination Error Messages 603

**Appendix 3 △ Using SSL under Windows 605**
What Is SSL? 605
Using SSL 606

SSL for SAS **607**
SSL for Windows **607**
Converting between PEM and DER File Formats **611**
SSL Language Elements **611**

**Appendix 4 △ Graphics Considerations 613**
Using TrueType Fonts with SAS/GRAPH Software **613**

**Appendix 5 △ Default Key Settings for Interactive SAS Sessions 615**
Default Key Definitions under Windows **615**
Keyboard Shortcuts within the SAS Main Window **616**
Keyboard Shortcuts within the Enhanced Editor **618**
Keyboard Shortcuts within Print Preview **621**

**Appendix 6 △ Recommended Reading 623**
Recommended Reading **623**

**Glossary 625**

**Index 639**

# What's New

## Overview

New and enhanced features in Base SAS improve ease of use and SAS performance under the Windows environment:

- SAS now runs under 32– and 64– bit Windows XP and Windows Server 2003 operating environments.
- SAS servers and PCs can use memory-based SAS libraries to process SAS data.
- Network security is enhanced by using Secure Sockets Layer (SSL).
- SAS is now able to sort data using the high-performance sorting tool SyncSort for Windows, by Syncsort, Incorporated, if you have SyncSort installed at your site.
- Sharing files between UNIX and Windows has been simplified.
- You can start SAS with the same destination printer that was specified when you ended your last SAS session.
- During initialization, SAS looks in the Windows user profile directory for a configuration file if you do not specify a configuration file at SAS invocation.
- The Enhanced Editor autosave files are now saved to the Application Data folder.
- Accessibility aids can now access many of the SAS windows and dialog boxes.

*Note:*

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

△

## Basic Operation of SAS

The basic operation of SAS has been enhanced as follows:

- You can store a customized configuration file in the Windows user profile folder. During invocation, SAS searches this folder for a configuration file named either SASV9.CFG or .SASV9.CFG. See “Creating a Customized Configuration File” on page 14.

- Performance can be improved by processing SAS data in memory-based SAS libraries. See “Memory-Based Libraries” on page 199.
- Use Secure Sockes Layer (SSL) for secure networks. See Appendix 3, “Using SSL under Windows,” on page 605 and the *SAS/CONNECT User’s Guide*.
- SAS can now sort data by using SyncSort for Windows if you have Syncsort installed at your site. See “Using Syncsort for Windows” on page 437.
- To use the same destination printer (not the default printer) from one SAS session to the next, use the PRTPERSISTDEFAULT system option when you start SAS. See “Changing the Printer” on page 169 and “PRTPERSISTDEFAULT System Option” on page 536.
- To send an e-mail attachment with records that contain more than 256 characters, the ATTACH email-option in the FILENAME, EMAIL statement now supports the LRECL and RECFM options. See “Using the DATA Step or SCL to Send E-mail” on page 44.
- When you attach the contents of the Results Viewer to an e-mail message, the file is attached according to its file type of either .html or .rtf. See “Sending the Contents of a Window by E-mail” on page 43.
- In 64–bit operating environments, the V6 engine is a read-only engine. See “Using SAS Files from Other Versions with SAS 9.1 for Windows” on page 136.
- New keyboard shortcuts are available for resizing the docking view and displaying property sheets from a window that contains a Tree view or a List view. See Appendix 5, “Default Key Settings for Interactive SAS Sessions,” on page 615.
- If a fatal error occurs during SAS initialization or termination and the MSG dialog box is not available or the SAS log is not open, error messages are written to the SAS console log. See “What If SAS Does Not Start?” on page 12 and “Ending Your SAS Session” on page 29.

## Accessibility

The following features are new:

- Many of the SAS windows and dialog boxes can be read by accessibility aids. See “Accessible Windows and Dialog Boxes” on page 74.
- The ACCESSIBILITY system option enables access to the Customize Tools dialog box Customize tab and some SAS Properties dialog boxes. See “Accessing the Standard or Fully Accessible User Interface” on page 76.
- Text can be enlarged by using the SYSGUIFONT system option. See “Enlarging Fonts” on page 78.
- Accessibility aids might have fewer problems reading menus if you exclude menu icons. See “Improving Access to Menus” on page 79.
- You can use the keyboard to resize the docking view. See “Resizing the Docking View in the Main SAS Window” on page 78.
- When a window (such as SAS Explorer) contains a List view, you can use the Sort Columns dialog box to sort the list by the detail information. See “Sorting Window List Views by a Specific Column” on page 78.
- You can resize the details columns in a List view by using the Column Settings dialog box. See “Resizing the Detail Columns of a List View” on page 79.

## Enhanced Editor

The following features are new:

- To open new or existing files in an Enhanced Editor window, you can select New Program and Open Program from the File menu. See "Opening Files" on page 83.
- The Enhanced Editor now recognizes XML files that have the .xml file extension. See "Associating File Extensions with File Types" on page 102.
- When you open a file in the Enhanced Editor, the tabs can be converted to spaces. See "Using Automatic Indenting and Tabs" on page 92.
- The full path and the filename of the last submitted program or catalog entry is available by using environment variables. See "Obtaining the Filename and Full Path of Submitted Programs or Catalog Entries" on page 95.
- Submit one or more lines of code without submitting your complete program by using the SUBTOP command. See "SUBTOP Command" on page 350.
- Line numbers can be turned on and off by using the NUMS command. See "Scrolling and Line Number Commands" on page 86.

The following features have been enhanced:

- The Enhanced Editor accepts lines of data in a DATALINES statement that are longer than 256 characters into one observaton without using FILENAME statement options. See "Submitting Your Program" on page 93.
- Autosave files are saved to the Application Data folder. See "Saving Files" on page 85.
- The default value for the Enhanced Editor Script appearance option is determined by the Windows regional options. See "Setting Appearance Options" on page 105.

The following feature has been removed:

- The Enhanced Editor no longer prompts you to reload a file on a disk that has been updated.

## Graphical User Interface

The following features are new in the SAS main window:

- The Universal Printing commands are available from the File menu when you specify the UPRINTMENUSWITCH system option. See "Introduction to Printing in SAS within the Windows Environment" on page 166.
- More windowing environment workspace is available by minimizing the docking view. See "Minimizing and Restoring the Docking View" on page 36.
- A maximum of 20 customized help files can be added to the Help menu. See "Adding Help to the Help Menu" on page 64.
- The Print and Copy toolbar buttons are always enabled if they are commands in a customized toolbar. See "Customizing and Saving a Toolbar for Use with a Particular Application or Window" on page 70.
- To print line numbers, page numbers, and to print in color, select the Options button in the Print dialog box. See "Printing Line Numbers, Page Numbers, and in Color" on page 173.
- The About SAS System dialog box includes information about SAS version and the Windows operating environment that you are currently using. See "Getting Help from the Help Menu" on page 73.

# SAS Language Elements

## Commands

The following SAS commands are new:

- ☐ When a SAS window (such as the SAS Explorer window) contains a List view, you can resize the columns by using the DLGCOLUMNSIZE command. See "DLGCOLUMNSIZE Command" on page 331.
- ☐ Sort the columns in a List view by using the DLGCOLUMNSORT command. See "DLGCOLUMNSORT Command" on page 331.
- ☐ Three new docking view commands enable the docking view to be minimized, restored, and resized. See "WDOCKVIEWMINIMIZE Command" on page 358, "WDOCKVIEWRESTORE Command" on page 359, and"WDOCKVIEWRESIZE Command" on page 359.
- ☐ To change the active window to the editor window that was last edited, use the WPGM command. See "WPGM Command" on page 367.

## Functions

The following have specific functionality in the Windows operating environment:

- ☐ The DINFO function, the DOPTNAME function, and the DOPTNUM function return directory information. See "DINFO Function" on page 390, "DOPTNAME Function" on page 392, and "DOPTNUM Function" on page 392.
- ☐ To delete an external file, use the FDELETE function . See "FDELETE Function" on page 393.
- ☐ Verify the existence of an external file by using either the FEXIST function or the FILEEXIST function. See "FEXIST Function" on page 394 and "FILEEXIST Function" on page 394.
- ☐ To assign or clear a fileref, use the FILENAME function. See "FILENAME Function" on page 395.
- ☐ Verify that a filref has been assigned by using the FILEREF function. See "FILEREF Function" on page 396.
- ☐ Obtain information about a file by using the FINFO function, the FOPTNAME function, and the FOPTNUM function. See "FINFO Function" on page 396, "FOPTNAME Function" on page 400, and "FOPTNUM Function" on page 402.
- ☐ To assign or clear a libref, use the LIBNAME function. See "LIBNAME Function" on page 403.

## Statements

The following statements have been enhanced:

- ☐ Share files between the UNIX and Windows operating environments by using the TERMSTR host option. See "FILE Statement" on page 443, and "INFILE Statement" on page 453.
- ☐ ^Z is interpreted as character data and not as an end-of-file marker when you specify the IGNOREDOSEOF host option. See "%INCLUDE Statement" on page 451, "FILENAME Statement" on page 445, "FILE Statement" on page 443, and "INFILE Statement" on page 453.

## System Options

The following system options are new:

- Accessibility features are extended for the Customize Tools dialog box and for some Properties dialog boxes. See "ACCESSIBILITY System Option" on page 482.
- If you create a customized table of contents and index for the SAS Help and Documentation, use the HELPTOC and HELPINDEX system options to specify the file location. See "HELPTOC System Option" on page 511 and "HELPINDEX System Option" on page 507.
- Use memory-based SAS libraries to improve SAS performance by using the MEMBLKSZ and MEMMAXSZ system options. See "MEMBLKSZ System Option" on page 519 and "MEMMAXSZ System Option" on page 522.
- Improve the access to menus by accessibility aids when you use the NOMENUICONS system option. See "Improving Access to Menus" on page 79
- The same destination printer can persist from one SAS session to another by using the PRTPERSISTDEFAULT system option. See "PRTPERSISTDEFAULT System Option" on page 536
- Stop the SLEEP window from displaying when you use the SLEEP function or the WAKEUP function by specifying the NOSLEEPWINDOW system option. See "SLEEPWINDOW System Option" on page 551.
- If you have SyncSort for Windows installed at your site, use these system option to specify sort options:
  - SORTPGM system option specifies to sort by using SyncSort for Windows. See "SORTPGM System Option" on page 555.
  - The SORTANOM system option enables you to specify SyncSort for Windows options. See "SORTANOM System Option" on page 551.
  - Use SORTCUT and SORTCUTP system options to specify the number of observations and bytes above which SyncSort for Windows is used instead of the SAS sort. See "SORTCUT System Option" on page 552 and "SORTCUTP System Option" on page 553.
  - Use the SORTDEV system option to specify a temporary directory for files that are created by SyncSort for Windows. See "SORTDEV System Option" on page 554.
  - The SORTPARM system option enables you to specify SyncSort parameters. See "SORTPARM System Option" on page 555.
- SSLCERTISS, SSLCERTSERIAL, SSLCERTSUBJ, SSLCLIENTAUTH, and SSLCRLCHECK are new system options that support Secure Sockets Layer (SSL) authentication. See "SSLCERTISS System Option" on page 559, "SSLCERTSERIAL System Option" on page 559, "SSLCERTSUBJ System Option" on page 560, "SSLCLIENTAUTH System Option" on page 561, and "SSLCRLCHECK System Option" on page 562.
- To enlarge text for easier readability, use the SYSGUIFONT system option. See "Enlarging Fonts" on page 78.
- The Universal Print commands are available from the File menu. See "UPRINTMENUSWITCH System Option" on page 570.

The following system option has been enhanced:

- The new default value for the SORTSIZE system option is MAX. See "SORTSIZE System Option" on page 556.

The EMAILID and EMAILPW options, which were previously available only in the Windows environment, are now available in all operating environments. See *SAS Language Reference: Dictionary*.

## Macros

The SYSSCPL automatic macro variable returns a value for the Windows XP and Windows Server 2003 operating environments. See "Automatic Macro Variables" on page 581.

P A R T 1

# Running SAS under Windows

*Chapter* **1**. . . . . . . . . . **Getting Started** *3*

*Chapter* **2**. . . . . . . . . . **Interacting with SAS under Windows** *31*

*Chapter* **3**. . . . . . . . . . **Using the SAS Editors** *81*

*Chapter* **4**. . . . . . . . . . **Using SAS Files** *117*

*Chapter* **5**. . . . . . . . . . **Using External Files** *145*

*Chapter* **6**. . . . . . . . . . **Managing SAS Output** *165*

*Chapter* **7**. . . . . . . . . . **Performance Considerations** *195*

CHAPTER

1

# Getting Started

*SAS: Exploiting the Power of Windows* **4**
- *SAS Runs in Enterprise Environments* **4**
- *An Integral Part of Your Windows Environment* **5**
- *Compatible and Maintainable* **6**
  - *Read and Write SAS Data Sets from Previous Releases* **6**
  - *Use the Graphical Interface or the Command Line Interface* **6**

*Starting SAS* **6**
- *Use SAS Interactively or in Batch Mode* **6**
- *Starting from the Start Menu* **6**
- *Starting from Custom Shortcuts or Program Items* **7**
- *Starting from the Run Dialog Box or a Command Prompt* **7**
  - *Specifying the SAS Configuration File* **7**
  - *Using the Run Dialog Box* **7**
  - *Using the SAS Command from the Command Prompt* **7**
- *Starting from a SAS File* **8**
- *Submitting a Batch SAS Job* **8**
  - *Ways to Submit a Batch SAS Job* **8**
  - *The Status Window* **8**
  - *Cancelling a Batch Job* **8**
  - *Windowing Procedures in a Batch Job* **9**
- *Starting the Program Editor When SAS Starts* **9**
- *Determining the Current Folder When SAS Starts* **9**
- *Sample SAS Session* **10**
- *What If SAS Does Not Start?* **12**

*Files Used by SAS* **12**
- *Introduction To Files Used by SAS* **12**
- *SAS Configuration Files* **13**
  - *Purpose of Configuration Files* **13**
  - *The Default Configuration File* **13**
  - *Specifying System Options in a Configuration File* **13**
  - *Creating a Customized Configuration File* **14**
  - *Starting SAS with an Alternate Configuration File* **15**
  - *How SAS Finds and Processes Configuration Files* **15**
  - *Processing Options Specified by Additional CONFIG Options* **18**
- *SAS Autoexec File* **18**
  - *Introduction to the SAS Autoexec File* **18**
  - *The Default Autoexec File* **18**
  - *Locating a Renamed Autoexec File* **18**
  - *Uses for the Autoexec File* **19**
  - *Suppressing the Autoexec File* **19**
- *Profile Catalog* **19**

*Introduction to the Profile Catalog* **19**
*The Default Profile Catalog* **20**
*Changing the Location of the Profile Catalog* **20**
*Deleting the Profile Catalog* **20**
*Work Data Library* **21**
*Introduction to the Work Data Library* **21**
*The Default Work Folder* **21**
*Specifying the Location of the Work Data Library* **21**
*Temporary Subfolders* **21**
*Deleting the Work Folder* **21**
*SAS Registry Files* **21**
*SAS Default Folder Structure* **22**
*Submitting SAS Code* **23**
*Introduction to Submitting SAS Code* **23**
*Submitting Code from the Enhanced Editor or Program Editor* **23**
*Submitting Code from the SAS NOTEPAD Text Editor* **23**
*Submitting Code from the Clipboard* **23**
*Submitting Code by Dragging and Dropping* **24**
*Introduction to Submitting Code by Dragging and Dropping* **24**
*Dragging Text from Other Windows* **24**
*Dragging Files in an Interactive Session* **24**
*Dragging Files to Start a Batch Session* **24**
*Submitting Code Stored in Registered SAS File Types* **24**
*Interrupting Your SAS Session* **25**
*Running Windows or MS-DOS Commands from within SAS* **25**
*Overview of Running Windows or MS-DOS Commands from within SAS* **25**
*Running Windows Commands Using the X Statement or the X Command* **26**
*Using a DATA Step to Issue Conditional Operating System Commands Conditionally* **26**
*XWAIT System Option* **27**
*XSYNC System Option* **27**
*Comparison of the XWAIT and XSYNC System Options* **28**
*Terminating a SAS Process* **28**
*Ending Your SAS Session* **29**

## SAS: Exploiting the Power of Windows

### SAS Runs in Enterprise Environments

The Windows enterprise environment provides a flexible, easy-to-use working environment by which you can integrate SAS into your enterprise solutions. The following Windows environments are supported in SAS 9.1:

**Table 1.1** Supported Windows Environments for SAS 9.1

| Processor Bus Size | Operating System | Operating Environment |
|---|---|---|
| 32-bit | Windows NT | Workstation<br>Server<br>Server Enterprise Edition<br>Terminal Server Edition |
| | Windows 2000 | Professional Edition<br>Server<br>Advanced Server<br>Datacenter Server |
| | Windows XP | Professional Edition |
| | Windows Server 2003 | Standard Server<br>Datacenter Server<br>Enterprise Server<br>Web Server |
| 64-bit | Windows XP | Professional Edition |
| | Windows Server 2003 | Enterprise Server<br>Datacenter Server |

## An Integral Part of Your Windows Environment

SAS under Windows is designed to let you complete your data– and computation-intensive tasks while integrating with the Windows applications that are already in place on your desktop and within your enterprise. SAS supports information sharing through the most powerful tools and techniques that Windows has to offer, including

- OLE
- Dynamic Data Exchange (DDE)
- Open Database Connectivity (ODBC)
- e-mail system

- □ Lotus Notes
- □ pipes and named pipes
- □ the Windows clipboard.

## Compatible and Maintainable

### Read and Write SAS Data Sets from Previous Releases

SAS can read and write SAS data sets that were created by earlier releases of SAS.
However, in order to bridge the upgrades in the SAS catalog architecture and differences in the operating environment structure, catalogs must be converted from earlier formats (such as Release 6.12 under Windows) to SAS 9.1 format using the transport procedures CPORT and CIMPORT.

### Use the Graphical Interface or the Command Line Interface

You can still use the command line as you did in previous releases. However, you can also use the graphical user interface (GUI) to issue commands. Most existing SAS commands and windows are available through the GUI. In some cases, you select operations through dialog boxes and various other GUI controls.

## Starting SAS

### Use SAS Interactively or in Batch Mode

When running SAS under Windows, you can start an interactive session to submit programs and view the resulting output, or you can execute batch SAS jobs, whose output you can view later.

By default, invoking SAS begins an interactive SAS session. If you have a SAS program that you want to submit as a batch job, specify the SYSIN system option with the name of the SAS program file when you invoke SAS.

When you start SAS in an interactive session for the first time, you are asked if you want to learn some basic tasks by taking the Getting Started Tutorial. To start the tutorial, click **`Start Tutorial`**. If you do not want to be prompted to take the tutorial, select **`Don't show this dialog again`**. You can start the tutorial at any time by selecting

Help ► Getting Started with SAS Software

### Starting from the Start Menu

To start SAS from the Windows Start Menu:

1 Click **`Start`**.

2 Select **`Programs`**.

3 Select **`SAS`**.

4 Select **`SAS 9.1`** .

## Starting from Custom Shortcuts or Program Items

During installation, the Setup program automatically creates a program item in the Start menu that you can use to start SAS. However, you can create multiple SAS items within a folder to represent several differently configured SAS sessions. Also, if you want SAS to start every time you start Windows, you can place a program item or shortcut in the Startup folder. For information about creating shortcuts, see your Windows documentation.

After you have created a shortcut to SAS, you can append system options to the SAS command. To do this,

1 Open the SAS Properties window and click the **`Shortcut`** tab.

2 In the **`Target`** field, append the system options to the SAS command. Remember that double quotation marks are required around path names. For example,

```
"c:\program files\sas\sas 9.1\sas.exe" -config "c:\mydir\sasv9.cfg"
```

Display 1.1 on page 10 shows an example of a SAS item from the Start menu under Windows 2000 Professional.

## Starting from the Run Dialog Box or a Command Prompt

### Specifying the SAS Configuration File

If you start SAS by using a command line (either from the Run dialog box or the Command Prompt window), you might want to specify the SAS configuration file location through the CONFIG system option. Even if you use the default configuration file SASV9.CFG, specify the file to ensure that SAS uses the configuration file that you want. For more information about how SAS searches for the configuration file, see "How SAS Finds and Processes Configuration Files" on page 15.

When the WORK and SASUSER system options are set, the Work and Sasuser data libraries reside in the specified paths regardless of the path from which you invoke SAS. For more information about the Sasuser data library, see "Profile Catalog" on page 19. For more information about the Work data library, see "Work Data Library" on page 21

### Using the Run Dialog Box

To start an interactive session by using the **`Run`** dialog box

1 Select

Start ▶ Run

2 In the **`Open`** field, type the path and the exact name of the program file, including the extension and options.

3 Click **`OK`**.

For example, if SAS is installed in the default folder c:\Program Files\SAS\SAS 9.1, you type **`c:\program files\sas\sas 9.1\sas.exe`**, and the options that you want to specify.

### Using the SAS Command from the Command Prompt

You can start either an interactive SAS session or a batch SAS job by typing **`SAS`** at the command prompt. For example, the following command starts an interactive session, specifies the page size and line size, and indicates the location of the SAS configuration file:

```
c:\program files\sas\sas 9.1\sas.exe -ls 80 -ps 60
   -config c:\program files\sas\sas 9.1\sasv9.cfg
```

This command starts a batch SAS job in a similar manner:

```
c:\program files\sas\sas 9.1\sas.exe
   -sysin c:\mysas\programs\prog1.sas
   -config c:\program files\sas\sas 9.1\sasv9.cfg
```

*Note:* These examples are displayed on multiple lines because of space limitations. When you enter a command from the command prompt, the command must be on one line. △

## Starting from a SAS File

There are two ways to start SAS from a SAS program file in Windows Explorer.

- Double-click on a SAS program file
- Right-click on a SAS program file and select the appropriate action.

## Submitting a Batch SAS Job

### Ways to Submit a Batch SAS Job

There are several ways to submit a batch SAS job:

- Specify the SYSIN system option in the SAS command (issued from the command prompt or from the Run dialog box) to specify the SAS program to submit.
- Right-click a file that has a .sas or .sasv7bpgm file extension. From the pop-up menu, select **Batch Submit**.
- Use the Windows Explorer and drag your SAS program file icon (for the file that contains the SAS code) to the SAS.EXE file icon or shortcut.

### The Status Window

When you run SAS in batch mode, SAS displays a Status window for the SAS job that you submit. This window tells you the name of the SAS job that is running and where your log and procedure output files are written. This window remains open until the SAS job is complete.

If you do not want to see the Status window while your batch SAS job is running, invoke SAS with the ICON system option so that the Status window becomes an icon when your job is running. You can also minimize the Status window by clicking the **Icon** button when the window appears. The icon shows the busy cursor (usually an hourglass) while the SAS job is running. The icon disappears when the job is complete.

### Cancelling a Batch Job

You can cancel a batch job by either

- pressing CTRL+BREAK
- clicking **Cancel** in the Status window.

## Windowing Procedures in a Batch Job

You can run windowing procedures, such as those that are associated with SAS/GRAPH, SAS/INSIGHT, and SAS/ACCESS software, in a batch SAS job. When SAS reaches a point in your program where interaction is required, it opens the main SAS window.

## Starting the Program Editor When SAS Starts

The Enhanced Editor is the default editor that starts when you start SAS. If you prefer to use the Program Editor, use one of the following methods to start the Program Editor when SAS starts:

- Start SAS with the NOENHANCEDEDITOR system option:

  ```
  sas.exe -noenhancededitor
  ```

- Disable the Enhanced Editor in the Edit tab of the Preferences dialog box.

For additional information, see “Switching from the Enhanced Editor to the Program Editor” on page 108, “Edit Preferences” on page 59, and “ENHANCEDEDITOR System Option” on page 501.

## Determining the Current Folder When SAS Starts

SAS uses the current folder as the default location to read and write SAS files when you do not explicitly specify a different path.

You can specify a pathname to use for the current folder by using the SASINITIALFOLDER system option when you start SAS or use the following rules to determine the current folder:

1. If you use a program item or shortcut to start SAS and a path is specified in the Windows Properties **`Shortcut`** tab (**`Start in`** field), SAS uses that path as the current folder.
2. If you use a command to start SAS by using either the Run dialog box or a command line and the command contains a path to SAS.EXE, the current folder is the path that you specify as part of the SAS command, regardless of where Windows actually finds the SAS.EXE file.
3. If you use a command to start SAS and you do not specify a path as part of the SAS command, then the current folder is the path from which you issued the command.

If Windows cannot find the SAS.EXE file in the specified folder, the folder that is specified in the SAS command still becomes the current folder and Windows searches for the SAS.EXE file by using the Windows PATH environment variable.

For example, if you specify the following command, C:\MYSAS is the current folder, regardless of whether the SAS.EXE file is actually in that folder:

```
c:\mysas\sas.exe -config c:\mysas\sasv9.cfg
```

Under Windows NT, when you start SAS from a file as explained in “Starting from a SAS File” on page 8, the default current folder is the path to the operating system with \system32 appended to it. An example of a default current folder under Windows NT is **`C:\winnt\system32`**. Other operating systems default to the folder where the file is stored. You can specify the current folder when you start SAS.

For more information, see “Changing the SAS Current Folder” on page 37 and “SASINITIALFOLDER System Option” on page 546.

*Note:* Do not confuse the current folder with the Work data library. For more information about the Work data library, see "Work Data Library" on page 21. △

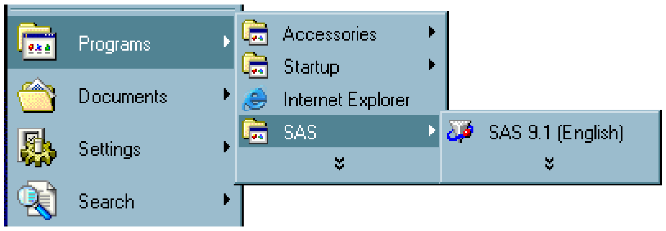
## Sample SAS Session

This section illustrates

- □ invoking SAS from the Start menu
- □ submitting a sample SAS program
- □ examining the program output
- □ ending the SAS session.

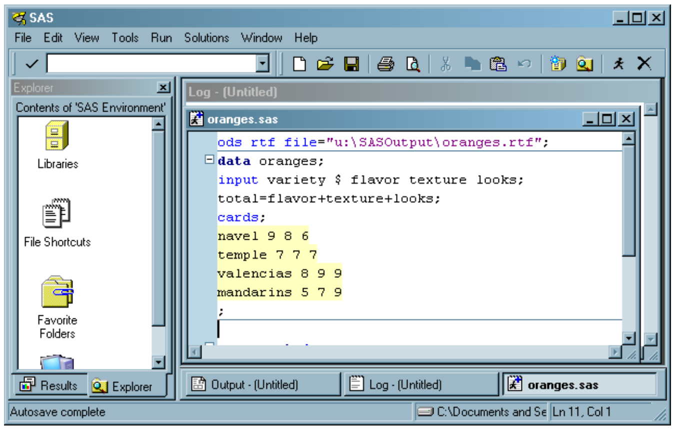
The following display shows a possible configuration of the SAS program item in the Start menu. Select **SAS 9.1** to start SAS.

**Display 1.1** Starting SAS



The following display shows the Enhanced Editor and Log windows with a sample SAS program that is ready to be submitted. This program creates a SAS data set called Oranges, which contains the results of a taste test on four varieties of oranges. The program sorts the data set by the total test score and prints the data set.

**Display 1.2** Submitting the Sample SAS Program



```
ods rtf file="u:\SASOutput\oranges.rtf";
data oranges;
input variety $ flavor texture looks;
total=flavor+texture+looks;
cards;
navel 9 8 6
temple 7 7 7
valencias 8 9 9
mandarins 5 7 9
;
```

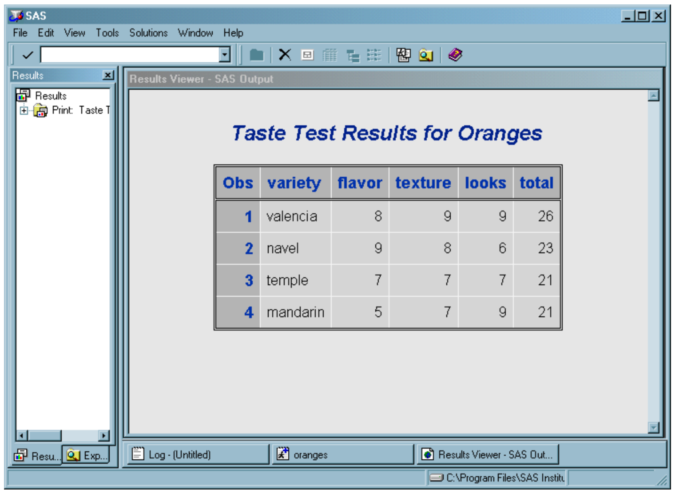
The following SAS code appears in the Enhanced Editor window:

```
ods rtf file="c:\em\oranges.rtf";
data oranges;
   input variety $ flavor texture looks;
   total=flavor+texture+looks;
   cards;
navel 9 8 6
```

```
temple 7 7 7
valencia 8 9 9
mandarin 5 7 8
;
proc sort data=oranges;
   by descending total;
run;
proc print data=oranges;
   title 'Taste Test Results for Oranges';
run;
ods rtf close;
```

After you submit the program, the output appears in the Results Viewer window as follows:

**Display 1.3** Looking at the Program Output



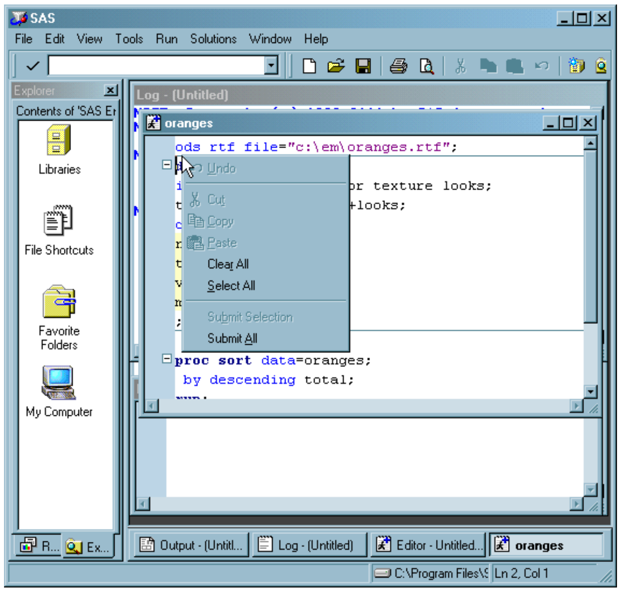
SAS
File Edit View Tools Solutions Window Help
Results
Results
Print: Taste T
Results Viewer - SAS Output
Resu... Exp...
Log - (Untitled)
oranges
Results Viewer - SAS Out...
C:\Program Files\SAS Institu

Taste Test Results for Oranges

| Obs | variety | flavor | texture | looks | total |
|---|---|---|---|---|---|
| 1 | valencia | 8 | 9 | 9 | 26 |
| 2 | navel | 9 | 8 | 6 | 23 |
| 3 | temple | 7 | 7 | 7 | 21 |
| 4 | mandarin | 5 | 7 | 9 | 21 |

The items in the SAS menu bar at the top of the main SAS window change, depending on which window is active within the SAS session. In addition, you can access window-specific pop-up menus, which offer the same menu choices. The pop-up menu in the following display was generated by clicking the right mouse button with the cursor in an Enhanced Editor window.

**Display 1.4** Pop-up Menu in the Enhanced Editor Window



```
ods rtf file="c:\em\oranges.rtf";
                          or texture looks;
                          +looks;

proc sort data=oranges;
 by descending total;
```

When you are ready to end your SAS session, double-click the SAS control menu (the small icon in the upper-left corner of the main SAS window) or click the **X** (in the upper-right corner) and click **OK** when the dialog box verifies your request.

*Note:* If you have disabled the **Confirm Exit of SAS** option in the Preferences dialog box, your SAS session ends without asking if you are sure you want to end the session. For more information about how to customize your SAS session, see "Setting Session Preferences" on page 57. △

## What If SAS Does Not Start?

If SAS does not start, the SAS log may contain error messages that explain the error. Any error message that SAS issues before the SAS log is initialized is written to the MSG window. Under Windows NT, the SAS console log is located in the c:\winnt\Profiles\*username*\Application Data folder. In all other Windows operating environments, the SAS console log is located in the c:\Document and Settings*userid*\Application Data folder. You can obtain the filename for the SAS Console Log from the Application Event Log. To open the application Event Log, submit **eventvwr** from the Run dialog box and click on **Application**.

# Files Used by SAS

## Introduction To Files Used by SAS

SAS uses many files while it is running; however, some of these files are especially important from a user's perspective. These files include the

- SAS configuration files (SASV*x*.CFG by default, where *x* is the release number)
- SAS autoexec file (AUTOEXEC.SAS by default)
- user profile catalog (Profile.sas7bcat)

- user printer profile catalog (Profile2.sas7bcat)
- Work data library ("SAS Temporary Files" folder in your system's designated TEMP area)
- SAS Registry Files.

## SAS Configuration Files

### Purpose of Configuration Files

The SAS *configuration file* enables you to specify SAS system options that are used to establish your SAS session. These system options indicate, among other things, the location of your SAS Help and Documentation files as well as the location of message files and the pathnames to SAS executable files. The SAS configuration file is particularly important because it specifies the folders that are searched for the various components of SAS products. You must have at least one configuration file in order for SAS to initialize; you can have multiple configuration files that are all processed while your SAS session begins. For a list of system options that you can use in your SAS configuration file, see "Summary of System Options for Windows" on page 470. For more information about system options, see Chapter 22, "System Options under Windows," on page 465 and "SAS System Options" in *SAS Language Reference: Dictionary*.

### The Default Configuration File

In previous releases of SAS, the default configuration file was stored in the !SASROOT folder. The !SASROOT folder is the folder in which you install SAS. Starting with SAS 9, SAS creates two default configuration files during installation. Both configuration files are named SASV9.CFG.

SAS stores one of these files in the !SASROOT folder and the other in the `!sasroot\nls\`*language-code* folder. The *language-code* is a two-letter language code for the SAS default language.

The SASV9.CFG file that is located in the !SASROOT folder contains a CONFIG system option that specifies the location of the configuration file for the SAS default language. The default system options that are used to start SAS are specified in the `!sasroot\nls\`*language-code*`\SASV9.CFG` file. For example, if SAS is installed in the default folder and the default language is English, the SASV9.CFG file in the !SASROOT folder contains
`-config "c:\program files\sas\sas 9.1\nls\en\sasv9.cfg"`.

SAS requires a configuration file, so you must use a SAS configuration file regardless of whether you are using interactive or batch mode.

SAS uses the default configuration file if you start SAS by double-clicking a registered SAS file type, such as .sas.

For more information about the !SASROOT folder, see "SAS Default Folder Structure" on page 22.

### Specifying System Options in a Configuration File

Any system option may be specified when you start SAS. It is often more convenient to place frequently used system options in a configuration file. The syntax for specifying system options in a SAS configuration file is discussed in "Syntax for System Options in the SAS Invocation or SAS Configuration File" on page 468.

You can edit the default configuration file to add to or change the system option settings, or you can create your own configuration file. "Creating a Customized Configuration File" on page 14 discusses how to modify your configuration file.

Your configuration file is divided into two sections. The first section specifies system options that are not updated by the SAS Setup application. The second section is used by the setup application for updating information about where SAS software is installed. The sections are divided by the following warning:

```
WARNING:   INSTALL Application edits below this line. User
           options should be added above this box comment.
           INSTALL Application maintains and modifies the
           following options; -SASAUTO, -SASHELP, -SASMSG
           -PATH, and -MAPS. It also maintains and modifies
           the following CONFIG variables with the -SET option;
           INSTALL, USAGE, LIBRARY, SAMPSIO, SAMPSRC, SASCBT,
           and SASEXT01--SASEXT50. It preserves all lines above
           the line containing 'DO NOT EDIT BELOW THIS LINE'.
```

The setup application deletes all data below this warning but does not affect the options that are specified above it. The SET system option defines the following SAS environment variables: SASROOT, SASEXT0, SASFOLDER, MYSASFILES, SASCFG, SASAUTOS, SAMPSIO, SAMPSRC, EISIMAGE, and INSTALL. The setup application appends the following system options below this warning: SASUSER, WORK, HELPLOC, DMSEXP, APPLETLOC, TEXTURELOC, RESOURCESLOC, JREOPTIONS, SASSCRIPT, SASHELP, MSG, and PATH.

***CAUTION:***

**To avoid corrupting your configuration file, use a SAS text editor or an ASCII text editor to edit your configuration file.** The text editor that you choose to edit the configuration file is important to preserve some of the special character formatting in the file. The recommended method is to edit your configuration file by using a SAS text editor (such as the Enhanced Editor) and save it by using the Save As dialog box. If you do not use a SAS text editor, be sure to use another ASCII text editor (such as Windows Notepad). Do not use a specialized editor such as the WordPad application or Microsoft Word. Using such editors can insert carriage control characters into your configuration file or corrupt the characters that are there. △

## Creating a Customized Configuration File

When you install SAS, a SASV9.CFG file is created in the **`!SASROOT\nls\`***language-code* folder. The *language-code* is a two-letter language code for the SAS default language. You can specify your own file to act as the configuration file, overriding the default file, SASV9.CFG.

The filename that you choose must follow the file naming conventions for the Windows operating environment. The file extension must be .CFG.

When you use your own configuration file instead of the default configuration file, you must add several required system options. For example, you must either use the SET system option to define the environment variable, !SASROOT, or define SASROOT as a Windows environment variable.

To ensure that all required system options are defined in your configuration file, copy the default file (**`!SASROOT\nls\`***language-code***`\SASV9.CFG`**) and modify the copy instead of creating your own file from scratch.

You can create a customized configuration file and name the file either SASV9.CFG or .SASV9.CFG in your Windows user profile folder. During SAS invocation, SAS looks for either of these files in the Windows user profile folder if the -CONFIG options is not specified. Under Windows NT, the Windows user profile folder is c:\winnt\Profiles\*username*\Personal. Under all other Windows operating environments, the user profile folder is c:\Documents and Settings*username*\My Documents.

## Starting SAS with an Alternate Configuration File

When you use a file that is located in a different folder or under a different name as your default configuration file, you must tell SAS where to find the configuration file. Use the CONFIG system option to specify the location of this configuration file. For example, the **Target** field of the SAS Properties dialog box might contain

```
''c:\program files\sas\sas 9.1\sas.exe -config c:\mysas\mysasconfig.CFG''
```

If SAS cannot find the configuration file, an error message is displayed, and SAS does not initialize.

For more information about the CONFIG system option, see "Processing Options Specified by Additional CONFIG Options" on page 18 and "CONFIG System Option" on page 494.

## How SAS Finds and Processes Configuration Files

When you invoke SAS, SAS automatically searches several locations for configuration options that can affect your SAS session. SAS looks in the following areas and processes them in this order:

SAS_SYS_CONFIG operation system environment variable
: This environment variable, if defined, must resolve to a valid configuration file. In a multi-user Windows system, this environment variable would most likely be defined as a system environment variable (instead of as a user environment variable) so that it is processed for all users on that system. Use the SAS_USER_CONFIG user environment variable to specify a user-specific configuration file.

files specified by CONFIG system options
: In the SAS invocation command, you can specify one or more -CONFIG options with the names of the configuration files that you want to use. You must include a separate -CONFIG option for each file that you want to specify.

SASV*x*.CFG in the folder where SAS.EXE resides
: SAS looks for a file that is named SASV*x*.CFG (*x* is the SAS version number) in the folder that contains the SAS.EXE file *only if* you do not specify a -CONFIG option at SAS invocation. This configuration file contains only a CONFIG system option that specifies the configuration file for the SAS default language.

.SASV*x*.CFG in the Windows user profile folder
: SAS looks for a file that is named .SASV*x*.CFG (*x* is the SAS version number) in the Windows user profile folder only if you do not specify a -CONFIG option at SAS invocation. Under Windows NT, the Windows user profile folder is **c:\winnt\Profiles\*username*\Personal**. Under all other Windows operating environments, the Windows user profile is **c:\Documents and Settings*username*\My Documents**.

SASV*x*.CFG in the Windows user profile folder
: SAS looks for a file that is named SASV*x*.CFG (*x* is the SAS version number) in the Windows user profile folder only if you do not specify a -CONFIG option at SAS invocation. Under Windows NT, the Windows user profile folder is c:\winnt\Profiles\*username*\Personal. Under all other Windows operating environments, the Windows user profile is c:\Documents and Settings*username*\My Documents.

SASV*x*.CFG in the current folder
: SAS looks for a file that is named SASV*x*.CFG (*x* is the SAS version number) in the current folder *only if* you do not specify a -CONFIG option at SAS invocation.

SAS_USER_CONFIG operating system environment variable
: This environment variable, if defined, must be a path to a valid SAS configuration file. In a multi-user Windows system, this environment variable would likely be defined as a user environment variable (instead of as a system environment variable) so that it is processed only for the current user on that system. Use the SAS_SYS_CONFIG system environment variable to specify a system-wide configuration file.

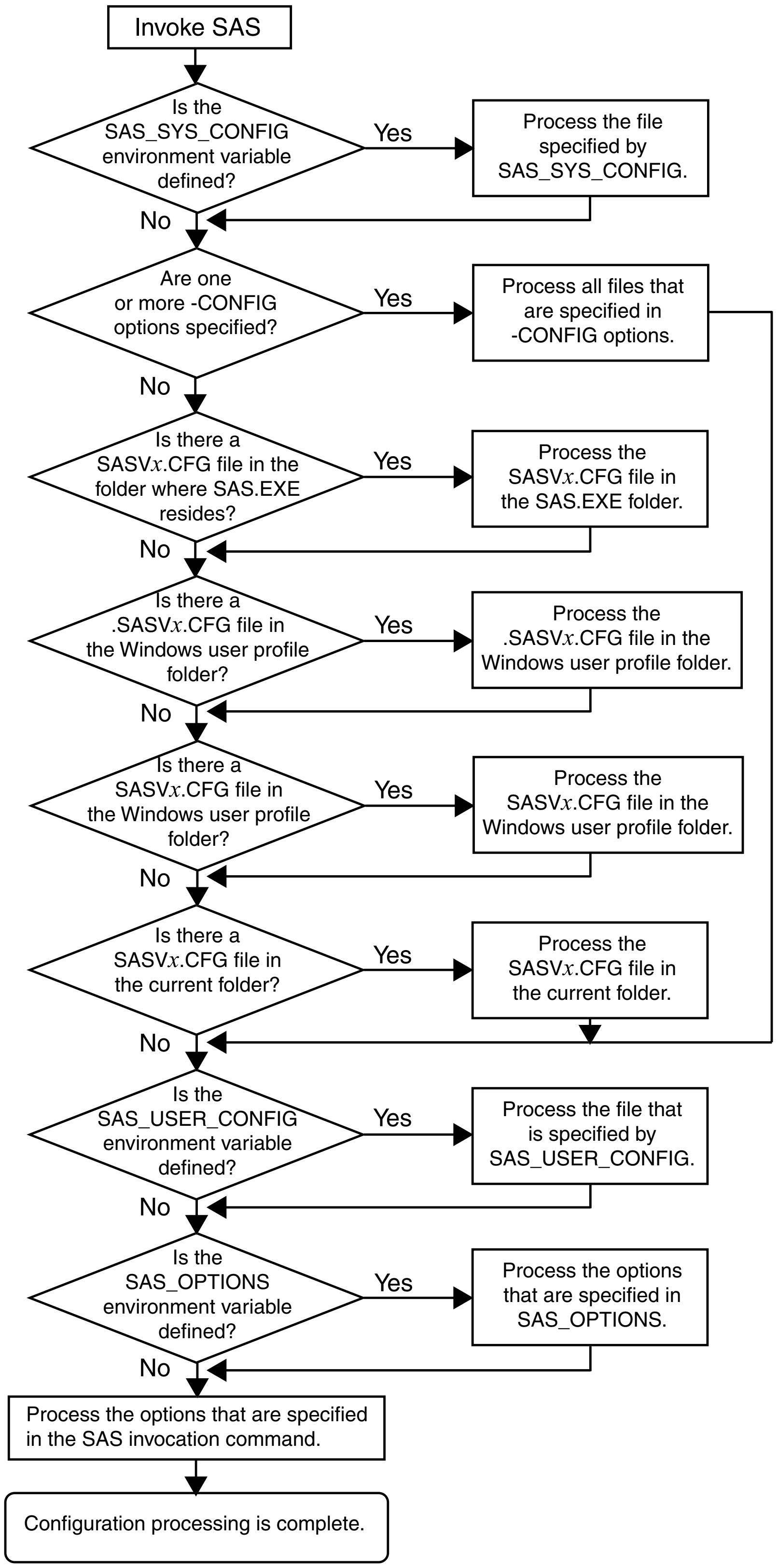
SAS_OPTIONS operating system environment variable
: This environment variable, if defined, contains a string of option specifications for any other SAS system options that you want to process each time that you invoke SAS. For example, this environment variable might contain **`-obs 2m -sasinitialfolder c:\myfolder -linesize max`**.

SAS invocation command line
: You can specify additional system options in the command that you use to invoke SAS. These system options always override option values that are set within any of the configuration files.

If you start SAS from the Windows Start menu, the last configuration file to be processed is the one that is specified in the Start menu shortcut for SAS. The options specified in this configuration file override any options that have previously been processed. By default, the Start menu shortcut specifies
**`-config !SASROOT\nls\`***language-code***`\SASV`***x***`.CFG`**

SAS uses the default configuration file if you start SAS by double-clicking on a registered SAS file type, such as .sas or .sas7bpgm.

**Figure 1.1** Order of Processing for SAS Configuration Files


Invoke SAS
Is the SAS_SYS_CONFIG environment variable defined?
Yes
Process the file specified by SAS_SYS_CONFIG.
No
Are one or more -CONFIG options specified?
Yes
Process all files that are specified in -CONFIG options.
No
Is there a SASV*x*.CFG file in the folder where SAS.EXE resides?
Yes
Process the SASV*x*.CFG file in the SAS.EXE folder.
No
Is there a .SASV*x*.CFG file in the Windows user profile folder?
Yes
Process the .SASV*x*.CFG file in the Windows user profile folder.
No
Is there a SASV*x*.CFG file in the Windows user profile folder?
Yes
Process the SASV*x*.CFG file in the Windows user profile folder.
No
Is there a SASV*x*.CFG file in the current folder?
Yes
Process the SASV*x*.CFG file in the current folder.
No
Is the SAS_USER_CONFIG environment variable defined?
Yes
Process the file that is specified by SAS_USER_CONFIG.
No
Is the SAS_OPTIONS environment variable defined?
Yes
Process the options that are specified in SAS_OPTIONS.
No
Process the options that are specified in the SAS invocation command.
Configuration processing is complete.

### Processing Options Specified by Additional CONFIG Options

You can also specify additional –CONFIG options within any configuration file. When SAS encounters a –CONFIG option, SAS immediately processes the options in that named file and then returns to process the remainder of the current file. SAS options that are encountered later in the processing always override those options that are specified earlier. For example, if you specify -ICON in the file that is specified by the SAS_SYS_CONFIG environment variable, and then –NOICON in the file that is specified by the SAS_USER_CONFIG environment variable, the –NOICON option is used. Since the options that you specify in the SAS invocation command are always processed last, those option values will always override the option values that are specified in configuration files. Figure 1.1 on page 17 illustrates the flow of the SAS configuration file processing.

For more information about the CONFIG system option, see “CONFIG System Option” on page 494.

## SAS Autoexec File

### Introduction to the SAS Autoexec File

The SAS *autoexec file* contains SAS statements that are executed immediately after SAS initializes and before any user input is accepted. These SAS statements can be used to invoke SAS programs automatically, set up certain variables for use during your SAS session, or set system options.

Use a SAS text editor to create your autoexec file. The text editor that you choose to create the autoexec file is important. The recommended method is to create the file by using a SAS text editor (such as the Enhanced Editor window) and save it using the Save As dialog box. If you do not use the SAS text editor, be sure to use another ASCII text editor (such as Windows Notepad). Do not use a specialized editor such as the WordPad application or Microsoft Word. Using such an editor can insert special carriage control characters into your autoexec file that SAS cannot interpret when it tries to execute the statements in the file.

### The Default Autoexec File

Unlike the configuration file, a SAS autoexec file is not required in order to run SAS. But, if you do have an autoexec file, the default name is AUTOEXEC.SAS. SAS uses the following search order to find the AUTOEXEC.SAS file:

1. Search the current folder.
2. Search the paths that are specified by the Windows PATH environment variable.
3. Search the root folder of the current drive.
4. Search the folder that contains the SAS.EXE file.

If an AUTOEXEC.SAS file is not present in one of these folders and if you did not specify the -AUTOEXEC option on the command line or within any of your configuration files, then SAS assumes that there is no autoexec file to process. For more information, see “AUTOEXEC System Option” on page 485.

### Locating a Renamed Autoexec File

You do not have to name your autoexec file AUTOEXEC.SAS, but if you name it something else, you must use the AUTOEXEC system option to tell SAS where to find the autoexec file. For example, you can specify the following option after the path specification for the SAS.EXE file in the **`Target`** field of the SAS Windows shortcut:

```
-autoexec c:\mysasfiles\init.sas
```

If the specified autoexec file is not found, an error message is displayed, and SAS terminates.

### Uses for the Autoexec File

The autoexec file is a convenient way to execute a standard set of SAS program statements each time that you invoke SAS. You may want to include OPTIONS, LIBNAME, or FILENAME statements, or any other SAS statements and system options that you want the system to execute each time you invoke a SAS session. For example, if you want to specify a script file for SAS/CONNECT software, you can place the following statement in the AUTOEXEC.SAS file:

```
filename rlink 'c:\program files\sas\sas 9.1\connect\saslink\startSession.scr';
```

Or you can use the OPTIONS statement to set the page size and line size for your SAS output and use several FILENAME statements to set up filerefs for commonly accessed network drives, as in the following example:

```
options linesize=80 pagesize=60;
filename saledata 'f:\qtr1';
filename custdata 'l:\newcust';
filename invoice 'o:\billing';
```

Other system options, in addition to the AUTOEXEC system option, provide ways to send SAS information as it is initializing. These options are listed below in the order in which they are processed:

1. CONFIG (at SAS invocation only)
2. AUTOEXEC
3. INITCMD
4. INITSTMT
5. SYSIN

For more information about the CONFIG, AUTOEXEC, INITSTMT, and SYSIN system options, see "SAS System Options under Windows" on page 467. For more information about the INITCMD system option, see *SAS Language Reference: Dictionary*.

### Suppressing the Autoexec File

If you have an AUTOEXEC.SAS file in your current folder but do not want SAS to use it, specify the NOAUTOEXEC option in the SAS command, as in the following example:

```
c:\program files\sas\sas 9.1\sas.exe -noautoexec
```

## Profile Catalog

### Introduction to the Profile Catalog

Each time that you invoke a SAS session, SAS checks the Sasuser data library for your user profile catalog (named Sasuser.Profile), which defines the start-up profile for your SAS session, including key definitions, display configurations, and other personal customizations. If you invoke SAS without accessing an existing profile catalog, SAS creates one with the default key definitions and window configuration.

If Sasuser.Profile does not exist and Sashelp.Profile (in the Sashelp data library) does exist, SAS copies Sashelp.Profile to Sasuser.Profile before invoking a SAS session.

The profile catalog is not re-created if it already exists. Any customizations (such as key definitions or color modifications) that are defined during subsequent sessions are stored in your profile catalog in the specified folder.

## The Default Profile Catalog

The default configuration file for SAS specifies the SASUSER system option as follows:

**Table 1.2** Default SASUSER location for the Windows Operating Environments

| Windows NT | Windows 2000, Windows XP, and Windows Server 2003 |
|---|---|
| `-sasuser "c:\winnt\Profiles\`*`username`*`\Personal\My SAS Files\9.1` | `-sasuser "c:\Documents and Settings\`*`username`*`\My Documents\My SAS Files\9.1"` |

## Changing the Location of the Profile Catalog

Use the SASUSER system option to specify a location for the profile catalog other than the default (which is a folder named \My SAS Files\9.1). This option is useful if you want to customize your SAS sessions when sharing a machine with other users or if users are accessing SAS from a network.

The SASUSER system option takes the following form:

-SASUSER ("*library-specification*")

Parentheses () are used to specify multiple library-specifications, and quotes (") are used when special characters and spaces are used in the library-specification.

If *library-specification* (which specifies a valid Windows pathname) does not exist, SAS attempts to create it. For example, if you specify the following option, a profile catalog is created in a folder named MYUSER that resides in the root folder of the C: drive:

```
-sasuser "c:\myuser"
```

For more information, see “SASUSER System Option” on page 547.

## Deleting the Profile Catalog

When you delete your profile catalog, you lose the key definitions, window configurations, and option settings that you might have defined, as well as any other entries that you saved to your profile catalog. In addition, any text that you stored in NOTEPAD windows is erased. For this reason, it is a good idea to make a backup copy of your profile catalog after making significant modifications to your SAS session settings.

## Work Data Library

### Introduction to the Work Data Library

SAS requires some temporary disk space during a SAS session. This temporary disk space is called the *Work data library*. By default, SAS stores SAS files with one-level names in the Work data library, and these files are deleted when you end your SAS session. You can change the Work data library in which SAS files that have one-level names are stored. For more information, see "Using the User Libref" on page 134.

### The Default Work Folder

The default configuration file for SAS specifies the WORK system option to be a folder in your system's designated temporary area (as defined by the TEMP environment variable). For example: **`!TEMP\SAS Temporary Files`**.

To determine TEMP environment variable, refer to the System Properties dialog box that you access from the Control Panel.

For more information about using the Work data library and overriding the default location, see "Using the Work Data Library" on page 133.

### Specifying the Location of the Work Data Library

The WORK system option controls the location of the Work data library. You can specify the WORK option in your SAS configuration file or when you invoke SAS. Usually, you use the WORK option that is specified in the default configuration file.

### Temporary Subfolders

Because you can run multiple SAS sessions at one time, SAS creates temporary subfolders under the folder that you specify with the WORK option. These temporary subfolders are created in the unique form _TD*nnnnnnnnnn*, where TD means temporary folder and *nnnnnnnnnn* is the process ID for each SAS session. These subfolders enable multiple SAS sessions to be invoked, each using the same configuration file, and they prevent the Work folder from being shared. SAS creates any temporary files that are required within each temporary folder. As with all temporary files that are created in the Work data library during a SAS session, these temporary folders are deleted when you end the SAS session. If SAS terminates abnormally, you may need to delete the temporary files.

### Deleting the Work Folder

If SAS terminates abnormally, determine if the Work library was deleted. If not, remove it by using Windows commands.

*Note:* Do not attempt to delete the Work folder while SAS is running. △

You can verify the location of the current Work folder by opening the Libraries folder in the SAS Explorer window. Click the right mouse button on the Work folder and select **`Properties`** from the pop-up menu.

## SAS Registry Files

The SAS registry files are used to store information about the SAS session applications. The registry entries can be customized by using the SAS registry editor or by importing the registry files. To invoke the SAS registry editor, select

Solutions ► Accessories ► Registry Editor

***CAUTION:***

**Incorrect registry entries can corrupt your SAS registry.** Registry customization is generally performed by more advanced users who have experience and knowledge of SAS and their operating environment. △

## SAS Default Folder Structure

The SAS Setup program creates a number of subfolders during the installation process. Understanding the organization of the SAS folders can help you to use SAS more efficiently.

The root folder of SAS is the folder in which you install SAS. Within SAS, this folder has the logical name !SASROOT. If you use the default provided by SAS, this folder is **`c:\Program Files\SAS\SAS 9.1`** but this is not required. (The examples in this document assume the !SASROOT folder is called **`c:\Program Files\SAS\SAS 9.1`**.)

SAS creates a folder for shared components, such as the Enhanced Editor and images, that are used by other SAS products. For SAS Version 8, the default path for shared components is **`c:\Program Files\SAS Institute\Shared Files`**. For SAS 9 and 9.1, the default path for shared files is **`c:\Program Files\SAS\Shared Files`**.

One important subfolder of the !SASROOT folder is the CORE subfolder. The CORE subfolder contains many subfolders, three of which are described here:

!SASROOT\CORE\RESOURCE
: contains SAS resources such as fonts and images.

!SASROOT\CORE\SAMPLE
: contains the SAS sample programs.

!SASROOT\CORE\SASINST
: contains the installation process software.

For each SAS product that is installed, the following subfolders might be created (not all products contain all of these folders):

!SASROOT\*product*\SASEXE
: contains the SAS executable files.

!SASROOT\*product*\SASHELP
: contains many specialized catalogs and files.

!SASROOT\*product*\SASMACRO
: contains SAS autocall macro files.

!SASROOT\*product*\SASMSG
: contains the SAS message files.

!SASROOT\*product*\SAMPLE
: contains the Sample Library programs.

!SASROOT\*product*\SASTEST
: contains Test Stream programs.

!SASROOT\*product*\SASMISC
: contains miscellaneous external files shipped with the product.

Some products, such as SAS/CONNECT software, also have other subfolders that are associated with them. For details about each product's structure, see the specific SAS product documentation.

For more information about how the SAS folders are configured at your site, contact your SAS Support Consultant.

# Submitting SAS Code

## Introduction to Submitting SAS Code

SAS under Windows provides several methods for you to submit your SAS programs for processing. SAS supports a variety of work strategies, whether you run SAS interactively or in batch, and in conjunction with other Windows programs or as a stand-alone application.

## Submitting Code from the Enhanced Editor or Program Editor

To submit SAS code that you have typed into the Enhanced Editor or Program Editor window, you issue the SUBMIT command. SAS provides several ways to do this:

- Press F8 when the editor window is active.
- Click the Submit toolbar button.
- Enter **`submit`** in the command bar.
- From the Run pull-down menu, select **`Submit`**.

You can use the SUBTOP command from either the command line or the **`Run`** pull-down menu to submit one or more lines of your SAS code. For more information, see "SUBTOP Command" on page 350.

## Submitting Code from the SAS NOTEPAD Text Editor

SAS allows you to submit SAS code that you have typed into the SAS NOTEPAD text editor. NOTEPAD may be invoked by selecting

Tools ► Text Editor

when the Enhanced Editor is disabled. SAS provides several ways to submit the code in NOTEPAD:

- Click the Submit toolbar button.
- Enter **`submit`** in the command bar.
- From the menu, select

  Run ► Submit

## Submitting Code from the Clipboard

Using the Program Editor, you can submit SAS code that you copied from another Windows application (such as an editor or word processor) or from the SAS Help and Documentation. When you copy text from another Windows application, that text is stored in the Windows clipboard.

From the Run pull-down menu in the Program Editor window, select **`Submit clipboard`**. The code is submitted from the clipboard directly to SAS (without appearing in the Program Editor window). Notes and results are sent to the SAS Log and Output window, respectively. You can still issue the RECALL command (or press F4) to recall the submitted program into the Program Editor window.

You can also use the GSUBMIT command to submit SAS code that is stored in the clipboard. For more information, see "GSUBMIT Command" on page 345.

## Submitting Code by Dragging and Dropping

### Introduction to Submitting Code by Dragging and Dropping

You can drag SAS programs from other Windows applications onto an open SAS session and submit them. You can also drag files that contain SAS code and drop them on an open SAS session to submit them.

### Dragging Text from Other Windows

If you drag text from another Windows application or SAS window to the Enhanced Editor or the Program Editor window, that text is moved to the window by default. It is not submitted until you press F8 or issue the SUBMIT command.

However, you can override this default action by dragging the text using the *right* mouse button (if the application supports nondefault dragging). When you drop the selection on the Enhanced Editor or the Program Editor window, a menu appears and you can choose between moving the code or copying the code. The menu for the Program Editor also enables you to submit the code.

### Dragging Files in an Interactive Session

By using the My Favorite Folders window, you can access files that exist outside the SAS environment. Files that contain SAS code can be dragged into your interactive SAS session for execution. Access the My Favorite Folders window by using the **`View`** menu.

If you drop a file that contains SAS code on the Enhanced Editor window or the Program Editor window, that code is included in the window (but not submitted). If you drop the file on the Log or Output window or on a minimized SAS session, the code is automatically submitted.

When you minimize a SAS session, its icon appears on the Windows task bar. You cannot drop a file onto the task bar. Instead, you can drag the file to the SAS icon on the task bar and hold it there, without releasing the mouse button. After about one second, the SAS window resumes its normal size. Then you can drop the file on the open SAS session.

Dropping the file C:\MYPROG.SAS onto a window (other than the Enhanced Editor or Program Editor windows) of an open SAS session is the same as issuing this command:

```
gsubmit "%include 'c:\myprog.sas'";
```

You can submit more than one file at once by selecting a group of files that contain SAS programs and then dropping them onto the open SAS session. The order in which the programs are run when they are submitted as a group is determined by Windows. Therefore, if order is important, you should drop each program file separately.

If SAS is busy when you drop a SAS program icon, the dropped file is ignored. The only indication that the dropped file was ignored is a warning beep.

### Dragging Files to Start a Batch Session

If you drag a file that contains SAS code and drop it onto a SAS program icon, SAS starts in batch mode and submits the code for processing.

## Submitting Code Stored in Registered SAS File Types

During installation, the SAS Setup procedure registers certain file types with Windows to invoke specified actions when you double-click those types of objects. For example, files that have a file extension of .SAS are registered as SAS programs. These registered file types are displayed in Windows with a special icon, as shown here:

When you double-click a file that has this extension (or that has this icon) SAS is invoked and the contents of the file are included in the Enhanced Editor or Program Editor window. The SAS code that is contained in the file is not processed until you submit it (for example, by pressing F8 or by clicking the Submit tool). If you already have a SAS session running, double-clicking a file begins a second SAS session; it does not use the already-existing session.

SAS uses the default configuration file if you start SAS by double-clicking a registered SAS file type, such as .sas or .sas7bpgm.

## Interrupting Your SAS Session

You can click the circled exclamation point (!) in the toolbar or press CTRL+BREAK to interrupt processing in your SAS session. Depending on what tasks SAS is performing at the time of the interrupt, you can cancel submitted statements or cancel an upload or download request. SAS prompts you with various choices (such as to continue the interrupt or cancel it) in a dialog box.

*Note:* Depending upon what tasks are in progress when you interrupt your session, SAS may require several seconds to stop processing. △

SAS also supports the common Windows methods of issuing interrupts: you can click the Control menu icon and choose to close the application, or you can select **`Close`** from the pop-up menu for SAS on the Windows Task Bar (or **`End Task`** from within the Windows Task Manager). If you use either of these methods, SAS displays a dialog box to allow you to verify your selection. Note that the task might not close until SAS has completed processing.

## Running Windows or MS-DOS Commands from within SAS

### Overview of Running Windows or MS-DOS Commands from within SAS

You can execute Windows or MS-DOS commands from within SAS by using the X statement or the X command. You can also use the CALL SYSTEM statement or the SYSTASK statement from within a DATA step. Windows or MS-DOS commands can be issued 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. When you run a command as a synchronous task, the command must complete before another task can run.

To issue a command asynchronously, use either the SYSTASK statement with the NOWAIT option or specify the NOXSYNC system option. To issue a command synchronously, use either the SYSTASK statement with the WAIT option or specify the XSYNC system option. For more information about running asynchronous commands using the SYSTASK statement, see “SYSTASK Statement” on page 458.

## Running Windows Commands Using the X Statement or the X Command

You can use the X statement or the X command to run Windows commands. The X statement can be run outside of a DATA step. You can enter the X command in the command bar or any SAS command line.

The X statement is similar to the X command in the SAS windowing environment. The major difference between the two is that the X statement is submitted like any SAS statement; however, the X command is issued as a windowing environment command. This section uses the X statement in its examples, but the information applies to the X command as well.

When you submit the X statement you exit your SAS session temporarily and gain access to the Windows command processor. The X statement has the following syntax:

**X** <'*command*'>;

The optional *command* argument is used either to issue an operating system command or to invoke a Windows application such as Notepad. This discussion concentrates on using the X statement to issue operating system commands; however, you should be aware that the X statement can also be used to invoke Windows applications.

If you want to run only one operating system command, include the command as an argument to the X statement. When you submit the X statement, the command is executed, and you cannot issue any additional commands.

If you want to run several operating system commands, submit the X statement without an argument. A command prompt appears where you can issue an unlimited number of operating system commands. Remember, any environment variables you define are not available to SAS. If you submit an X statement or command without a *command* argument, type EXIT to return to your SAS session.

The X command is a global SAS statement; therefore, it is important to realize that you cannot conditionally execute the X command. For example, if you submit the following code, the X statement is executed:

```
data _null_;
   answer='n';
   if upcase(answer)='y' then
      do;
         x 'md c:\extra';
      end;
run;
```

In this case, the C:\EXTRA folder is created regardless of whether the value of ANSWER is equal to **'n'** or **'y'**.

## Using a DATA Step to Issue Conditional Operating System Commands Conditionally

If you want to issue operating system commands conditionally, use the CALL SYSTEM routine, as in the following example:

```
options noxwait;
data _null_;
   input flag $ name $8.;
   if upcase(flag)='Y' then
      do;
         command='md c:\'||name;
         call system(command);
      end;
   datalines;
Y mydir
Y junk2
N mydir2
Y xyz
;
```

This example uses the value of the variable FLAG to conditionally create folders. After the DATA step executes, three folders have been created: C:\MYDIR, C:\JUNK2, and C:\XYZ. The C:\MYDIR2 folder is not created because the value of FLAG for that observation is not **`Y`**.

For more information about the CALL SYSTEM routine, see "CALL SYSTEM Routine" on page 387 and the section on the CALL SYSTEM routine in *SAS Language Reference: Dictionary*.

## XWAIT System Option

The XWAIT system option controls whether you have to type EXIT to return to your SAS session after an X statement or X command has finished executing a MS-DOS command. (The XWAIT system option is not used if an X statement is issued without a *command* argument or if the X statement invokes a Windows application such as Notepad.) This option and its negative form operate in the following ways:

| | |
|---|---|
| XWAIT | specifies that you must type EXIT to return to your SAS session. This is the default value. |
| NOXWAIT | specifies that the command processor automatically returns to the SAS session after the specified command is executed. You do not have to type EXIT. |

If you issue an X statement or X command without a *command* argument, you must type EXIT to return to your SAS session, even if NOXWAIT is in effect.

When a window created by an X statement is active, reactivating SAS without exiting from the command processor causes SAS to issue a message box containing the following message:

```
The X command is active. Enter EXIT at
the prompt in the X command window to
reactivate this SAS session.
```

If you receive this message box, click **`MS-DOS Prompt`** on the Windows Task Bar. Enter the EXIT command from the prompt to close the window and return to your SAS session.

## XSYNC System Option

The XSYNC system option specifies whether the operating system command you submit executes synchronously or asynchronously with your SAS session. This option and its negative form operate in the following ways:

| | |
|---|---|
| XSYNC | specifies that the operating system command execute synchronously with your SAS session. That is, control is not returned to SAS until the command has completed. You cannot return to your SAS session until the command prompt session spawned by the CALL SYSTEM statement, the X command, or the X statement is closed. This is the default. |

| | |
|---|---|
| NOXSYNC | specifies that the operating system command execute asynchronously with your SAS session. That is, control is returned immediately to SAS and the command continues executing without interfering with your SAS session. With NOXSYNC in effect, you can execute a CALL SYSTEM statement, an X command, or an X statement and return to your SAS session without closing the window spawned by the X command or X statement. |

Specifying NOXSYNC can be useful if you are starting applications such as Notepad or Excel from your SAS session. For example, suppose you submit the following X statement:

```
x notepad;
```

If XSYNC is in effect, you cannot return to your SAS session until you close the Notepad. But if NOXSYNC is in effect, you can switch back and forth between your SAS session and the Notepad. The NOXSYNC option breaks any ties between your SAS session and the other application. You can even end your SAS session; the other application stays open until you close it.

## Comparison of the XWAIT and XSYNC System Options

The XWAIT and XSYNC system options have very different effects. An easy way to remember the difference is the following:

| | |
|---|---|
| XWAIT | means the command prompt session waits for you to type EXIT before you can return to your SAS session. |
| XSYNC | means SAS waits for you to finish with the other application before you can return to your SAS session. |

The various option combinations are summarized in Table 1.3 on page 28.

**Table 1.3** Combining the XWAIT and XSYNC System Options

| Options in Effect | Result |
|---|---|
| XWAIT<br>XSYNC | The command prompt window waits for you to type EXIT before closing, and SAS waits for the application to finish. |
| XWAIT<br>NOXSYNC | The command prompt window waits for you to type EXIT before closing, and SAS does not wait for the application to finish. |
| NOXWAIT<br>XSYNC | The command prompt window closes automatically when the application finishes, and SAS waits for the application to finish. |
| NOXWAIT<br>NOXSYNC | The command prompt window closes automatically when the application finishes, and SAS does not wait for the application to finish. |

# Terminating a SAS Process

You can terminate a SAS process using several methods. A SAS server is a specific type of SAS process.

*Note:* Before you terminate SAS using one of the following methods, you should try to end the process using one of the methods described in “Ending Your SAS Session” on page 29 or in the documentation for the SAS server. △

If the SAS process was instantiated as a Windows service, then you can terminate the process using one of the following methods:

- at the command prompt, submit one of the following commands:
  - **net stop <*service name*>**where *service name* is the name of the Windows service.
  - **sc <*server*> stop <*service name*>**where *server* is in the form "\\ServerName" and *service name* is the name of the Windows service.
- in the Microsoft Management Console Services snap-in, select the service that you want to terminate and select **Stop**.

To terminate a SAS process, use one of the following methods:

- if you are using Windows XP, at the command prompt submit

  ```
  taskkill/pid <process id>
  ```

  where *process id* is the SAS process ID. You can get this process ID from the output of the **tasklist** command.
- in the Windows Task Manager, select the process and click End Process.

***CAUTION:***
**Using the taskkill command or the Windows Task Manager to terminate a SAS process might result in data loss or data corruption.** △

## Ending Your SAS Session

You can end your SAS session using several methods, including

- selecting **Close** from the control menu of the main SAS window
- selecting Cancel in the Status window. This window appears when you are running in batch mode.
- double-clicking on the control menu of the main SAS window, or clicking on the **X** in the upper-right corner of the main SAS window
- issuing the BYE or ENDSAS command from a SAS command line
- submitting an ENDSAS statement
- closing the SAS session from the Task List by selecting the session process (the process name differs depending on how you started SAS) and selecting **End Task**
- selecting **Exit SAS** from the File pull-down menu in the main SAS window menu bar
- selecting **Exit** from the File pop-up menu
- pressing the ALT+F4 accelerator-key combination that is defined by Windows.

If SAS terminates with errors the SAS log might contain error messages that explain the failure. Any error message that SAS issues before the SAS log is initialized are written to the MSG window if it is available or to the SAS console log, which is a Windows file. Under Windows NT, the SAS console log is located in the c:\winnt\Profiles\*username*\Application Data folder. In all other Windows operating systems, the SAS console log is located in the c:\Document and Settings*userid*\Application Data folder. You can obtain the filename for the SAS console log from the Application Event Log. To open the application Event Log, submit **eventvwr** from the Run dialog box and click **Application**.

CHAPTER
2

# Interacting with SAS under Windows

*Overview of the SAS Interface* **33**
*The SAS Windowing Environment* **33**
*Understanding Components of the Main SAS Window* **33**
*Getting Help for the Main SAS Window* **35**
*Working within Your SAS Session* **35**
*Using the Docking View* **35**
*Introduction to the Docking View* **35**
*Docking and Undocking Windows* **35**
*Resizing the Docking View* **36**
*Minimizing and Restoring the Docking View* **36**
*Enabling and Disabling the Docking View* **36**
*Using the Window Bar* **36**
*Using Menus* **37**
*Changing the SAS Current Folder* **37**
*What Is the Current Folder?* **37**
*Interactively Selecting a New Current Folder* **38**
*Using SAS Statements to Change the Current Folder* **38**
*Issuing SAS Commands* **38**
*Using Menus to Issue Commands* **38**
*Using the Toolbar to Issue Commands* **39**
*Using the Command Bar to Issue Commands* **39**
*Using the Command Line to Issue Commands* **40**
*Sending E-Mail Using SAS* **40**
*Overview of Sending E-Mail* **40**
*Initializing E-Mail* **41**
*Using Your E-Mail Software to Send Mail* **41**
*Using the SAS Send Mail Dialog Box* **42**
*Sending the Contents of a Window by E-mail* **43**
*Using the DATA Step or SCL to Send E-mail* **44**
*Example of Sending E-Mail from the DATA Step* **47**
*Example of Sending E-Mail Using SCL Code* **49**
*Example of Sending E-Mail Using SMTP* **51**
*Saving Windows to External Files* **51**
*Clearing the Window and Filename* **52**
*Defining Keys* **52**
*Navigating with Microsoft IntelliMouse* **53**
*Using the Clipboard* **53**
*Selecting and Copying Text* **54**
*Selecting and Copying in Nontext Windows* **55**
*Pasting Bitmapped Information into Your SAS Session* **55**
*Submitting SAS Code from the Clipboard* **55**

*Creating Text Highlighting and Special Characters* **56**
*Special Character Attributes* **56**
*Alternate ASCII Characters* **56**
*Customizing Your SAS Session* **56**
*Selecting Fonts* **57**
*Setting Session Preferences* **57**
*Introduction to Setting Session Preferences* **57**
*Using the Preferences Dialog Box* **58**
*General Preferences* **58**
*View Preferences* **59**
*Edit Preferences* **59**
*Results Preferences* **60**
*Web Preferences* **61**
*Advanced Preferences* **61**
*Customizing Your Windowing Environment with Commands* **62**
*Customizing Window Positions* **62**
*Changing the Window Colors* **63**
*Customizing Your Windowing Environment with System Options* **63**
*Changing the Size and Placement of the Main SAS Window* **64**
*Changing the Title of Your SAS Session* **64**
*Adding Help to the Help Menu* **64**
*Minimizing Your SAS Session* **64**
*Adding Applications to the Tools Menu* **64**
*Setting the Initial Path For the Current Folder and the Paths Specified in the Open and Save Dialog Boxes* **65**
*Displaying a Custom Logo Screen during SAS Invocation* **65**
*Adding User-Defined Icons to SAS* **65**
*Enabling Web Enhancements in SAS* **66**
*Customizing the Toolbar* **66**
*Introduction to Customizing the Toolbar* **66**
*Using the Customize Tools Dialog Box* **66**
*Setting General Toolbar Preferences* **66**
*Customizing a Toolbar* **67**
*Adding a Tool to the Toolbar* **69**
*Removing a Tool from the Toolbar* **70**
*Customizing and Saving a Toolbar for Use with a Particular Application or Window* **70**
*Resetting the Tools to the Default Settings* **71**
*Examples of Useful Tools You Can Create* **71**
*Accessing Online Help and Documentation* **72**
*Using Microsoft HTML Help* **72**
*Getting Help from the Command Bar* **72**
*Getting Help in the Dialog Boxes* **73**
*Getting Help for a SAS Product* **73**
*Getting Help from the Help Menu* **73**
*Getting to SAS Institute (and Other Web Sites) from within SAS* **74**
*Accessibility Features in SAS under Windows* **74**
*Introduction to Accessibility Features in SAS under Windows* **74**
*Accessible Windows and Dialog Boxes* **74**
*The ACCESSIBILITY System Option* **76**
*Accessing the Standard or Fully Accessible User Interface* **76**
*Using the Accessible Customize Tools Dialog Box* **77**
*Using Accessible Properties Dialog Boxes* **77**
*Enlarging Fonts* **78**
*Enlarging Icons* **78**

*Resizing the Docking View in the Main SAS Window* **78**
*Sorting Window List Views by a Specific Column* **78**
*Resizing the Detail Columns of a List View* **79**
*Improving Access to Menus* **79**

## Overview of the SAS Interface

### The SAS Windowing Environment

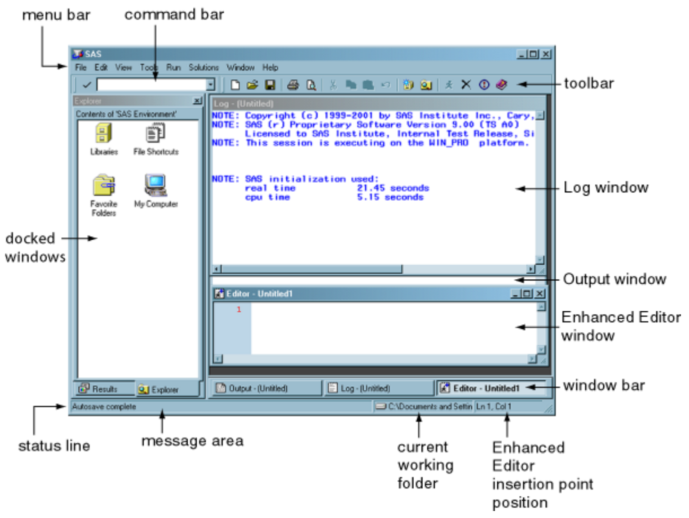
The SAS windowing environment refers to the windows that open in the main SAS window. You access the main SAS window when you start SAS from your PC or from a terminal emulator.

Windows in client software that do not access the main SAS window, such as Enterprise Guide or Java Display Manager System (JDMS), are documented in their product documentation. For more information about Enterprise Guide, see the Help in Enterprise Guide. For more information about JDMS, see *JDMS Reference*.

### Understanding Components of the Main SAS Window

The main SAS window contains all other SAS application windows. The main SAS window is completely configurable, allowing you to use its features in a way that is productive for you. The following display shows the main SAS window as it appears when you first start SAS. This section briefly describes the features of the window.

**Figure 2.1** The Main SAS Window


```
NOTE: Copyright (c) 1999-2001 by SAS Institute Inc., Cary,
NOTE: SAS (r) Proprietary Software Version 9.00 (TS A0)
      Licensed to SAS Institute, Internal Test Release, Si
NOTE: This session is executing on the WIN_PRO  platform.



NOTE: SAS initialization used:
      real time           21.45 seconds
      cpu time            5.15 seconds
```

The following are the primary components of the main SAS window:

menu bar
: presents the menus available for the active SAS application window. As you switch between application windows, the menu bar changes.

  Similarly, the pop-up menus that appear when you click the right mouse button inside an application window are customized for that window.

command bar

provides a way to quickly enter any SAS command. The command bar retains a list of the commands that you enter.

To repeat a command that you previously issued, either type until the command appears in the command bar or select the command from the list box and click the check mark button. Pressing ENTER will also submit a command.

To switch the keyboard focus to the command bar, press F11 (the function key defined as COMMAND FOCUS).

toolbar

provides quick access to the commands you perform most often. The toolbar is completely configurable and can contain up to 30 tools.

You can associate different sets of tools with different SAS application windows. When you create a tool, you specify the tool button, the commands associated with the tool, Help text displayed on the status line, and the tip text. The bitmap browser provides images that you can use to represent your commands on the toolbar.

windowing environment

contains a workspace to open windows within the main SAS window. Certain windows, such as windows that are used for navigation, can dock to the left side of the main SAS window when **`Docking view`** is enabled from the Preferences dialog box. Windows that cannot dock to the main SAS window open to the right of the docking area. In the above figure, the Log window, the Output window, the Enhanced Editor window, and the docked windows are all part of the windowing environment. For more information about using dockable windows, see "Using the Docking View" on page 35.

window bar

is located at the bottom of the main SAS window and provides easy access to any window within the main SAS window.

When a window opens, a button that represents that window is placed in the window bar. Whenever you want access to a window, click the button for that window. That window then becomes the active window.

You can load a file into an application by dragging a file to the window bar button for the application (making the application the active window), and then continue dragging the file into the application window.

The window bar can be enabled or disabled from either the Preferences dialog box or the window bar pop-up menu.

status line

contains a message area, the current folder for SAS, and the Enhanced Editor insertion point position.

The message area displays Help text for menus and tools, as well as messages that are specific to SAS application windows.

The current folder area displays the name of the current working folder. To change the current folder, double-click the current folder area. For more information, see "Changing the SAS Current Folder" on page 37.

The Enhanced Editor insertion point position displays the current line and column when the Enhanced Editor is the active window.

The status line can be enabled and disabled from the Preferences dialog box. The message line, the current folder, and the insertion point position can be enabled and disabled from the either the Preferences dialog box or the status line pop-up menu. When the message line is disabled, messages appear in the active window.

## Getting Help for the Main SAS Window

SAS provides help for the main SAS window using screen tips and status line messages.

For a description of a menu or a menu item:

1 Select the menu or menu item
2 As the mouse pointer passes over the menu or menu item, a description of the item appears in the message area of the status line.

For toolbar help, place the mouse pointer over a button. A pop-up ScreenTip appears below the mouse pointer and a longer description appears in the message area.

For help on other parts of the main SAS window, such as tabs, buttons, and the status line:

1 Place the mouse pointer over the item.
2 Hold the mouse pointer over the item for a few seconds. A pop-up ScreenTip appears below the mouse pointer. When you place the mouse pointer over a window bar button, the ScreenTip contains the window name.

To customize ScreenTip and status line help text for commands available from the toolbar see "Customizing the Toolbar" on page 66.

To enable or disable command bar or toolbar ScreenTips, you can use either the **`Show ScreenTips on toolbars`** option in the Customize dialog box or type the TOOLTIP command in the command bar.

All other ScreenTips can be enabled or disabled using the **`ScreenTip`** option in the Preferences dialog box **`View`** tab or by using the WSCREENTIPS command. For more information on enabling and disabling ScreenTips, see "Setting Session Preferences" on page 57,"Customizing the Toolbar" on page 66, "WSCREENTIPS Command" on page 369, and "TOOLTIPS Command" on page 354.

# Working within Your SAS Session

## Using the Docking View

### Introduction to the Docking View

The Docking View allows for easy navigation within the main SAS window. When the docking view is enabled, windows that can be docked (integrated with the main SAS window) such as the Explorer and the Results windows, appear on the left side of the main SAS window. When you open an object from a docked window, the opened object appears to the right of the docking area.

Each docked window has a tab at the bottom of the docking area for easy access to the window. When the number of docked windows is large enough so that you cannot identify the tabs, a left and right arrow are displayed for you to navigate through the docked windows.

### Docking and Undocking Windows

To dock or undock individual windows:

1 Select the window to make it the active window
2 Toggle the **`Docked`** menu item by selecting

   Window ► Docked

For information about setting docking view preferences, see "View Preferences" on page 59. To use a command to dock and undock the docking view, see "WDOCKVIEW Command" on page 358.

### Resizing the Docking View

Docked windows cannot be individually moved or resized.
To enlarge or contract the docking area:

1 Place the mouse pointer over the split bar between the docking area and the remaining portion of the main SAS window.
2 Press and hold down the left mouse button.
3 Move the mouse to the left or right to resize the docking area.

You can also resize the docking view by using the WDOCKVIEWRESIZE command. For more information, see "Resizing the Docking View in the Main SAS Window" on page 78 and "WDOCKVIEWRESIZE Command" on page 359.

### Minimizing and Restoring the Docking View

To minimize a docked window, do one of the following:

- Right-click the window title and select **`Minimize`**.
- Type **`wdockviewminimize`** in the command bar.

To restore the docked window, do one of the following:

- Click the **`Docking view`** button on the window bar.
- Type **`wdockviewrestore`** in the command bar.

For more information, see "WDOCKVIEWMINIMIZE Command" on page 358 and "WDOCKVIEWRESTORE Command" on page 359.

### Enabling and Disabling the Docking View

To enable or disable the docking view, do one of the following:

- Type WDOCKVIEW in the command bar without any arguments to toggle the docking view.
- Use the Preferences dialog box **`View`** tab:
  1 Select

     Tools ► Options ► Preferences ► View

  2 Select (enable) or deselect (disable) the **`Docking view`** check box.

## Using the Window Bar

The window bar, similar to the Windows taskbar, is a reserved space at the bottom of the main SAS window that is used to display a button for each opened window within SAS, providing immediate access to any opened window. When you click a button in the window bar, that window becomes the active window and appears on top of all other windows. When you click the button for the active window, the window is minimized.

Each button on the window bar has a menu that is associated with it. To access the menu, place the mouse pointer over the button and click the right mouse button.

You can enable and disable the window bar by using one of the following:

- the Preferences dialog box `View` tab
- the status line pop-up menu
- the window bar pop-up menu
- typing `wwindowbar` in the command bar

When you place the cursor over a window bar button, a ScreenTip pops up with the name of the window, or for an editor, the name of the opened file. You enable ScreenTips by using the Preferences dialog box or by typing `wscreentips` in the command bar. For more information about enabling ScreenTips, see "View Preferences" on page 59 and "WSCREENTIPS Command" on page 369.

By using drag-and-drop editing, you can use an application's window bar button to load a file into an application, such as the Enhanced Editor, that accepts file input. To do this

1. Drag the file on top of the application's button on the window bar, which makes the application the active window.
2. Drag the file to the application window.
3. Release the mouse button to load the file into the application.

If you attempt to drop a file onto a windowbar button, SAS issues an error message.

## Using Menus

You can access SAS commands, tools, and options by selecting them from the menus at the top of the main SAS window or by using the pop-up menus within application windows. The menus display options that are available to the active window. To access a pop-up menu for a particular window, click the *right* mouse button anywhere within the window. The pop-up menu that appears contains the menu items that are available for that particular window.

Some SAS windows (such as the Explorer window) along with the main SAS window can contain objects that have their own pop-up menus when you right-click an object. For example, the command bar, the toolbar, and the status line each have a pop-up menu. In these windows, the pop-up menu is specific to the selected object.

## Changing the SAS Current Folder

### What Is the Current Folder?

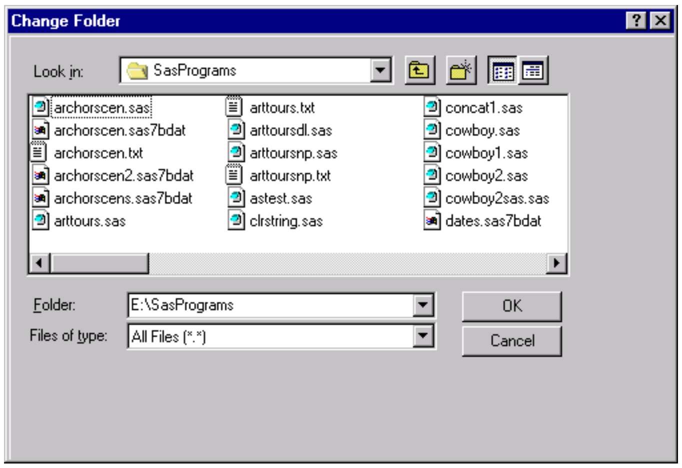
The current folder is the operating environment folder to which many SAS commands and actions apply. The current folder is displayed in the status line at the bottom of the main SAS window. By default, SAS uses the folder that is designated by the SASUSER system option in the SAS configuration file as the current folder when you begin your SAS session. You can specify a different default current folder by changing the `Start In` field that is available in the Properties tab for the SAS program shortcut or by specifying the SASINITIALFOLDER system option during SAS invocation. For more information, see "SASINITIALFOLDER System Option" on page 546.

Under Windows NT, SAS can be started by right-clicking or double-clicking a SAS file from Windows NT Explorer. The default current folder is the path to the operating system with \system32 appended to it. An example of a default current folder under Windows NT is c:\winnt\system32. Under other operating systems, the default is the folder where the file is stored.

## Interactively Selecting a New Current Folder

To change the SAS current folder during your SAS session, double-click the current folder in the status line. Then use the Change Folder dialog box (shown in the following display) to select a new current folder.

**Display 2.1** The Change Folder Dialog Box



If you organize your files so that each project has its own folder, then this Change Folder dialog box enables you to quickly switch between projects. As you select different projects, the dialog box stores in the **Folder** list box the directories that you select.

## Using SAS Statements to Change the Current Folder

You can change the current drive and folder by submitting the change directory (CD or CHDIR) command with the X statement in SAS. SAS intercepts the change directory command and then changes drive commands and changes its current folder.

For example, the following statements change the current folder for your SAS session to the MYDATA folder and G:\SALES\JUNE folder, respectively:

```
x 'cd \mydata';
x 'cd g:\sales\june';
```

To change the current drive, you can submit a change drive command (the drive letter followed by a colon) such as the following:

```
x 'a:';
```

## Issuing SAS Commands

### Using Menus to Issue Commands

Many commands are already assigned to menu items for the windows in which they apply. For example, selecting the **Run** pull-down menu and then selecting **Submit** is the same as typing **submit** in the command bar.

The items in the menu bar and pop-up menu vary depending on the active window. This ensures that each menu item is valid and appropriate for the currently active window.

## Using the Toolbar to Issue Commands

When you start SAS, by default, the toolbar appears at the top of the main SAS window. The toolbar provides a convenient way to issue commands that you use often. The toolbar commands are specific for the active SAS window.

To submit a command by using the toolbar, click the tool button that represents the command that you need.

To learn which tools perform what commands, position the mouse pointer over a tool briefly to reveal the ScreenTip for that tool.

To undock the toolbar to use it in a separate window or to dock the window to the main SAS window

1 Position the mouse pointer over the toolbar (not over a tool).

2 Press and hold down the right mouse button.

3 Select `Docked`.

You can add or change the tools that are defined in the toolbar and customize the toolbar for an application. For more information, see "Customizing the Toolbar" on page 66.

## Using the Command Bar to Issue Commands

The command bar, as shown in Figure 2.1 on page 33, is an integrated command line that offers a central location from which you can type any SAS command, as long as the command is valid for the active window. The command bar can also be undocked and appear in a separate window. It can be moved anywhere on your desktop. If you type a command that is not valid for the active window, an error message is displayed on the status line.

To move your insertion point position to the command bar, press F11.

SAS stores the commands that you type in the command bar from session to session, and you can easily retrieve previously issued commands by selecting them from the command list. The default number of commands to save is 15, but you can save from 0 to 50 commands. To change the number of commands to save in the command bar

1 Select

Tools ► Customize ► Toolbars

2 Press the up or down arrow in **`Number of commands saved`**.

By default, SAS stores the commands in the order of most frequently used. To store commands in the order of most recently used

1 Select

Tools ► Customize ► Toolbars

2 Select **`Sort commands by most recently used`**.

You can also retrieve previously issued commands by using the AutoComplete feature. When you start to type in the command bar, SAS completes the command that best matches the command that you are entering. When the command that you want appears in the command bar, press ENTER. To enable AutoComplete

1 Select

Tools ► Customize

2 Select **`Use AutoComplete`**.

3 Click **`OK`**.

To dock or undock the command bar

1 Position the mouse pointer over the command bar (the mouse pointer should not be in the text field).

2 Press and hold down the right mouse button.

3 Select **`Docked`**.

To customize the command bar by using a command, see "COMMAND Command" on page 328. For more information about the Customize Tools Toolbars tab, see "Setting General Toolbar Preferences" on page 66.

### Using the Command Line to Issue Commands

You can toggle the command line so that each window contains a command line. Commands that you enter in a window apply only to that window. For example, the INCLUDE command applies in the Enhanced Editor window, but not in the Log window.

To toggle the command line use the COMMAND command without arguments. You can also select **`Command Line`** from the Preferences dialog box **`View`** tab.

For more information, see "Setting Session Preferences" on page 57 and "COMMAND Command" on page 328.

## Sending E-Mail Using SAS

### Overview of Sending E-Mail

You can use SAS to send electronic mail either interactively (using a dialog box) or programmatically (using SAS statements in a DATA step or SCL). SAS supports three types of electronic mail interfaces:

- MAPI (Mail API, such as Microsoft Exchange, Lotus Notes Version 5, and Eudora)
- VIM (Vendor Independent Mail, such as Lotus cc:Mail and Lotus Notes Version 4)
- SMTP (Simple Mail Transfer Protocol).

You might need to install an e-mail client that supports one of these protocols before you can use SAS e-mail support. Also, although you can use SAS to send messages, you must use your e-mail program to view or read messages.

When you send mail interactively, SAS automatically includes the contents of the active window as an attachment to your e-mail. Depending on the contents of the active window, the attachment can be a text file (.TXT), a bitmap (.BMP), an HTML file (.HTML), or an RTF file (.RTF).

SMTP is available only by using the FILENAME statement for e-mail. If you specify SMTP as the e-mail system in the EMAILSYS system option and you are using an e-mail dialog box, the MAPI e-mail system is used. For information about using SMTP, see *SAS Language Reference: Concepts* as well as the FILENAME Statement, EMAIL (SMTP) Access Method, and the EMAILHOST and EMAILPORT system options in *SAS Language Reference: Dictionary*.

## Initializing E-Mail

To send e-mail from within SAS, use the following system options that are appropriate for your e-mail system in the SAS configuration file or when you invoke your SAS session:

-EMAILSYS MAPI | VIM | SMTP
: specifies which e-mail interface to use. By default, SAS uses MAPI. The SMTP interface is available only when you send e-mail programmatically. SMTP is not available when you use either your e-mail program native dialog box or the SAS e-mail dialog box.

*Note:* The directory that contains the e-mail DLL file (for example, MAPI32.DLL or VIM32.DLL) must be specified in your Windows PATH environment variable. SAS will use the first e-mail DLL that it finds for the interface that you specify. △

-EMAILDLG NATIVE | SAS
: specifies whether to use the native e-mail interactive dialog box that is provided by your e-mail application or the e-mail interface that is provided by SAS. SAS uses the native dialog box by default.

-EMAILHOST *SMTP server*
: specifies the domain name for the SMTP server that supports e-mail access for your site. This option is necessary only if you are using the SMTP e-mail interface.

-EMAILAUTHPROTOCOL *authorization protocol*
: specifies the authorization protocol to use in SMTP email. The default authorization protocol is LOGIN.

-EMAILID *VIM e-mail login ID* | *MAPI profile* | *e-mail address*
: specifies either your VIM e-mail login ID, the MAPI profile that you use to access the underlying e-mail system, or a fully qualified e-mail address if you are using SMTP. If any of these values contain spaces, you must enclose them in double quotation marks.

-EMAILPORT *port number*
: specifies the port number to which the SMTP server is attached. This option is necessary only if you are using the SMTP e-mail interface.

-EMAILPW "*password*"
: specifies your e-mail login password, where *password* is the login password for your login name. If *password* contains spaces, you must enclose the password in double quotation marks.

For example, if your login ID is **`J.B. Smith`** and your password is **`rosebud`**, your SAS invocation might look like this:

```
c:\sas\sas.exe -emailsys vim
               -emailid "J.B. Smith"
               -emailpw rosebud
```

## Using Your E-Mail Software to Send Mail

The default value of the EMAILDLG system option is NATIVE, which enables you to use your own e-mail software when you send e-mail interactively from within SAS. To send e-mail by using your own e-mail software, do one of the following:

- □ Select

  File ▶ Send Mail

- Type DLGSMAIL in the command bar.

Your e-mail software provides the interface to send mail.

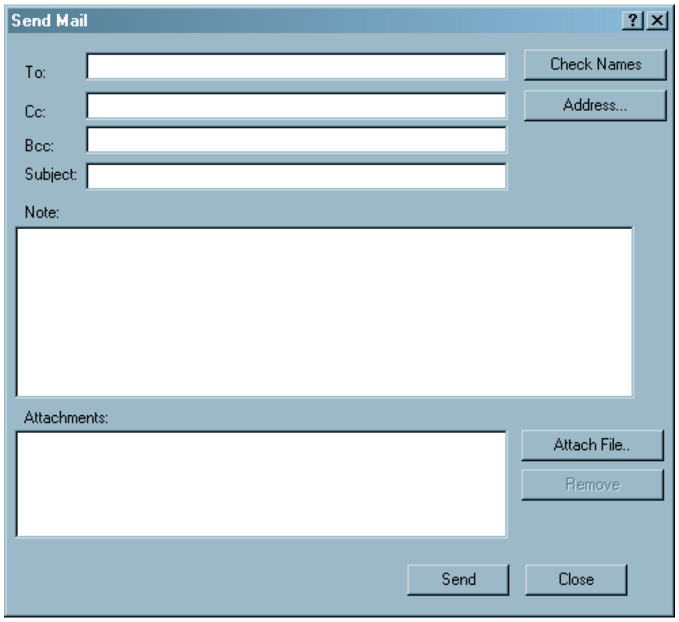
## Using the SAS Send Mail Dialog Box

If the value of the EMAILDLG system option is set to SAS, you can send electronic mail by using the Send Mail dialog box that is provided by SAS, as shown in the display.

To send e-mail system by using the SAS Send Mail dialog box, select

File ▶ Send Mail

**Display 2.2** Send Mail Dialog Box



The Send Mail dialog box contains the following fields:

**To**
the primary recipients of your e-mail. You must specify one or more e-mail addresses that are valid for your mail system before you can send e-mail. Separate multiple recipients with a semicolon (;).

**Cc**
the e-mail addresses of any users that you want to receive a copy of the mail that you are sending. You can leave this field blank if you want. Separate multiple recipients with a semicolon (;).

**Bcc**
specifies the recipients who will receive a copy of the e-mail. These addresses will not be visible to those individuals in the TO and CC options.

**Subject**
the subject of your message. You can leave this field blank.

**Note**
the body of your message. You can copy text from SAS application windows or other Windows applications and paste it here (using the CTRL+C and CTRL+V

accelerator key combinations). If your note text exceeds the window space that is provided, you can scroll backward and forward by using the arrow keys, or you can use the PgUp and PgDn keys.

Some e-mail systems currently limit the note length to 32K (or 32,768 characters). Text that you type in the **`Note`** area will automatically wrap at the right side.

For large amounts of text, attach a text file as described below.

**`Attachments`**

icons and names of any files that you want to send with the message. You or the recipient can open an attached file by double-clicking its icon, provided that its file extension has a File Manager association with a Windows application (for example, the .TXT extension might be associated with Notepad).

To open a file selection dialog box that you can use to select files to attach, click **`Attach File`**.

To remove an attachment, select the file's icon in the **`Attachments`** field and click **`Remove`**.

*Note:* The attached files are sent as they exist on the disk; that is, if you edit a file before attaching it to an e-mail message, the *saved* version of the file is sent with the message. △

To verify whether the addresses that you specified in the **`To`** and **`Cc`** fields are valid, click **`Check Names`**. If one or more of the addresses is ambiguous (that is, the mail program cannot locate them in the address books) SAS displays an error message and highlights the first ambiguous address.

Note that an ambiguous address is not necessarily invalid. It is possible to send mail to recipients who are outside your immediate local-area-network (LAN) by using gateways, even though the addresses might not be resolved by using **`Check Names`**.

Whether an address is considered invalid or ambiguous depends on the e-mail program that you are using and on the configuration of your network. For example, suppose you want to send e-mail to a colleague on the Internet. Your LAN might have a gateway to the Internet that enables you to address the mail to **`JBrown@rhythm.com at Internet`** (where **`at`** is the gateway directive keyword and **`Internet`** is the name of a gateway on your LAN). Because your mail program uses the **`at`** keyword to direct your message to the **`Internet`** gateway, the address is considered valid. However, when you click **`Check Names`**, the address is considered ambiguous because the final destination address cannot be resolved by using the local address book. You can still click **`Send`** to send the message without an error.

Clicking **`Address`** invokes the address book facility for your e-mail program, provided that the facility is accessible.

## Sending the Contents of a Window by E-mail

When you send mail from within SAS, SAS automatically attaches to the e-mail the contents of the active window. The type of file that SAS creates depends on the active window:

**Table 2.1**

| SAS creates this file type... | When the active window is... |
|---|---|
| .txt | Log<br>Output<br>Enhanced Editor<br>Program Editor |
| .bmp | Explorer<br>Graphics<br>Results |
| .htm | Results Viewer window and the output type is HTML |
| .rtf | Results Viewer window and the output type is RTF |

## Using the DATA Step or SCL to Send E-mail

By using the EMAIL access method, you can use the DATA step or SCL to send electronic mail from within SAS. This has several advantages:

- □ You can use the logic of the DATA step or SCL to subset e-mail distribution based on a large data set of e-mail addresses.
- □ You can automatically send e-mail upon completion of a SAS program that you submitted for batch processing.
- □ You can direct output through e-mail based on the results of processing.
- □ You can send e-mail messages from within a SAS/AF FRAME application, customizing the user interface.

In general, DATA step or SCL code that sends electronic mail has the following components:

- □ a FILENAME statement that contains the EMAIL device-type keyword
- □ options that are specified in the FILENAME or FILE statement that indicate the e-mail recipients, subject, and any attached files
- □ PUT statements that contain the body of the message
- □ PUT statements that contain special e-mail directives (of the form !EM_*directive*!) that can override the e-mail attributes (TO, CC, BCC, SUBJECT, ATTACH) or perform actions (such as SEND, ABORT, and NEWMSG).

To send e-mail by using the DATA step or SCL, you must be signed on to your e-mail program.

The FILENAME statement syntax to send e-mail is

**FILENAME** *fileref* EMAIL '*address*' <*email-options*>;

where

*fileref*
: is a valid fileref.

EMAIL
: is the device-type keyword that indicates that you want to use e-mail.

'*address*'

is the valid destination e-mail address that you want to send mail to. You must enclose the address in quotation marks. Specifying an address as a FILENAME statement argument is optional if you specify the TO= e-mail option or the PUT statement !EM_TO! directive, which will override an address specification.

*email-options*
can be any of the following:

*Note:* Each e-mail option can be specified only in a FILENAME statement that overrides the corresponding SAS system option. △

EMAILID= *e-mail login ID* | *MAPI profile* | *e-mail address*
specifies your e-mail login ID, MAPI profile name or your SMTP e-mail address that is used to access the underlying e-mail system. If you specify MAPI in the EMAILSYS system option, specify your profile name. If the value contains a space, enclose the name in double quotation marks. This e-mail option can be specified in the FILENAME statement that overrides the SAS system option.

EMAILPW=“*password*”
specifies your e-mail login password, where *password* is the login password for your login name. If *password* contains a space, enclose it in double quotation marks. This e-mail option can be specified in the FILENAME statement that overrides the SAS system option.

EMAILSYS=MAPI | VIM | SMTP
SAS supports three types of e-mail interfaces:

MAPI
Mail API is the interface that is supported by Windows operating environments, which is used by Microsoft Exchange. MAPI is the default.

VIM
Vendor Independent Mail, such as Lotus or cc:Mail.

SMTP
Simple Mail Transfer Protocol.

TO=*to-address*
specifies the primary recipients of the e-mail. If an address contains more than one word, you must enclose the address in double quotation marks. If you want to specify more than one address, you must enclose the group of addresses in parentheses and each address in double quotation marks. For example, valid TO values are

```
to="John Smith"
to=("J. Callahan" "P. Sledge")
```

CC=*cc-address*
specifies the recipients who will receive a copy of the e-mail. If an address contains more than one word, you must enclose the address in double quotation marks. If you want to specify more than one address, you must enclose the group of addresses in parentheses and each address in double quotation marks. For example, valid CC values are

```
cc="John Smith"
cc=("J. Callahan" "P. Sledge")
```

BCC=*bcc-address*
specifies the recipients who will receive a copy of the e-mail. These addresses will not be visible to those individuals in the TO and CC options. If an

address contains more than one word, you must enclose the address in double quotation marks. If you want to specify more than one address, you must enclose the group of addresses in parentheses and each address in double quotation marks. For example, valid BCC values are

```
bcc="John Smith"
bcc=("J. Callahan" "P. Sledge")
```

SUBJECT=*subject*
specifies the subject of the message. If the subject text is longer than one word (that is, it contains at least one blank space), you must enclose the text in double quotation marks. You must also use quotation marks if the subject contains any special characters. For example, **`subject=Sales`** and **`subject="June Report"`** are valid subjects.

ATTACH=*filename.ext*
ATTACH=(*filename.ext* <LRECL=*record-length*> <RECFM=*record-format*>)

*filename.ext*
specifies the full path and filename of one or more files to attach to the message.

LRECL=*record-length*
specifies the record length (in bytes). Under Windows, the default is 256. The value of *record-length* can range from 1 to 1,048,576 (1 megabyte).

RECFM=*record-format*
controls the record format. The following values are valid under Windows:

| | |
|---|---|
| F | indicates fixed format. |
| N | indicates binary format and causes the file to be treated as a byte stream. |
| P | indicates print format. |
| S370V | indicates the variable S370 record format (V). |
| S370VB | indicates the variable block S370 record format (VB). |
| S370VBS | indicates the variable block with spanned records S370 record format (VBS). |
| V \| D | indicates variable format. This is the default. |

The S370 values are valid with z/OS files only. That is, files must be binary, have variable-length records, and be in EBCDIC format. If you want to use a fixed-format z/OS file, first copy it to a variable-length, binary z/OS file.

If you want to attach more than one file or if you want to specify a record length and record format, you must enclose the group of filenames in parentheses and each filename in double quotation marks. For example, valid values for file attachments are

```
attach=opinion.txt
attach=("june2004.txt" "july2004.txt")
attach=("home.html" recfm=v lrecl=372);
```

If your e-mail system is SMTP, see “FILENAME Statement, EMAIL (SMTP) Access Method” in *SAS Language Reference: Dictionary* for additional ATTACH arguments.

Options that you specify in a FILE statement override any corresponding options that you specified in the FILENAME statement. In your DATA step, after using the

FILE statement to define your e-mail fileref as the output destination, use PUT statements to define the body of the message. For an example of using e-mail options in the FILE statement, see Example Code 2.2 on page 48.

You can also use PUT statements to specify e-mail directives that change the attributes of your electronic message or perform actions with it. You can specify only one directive in each PUT statement; each PUT statement can contain only the text that is associated with the directive that it specifies. The following are the directives that change your message attributes:

!EM_TO! *addresses*
: replaces the current primary recipient addresses with *addresses*. If a single address contains more than one word, you must enclose that address in quotation marks. If you want to specify more than one address, you must enclose each address in quotation marks and the group of addresses in parentheses.

!EM_CC! *addresses*
: replaces the current copied recipient addresses with *addresses*. If you want to specify more than one address, you must enclose each address in quotation marks and the group of addresses in parentheses.

!EM_BCC! *addresses*
: replaces the current copied recipient addresses with *addresses*. These recipients are not visible to those in the !EM_TO! or !EM_CC! addresses. If you want to specify more than one address, you must enclose each address in quotation marks and the group of addresses in parentheses.

!EM_SUBJECT! '*subject*'
: replaces the current subject of the message with *subject*.

!EM_ATTACH! *filename.ext*
: replaces the names of any attached files with *filename.ext*. If you want to specify more than one file, you must enclose each filename in quotation marks and the group of filenames in parentheses.

Here are the directives that perform actions:

!EM_SEND!
: sends the message with the current attributes. By default, SAS sends a message when the fileref is closed. The fileref closes when the next FILE statement is encountered or the DATA step ends. If you use this directive, SAS sends the message when it encounters the directive, and again at the end of the DATA step. This directive is useful for writing DATA step programs that conditionally send messages or use a loop to send multiple messages.

!EM_ABORT!
: aborts the current message. You can use this directive to stop SAS from automatically sending the message at the end of the DATA step. By default, SAS sends a message for each FILE statement.

!EM_NEWMSG!
: clears all attributes of the current message that were set by using PUT statement directives.

## Example of Sending E-Mail from the DATA Step

Suppose you want to share a copy of your SAS configuration file with your coworker Jim, whose user ID is **`JBrown`**. The following example code shows how to send the file with the DATA step.

**Example Code 2.1** Sending a File with the DATA Step

```
filename mymail email "JBrown"
         subject="My SASV9.CFG file"
         attach="c:\sas\sasV9.cfg";
data _null_;
   file mymail;
   put 'Jim,';
   put 'This is my SAS configuration file.';
   put 'I think you might like the';
   put 'new options I added.';
run;
```

The following example code sends a message and attaches a file to multiple recipients, and specifies the e-mail options in the FILE statement instead of the FILENAME statement.

**Example Code 2.2** Attaching a File and Specifying Options in the FILE Statement

```
filename outbox email "Ron B";
data _null_;
   file outbox
   /* Overrides value in */
   /* filename statement */
   to=("Ron B" "Lisa D")
   cc=("Margaret Z" "Lenny P")
   subject="My SAS output"
   attach="c:\sas\results.out"
   ;
   put 'Folks,';
   put 'Attached is my output from the SAS';
   put 'program I ran last night.';
   put 'It worked great!';
run;
```

You can use conditional logic in the DATA step to send multiple messages and control which recipients get which message. For example, suppose you want to send customized reports to members of two different departments. The following example code shows such a DATA step.

**Example Code 2.3** Sending Customized Messages Using the DATA Step

```
filename reports email "Jim";
data _null_;
   file reports;
   length name dept $ 21;
   input name dept;
   /* Assign the TO attribute */
   put '!EM_TO!' name;
   /* Assign the SUBJECT attribute */
   put '!EM_SUBJECT! Report for ' dept;
   put name ',';
   put 'Here is the latest report for ' dept '.';
   if dept='marketing' then
      put '!EM_ATTACH! c:\mktrept.txt';
   else  /* ATTACH the appropriate report */
      put '!EM_ATTACH! c:\devrept.txt';
```

```
    /* Send the message. */
    put '!EM_SEND!';
    /* Clear the message attributes.*/
    put '!EM_NEWMSG!';
    /* Abort the message before the */
    /* RUN statement causes it to   */
    /* be sent again.               */
    put '!EM_ABORT!';
    cards;
Susan          marketing
Jim            marketing
Rita           development
Herb           development
;
run;
```

The resulting e-mail message, and its attachments, are dependent on the department to which the recipient belongs.

*Note:* You must use the !EM_NEWMSG! directive to clear the message attributes between recipients. The !EM_ABORT! directive prevents the message from being automatically sent at the end of the DATA step. △

The following example code shows how to send a message and attach a file to multiple recipients. It specifies the e-mail options in the FILENAME statement instead of in the FILE statement. This overrides the values for the SAS system options EMAILID, EMAILPW, and EMAILSYS.

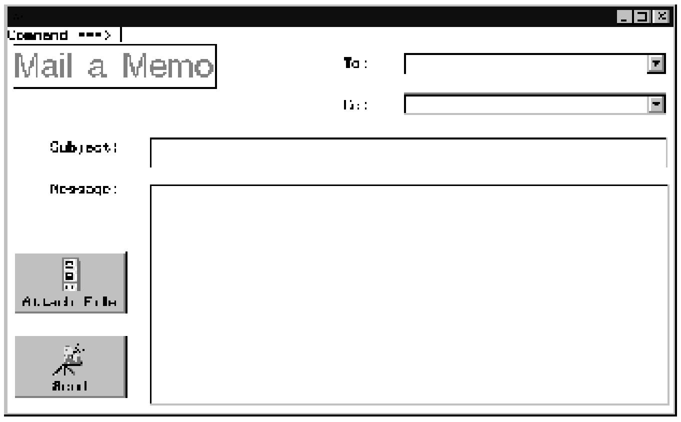
**Example Code 2.4** Overriding SAS System Options

```
filename outbox email "Ron B" emailsys=VIM
 emailpw="mypassword" emailid="myuserid";
data _null_;
  file outbox
        /* Overrides value in */
        /* filename statement */
     to=("Ron B" "Lisa D")
     cc=("Margaret Z" "Lenny P")
     subject="My SAS output"
     attach="c:\sas\results.out"
     ;
  put 'Folks,';
  put 'Attached is my output from the SAS';
  put 'program I ran last night.';
  put 'It worked great!';
run;
```

## Example of Sending E-Mail Using SCL Code

The following example is the SCL code that generates a FRAME entry that is designed for e-mail. The FRAME entry might look similar to the one shown in the following display.

**Display 2.3** An Example E-Mail FRAME Entry



The FRAME entry has objects that enable the user to type the following information:

| | |
|---|---|
| MAILTO | the user ID to send mail to. |
| COPYTO | the user ID to copy (CC) the mail to. |
| ATTACH | the name of a file to attach. |
| SUBJECT | the subject of the mail. |
| LINE1 | the text of the message. |

The following example code shows the FRAME entry that also contains SEND button that invokes this SCL code (marked by the **send** label).

**Example Code 2.5** Invoking SCL Code from a FRAME Entry

```
send:
   /* set up a fileref */
   rc = filename('mailit','userid','email');
   /* if the fileref was successfully set up,
      open the file to write to */
   if rc = 0 then do;
      fid = fopen('mailit','o');
      if fid > 0 then do;
       /* fput statements are used to
          implement writing the mail and
          the components such as subject,
          who to mail to, etc. */
   fputrc1= fput(fid,line1);
   rc = fwrite(fid);
   fputrc2= fput(fid,'!EM_TO! '||mailto);
   rc = fwrite(fid);
   fputrc3= fput(fid,'!EM_CC! '||copyto);
   rc = fwrite(fid);
   fputrc4= fput(fid,'!EM_ATTACH! '||attach);
   rc = fwrite(fid);
   fputrc5= fput(fid,'!EM_SUBJECT! '||subject);
   rc = fwrite(fid);
   closerc= fclose(fid);
      end;
   end;
```

```
return;
cancel:
   call execcmd('end');
return;
```

## Example of Sending E-Mail Using SMTP

The following examples show how you can send e-mail by using SMTP from a DATA step and how you can send your ODS HTML output as HTML and not as an attachment to your e-mail.

To use SMTP you need an SMTP e-mail server that you can access.

To configure SAS to use SMTP, add these system options to your configuration file:

- -emailsys SMTP
- -emailhost *your.email.server.com*
- -emailport 25.

Ask your system administrator for the location of *your.email.server.com*. Port 25 is the most common port.

The following code uses the FILENAME statement and a DATA step to send e-mail:

```
filename mymail email from="yourid@email.com"
                      to="id1@.emailaddr.com" "id2@emailaddr.com")
                      subject="Put Subject Here"
                      content_type="text/html";
data _null_;
   file mymail;
   put 'hello world';
run;
quit;
```

You can also send an attachment by using the ATTACH e-mail option in the FILENAME statement. Compress non-textual attachments such as SAS data sets, bitmap files, and HTML files before using the ATTACH e-mail option.

You can also use SMTP to send HTML output without using an attachment:

```
filename temp1 email to=("yourid@email.com")
                     from="wileycoyote@acme.com"
                     subject="HTML OUTPUT"
                     content_type="text/html";
ods html body=temp1 style=default;
proc print data=sashelp.class;
run;
ods html close;
```

## Saving Windows to External Files

You can save any text editor window, such as the Enhanced Editor window, the Program Editor window, or other SAS windows, such as the Log, Output, or Results Viewer windows to an external file.

To save the contents of the active window to a file:

1 Either click the Save button (the diskette) or select the **`File`** pull-down menu and select **`Save`**. If you have previously saved the contents of this window to a file (and the filename is part of the window title), SAS saves the contents to the file that you specified previously. If you have not saved the window contents during this session, then SAS displays the Save As dialog box.

If you have previously saved the window contents but now want to save the window contents to a different file, type **`dlgsave`** in the command bar or select

File ▶ Save As

***CAUTION:***

**Using Save instead of Save As from the File menu to save a file causes SAS to overwrite or append the file.** Always use Save As when you want to save the contents of the editor to a new file. If you open a text file in the editor window, whether you use the Open dialog box or the INCLUDE command, the editor title bar displays the name of the file that you opened. When you select the **`File`** pull-down menu and then the **`Save`** item, SAS overwrites or appends the file of that name with the current contents of the editor. △

2 Select or name the file in which to store the window contents. Optionally, you can select a file type from the **`Save file as type`** list. SAS saves most file types as plain text and assigns different file extensions based on the type you select; the exception is the RTF file type, which SAS saves in rich text format (RTF).

If you select a file type from the list, SAS remembers that selection and presents it as the default type the next time that you save a new file in that window.

## Clearing the Window and Filename

To clear a SAS window of its contents and saved filename (if it has one), do one of the following

- □ Press CTRL + E
- □ Select **`Clear All`** from the **`Edit`** menu
- □ Click the New (the blank page) button
- □ Select the **`File`** pull-down menu and select either **`New`** or **`New Program`**
- □ Type CLEAR in the command bar and press ENTER.

If the contents of the window have not been saved, SAS prompts you to save the contents before it clears the window.

## Defining Keys

To display the key definitions that are active for the SAS session (that is, the DMKEYS entry in your Sasuser.Profile catalog), either type KEYS in the command bar or select

Tools ▶ Options ▶ Keys

These key definitions apply to the basic SAS windows, such as the Enhanced Editor, Output, and Log windows. For a list of default keys, see "Default Key Definitions under Windows" on page 615 and "Keyboard Shortcuts within the Enhanced Editor" on page 618.

To define or redefine a key within SAS:

1 Click the mouse pointer in the **`Definition`** column across from the key or mouse button that you want to define.

2 Type the command or commands that you want to associate with that key or button.

The definition must be a valid SAS command or sequence of commands. When you specify a sequence of commands, separate the commands with a semicolon ( ; ). For example, if you want to define the Ctrl + H key sequence to maximize a window and recall the last submitted program, specify the following commands in the **`Definitions`** column next to **`CTL H`**:

```
zoom; recall
```

SAS does not check the syntax of a command until it is used (that is, when the key is pressed). If you misspell a command or type an incorrect command, you do not discover your error until you use the key and receive an error message that indicates that the command was unrecognized.

Key definitions are stored in your Sasuser.Profile catalog. SAS creates a profile catalog each time you invoke SAS with a different value for the SASUSER option. Changes that you make to one profile catalog are not reflected in any other catalog. However, you can use the COPY command from the KEYS window or the CATALOG procedure to copy key definition members to other profile catalogs. For more information, see the CATALOG procedure in *Base SAS Procedures Guide*.

Although SAS enables you to define any key that is listed in the KEYS window, Windows reserves some keys for itself to maintain conformity among Windows applications. These reserved keys are not shown in the KEYS window.

Other SAS products have their own key definitions. Use the pull-down menus in the specific product window to access key definitions for specific products.

## Navigating with Microsoft IntelliMouse

SAS provides support for the Microsoft IntelliMouse. The IntelliMouse is a modified mouse that includes a rotation wheel (wheel control) that enables new forms of navigation. The IntelliMouse works within the SAS windows that use a vertical scrollbar to scroll the window contents.

With the IntelliMouse, you can use the mouse to scroll instead of interacting with the navigational controls in the SAS windows. To scroll with the IntelliMouse, you rotate the wheel control forward or backward, which is equivalent to pressing the up arrow or down arrow on the scrollbar.

The IntelliMouse also supports AutoScroll. To initiate AutoScroll, click the mouse wheel and then move the mouse away from the origination point. The contents of the window starts to scroll in the direction that you move the mouse. The farther away you move the mouse from the origination point, the faster the contents scroll. Pressing a key, clicking a mouse button, or rotating the mouse wheel terminate AutoScroll mode.

You can modify IntelliMouse settings through the Windows Control Panel mouse settings. For more information about IntelliMouse, see the Microsoft documentation.

## Using the Clipboard

The Windows clipboard enables you to exchange text and graphics between applications. You can also submit SAS code that is stored on the clipboard. The clipboard uses operating environment memory as an intermediate storage buffer for exchanging text and graphics. With the clipboard, you can move text between

- windows within SAS
- SAS and other Windows applications
- two SAS sessions
- two SAS processes.

SAS communicates with the clipboard using these formats:

SAS text format

preserves the text and color attributes between SAS sessions. This format is understood by SAS, but not by other Windows applications.

Windows text format
: is understood by most Windows applications and is called the ANSI text format.

RTF text format
: encapsulates text font and highlighting attributes when copying text between applications that both support RTF format. SAS can only cut and copy text in RTF format; you *cannot* paste RTF text into a SAS window.

Unicode text format
: is a universal text format that can represent all possible characters. The format is also referred to as a wide-character format. For more information about this text format, see your Microsoft documentation.

Windows bitmap format
: is for graphics. This format is understood by most Windows applications and is called the BITMAP format.

Windows metafile (WMF) format
: is used in many SAS applications, such as the Graphics Editor in SAS/GRAPH software, SAS/QC software's ISHIKAWA procedure, and SAS/INSIGHT software. The metafile format provides more information about the image than the bitmap format and is sometimes called the PICTURE format.

DIB (Device Independent Bitmap) format
: is used with color bitmap files. When a bitmap is stored in the DIB format, colors correspond correctly from one device to another.

These formats enable you to copy text and SAS bitmapped information (for example, from a graphic) to another application. You can also use the Print Screen and ALT+ Print Screen keys to copy information from your SAS session to the clipboard. Pressing the Print Screen key places the entire display in bitmap form on the clipboard. Pressing ALT+Print Screen places just the SAS session (including any menus and scroll bars) or the active dialog box (if any) on the clipboard.

You can use the clipboard only if both the source and destination applications provide support for the clipboard facility and for the format you are using. Note that whereas some operating environments allow multiple paste buffers, SAS uses the Windows clipboard, which is a single buffer.

## Selecting and Copying Text

For windows that contain text, such as the Enhanced Editor, Notepad, Log, Output, and KEYS windows, you can hold down either the left mouse button or ALT + the left mouse button and drag the mouse to mark the area that you want to cut or copy. Holding down the left mouse button when you are selecting multiple lines selects whole lines of text. Holding down ALT + the left mouse button selects a rectangular block or column of text. The text area is immediately marked in reverse video while you are dragging the mouse. In text windows, you can scroll while you are dragging the mouse by moving the mouse pointer beyond the border of the window in the direction that you want to scroll. To extend the selection of a text area, use the SHIFT key + the left mouse button. Release the mouse button when you have included all the text you want to copy.

To copy marked text to the clipboard, do one of the following:

- press Ctrl + C
- click the Copy button (the double document)
- select the **`Edit`** pull-down menu and then select **`Copy`**.

To paste text that is stored on the clipboard, position the insertion pointer in a text area of a window and do one of the following:

- □ press Ctrl + V
- □ click the Paste button (the clipboard and document)
- □ select the **Edit** pull-down menu and then select **Paste**.

The text from the clipboard is pasted to the area that you indicate. If there is already an area of selected text within the target window, the selected text is replaced with contents of the clipboard. You can paste text only into SAS windows that accept text input, such as the Enhanced Editor or the SAS Notepad.

## Selecting and Copying in Nontext Windows

For windows such as SAS/GRAPH windows, an area is marked by a box, not by reverse video. The box indicates that the area you are marking is in bitmap format. After you finish marking an area, you can copy it to the clipboard. If the window you are working in has no **Edit** pop-up menu, you can use the following key combinations to perform the copy and paste functions:

| | |
|---|---|
| CTRL+C | copies the selection to the clipboard. |
| CTRL+V | pastes the contents of the clipboard. |

## Pasting Bitmapped Information into Your SAS Session

Some windows, such as the BUILD: DISPLAY window for FRAME entries in SAS/AF software, allow you to paste bitmaps into the window. For more information, see “Pasting an OLE Object from the Clipboard” on page 243.

Also, you can paste bitmaps into the SAS/GRAPH window to import graphics. For more information, see “Importing Graphics from Other Applications” on page 188.

## Submitting SAS Code from the Clipboard

SAS enables you to use the Windows clipboard to submit SAS code. This feature can be used to copy or cut SAS code from another application, such as the Windows Notepad or another text editor, and submit it to SAS for execution. This is also convenient for submitting the sample programs that are available in SAS Help and Documentation.

To submit SAS code that is stored on the clipboard, select the **Run** pull-down menu and then select **Submit clipboard** when the Enhanced Editor window or the Program Editor window is active. Alternatively, you can use the GSUBMIT command from the command line, with the following syntax:

```
gsubmit buf=default
```

The GSUBMIT command can be used to submit SAS code that stored on the clipboard even if the editor window is not the active window (or is closed). If you use the GSUBMIT command often, you may want to define an icon for it in the tool bar, or assign the GSUBMIT command to a function key. For more information about how to define buttons, see “Customizing the Toolbar” on page 66.

If you submit SAS code from the Windows clipboard while a procedure RUN group is active, the submit will fail. You can submit this code by copying the code to a new Enhanced Editor window and then submitting the code.

## Creating Text Highlighting and Special Characters

### Special Character Attributes

The SAS Notepad and SAS/AF applications let you use extended color and highlight attributes for text. To access these attributes, press the ESC key and the appropriate letter or number to toggle a color or attribute. With this feature, you can alter the color or attributes of entire lines or individual words or letters. Valid colors and attributes, as well as the keys that you use to implement them, are listed in Table 2.2 on page 56 and Table 2.3 on page 56. You can type the letters for the colors in either uppercase or lowercase letters.

**Table 2.2** Extended Color Key Sequences

| Key | Color | Key | Color |
|---|---|---|---|
| ESC+A | gray | ESC+B | blue |
| ESC+C | cyan | ESC+G | green |
| ESC+K | black | ESC+M | magenta |
| ESC+N | brown | ESC+O | orange |
| ESC+P | pink | ESC+R | red |
| ESC+W | white | ESC+Y | yellow |

**Table 2.3** Extended Attribute Key Sequences

| Key | Description |
|---|---|
| ESC+0 | turns off all highlighting attributes. |
| ESC+2 | turns on the underline attribute. |
| ESC+3 | turns on the reverse-video attribute. |

### Alternate ASCII Characters

If you want to create alternate ASCII characters such as foreign language characters, you can use the ALT key in combination with the ASCII character code. Use the numeric keypad and press the Num Lock key to enter the character code. For a list of ASCII character codes and instructions about how to use the ALT key sequences, see your Microsoft documentation.

## Customizing Your SAS Session

## Selecting Fonts

To change the font for button text and descriptive text elements, use the SYSGUIFONT system option either in the configuration file or at the command prompt when you start SAS.

To choose a different font or point size for text in SAS windows, open the Fonts dialog box by using the DLGFONT command or by selecting

Tools ► Options ► Fonts

To change the font in the Enhanced Editor, select

Tools ► Options ► Enhanced Editor

and click the **`Appearance`** tab.

The fonts that are available for SAS windows depend on the monospace fonts that you have installed under Windows. For example, you might have the Courier font and Lucida Console font available.

When you select a font or point size, the Font dialog box and the Enhanced Editor Options dialog box display a sample of the font that you have selected. For more information about selecting fonts for the Enhanced Editor, select

Help ► Using This Window

or press F1 when the Enhanced Editor is the active window.

When you install SAS, the Setup program automatically installs a TrueType font, named SAS Monospace, that is designed specifically for use with SAS. This font, in combination with the Sasfont display font, ensures that tabular output is formatted properly whether you view it in the Output window, print it, or copy it to another Windows application.

By default, SAS uses the SAS Monospace font to produce printed output. In addition, any text that you cut, copy, or drag from a SAS window to paste into another Windows application will be formatted with the SAS Monospace font.

You cannot use the **`Fonts`** item to select SAS/GRAPH fonts.

***CAUTION:***

**Beware of changing certain display characteristics on low-resolution displays.** If you select large font sizes on some monitors, you may not be able to see all the text in your SAS windows at one time. In windows that have no scroll bars, large font sizes can hide some choices, causing them to be invisible. For these types of displays, large font sizes are not recommended. This same problem can occur if you change the Windows Appearance properties and select a thick window border. On low-resolution displays, you should not use thick window borders. △

## Setting Session Preferences

### Introduction to Setting Session Preferences

You can configure your SAS session to accommodate the way that you like to work. For example:

- □ You can use the command bar in a separate, movable window.
- □ You can set preferences for scrolling behavior and window appearance.
- □ You can set a preferred Web browser to use when viewing internet Web pages or HTML output.

The following sections describe the Preferences dialog box and how to use these settings to control your SAS session.

## Using the Preferences Dialog Box

To customize your SAS session, open the Preferences dialog box in one of the following ways:

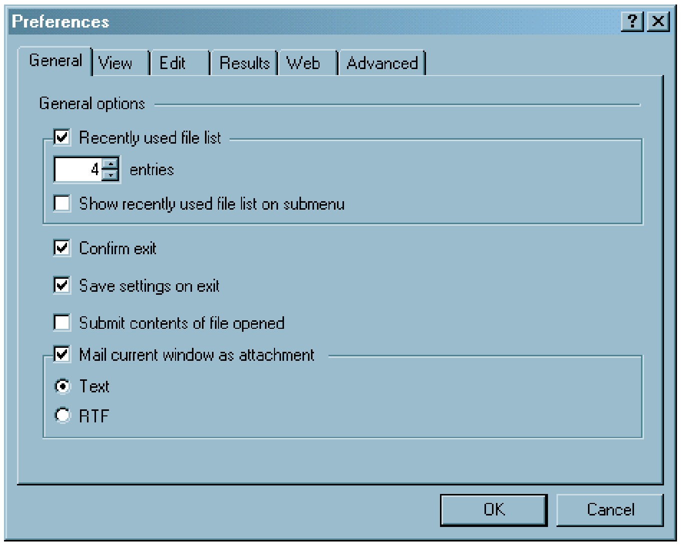
- Type **`dlgpref`** in the command bar
- Select

  Tools ▶ Options ▶ Preferences

The Preferences dialog box contains tabs that separate the session settings into categories. Click the tabs for each sheet to navigate to the settings that you want to change, and then select the options that you want. When you are finished, click **`OK`**.

The settings that you select are saved from session to session in the Sasuser.Profile catalog by their respective pages, except for the Results tab. The entries in the Sasuser.Profile are GENWSAVE, VIEWWSAVE, EDITWSAVE, WEBWSAVE, and ADVWSAVE. The Results tab settings are saved in the SAS registry, so they are not moved to another machine when the Sasuser.Profile catalog is copied.

**Display 2.4** Preferences Dialog Box (showing the General tab)



General options

- [✓] Recently used file list
  - 4 entries
  - [ ] Show recently used file list on submenu
- [✓] Confirm exit
- [✓] Save settings on exit
- [ ] Submit contents of file opened
- [✓] Mail current window as attachment
  - [● Text]
  - RTF

## General Preferences

The General tab enables you specify the general options that control how your SAS session works. The following are the General options:

**`Recently used file list`**
: specifies whether SAS retains a list of the files that you have accessed. If this option is selected, you can specify in the **`entries`** field up to 30 files that you want to retain. **`Show recently used file list on submenu`** specifies whether the files will be displayed from the **`Recent Files`** submenu that you access from the **`File`** menu. If **`Show recently used file list on submenu`** is not selected, the files are displayed in the **`File`** menu. Each time that you access a file from an editor window, the filename is added to the list.

**`Confirm exit`**

specifies whether you want SAS to prompt you for confirmation before you end your SAS session.

**`Save settings on exit`**
specifies whether SAS should automatically save your settings when you exit your SAS session.

**`Submit contents of file opened`**
specifies whether you want to submit the contents of all files that you open to SAS.

**`Mail current window as attachment`**
specifies whether the active window should be automatically included as an e-mail attachment when you initiate electronic mail from within SAS. If you select this option, then you can also specify whether the attachment should be formatted as plain text or as RTF (rich text format, which retains font and color information).

## View Preferences

The View tab lets you specify the options that control the appearance of your SAS session. The View options include

**`Window`**
specifies whether your SAS windows contain scroll bars and a command line. You can also enable or disable ScreenTips (the helpful hints that appear when you position your mouse pointer over window controls).

**`Show`**
specifies whether to show certain aspects of the SAS interface, including the following settings:

> **`Docking View`**
> specifies whether to enable the docking area so that windows that can be docked appear on the left side of the main SAS window.
>
> **`Window Bar`**
> specifies whether to display the window bar at the bottom of the main SAS window.
>
> **`Status line`**
> specifies which aspects of the status line, if any, you want to have visible in your session. **`Display message lines`** specifies whether to display the message area of the status line. **`Display current folder`** specifies whether to display the SAS current folder area. **`Display cursor position`** specifies whether to display the line and column position of the Enhanced Editor insertion point.

## Edit Preferences

The Edit tab controls options that affect the SAS text editors, including:

**`Overtype mode`**
specifies whether to insert text or type over on existing text when you type text in a SAS application window. You can also toggle the overtype mode by pressing the Insert key on your keyboard. *Overtype mode is not available for the Enhanced Editor.*

**`Autosave every` *`n`* `minutes`**
specifies whether to automatically save the contents of the editor, and how often to save it.

The Enhanced Editor contents are saved as **Autosave of *filename*.$AS** in the operating environment Application Data folder. Under Windows NT, the pathname for the Application Data folder is c:\winnt\Profiles\*username*\Application Data\SAS\EnhancedEditor. Under all other Windows operating environments, the pathname for the Application Data folder is c:\Documents and Settings*username*\Application Data\SAS\EnhancedEditor.

The Program Editor contents are saved to **pgm.asv** in the current active folder so that you can recover your work in the event that your SAS session ends without enabling you to save the contents of the editor.

**Enable unmarking with navigation keys**
enables you to unmark text by using the UP, DOWN, LEFT, and RIGHT navigation keys.

**Use Enhanced Editor**
specifies whether the Enhanced Editor is the primary editor. If this check box is not selected, the Program Editor opens when SAS starts.

## Results Preferences

The Results tab enables you to configure how you would like to view your program output results. The Results tab options include the following:

**Listing**
specifies to display program output in the Output window.

**HTML**

**Create HTML**
specifies to display program output in HTML format.

**Folder**
specifies a folder to store HTML output files. You can either type a folder name or click **Browse** to search for a folder. This setting is available only when the **Use WORK folder** setting is not selected.

**Use WORK folder**
specifies to store HTML output files in the Work folder. The Work folder is a temporary folder that is deleted when SAS closes.

**Style**
enables you to choose the appearance of the program output. For more information about styles, see the TEMPLATE procedure in *SAS Output Delivery System: User's Guide*.

**View results as they are generated**
specifies whether to update the browser with the latest generated HTML output.

**View results using**
specifies a browser to view HTML program output. **Internal browser** is available if Microsoft Internet Explorer is installed. When **Internal browser** is selected, SAS displays HTML output using the Results Viewer.

If you select **Preferred browser**, your HTML output appears using the browser that is specified by the **Preferred browser - Other** text field of the Preferences dialog box **Web** tab.

*Note:* If you select **Use default** in the Preferences dialog box **Web** tab, your output is displayed using the browser that is registered with Windows. △

## Web Preferences

The Web tab enables you to specify your preferred Web browser for use within your SAS session. These preferences are used whenever you issue the WBROWSE command (either directly or by selecting a **`Help`** menu item or toolbar button that issues the command). For more information, see "WBROWSE Command" on page 356. You can specify the following Web options:

**`Preferred browser`**
specifies the preferred Web browser to use when accessing Web information from within SAS. By default, SAS uses the browser that is installed on your system and registered with Windows as the default browser. To use a browser other than the default, select the **`Other`** radio button and either type a path to the Web browser or click **`Browse`** to search for the path to the Web browser.

**`Start page`**
specifies the default Web page to which to navigate when invoking the web browser within SAS. By default, the browser navigates to http://www.sas.com (the SAS Institute home page on the World Wide Web).

## Advanced Preferences

The Advanced tab enables you to specify options that can affect your SAS session, including scrolling policy and other miscellaneous behavior. The Advanced options include

**`Scrolling Options`**
specifies the number of lines that the Log and Output windows scroll when information is written to them. The default value for the Log window is 8.

When you select **`Scroll page`**, the Output window will not display any lines until an entire page is written.

When **`Scroll max`** is selected, no output will be written to the window until the procedure is complete.

If **`Scroll lines`** is selected and the Output window is full, the Output window scrolls the number of lines specified in the **`Scroll lines`** box. The default value is 0. If the value is 0, no output is written to that window while statements are executing, thus providing the best performance.

Scrolling can increase the length of time that SAS takes to run your program. The less that the Log and Output windows have to scroll, the faster your program will run.

You can also set these values by using the Editor Options window or the AUTOSCROLL command. For more information, see "AUTOSCROLL Command" on page 325 and the SAS Help and Documentation.

**`Other`**
These are miscellaneous options settings:

**`Hide cursor in non-input windows`**
specifies that the insertion point will not appear in windows that do not require text input, such as some SAS/AF programs.

**`Disable scroll bar focus`**
specifies that the scroll bar does not become the selected window component when you click it. This eliminates flashing problems that can occur in some SAS applications.

## Customizing Your Windowing Environment with Commands

### Customizing Window Positions

In the default display configuration of an interactive session (shown in Figure 2.1 on page 33) the main SAS window displays the Explorer and Results windows as docked windows, and the Log, Enhanced Editor, and Output windows in the remaining SAS workspace.

Using the Windows menu, you can position SAS windows in the same manner as other Windows applications:

- **`Minimize (Restore) All Windows`**
- **`Cascade`**
- **`Tile Vertically`**
- **`Tile Horizontally`**
- **`Resize`**

While the default configuration is sufficient for efficient SAS use, you may want to open a few more windows for easy access and rearrange the windows within the main SAS window. For example, you may want to keep the My Favorite Folders window open, but minimized, and the windows arranged in a mosaic pattern so you can see all of them at once.

To accomplish this configuration

1 Open the My Favorite Folders by selecting

   View ▶ My Favorite Folders

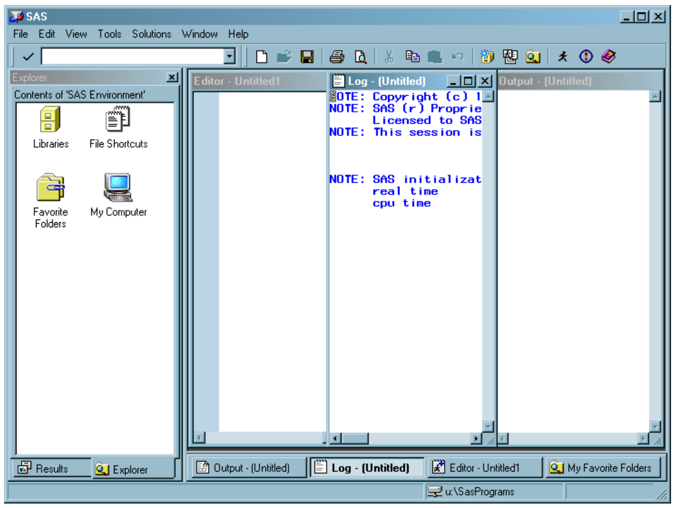
2 Select the minimize button in the window title bar for the My Favorite Folders window

3 Select

   Windows ▶ Tile Vertically

The following display shows the resulting main SAS window:

**Display 2.5** Customized SAS Session



```
OTE: Copyright (c) 1
NOTE: SAS (r) Proprie
      Licensed to SAS
NOTE: This session is


NOTE: SAS initializat
      real time
      cpu time
```

In addition, you can undock windows so that all windows can be positioned where you want them or you can minimize the docking view. For more information about the docking view, see “Using the Docking View” on page 35.

For a list of SAS commands used to control the appearance of the main SAS window, see “SAS Commands That Control the Main SAS Window” on page 316.

### Changing the Window Colors

Changing the color of window components is a shared responsibility of Windows and SAS. You change the color of most standard window parts by changing the properties of the Windows desktop. Several window element colors are controlled by SAS (such as the color of error message text in the Log).

To change a window component that is controlled by SAS do one of the following

- Enter **`sascolor`** in the command bar
- Select

  Tools ► Options ► Colors

Use the SASCOLOR window to choose the colors for specific elements.

Close and reopen any active windows for new color settings to take effect.

For more information, see the SAS Help and Documentation for the SASCOLOR window.

## Customizing Your Windowing Environment with System Options

Several SAS system options are available to control the default windowing environment within SAS. The most commonly used options are the following:

AWSDEF
: specifies the location and dimensions of the main SAS window when SAS initializes.

AWSTITLE
: specifies the text for the main SAS window title bar.

HELPREGISTER
: enables you to add Help to the main SAS window Help pull-down menu in order to access custom Help. See “HELPREGISTER System Option” on page 509.

ICON
: minimizes the SAS window when SAS initializes.

REGISTER
: enables you to add applications to the main SAS window **`Tools`** pull-down menu so that you can execute them by clicking their names.

SASINITIALFOLDER
: specifies the pathnames to set for the current folder and the default folder for the Open and Save As dialog boxes when SAS starts.

SPLASHLOC and NOSPLASH
: specifies the pathname or the dynamic link library name of the logo screen that is to appear at the start of a SAS session, or it specifies to suppress the logo screen.

USERICON
: specifies user-defined icons to be incorporated into SAS/AF applications.

WEBUI
: specifies to enable use of the pointer to select an object and a single click to invoke the object’s default action.

These system options can be specified in your SAS configuration file or in the SAS command when you start SAS from a command prompt. Some are also valid in an OPTIONS statement. For details about the syntax of these options and about where you can specify them, see “SAS System Options under Windows” on page 467. For a comprehensive list of these options, see “SAS System Options That Control the Main SAS Window” on page 315.

## Changing the Size and Placement of the Main SAS Window

The AWSDEF system option enables you to control the placement and size of the main SAS window when SAS initializes. If you want your SAS session always to occupy the upper-left quarter of your screen, specify the following AWSDEF option in your SAS configuration file:

```
-awsdef 0 0 50 50
```

For more information, see “AWSDEF System Option” on page 487.

## Changing the Title of Your SAS Session

By default, the main SAS title bar contains the text **`SAS`**. If you want a different title, you can use the AWSTITLE system option. For example, to set the title to **`My SAS Session`**, specify the following option in your SAS configuration file:

```
-awstitle "My SAS Session"
```

For more information, see “AWSTITLE System Option” on page 489.

## Adding Help to the Help Menu

The HELPREGISTER system option enables you to access customized help from the main SAS window Help menu. You can add up to 20 WinHelp (.hlp), HTML (.htm), or Microsoft HTML Help (.chm) files to the Help menu. HELPREGISTER system option arguments enable you to

- □ link to a topic within a Help file
- □ customize the text that appears in the Help menu
- □ customize the text that appears in the message line when you position the pointer over the Help menu item.

To add multiple Help files to the Help menu, use multiple HELPREGISTER system options either in your configuration file or at the command prompt when you start SAS. The following example adds the Help file My Help.htm to the Help menu:

```
sas -helpregister "My Help" c:\mysashelp\myHelp.htm html
```

For more information, see “HELPREGISTER System Option” on page 509.

## Minimizing Your SAS Session

The ICON system option causes SAS to be minimized at invocation. If you are running a batch job, you might want to use this system option to save space on your screen. For more information, see “ICON System Option” on page 513.

## Adding Applications to the Tools Menu

The REGISTER system option enables you to add names of applications to the **`Tools`** pull-down menu of the main SAS window. You can execute one of these applications by clicking its name. The REGISTER system option takes as arguments a menu name and an operating environment command or a path specification for an executable file. You can also specify a working folder.

The following is an example that adds a command to print the contents of the SAS folder:

```
-register "Contents of SAS"
          "dir c:\program files\sas"
```

When you click **`Contents of SAS`** in the **`Tools`** pull-down menu, the output of the Windows DIR command is displayed in a command prompt window.

The following is an example of adding an .EXE file to the menu along with a specification of a working folder of C:\EXDATA:

```
-register "Excel" "excel.exe" "c:\exdata"
```

This adds **`Excel`** to the menu. When you click **`Excel`**, the file EXCEL.EXE is invoked.

The REGISTER system option is valid only as an invocation option (that is, in a SAS configuration file or in the SAS invocation command). For more information, see "REGISTER System Option" on page 539.

## Setting the Initial Path For the Current Folder and the Paths Specified in the Open and Save Dialog Boxes

If you want to start SAS with a current folder other than the default current folder, use the SASINITIALFOLDER system option when you start SAS. The pathname that you specify in the SASINITALFOLDER option sets the initial current folder as well as the initial pathname for the Open and Save As dialog boxes.

You can specify the SASINITIALFOLDER option either on the command line when you start SAS or in a configuration file. For example, you might specify **`sas -sasinitialfolder "c:\mySasFiles"`** to start SAS.

For more information, see "SASINITIALFOLDER System Option" on page 546.

## Displaying a Custom Logo Screen during SAS Invocation

To display your own logo when SAS starts

1 Create the logo that you want to display and save it either as a Windows bitmap (which has a .bmp file extension), or compile it as a resource and build it into a dynamic link library.

2 When you invoke SAS, specify the -SPLASHLOC system option with the full pathname of the file that contains your bitmap. If the bitmap is in a DLL, you must specify the resource number as well. The default resource number is 1.

For example, if your logo is stored in C:\MYBMPS\SPLASH.BMP specify the following SPLASHLOC system option:

```
-splashloc c:\mybmps\splash.bmp
```

If your logo is stored in C:\MYDLLS\OPENING.DLL as resource 101, you specify the following SPLASHLOC system option:

```
-splashloc c:\mydlls\opening.dll 101
```

For more information, see "SPLASHLOC System Option" on page 558.

## Adding User-Defined Icons to SAS

The USERICON system option enables you to add your own icons to SAS. These icons can be used with SAS/AF and SAS/EIS applications. The syntax for the USERICON system option is as follows:

-USERICON *icon-resource-file number-of-icons*

The *icon-resource-file* argument specifies the full path to a dynamic link library (DLL) file that contains the user icons. The *number-of-icons* argument specifies the number of icons found in the resource file. For example, the following system option specifies that there are four icons located in an icon resource file named ICONS.DLL found in the C:\JUNK folder:

```
-usericon c:\junk\icons.dll 4
```

The DLL that is used as the icon resource file must be created using the Win32 Software Development Kit (and must therefore be 32-bit). For more information about how to build a resource file, refer to the documentation for the Microsoft Win32 Software Development Kit.

You can incorporate icons into your SAS/AF and SAS/EIS applications using a FRAME entry. For more information, see "USERICON System Option" on page 571 and refer to the SAS Help and Documentation for SAS/AF software and SAS/EIS software.

### Enabling Web Enhancements in SAS

If you have Microsoft Internet Explorer 5.0 (IE) or greater installed, the WEBUI system option enables some SAS windows, such as the SAS Explorer window, to work like an IE web page where pointing to an object selects the object and a single mouse-click invokes the default action.

To select a range of objects, press and hold down the SHIFT key, and point to the first and last objects in the group.

To select multiple items, press and hold down the CTRL key, and point to individual items in the group.

## Customizing the Toolbar

### Introduction to Customizing the Toolbar

SAS assigns several commonly used commands to the buttons for your convenience. You might find that the commands you use most often are different than the ones assigned to the toolbar by default. Or, you might want to create a toolbar to use with a specific application window. This section describes how to customize the toolbar settings.

### Using the Customize Tools Dialog Box

You customize all toolbar settings using the Customize tools dialog box. To open the Customize Tools dialog box, do one of the following:

- Enter TOOLEDIT in the command bar.
- Select

  Tools ▶ Customize

Use the **Toolbars** tab for general toolbar settings and the **Customize** tab to define tools on the toolbar.

### Setting General Toolbar Preferences

The **Toolbars** tab has settings to control the behavior and appearance of the toolbar. Tool options include:

**`General`**
specifies button appearance and Help options. These options include:

**`Large icons`**
specifies whether to use the set of large buttons on the toolbar. This is useful for high-resolution displays.

**`Show ScreenTips on toolbars`**
specifies whether to display a brief button description when you place the pointer over the button.

**`Toolbars`**
specifies whether to display the toolbar and command bar. These options include:

**`Application Toolbar`**
specifies whether to display the toolbar for the active application.

**`Command Bar`**
specifies to display the command bar and enable the options to use the command bar.

When **`Use AutoComplete`** is selected, SAS stores commands that were entered previously and completes the command once you start typing the command.

Select **`Sort commands by most recently used`** to display commands in the command bar list by the most recently entered command. If this setting is not selected, the commands are ordered by the most frequently used.

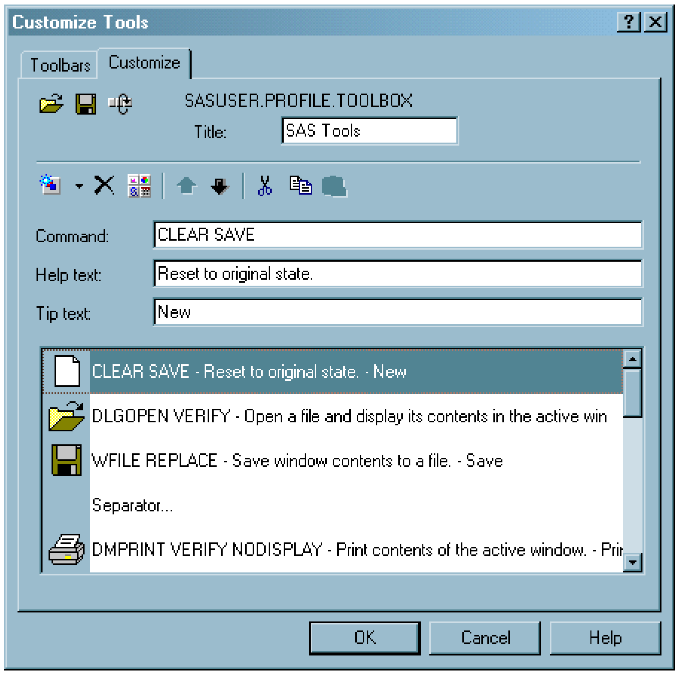
In **`Number of commands saved`**, type the number of commands to save in the command bar list. Valid values range between zero and 50. The default is 15.

When you have configured the **`Toolbars`** tab, either click **`Customize`** to complete your customization or click **`OK`** to close the dialog box.

## Customizing a Toolbar

The **`Customize`** tab, as shown in the following display, enables you to add, delete, and modify commands on the toolbar.

**Display 2.6** Customize Tab of the Customize Tools Dialog Box



SASUSER.PROFILE.TOOLBOX

Title: SAS Tools

Command: CLEAR SAVE

Help text: Reset to original state.

Tip text: New

CLEAR SAVE - Reset to original state. - New

DLGOPEN VERIFY - Open a file and display its contents in the active win

WFILE REPLACE - Save window contents to a file. - Save

Separator...

DMPRINT VERIFY NODISPLAY - Print contents of the active window. - Prir

The following explains each of the buttons (commands) and fields:

Open
: Opens a toolbar file.

Save
: Saves a toolbar file.

Restore
: Restores a toolbar to the default settings.

**`Title`**
: Displays the title text which appears in the title bar when the toolbar is undocked.

Add a tool
: Adds a tool or a separator space to the toolbar. This tool has two parts. When you click the left button a blank tool is added to the toolbar. When you click the down arrow, you can select to add either a **`Blank tool`** or **`Separator`**. Windows that define an action set (for example, Explorer) have a selection for **`Action`**.

Remove tool
: Deletes the selected tool from the toolbar list.

Change icon
: Opens the Bitmap Browser for you to select a new icon for the selected tool.

Move tool up
: Moves a tool up one position in the toolbar list.

Move tool down
: Moves a tool down one position in the toolbar list.

Cut
: Deletes the selected icon from the toolbar list and places it in the clipboard.

Copy

Places a copy of the selected icon in the clipboard.

Paste
: Copies an icon from the clipboard to the selected tool in the toolbar list.

**`Command`**
: displays the command for the tool that is selected in the toolbar list. You can modify the command in the **`Command`** box.

**`Help Text`**
: displays the Help text that appears in the status bar message area when the pointer is placed over the button in the toolbar. You modify the Help text in the **`Help Text`** box.

**`Tip Text`**
: displays the ScreenTip that appears under the button when the pointer is placed over the button in the toolbar. You modify the tip text in the **`Tip Text`** box.

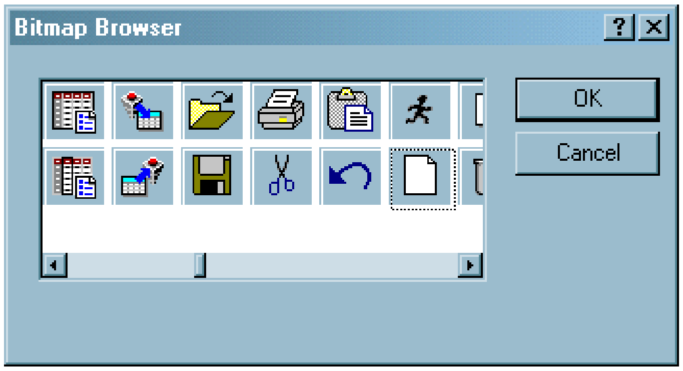
Toolbar list
: lists the buttons, commands, Help text and separators that are defined in the toolbar.

## Adding a Tool to the Toolbar

To add a tool to the toolbar:

1. Do one of the following:
   - Click the Add tool button to add a blank tool to the toolbar list. Enter a SAS command in the **`Command`** box.
   - For windows that have a set of predefined tools, such as the Explorer window or the My Favorite Folders window, click the Add tool down arrow and select **`Action`**. From the Add Action dialog box, select an action. This adds a new action to the toolbar. You can enter multiple commands separated by semicolons.
   - Click the Add tool down arrow and select **`Separator`** to add a separator to the toolbar list.
2. Click the Bitmap Browser button to select an icon for the tool. When you have selected an icon, click **`OK`**.
3. Type Help text in the **`Help Text`** box.
4. Type ScreenTip text in the **`Tip Text`** box.
5. Position the tool in the toolbar list by clicking the Move tool up and Move tool down buttons.
6. When you are finished, click Save. In the Save Tools dialog box, type the library, catalog, and toolbox name. Then click **`OK`**.

**Display 2.7** Bitmap Browser Dialog Box



## Removing a Tool from the Toolbar

To remove a tool from the toolbar:

1. Select the tool in the toolbar list that you want to remove.
2. Click Remove tool.
3. When you are finished, click Save.

## Customizing and Saving a Toolbar for Use with a Particular Application or Window

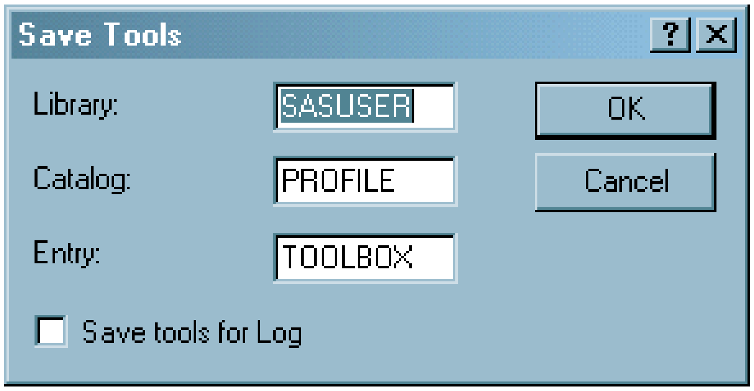
Before you add a command to a toolbar, ensure that the command is available from a pull-down menu. Buttons are enabled only if the command is available from a pull-down menu, with the exception of the Print and Copy commands, which are always enabled.

Use the following procedure to customize a toolbar to use with a particular application or window:

1. Click in the application or window to make it the active window.
2. Customize the toolbar by adding and removing tools as described in previous sections.
3. When you are finished customizing the toolbar, click Save. The Save Tools dialog box appears (as shown in Display 2.8 on page 71).
4. SAS completes the libref, catalog, and entry fields. Select the **`Save tools for`** ***`window`*** check box, where *window* is the active window, and then click **`OK`**.

   When you select the **`Save tools for`** ***`window`*** check box, the toolbar is associated with the particular application or window by using the same library, catalog, and entry name as the PMENU entry for the application or window. SAS first looks for toolbox entries in Sasuser.Profile before searching the application catalog.

**Display 2.8** Save Tools Dialog Box



Library: SASUSER
Catalog: PROFILE
Entry: TOOLBOX
☐ Save tools for Log

If you save the toolbar so that it is associated with a particular application, SAS automatically loads the tools when that application's window is active.

You can use the TOOLLOAD command to load your custom toolbar manually. For more information, see "TOOLLOAD Command" on page 352.

## Resetting the Tools to the Default Settings

To restore a toolbar to its default settings, click Restore Defaults. SAS asks you to confirm that you want to restore to the default tool settings. When you click **`Yes`**, the tools are reset to their original settings (the settings that were in place when SAS was installed).

If a SAS application defines a default toolbar for its application window, clicking the Restore Defaults button restores the settings for that toolbar.

## Examples of Useful Tools You Can Create

Suppose that you want to create a tool that opens the SAS Web page to the sample programs for Base SAS when the Enhanced Editor is the active window. You would perform the following steps:

1 Make the Enhanced Editor the active window.

2 In the Customize tab of the Customize Tools dialog box, click the **`Add tool`** toolbar button. This creates a template for a new tool in the list box.

3 In the **`Command`** field, type **`http://support.sas.com/techsup/sample/base_samples.html`**.

   In the **`Help Text`** field, type **`Sample programs for Base SAS on sas.com`**.

   In the **`Tip Text`** field, type **`sas.com sample programs`**.

4 Click the **`Change icon`** button. From the Bitmap Browser dialog box, select a bitmap that is appropriate for the sample programs on the SAS Web site and click **`OK`**.

5 Use the **`Move tool up`** and the **`Move tool down`** buttons to position the tool in the toolbar.

6 Click the **`Save the toolbar`** button to save the tool with your default tool configuration.

The following are some examples of other tools that you might find useful to create:

WEDIT; CLEAR; INCLUDE C:\SAS\MYPROGRAM.SAS
includes a program that you use often into the Enhanced Editor window for editing.

WEDIT; FILE C:\SAS\MYPROGRAM.SAS; CLEAR
: saves a SAS program after you finish editing it and clears the Enhanced Editor window.

WEDIT; CLEAR; INCLUDE C:\SAS\MYPROGRAM.SAS; SUBMIT
: includes and submits a SAS program that you use often.

WEDIT; CLEAR; INCLUDE C:\SAS\SIGNON.SAS; SUBMIT
: includes and submits a SAS program to sign on to a remote system. For example, to sign on to a remote MVS session, the SIGNON.SAS program might contain

```
options comamid=tcp remote=mytso;
libname remtdata 'mylib.mydata.monthly';
signon;
```

For more information about signing on to remote sessions, see the *SAS/CONNECT User's Guide*.

WEDIT; CLEAR; INCLUDE C:\SAS\DOWNLOAD.SAS; SUBMIT
: includes and submits a SAS program to download a data set from a remote session. Assuming that you have already signed on to the remote session, DOWNLOAD.SAS might contain:

```
proc download data=remtdata.june;
   /* where libname 'remtdata' is */
   /* already defined             */
run;
```

For more information about signing on to remote sessions, see the *SAS/CONNECT User's Guide*.

TOOLLOAD BAR SASUSER.PROFILE.MORTOOLS
: loads a different toolbar that contains another collection of tools.

## Accessing Online Help and Documentation

### Using Microsoft HTML Help

SAS Help and Documentation uses Microsoft HTML Help for easy navigation, indexing, and search capabilities. Microsoft Internet Explorer (IE) 5.00 and Microsoft HTML Help 1.3 or above are required. No action is required to configure SAS to use Microsoft HTML Help.

### Getting Help from the Command Bar

You can get Help for the active window and SAS language elements by using the HELP command in the command bar. The following table lists the HELP command arguments and the resulting display in the SAS Help and Documentation.

**Table 2.4** Types of Help Available Using the Command Bar

| Help Argument | SAS Help and Documentation Displays | Example |
|---|---|---|
| none | help for the active window | **`help`** |
| language element name and type | help on the specified language element | **`help libname statement`** |
| HELP | how to use the HELP command | **`help help`** |

## Getting Help in the Dialog Boxes

To access Help in a dialog box, click **`?`** at the top of the dialog box, then click the item you want information about. A pop-up window appears with a definition for the item. To close the pop-up window, click anywhere in the dialog box.

If a dialog box doesn't have the **`?`** button, look for a Help button or press F1.

## Getting Help for a SAS Product

To access help information about the SAS product associated with the currently active window, do one of the following:

- □ Click the Help button (the book with the question mark).
- □ Press the F1 function key.
- □ Select the **`Help`** menu and **`Using This Window`**. (For example, if you click the Help button and the active window is a SAS/GRAPH window, the SAS Help and Documentation displays help information about SAS/GRAPH software.)

Complete documentation for installed SAS products is available from the **`SAS Products`** entry in the SAS Help and Documentation table of contents.

## Getting Help from the Help Menu

The **`Help`** menu is always available within your SAS session. Here are descriptions of the help topics available from the **`Help`** menu:

**`Using this Window`**
Help information that is relevant to the active window. This is the same as clicking the Help button or pressing the F1 key.

**`SAS Help and Documentation`**
tutorials and sample programs to help you learn how to use SAS, comprehensive documentation for all products installed at your site, and information about contacting SAS for additional support.

**`Getting Started with SAS Software`**
opens a tutorial that will help you get started with SAS.

**`Learning SAS Programming`**
open the SAS Online Tutor, if it is installed, to help you develop your SAS programming skills. SAS Online Tutor is a separately licensed product.

**`SAS on the Web`**

provides links to useful areas on the SAS Institute web site, including technical support, frequently asked questions, sending feedback to SAS, and the SAS homepage.

**`About SAS System`**
opens the About SAS System dialog box which provides software levels for SAS and Windows, and your hardware information. You can also access SAS legal information and site information. The **`System Info`** button opens the Microsoft System Information window.

### Getting to SAS Institute (and Other Web Sites) from within SAS

SAS is configured to launch your local Web browser to view HTML files. You can invoke your Web browser several ways:

- □ Type a URL (uniform resource locator) in the command bar. SAS launches the browser that you specified in the Preferences dialog box Web tab.
- □ Type **`wbrowse`** in the command bar. This opens the browser to the SAS home page or another default URL that you specify in the Preferences dialog box Web tab. For more information, see "WBROWSE Command" on page 356.

Note that you can access web pages on the Internet (such as the SAS Institute home page) only if your workstation is connected to a network that allows such access.

## Accessibility Features in SAS under Windows

### Introduction to Accessibility Features in SAS under Windows

SAS under Windows includes the following accessibility and compatibility features that improve the usability of SAS for users with disabilities. These features are related to accessibility standards for electronic information technology adopted by the U.S. Government under Section 508 of the U.S. Rehabilitation Act of 1973, as amended.

- □ "Accessible Windows and Dialog Boxes" on page 74 lists the accessible SAS windows and dialog boxes.
- □ "The ACCESSIBILITY System Option" on page 76 explains how to use the standard user interface and the fully accessible interface.
- □ To enhance the readability of SAS windows, see "Enlarging Fonts" on page 78 and "Enlarging Icons" on page 78.
- □ You can use the keyboard, menus, and commands to access and modify the docking view and the SAS Explorer list view. See
  - □ "Resizing the Docking View in the Main SAS Window" on page 78
  - □ "Sorting Window List Views by a Specific Column" on page 78
  - □ "Resizing the Detail Columns of a List View" on page 79
- □ If your accessibility aid has difficulty reading menus, see "Improving Access to Menus" on page 79.

### Accessible Windows and Dialog Boxes

In addition to the main SAS window, the following table lists the SAS windows and dialog boxes that are compliant with Section 508 of the U.S. Rehabilitation Act of 1973:

**Table 2.5** Accessible SAS Windows and Dialog Boxes

| Window or Dialog Box | Related Window or Dialog Box |
|---|---|
| About SAS System | Legal Notices |
| | Siteinfo |
| Change Folder | none |
| Customize Tools, Customize tab | Bitmap Browser |
| | Open Tools |
| | Save Tools |
| DDE Triplet in Clipboard | none |
| Enhanced Editor | Find |
| | Replace |
| | Run Keyboard Macro |
| | Keyboard Macros |
| | Assign Keys |
| | Create Keyboard Macro |
| | Edit Keyboard Macro |
| | Import Keyboard Macros |
| | Export Keyboard Macros |
| | Add Abbreviation |
| | Enhanced Editor Options |
| | User Defined Keywords |
| | SAS Extensions |
| | Enhanced Editor Keys |
| | Assign Keys |
| Explorer | Find |
| | New Library |
| | Catalog Create |
| | Explorer Options |
| | Properties |
| | Catalog |
| Export Wizard | all subsequent dialog boxes |
| Find | none |
| Font | none |
| Import Wizard | all subsequent dialog boxes |
| Libraries | New Library |
| | Modify Library |
| Log | Log Options |
| My Favorite Folders | none |
| NOTEPAD | none |
| Open | none |
| Output | Output Options |

| Window or Dialog Box | Related Window or Dialog Box |
|---|---|
| Preferences | none |
| Print<br>(Windows and Universal Print dialog boxes) | Page Setup<br>Print Abort<br>Print Preview<br>Print Setup<br>(Windows and Universal Print dialog boxes) |
| Program Editor | none |
| Replace | none |
| Results | Results Properties<br>Rename<br>Save as Object<br>Templates and resulting dialog boxes |
| Results Viewer - SAS Output | none |
| Run | none |
| SAS System Options | none |
| Save | none |
| Send mail | none |
| Windows | none |

## The ACCESSIBILITY System Option

### Accessing the Standard or Fully Accessible User Interface

The ACCESSIBILITY system option enables you to specify either the standard user interface or the fully accessible user interface. The standard user interface enables accessibility aids to read components of most of the windows and dialog boxes that are listed in the previous section. The fully accessible user interface enables accessibility aids to read components of all of the windows that are specified in the previous section.

The fully accessible user interface adds buttons to these dialog boxes so that all commands and tabbed pages are accessible by using the keyboard:

- the Customize Tools dialog box Custom tab
- some SAS Properties dialog boxes.

You specify the ACCESSIBILITY system option either in your configuration file or at the command prompt when you start SAS. Valid values for ACCESSIBILITY are

standard
: specifies to use the preferred user interface that is not fully accessible. This is the default.

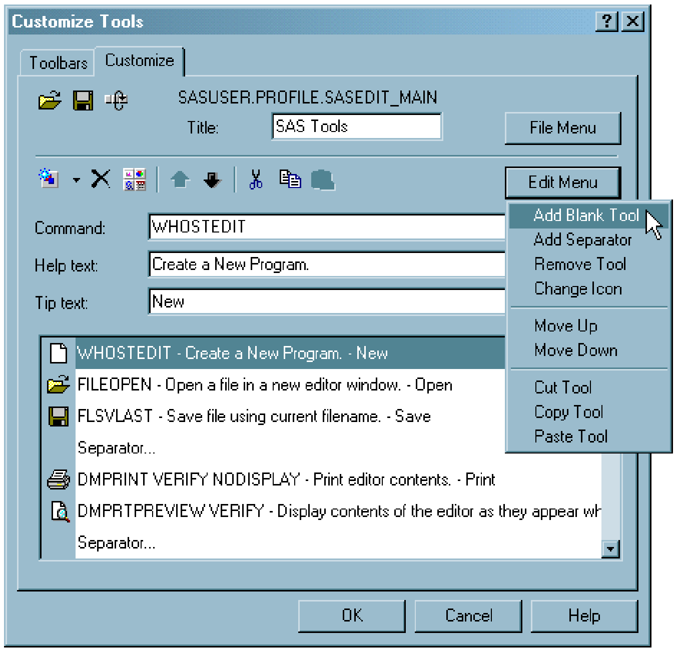
extended
: specifies to use the fully accessible user interface.

For more information, see "ACCESSIBILITY System Option" on page 482.

## Using the Accessible Customize Tools Dialog Box

The Customize Tools dialog box Customize tab provides command buttons for file and edit commands, such as Open a toolbar and Add tool. When you start SAS with the ACCESSIBILITY system option set to **`extended`**, SAS adds a **`File Menu`** button and an **`Edit Menu`** button to the Customize tab. These buttons enable you to use the keyboard to issue the commands that are available from the File and Edit menus. commands .

**Display 2.9** The Accessible Customize Tools Dialog Box



SASUSER.PROFILE.SASEDIT_MAIN

Title: SAS Tools

Command: WHOSTEDIT

Help text: Create a New Program.

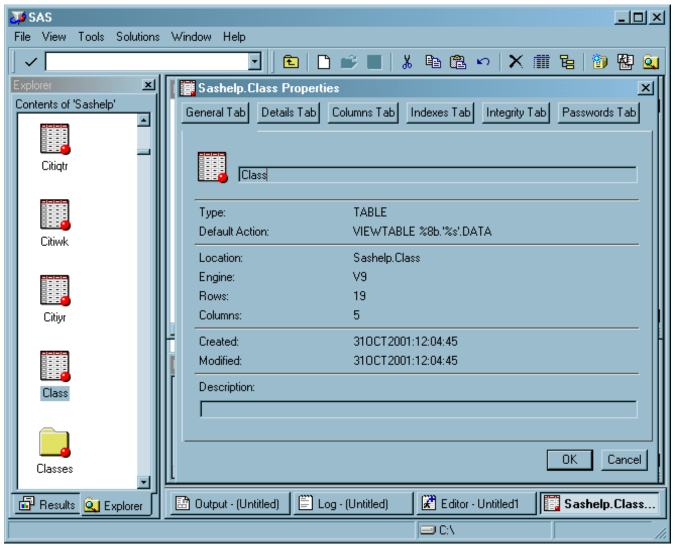
Tip text: New

```
WHOSTEDIT - Create a New Program. - New
FILEOPEN - Open a file in a new editor window. - Open
FLSVLAST - Save file using current filename. - Save
Separator...
DMPRINT VERIFY NODISPLAY - Print editor contents. - Print
DMPRTPREVIEW VERIFY - Display contents of the editor as they appear wh
Separator...
```

## Using Accessible Properties Dialog Boxes

When the ACCESSIBILITY system option is set to **`extended`**, the tabbed pages in some SAS Property dialog boxes are accessible as buttons. You can press Ctrl + Page Up and Ctrl + Page Down to navigate through the Properties dialog box.

**Display 2.10** An Accessible Properties Dialog Box



Class

```
Type:               TABLE
Default Action:     VIEWTABLE %8b.'%s'.DATA

Location:           Sashelp.Class
Engine:             V9
Rows:               19
Columns:            5

Created:            31OCT2001:12:04:45
Modified:           31OCT2001:12:04:45

Description:
```

## Enlarging Fonts

To make text easier to read, you can enlarge the font by using the Font dialog box and the SYSGUIFONT system option.

To open the Font dialog box, type **`dlgfont`** in the command bar or select

Tools ▶ Options ▶ Fonts

Specify the font size in the **`Size`** box and click **`OK`**.

To enlarge fonts in button text and the descriptive text, such as the words **`Contents of SAS Environment`** in the SAS Explorer window, use the SYSGUIFONT system option either in the configuration file or at the command prompt when you start SAS. The following SAS command uses the Times New Roman font with a font size of 16:

```
sas -sysguifont "times new roman" 16
```

*Note:* You might need to maximize the SAS window in order to allow space for large fonts to be readable. △

## Enlarging Icons

To enlarge icons, do the following

1 Select

   Tools ▶ Customize

2 In the **`Toolbars`** tab, select **`Large icons`**.

## Resizing the Docking View in the Main SAS Window

To resize the docking view by using the keyboard, press Alt + W + S or select

Window ▶ Size Docking View

Alternatively, you can type **`wdockviewresize`** in the command bar.

Use the following keyboard sequences to resize the docking view:

- Press the right arrow ( -> )or left arrow ( <- )to move the split bar slightly to the right or to the left.
- Press Ctrl + right arrow ( -> )or Ctrl + left arrow ( <- )to move the split bar a larger amount to the right or to the left.
- Press Home to move the split bar completely to the left.
- Press End to move the split bar completely to the right.
- Press Return to accept the docking view size.
- Press Esc to cancel all docking view resizing.

For more information, see "WDOCKVIEWRESIZE Command" on page 359.

## Sorting Window List Views by a Specific Column

When a window contains a list view (such as SAS Explorer), you can sort the list by the detail information that is associated with the list by using the Sort Columns dialog box:

1 If the window is not already a list, select

View ▶ List

2 Type **`dlgcolumnsort`** in the command bar.

3 Select a column to sort by.

4 Click Sort.

5 Click Close.

For more information, see “DLGCOLUMNSORT Command” on page 331.

## Resizing the Detail Columns of a List View

You can resize the details columns in a List view by using the Column Settings dialog box. In order to do this, you must display the details.

1 If the window is not already a list, select

View ▶ List

2 If the details are not already displayed, select

View ▶ Details

3 Type **`dlgcolumnsize`** in the command bar.

4 Select a column to resize.

5 Press TAB and type the pixel width that you want for the column.

6 Do one of the following:

- Click Apply to view the changes without closing the dialog box. Once you have finished making your changes, press ENTER to close the dialog box.
- Click OK to apply the changes and close the dialog box.

The columns in the SAS System Options window cannot be resized.
For more information, see “DLGCOLUMNSIZE Command” on page 331.

## Improving Access to Menus

Applications that provide custom icons to menu items may impede the readability of menus by accessibility aids. When you specify the NOMENUICONS system option, SAS does not include menu icons for all windows. When menu icons are excluded, menus adhere to the standard Windows menu structure.

You can specify the NOMENUICONS system option in the configuration file, at SAS invocation, or by using the OPTIONS statement:

-MENUICONS | -NOMENUICONS

MENUICONS | NOMENUICONS

MENUICONS
specifies to include icons in menus

NOMENUICONS
specifies not to include icons in menus.

For example, **`sas -nomenuicons`** starts SAS and excludes menu icons. In an OPTIONS statement, you specify

```
options nomenuicons;
```

CHAPTER
3

# Using the SAS Editors

*Using the Enhanced Editor* **82**
- *Enhanced Editor Features* **82**
- *Using the Enhanced Editor Window* **82**
  - *Overview of the Enhanced Editor Window* **82**
  - *Opening Files* **83**
  - *Saving Files* **85**
  - *Using Multiple Views of the Same File* **85**
  - *Scrolling and Line Number Commands* **86**
  - *Moving the Insertion Point* **87**
  - *Selecting and Editing Text* **88**
  - *Dragging Text* **89**
  - *Finding and Replacing Text* **89**
  - *Checking for Coding Errors* **91**
  - *Using Automatic Indenting and Tabs* **92**
  - *Bookmarking Lines* **92**
  - *Using Abbreviations* **93**
  - *Submitting Your Program* **93**
  - *Obtaining the Filename and Full Path of Submitted Programs or Catalog Entries* **95**
  - *Using Keyboard Shortcuts* **97**
  - *Using Keyboard Macros* **97**
  - *Using Collapsible Code Sections* **100**
- *Creating Your Own Keywords* **101**
- *Associating File Extensions with File Types* **102**
- *Setting Enhanced Editor Options* **102**
  - *Opening the Enhanced Editor Options Window* **102**
  - *General Editor Options* **103**
  - *Appearance Options* **104**
  - *Setting Appearance Options* **105**
  - *Using Schemes* **106**
- *Using Keyboard Shortcuts to Customize the Enhanced Editor* **106**
  - *Assigning Keyboard Shortcuts* **106**
  - *Deleting Keyboard Shortcuts* **107**
  - *Resetting Keyboard Shortcuts to the Enhanced Editor Defaults* **107**
- *Enabling and Disabling the Enhanced Editor* **107**

*Using the Program Editor* **108**
- *Switching from the Enhanced Editor to the Program Editor* **108**
- *Opening Files* **108**
- *Using Line Numbers* **109**
- *Moving the Insertion Point* **109**
- *Using Tabs* **110**
- *Understanding Line Breaks* **110**

*Selecting Text* 110
*Deleting Text* 111
*Finding and Replacing Text* 111
*Dragging and Dropping Text* 112
*Drag Scrolling* 113
*Using Rich Text Format Text* 113
*Saving Files* 114
*Saving Program Editor Files Using Autosave* 114
*Understanding Unique Features of the Editor* 114

## Using the Enhanced Editor

### Enhanced Editor Features

While retaining some familiar Program Editor features, the Enhanced Editor enables you to

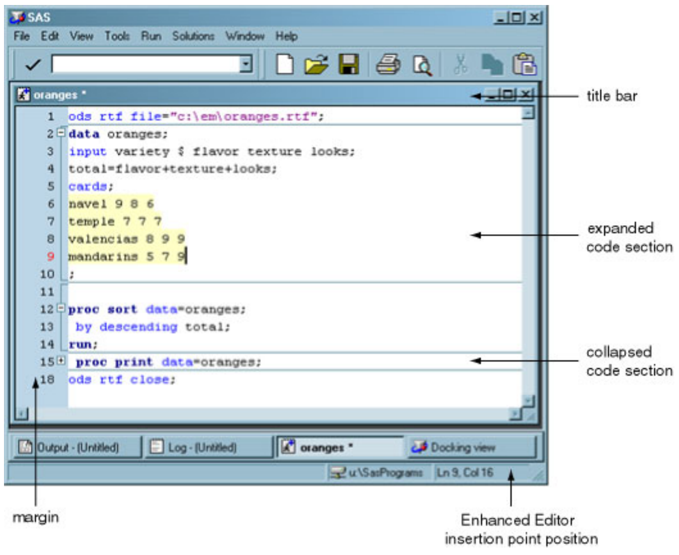
- use color-coding to identify SAS and SCL program elements as well as HTML and XML document elements. Color-coding settings can be saved in a color scheme.
- create and format your own keywords.
- automatically indent the next line when you press ENTER.
- view the high-level flow of your SAS program or see each detailed statement by expanding or contracting sections of SAS procedures, DATA steps, and macros.
- create macros that record and play back program editing commands by using the keyboard macro recorder.
- create shortcuts for typing in text using abbreviations.
- bookmark lines of code for easy access to different sections of your program or document.
- customize keyboard shortcuts for most Enhanced Editor commands.
- open multiple views of a files.
- access Help for the SAS language by placing the insertion point within the language element name and pressing F1.

### Using the Enhanced Editor Window

#### Overview of the Enhanced Editor Window

The parts of the Enhanced Editor window are shown in the following display:

**Display 3.1** Enhanced Editor Window



```
ods rtf file="c:\em\oranges.rtf";
data oranges;
input variety $ flavor texture looks;
total=flavor+texture+looks;
cards;
navel 9 8 6
temple 7 7 7
valencias 8 9 9
mandarins 5 7 9
;

proc sort data=oranges;
 by descending total;
run;
 proc print data=oranges;
ods rtf close;
```

title bar
: The title bar contains the Enhanced Editor icon and the name of the file. If the file is new, the filename is **`Editor Untitled`***x*, where *x* is a window number. An asterisk ( * ) in the title bar indicates that any changes to the file have not been saved.

expanded code section
: An expanded code section displays all of the code within the code section. It is indicated in the margin by the minus sign ( - ).

collapsed code section
: A collapsed code section displays only the signature line of code (the line of code that contains the keyword). It is indicated in the margin by the plus sign ( + ).

Enhanced Editor insertion point position
: Part of the main SAS window, the Enhanced Editor insertion point position displays the insertion point line and column position.

margin
: You use the margin on the left side of the Enhanced Editor window to

  - select one or more lines of text
  - expand and collapse code sections
  - display line numbers, code section brackets, and bookmarks.

You can move between Enhanced Editor windows by

- selecting an Enhanced Editor window
- typing **`wnextedit`** or **`wpgm`** in the command bar.

## Opening Files

The following table shows different methods of opening files in the Enhanced Editor.

**Table 3.1** Opening Files

| To do this... | Follow these instructions... |
|---|---|
| Open a new file | Do one of the following:<br>□ Type **wedit** in the command bar<br>□ Select<br>View ▶ Enhanced Editor<br>□ Click the **New Program** toolbar button<br>□ Select<br>File ▶ New Program |
| Open an existing file by using the Open dialog box | **1** Open the Open dialog box by using one of the following:<br>□ Click the **Open Program** toolbar button<br>□ Select<br>File ▶ Open Program<br>□ Type **fileopen** in the command bar.*<br>**2** Select the file.<br>**3** Click **Open**. |
| Open multiple individual files by using the Open dialog box | **1** Hold down the Ctrl key.<br>**2** Select the files.<br>**3** Click **Open**. |
| Open an existing file, bypassing the Open dialog box | Do one of the following:<br>□ Type **wedit** "*filename*" in the command bar.<br>□ Type **fileopen** "*filename*" in the command bar.* |
| Open multiple views of an opened file | **1** Make the file the active window.<br>**2** Select<br>Window ▶ New Window*<br>When you open multiple views of a file, changes that you make in any view of the file are made simultaneously in all views. |
| Append a file to an opened file | **1** Make the file that is to be appended the active window.<br>**2** Select<br>File ▶ Append.<br>**3** Select a folder and the file to append.<br>**4** Click **Open**. |

* The Enhanced Editor must be the active window.

*Note:* To change the default directory for the Open dialog box, either start SAS using the SASINITIALFOLDER system option or change the current working directory.

For more information, see “SASINITIALFOLDER System Option” on page 546 and “Changing the SAS Current Folder” on page 37. △

## Saving Files

An asterisk ( * ) that appears in an Enhanced Editor window title bar indicates that the editor contains text that has not been saved to disk. Enhanced Editor windows that display the name **`Editor Untitled`*`x`*** in the title bar are new files that have never been saved. The *x* indicates a window number.

To save the contents of the Enhanced Editor window, click the **`Save`** toolbar button (the diskette). If the file is to be saved for the first time, the Save As dialog box opens for you to name the file.

To save a file with a new name

1 Select

 File ▶ Save As

2 Select a folder in the **`Save in`** field.

3 Type a filename in the **`File name`** field.

4 Select a file type from the **`Save as type`** field.

5 Click **`OK`**.

*Note:* To change the default directory for the Save dialog box, either start SAS using the SASINITIALFOLDER system option or change the current working directory. For more information, see “SASINITIALFOLDER System Option” on page 546 and “Changing the SAS Current Folder” on page 37. △

If SAS ends unexpectedly, you can recover the contents of Enhanced Editor windows if the autosave feature is selected in the Preferences dialog box. The Enhanced Editor autosave files are saved in the operating environment Application Data folder with the filename **`Autosave of `*`filename`*`.$AS`**, where *filename* is the name of the file. Under Windows NT, the pathname for the Application Data folder is c:\winnt\Profiles\*username*\Application Data\SAS\EnhancedEditor. Under all other Windows operating environments, the pathname for the Application Data folder is c:\Documents and Settings*username*\Application Data\SAS\EnhancedEditor. For example, the path to the autosave file for MYPROGRAM.SAS in folder C:\TEMP would be

```
C:\Documents and Settings\myuserid\Application Data\SAS\EnhancedEditor
\Autosave of myprogram.$AS
```

SAS deletes autosave files when a file is saved, when the Enhanced Editor window closes, or when a SAS session ends normally. If you have renamed a file and SAS ends abnormally, you will find the autosaved file under the original filename.

For information about setting the autosave feature, see “Edit Preferences” on page 59 and “WAUTOSAVE Command” on page 355.

## Using Multiple Views of the Same File

You can see different parts of the same file simultaneously by opening multiple views of the same file. While you are working with multiple views, you are working with only one file, not multiple copies of the same file.

To open multiple views of the same file:

1 Make the file the active window.

**2** Select

Window ▶ New Window

The filename in the title bar is appended with a colon and a view number. For example, **`myfile.sas:1`** and **`myfile.sas:2`**.

Changes that you make to a file in one view, such as changing text or bookmarking a line, occur in all views simultaneously. Actions such as scroll bar movement, text selection, and expanding or contracting a section of code occur only in the active window.

## Scrolling and Line Number Commands

The Enhanced Editor supports a limited number of scrolling and line number commands that you may be familiar with from using the Program Editor. All Enhanced Editor commands can be typed only from the command bar.

The Enhanced Editor supports the following scrolling commands:

**Table 3.2** Scrolling Commands

| Command | Description |
|---|---|
| UP | Move one page toward the beginning of the file. |
| DOWN | Move one page toward the end of the file. |
| LEFT | Move one page to the left. |
| RIGHT | Move one page to the right. |

You can display line numbers either by selecting **`Show line numbers`** in the Enhanced Editor Options dialog box or by typing **`nums`** in the command bar. To suppress line numbers, either deselect **`Show line numbers`** or type **`nums`** again in the command bar.

All line number commands begin with a colon ( : ). A space is not required between the command and the number. The following line number commands are supported:

**Table 3.3** Line Number Commands

| Command | Description | Default Value of *n* | Maximum Value Allowed | Example |
|---|---|---|---|---|
| :I*n* | Insert *n* lines after the current line. | 1 | 9999 | :I4<br>Inserts 4 lines after the current line |
| :IA*n* | Insert *n* lines after the current line. | 1 | 9999 | :IA4<br>Inserts 4 lines after the current line |
| :IB*n* | Insert *n* lines before the current line. | 1 | 9999 | :IB2<br>Inserts 2 lines before the current line |

| Command | Description | Default Value of *n* | Maximum Value Allowed | Example |
|---|---|---|---|---|
| :D*n* | Delete *n* lines starting at the current line. | 1 | 9999 | :D3<br>Deletes 3 lines, starting with the current line. |
| :R*n* *m* | Repeat the block of *m* lines, starting with the current line, *n* times. A space is required between *n* and *m*. | 1 | 9999 | :R1 6<br>Repeats 6 lines, starting with the current line, one time. |

## Moving the Insertion Point

The Enhanced Editor accepts numerous key sequences for moving the insertion point, as shown in the following table. The key sequence Ctrl + G opens the Go To Line dialog box. All other key sequences move the insertion point as defined.

**Table 3.4** Key Sequences for Moving the Insertion Point

| Use this key sequence... | To move the insertion point.... |
|---|---|
| Up arrow | up one line |
| Down arrow | down one line |
| Left arrow | left by one character |
| Right arrow | right by one character |
| Page Down | down a page |
| Page Up | up a page |
| Home | to the beginning of the current line |
| Ctrl + Home or<br>Ctrl + Page Up | to the beginning of the document |
| End | to the end of the current line |
| Ctrl + End or<br>Ctrl + Page Down | to the end of the document |
| Ctrl + up arrow | toward the top of the file while scrolling up |
| Ctrl + down arrow | toward the bottom of the file while scrolling down |
| Ctrl + right arrow | to the start of the next word |
| Ctrl + left arrow | to the start of the previous word |
| Ctrl + ] | to the matching parenthesis or bracket |
| Ctrl + G | to a specific line number |
| Alt + up arrow | to the first visible line |
| Alt + down arrow | to the last visible line |
| Alt + right arrow | to the next case change or word boundary |

| **Use this key sequence...** | **To move the insertion point....** |
|---|---|
| Alt + left arrow | to the previous case change or word boundary |
| Shift + Tab | backward to the previous tab stop |

In addition to using key sequences, you can move the insertion point up by one page, down by one page, to the left by one page and to the right by one page by using the UP, DOWN, LEFT, and RIGHT commands in the command bar.

By default, when you click the mouse button past the end of a line, the insertion point is placed after the last character in a line.

To enable the Enhanced Editor to place the insertion point past the end of a line:

1 Select

Tools ► Options ► Enhanced Editor ► General

2 Select the **`Allow cursor movement past end of line`** check box.

3 Click **`OK`**.

## Selecting and Editing Text

Use the following mouse and keyboard shortcut actions to select and manipulate text.

**Table 3.5** Selecting Characters and Lines of Text

| **To select...** | **Do this...** |
|---|---|
| One or more lines of text using the margin | 1 Click and hold down the left mouse button in the margin on the first line of text that you want to select.<br>2 Still holding down the left mouse button, drag the mouse pointer within the margin to the last line that you want to select.<br>3 Release the left mouse button. |
| Single or multiple characters, or whole lines of text | 1 Click and hold down the left mouse button before the first character that you want to select.<br>2 Still holding down the left mouse button, drag the mouse pointer to the last character that you want to select.<br>3 Release the left mouse button. |

The following keyboard shortcuts are also available to select text.

**Table 3.6** Keyboard Shortcuts to Select Text

| **To...** | **Do this...** |
|---|---|
| Extend a selection in a particular direction | Press the Shift key and then press a directional arrow. |
| Extend a selection one character at a time | Press Shift + left arrow or right arrow |
| Unmark selected text | Press any directional key |

| To... | Do this... |
|---|---|
| Copy selected text | Press Ctrl + C or select Edit ▶ Copy |
| Cut selected text | Press Ctrl + X or select Edit ▶ Cut |
| Paste from the clipboard | Press Ctrl + V or select Edit ▶ Paste |
| Move selected text | **1** Place the mouse pointer over the selected text. **2** Click and hold down the left mouse button. The mouse pointer displays a vertical line. **3** Still holding down the left mouse button, drag the selected text and place the vertical line at the position where you want to place the text. **4** Release the left mouse button. |

Text that you select appears in reverse video.

For a complete list of selection keyboard shortcuts, see the Selection category in "Keyboard Shortcuts within the Enhanced Editor" on page 618.

*Note:* In addition to using commands from the Edit menu, you can use editing commands that are available from the pop-up menu when you click the right mouse button in the Enhanced Editor window. △

## Dragging Text

To move or copy text

1 Select the text, place the pointer over the selected text, and hold down the left mouse button.
2 To move the text
   a Drag the text to the location.
   b Release the left mouse button.
3 To copy the text
   a Press the Ctrl key.
   b Drag the text to the desired location.
   c Release the left mouse button.

To disable drag and drop editing

1 Select

   Tools ▶ Options ▶ Enhanced Editor ▶ General

2 Clear the **`Drag and drop text editing`** check box and click **`OK`**.

## Finding and Replacing Text

To find text

1 Open the Find dialog box by selecting

Edit ► Find

2 Supply the following information:

**`Find text`**
Type a text string to find. The initial value of this field is the last text string that was used in a search.

**`Find in`**
Click the **`Find in`** box to specify whether to search in the code only, in the comments only, or in both the code and comments.

**`Direction`**
Select either the **`Up`** or the **`Down`** option. **`Up`** specifies to search from the insertion point position toward the beginning of the file. **`Down`** specifies to search from the insertion point position toward the bottom of the file.

**`Match whole word only`**
Select the check box to specify that a match of the text must be a whole word and not part of a word.

**`Match case`**
Select the check box to specify that upper- and lowercase characters must match exactly.

**`Regular expression search`**
Select the check box to specify that the text string is a regular expression. A regular expression uses special characters as wildcards to search for a string or substring. For a selection of special characters that you can use in regular expressions, click the arrow that is located to the right of the **`Find text`** field.

3 Click **`Find Nex`**t.

To find and replace text

1 To search only within a subset of text, select the text.

2 Open the Replace dialog box by selecting

Edit ► Replace

3 Supply the following information:

**`Find text`**
Type a text string to find and replace. The initial value of this field is the last text string that was used in a search.

**`Replace with`**
Type the replacement string.

**`Find in`**
Click the **`Find in`** box to specify whether to search in the code only, in the comments only, or in both the code and comments.

**`Direction`**
Select either the **`Up`** or the **`Down`** option. **`Up`** specifies to search from the insertion point position toward the beginning of the file. **`Down`** specifies to search from the insertion point position toward the bottom of the file.

**`Match whole word only`**
Select this check box to specify that any match of the text must be a whole word and not part of a word.

**`Match case`**
Select this check box to specify that upper- and lowercase characters must match exactly.

**`Regular expressions`**
Select this check box to specify that the text string includes a regular expression. A regular expression uses special characters as wildcards to search for a string or substring. For a selection of special characters that you can use in regular expressions, click the right arrow that is located to the right of the **`Find text`** field.

4 Click **`Find Next`**.

5 If the text is found, click one of the following:

- □ **`Replace`** to replace this single occurrence of the text with the replacement string.
- □ **`Replace All`** to replace all occurrences of the text in the file with the replacement string.
- □ **`Replace in Selection`** to replace all occurrences of the text that is within selected text with the replacement string.

## Checking for Coding Errors

To assist you in finding coding errors, the Enhanced Editor

- □ color-codes program elements, quoted strings, and comments.
- □ searches for ending brackets or parentheses when you press Ctrl + ].
- □ searchs for matching DO-END pairs when you press Alt + [.

See the following table for suggestions about finding coding errors.

**Table 3.7** Hints about Finding Coding Errors

| To help you find... | Do this... |
|---|---|
| Undefined or misspelled keywords | In the Enhanced Editor Options dialog box Appearance tab, set the file elements **`Defined keyword`**, **`User defined keyword`**, and the **`Undefined keyword`** to unique color combinations.<br><br>When SAS recognizes a keyword, the keyword changes to the defined colors. You'll be able to easily spot undefined keywords by looking for the colors that you selected for undefined keywords. |
| Unmatched quoted strings | Look for one or more lines of the program that are the same color.<br><br>Text following a quotation mark remains the same color until the string is closed with a matching quotation mark. |
| Unmatched comments | Look for one or more lines of the program that are the same color.<br><br>Text that follows an open comment symbol ( /* )remains the same color until the comment is closed with a closing comment symbol ( */). |

| To help you find... | Do this... |
|---|---|
| Matching DO-END pairs | Place the cursor within a DO-END block and press Alt + [. The cursor moves first to the DO keyword. If one of the keywords is not found, the cursor remains as positioned. When both of the keywords exist, pressing Alt + [ moves the cursor between the DO-END keywords. |
| Matching parentheses or brackets | Place the cursor on either side of the parenthesis or bracket. Press Ctrl + ]. The cursor moves to the matching parentheses or bracket. If one is not found, the cursor remains as positioned. |
| Missing semi-colons ( ; ) | Look for keywords that appear in normal text. |

For a list of the components that you can color-code, open the Enhanced Editor Options dialog box. Select the Appearance tab. The components are listed in the **`File elements`** box. For more information about defining colors for program components, see “Setting Appearance Options” on page 105.

## Using Automatic Indenting and Tabs

When you press ENTER, you automatically indent the next line by the amount of space that the current line is indented. If you prefer not to use automatic indention:

1 Select

Tools ▶ Options ▶ Enhanced Editor ▶ General

2 In the **`Indentation`** box, select **`None`**.

In addition to automatic indenting, you can indent by using the TAB key. Pressing the TAB key moves the insertion point and any text to the right of the insertion point by the amount of space that you specified in the **`Tab size`** field of the Enhanced Editor Options dialog box.

Tab characters can be replaced by spaces either when you press the TAB key or when you open a file. To insert spaces instead of tab characters when you press the TAB key, select the **`Insert spaces for tabs`** check box. To replace tab characters with spaces when you open a file, select the **`Replace tabs with spaces on file open`** check box.

*Note:* Changing the tab size modifies tab settings to the new value in *all* Enhanced Editor windows. △

## Bookmarking Lines

When you bookmark a line, you create a line marker that is used to easily access that line. A vertical rectangle in the margin indicates that the line is bookmarked. Table 3.8 on page 92 shows the keyboard shortcuts that you can use with bookmarking.

**Table 3.8** Keyboard Shortcuts for Bookmarking Lines

| **To...** | **Press...** |
|---|---|
| Bookmark a line | Crtl + F2 on an unmarked line |
| Unmark a line | Ctrl + F2 on a marked line |

| To... | Press... |
|---|---|
| Go to the next bookmark | F2 |
| Go to the previous bookmark | Shift + F2 |

## Using Abbreviations

You can define a character string so that when you type it and then press the TAB key or the ENTER key, the string expands to a longer character string. For example, you could define the abbreviation *myv6sasfiles*, which would expand to *'c:\winnt\profiles\myid\personal\mysasfiles\v6';*. Abbreviations are actually macros that insert one or more lines of text.

To create an abbreviation

1 Press Ctrl + Shift + A or select

Tools ▶ Add Abbreviation

2 In the **Abbreviation** field, type the name of the abbreviation.

3 In the **Text to insert for abbreviation** field, type the text that the abbreviation will expand into.

4 Click **OK**.

To use an abbreviation, type the abbreviation. When an abbreviation is recognized, a tooltip displays the expanded text. Press the TAB key or the ENTER key to accept the abbreviation.

To modify an abbreviation

1 Press Ctrl + Shift + M or select

Tools ▶ Keyboard Macros ▶ Macros

2 Select the abbreviation from the list of current macros.

3 Click **Edit**.

4 Select the string in the **Keyboard Macro Contents** field.

5 Click **Modify**.

6 Type your modification in the Insert String dialog box and click **OK**.

7 Click **OK** in the Edit Keyboard Macros dialog box.

8 Click **Close** in the Macro dialog box.

To delete an abbreviation

1 Press Ctrl + Shift + M or select

Tools ▶ Keyboard Macros ▶ Macros

2 Select the abbreviation from the list of **Current Macros**.

3 Click **Delete**.

4 Click **Close**.

## Submitting Your Program

You can submit either a complete program or a specified number of lines of your program, beginning with the first line.

| **To submit a complete program...** | **Do this...** |
|---|---|
| When you open the program in the editor | Select the **`Submit`** check box in the Open dialog box. |
| From the Enhanced Editor | Do one of the following:<br>□ Click the Submit toolbar button<br>□ Press F3 or F8<br>□ Select<br>Run ▶ Submit<br>□ Type **`submit`** in the command bar. |

Use the SUBTOP command to submit either the first line or a specified number of lines of a program.

| **To submit ...** | **Do this...** |
|---|---|
| Only the top line of a program | Do one of the following:<br>□ Type **`subtop`** in the command bar.<br>□ Select<br>Run ▶ Submit Top Line |
| A specified number of lines, beginning with the first line | □ In the command bar, type **`subtop`** *n*, where **`n`** is the number of lines that you want to submit.<br>□ Select<br>Run ▶ Submit N Lines<br>and type the number of lines that you want to submit. |

For more information, see “SUBTOP Command” on page 350.

The Enhanced Editor Options dialog box provides the **`Clear text on submit`** setting for you to specify whether you want the contents of the Enhanced Editor window to be cleared after you submit your program. For more information, see “Setting Enhanced Editor Options” on page 102.

If a line of data in a DATALINES or CARDS statement is greater than 256 characters long, the data is read into one observation. You do not need to specify the LRECL option in the FILENAME statement as you do when you submit a long line of data using the Program Editor.

## Obtaining the Filename and Full Path of Submitted Programs or Catalog Entries

When you submit code or a catalog entry from the Enhanced Editor, the filename or catalog entry name and their respective folders are placed in these environment variables:

SAS_EXECFILEPATH
: contains the full path of the submitted program or catalog entry. The full path includes the folder and the filename.

SAS_EXECFILENAME
: contains only the name of the submitted program or the catalog entry name.

You can then extract the filename and full path for use in your SAS programs.

After the following DATA step runs and the data is sorted, the PRINT procedure includes the filename in the title and the full path in the footnote of the procedure output. The results are shown in Display 3.2 on page 96.

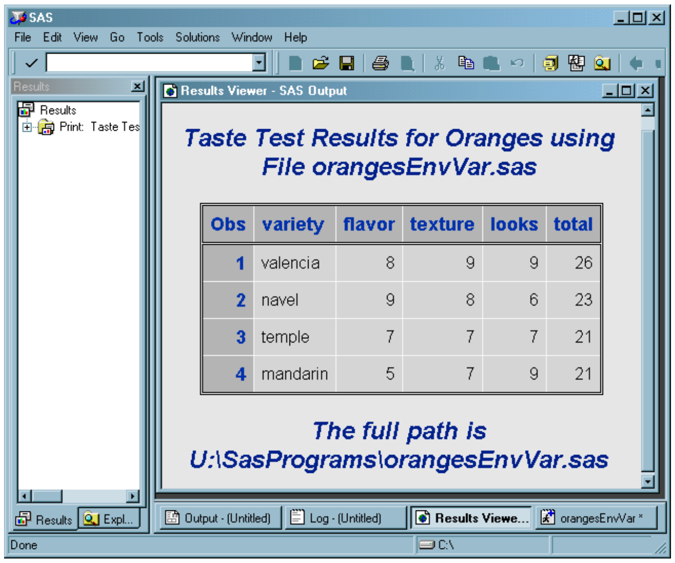
```
data oranges;
   input variety $ flavor texture looks;
   total=flavor+texture+looks;
cards;
navel 9 8 6
temple 7 7 7
valencias 8 9 9
mandarins 5 7 9
;

proc sort data=oranges;
   by descending total;
run;

proc print data=oranges;
title 'Taste Test Results for Oranges using File ' %sysget(SAS_EXECFILENAME);
footnote 'The full path is ' %sysget(SAS_EXECFILEPATH);
run;
```

The resulting output displays the filename in the title and the full path in the footnote:

**Display 3.2** Using an Environment Variable to Place a Filename in DATA Step Output



*Taste Test Results for Oranges using File orangesEnvVar.sas*

| Obs | variety | flavor | texture | looks | total |
|---|---|---|---|---|---|
| 1 | valencia | 8 | 9 | 9 | 26 |
| 2 | navel | 9 | 8 | 6 | 23 |
| 3 | temple | 7 | 7 | 7 | 21 |
| 4 | mandarin | 5 | 7 | 9 | 21 |

*The full path is U:\SasPrograms\orangesEnvVar.sas*

These environment variables are set only when code is submitted using the Enhanced Editor in the Windows environment. They are not set when you submit SCL code or when you submit code in a batch session.

However, when SAS is running in batch mode, you can obtain the full path (which includes the filename) by submitting **`%sysfunc(getoption(SYSIN))`**. The following macro can be used to obtain the full path in both a batch session and an interactive session by using the Enhanced Editor:

```
%let execpath=" ";
%macro setexecpath;
   %let execpath=%sysfunc(GetOption(SYSIN));
   %if %length(&execpath)=0
      %then %let execpath=%sysget(SAS_EXECFILEPATH);
%mend setexecpath;

%setexecpath;
%put &execpath;
```

You can also use the following %PUT macro statements to display the filename and full path in the SAS log:

```
%put Submitted file path is %sysget(SAS_EXECFILEPATH).;
%put Submitted file name is %sysget(SAS_EXECFILENAME).;
```

***CAUTION:***

**The values for these environment variables may be overwritten if subsequent programs are submitted while an interactive procedure is active.** The environment variables are set to the last submitted program. If a program starts an interactive procedure and subsequent programs are submitted while the interactive procedure is still active, the environment variables are set to the filename and full path of the latest submitted program. The filename and full path of the program that submitted the interactive procedure are no long available. △

## Using Keyboard Shortcuts

The Enhanced Editor provides extensive keyboard shortcuts for the Enhanced Editor. “Keyboard Shortcuts within the Enhanced Editor” on page 618provides a complete list of commands and their default keyboard shortcuts. The following table shows some of the more useful keyboard shortcuts.

**Table 3.9** Useful Keyboard Shortcuts

| To... | Press... |
|---|---|
| Get help for a SAS procedure | the mouse button and place the insertion point within the procedure name and press F1 |
| Add a new abbreviation | CTRL + Shift + A |
| Toggle expand current line | Alt + Num * |
| Collapse all code sections | Alt + Ctrl + Number pad - |
| Expand all code sections | Alt + Ctrl + Number pad + |
| Toggle marker on the current line | Ctrl + F2 |
| Go to the next marked line | F2 |
| Go to the previous marked line | Shift + F2 |
| Go to line | Ctrl + G |
| Go to the beginning of the file | Ctrl + Page Up |
| Go to the end of the file | Ctrl + Page Down |
| Convert selected text to uppercase | Ctrl + Shift + U |
| Convert selected text to lowercase | Ctrl + Shift + L |

For information about defining keyboard shortcuts, see “Using Keyboard Shortcuts to Customize the Enhanced Editor” on page 106.

## Using Keyboard Macros

A keyboard macro is a series of Enhanced Editor commands and instructions that you group together as a single command to accomplish a task automatically. Instead of manually performing a series of time-consuming, repetitive actions, you can create and run a single macro. You run a macro from the **Tools** menu or by using a keyboard shortcut. For information about defining a keyboard shortcut for using macros, see “Using Keyboard Shortcuts to Customize the Enhanced Editor” on page 106.

You can create a macro by recording it from the Enhanced Editor window:

1 Start recording either by pressing Alt + Shift + R or by selecting

Tools ▶ Keyboard Macros ▶ Record New Macro

2 Execute the sequence of actions to accomplish the task.

3 Stop recording either by pressing Alt + Shift + R or by selecting

Tools ▶ Keyboard Macros ▶ Stop Recording

4 If you want, define a keyboard shortcut to run the macro. For information, see “Using Keyboard Shortcuts to Customize the Enhanced Editor” on page 106.

An alternative way to create a macro is to add commands by using the Create Keyboard Macro dialog box. To do this

1 Open the Keyboard Macros dialog box by pressing Ctrl + Shift + M or by selecting

   Tools ▶ Keyboard Macros ▶ Macros

2 Click **Create** to open the Create Keyboard Macro dialog box.
3 Type the name of the macro in the **Keyboard macro name** field.
4 Type a description of the macro in the **Keyboard macro description** field.
5 The **Keyboard macro contents** box lists the commands in the order in which they will be executed.

   To add a command, select the command from the **Commands** box and click the Insert Selected Command button (the double arrow). Repeat this step until all commands are listed in the **Keyboard macro contents** box.

   You can reorder commands by selecting the command in the **Keyboard macro contents** box and clicking the UP or DOWN arrows.

   To delete a command, select the command and click **Delete**.
6 When all of the commands are present in the box and are ordered correctly, click **OK**.
7 In the Keyboard Macros dialog box, click **Close**.

To run a macro, use a keyboard shortcut or

1 Select

   Tools ▶ Keyboard Macros ▶ Run Macro

2 In the Run Macro dialog box, select a macro and click **Run**.

To edit a macro

1 Open the Keyboard Macro dialog box by pressing Ctrl + Shift + M or by selecting

   Tools ▶ Keyboard Macros ▶ Macros

2 Select a macro and click **Edit** to open the Edit Keyboard Macro dialog box.
3 To add a command, select a command from the **Commands** box and click the Insert Selected Command button.

   To modify a command, first ensure that you can modify the command by selecting the command from the **Keyboard macro contents** box. If a command can be modified, the **Modify** button is active. Click **Modify** to open a dialog box that will enable you to modify the command.

   To delete a command, select the command in the **Keyboard macro contents** box and click **Delete**.

   To reorder a command, select the command and click the up or down button.
4 When the commands are in the correct order, click **OK**.

To delete a macro

1 Press Ctrl + Shift + M or select

   Tools ▶ Keyboard Macros ▶ Macros

2 Select the macro and click **Delete**.
3 Click **Yes** in the Delete Macro dialog box.
4 Click Close.

This example lists the steps to create an RSUBMIT statement, an ENDRSUBMIT statement, a blank line between these statements, and tabs by the amount that you specified **Tabs size** field in the Enhanced Editor Options dialog box.

**1** Select

View ▶ Enhanced Editor

**2** Select

Tools ▶ Keyboard Macros ▶ Record New Macro

and click **OK** in the message box.

**3** In the Enhanced Editor window:

**a** Press ENTER.
**b** Type **rsubmit;**
**c** Press ENTER.
**d** Press ENTER.
**e** Type **endrsubmit;**
**f** Press the UP arrow.
**g** Press the TAB key.
**h** Select

Tools ▶ Keyboard Macros ▶ Stop Recording

The resulting macro contains the following commands:

```
Insert carriage return
Insert character ['r']
Insert character ['s']
Insert character ['u']
Insert character ['b']
Insert character ['m']
Insert character ['i']
Insert character ['t']
Insert character [';']
Insert carriage return
Insert carriage return
Insert character ['e']
Insert character ['n']
Insert character ['d']
Insert character ['r']
Insert character ['s']
Insert character ['u']
Insert character ['b']
Insert character ['m']
Insert character ['i']
Insert character ['t']
Insert character [';']
Move cursor up
Insert character ['|']
```

Keyboard macros can be shared by multiple users. You can import or export them to or from a folder by using the Keyboard Macros dialog box. To open the dialog box, select

Tools ▶ Keyboard Macros ▶ Macros

To import a keyboard macro, click **Import**, select a folder from the **Look in** box and a filename from the **File name** box. Then click **OK**.

To export a keyboard macro, click **Export**, select a folder from the **Look in** field, type a filename in the **File name** box, and click **OK**.

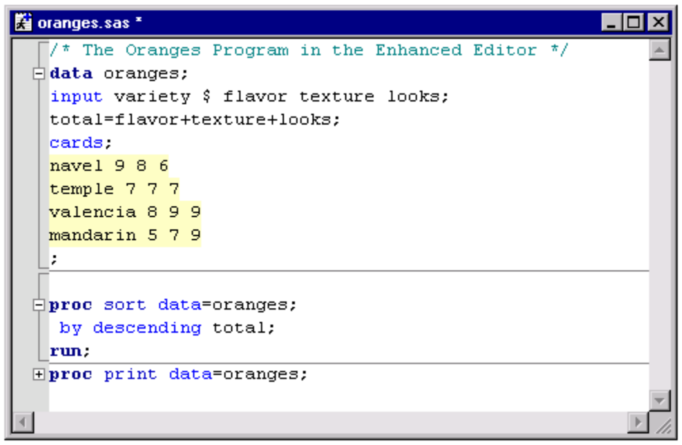
## Using Collapsible Code Sections

Collapsible code sections enable you to expand or collapse one or more sections of code. A section begins with a step keyword, a comment, or spaces above a section word or comment. A section ends with the next step keyword, a comment, or space above the next section word or comment. Step keywords include the DATA statement, the PROC statement, and the %MACRO statement. The *signature line* is the line in which the step keyword appears.

An expanded section is indicated by a minus sign in the margin next to the signature line. To collapse a section, click the minus sign.

A collapsed section is indicated by a plus sign in the margin, and the signature line is the only line of code that is displayed. To expand a section, click the plus sign.

**Display 3.3** The Enhanced Editor When Collapsible Code Segments Are Enabled



```
/* The Oranges Program in the Enhanced Editor */
data oranges;
input variety $ flavor texture looks;
total=flavor+texture+looks;
cards;
navel 9 8 6
temple 7 7 7
valencia 8 9 9
mandarin 5 7 9
;

proc sort data=oranges;
 by descending total;
run;
proc print data=oranges;
```

Brackets in the margin and a section line across the editor window mark the beginning and end of a section. If you do not want to see either the brackets or the section line, you can suppress them by using the Enhanced Editor Options dialog box.

To disable collapsible code sections, or brackets and lines, select

Tools ▶ Options ▶ Enhanced Editor ▶ General

and select the appropriate settings:

- Clear the **Collapsible code sections** check box to disable collapsible code sections.
- Clear the **Show section lines in text** check box to disable section lines in the editor.
- Clear the **Show section brackets in margin** check box to suppress section brackets in the margin.

The following rules apply when you select and edit collapsed segments:

- Selecting a line from the margin that includes a collapsed segment includes all text within the collapsed segment.
- Selecting a line of text by dragging the mouse over the text or by using the keyboard selects only that line of text.
- Selecting text from above a collapsed section and into a collapsed signature line copies text from the start of the selection to the end of the selection. The selection includes any hidden lines above the signature line up to the end of the selection.

- Selecting text on a signature line down into another section marks text from the beginning of the selection to the end of the selection and includes hidden text below the signature line.
- Any keystroke on a signature line expands the section.
- Pasting into a signature line or section expands the section.
- Typing something above a section that affects the section, such as comments or quotes, expands the section.
- Pressing ENTER at the beginning of a signature line adds code at the beginning of the section.
- Pressing ENTER at the end of a signature line adds code at the end of the section.
- Selecting **`Undo`** will not undo the collapse and expand commands.
- When you search for text and the text is found within a collapsed segment, the segment expands.
- When text within a collapsed segment is to be replaced, the segment expands.
- When text within a collapsed segment is to be replaced and you select **`Replace All`**, collapsed sections do not expand.

## Creating Your Own Keywords

In addition to the many SAS program file elements that you can format within the Enhanced Editor, you can create user-defined keywords for programming elements such as SAS procedure statements, variables, and user-defined formats. The appearance of user-defined keywords overrides the appearance of identical keywords that are defined in the SAS language.

Use the following rules for naming keywords:

1. The first letter must be a letter ( A, B, C, ... Z) or an underscore ( _ ).
2. Subsequent characters can be letters, numeric digits ( 0, 1, ... 9) or underscores.
3. You can use uppercase or lowercase letters. Keywords are not case-sensitive.
4. Blanks are not allowed in keyword names.

To create and format user-defined keywords, use the Enhanced Editor Options dialog box as follows:

1. Make an Enhanced Editor window the active window.
2. Select

   Tools ▶ Options ▶ Enhanced Editor

3. From the General tab, click **`User Defined Keyword`**.
4. Click **`Add`**.
5. Replace **`NewKeyword`** with your keyword.
6. Click **`OK`**.
7. Select the Appearance tab.
8. In the **`File elements`** box, select **`User defined keyword`**.
9. Select a font, foreground color, and background color.
10. Click **`OK`**.

To rename a keyword from the User Defined Keywords dialog box

1. Select the keyword.
2. Click **`Rename`**.
3. Rename the keyword.

4 Click **OK**.

To delete a keyword from the User Defined Keywords dialog box

1 Select the keyword.

2 Click **Delete**.

3 Click **OK**.

## Associating File Extensions with File Types

The following table lists the default file extensions for the types of files that are recognized by the Enhanced Editor.

**Table 3.10** Default File Extensions Recognized by the Enhanced Editor

| File Type | Default File Extension |
|---|---|
| SAS Program File | .sas |
| SCL Program File | .scl |
| HTML Document | .htm, .html, .xml |

To associate other file extensions with SAS and SCL programs, and HTML and XML documents, do the following:

1 Open the Enhanced Editor Options dialog box by selecting
Tools ▶ Options ▶ Enhanced Editor

2 Click the General tab.

3 Select a file type from the **File type** box.

4 Click **File Extensions** and then click **Add**.

5 Type the file extension and press ENTER.

To rename a file extension

1 Select the file extension.

2 Click **Rename**.

3 Type the new file extension and press ENTER.

To delete a file extension, select the file extension and click **Delete**.
To revert to the default file extensions, click **Default**.

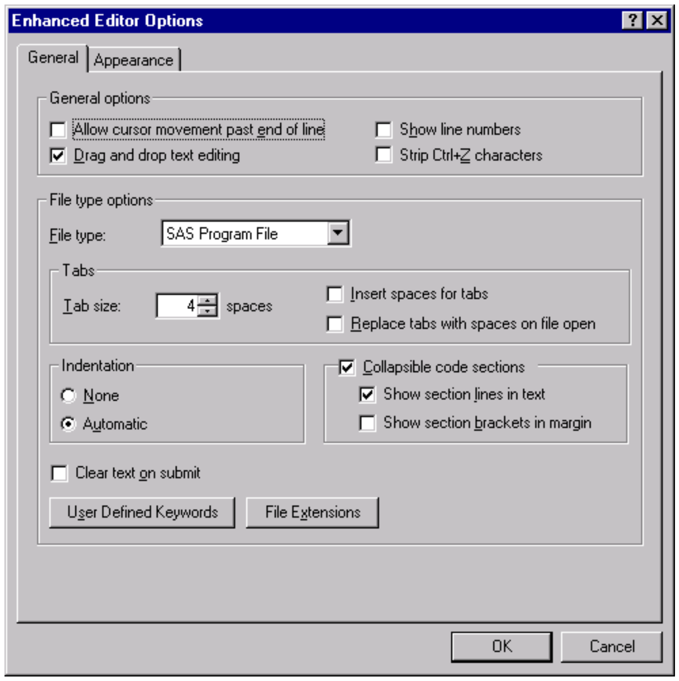
## Setting Enhanced Editor Options

### Opening the Enhanced Editor Options Window

To open the Enhanced Editor Options window from the menu, ensure that an Enhanced Editor window is the active window and select

Tools ▶ Options ▶ Enhanced Editor

**Display 3.4** Editor Options Dialog Box


Enhanced Editor Options

General | Appearance

General options

Allow cursor movement past end of line

[✓ Drag and drop text editing]

Show line numbers

Strip Ctrl+Z characters

File type options

File type: SAS Program File

Tabs

Tab size: 4 spaces

Insert spaces for tabs

Replace tabs with spaces on file open

Indentation

None [✓ Automatic]

[✓ Collapsible code sections]

[✓ Show section lines in text]

Show section brackets in margin

Clear text on submit

User Defined Keywords | File Extensions

OK | Cancel

.

Click the tabs that are located along the top of the dialog box to navigate to the settings that you want to change, and then select the options that you want. When you are finished, click **`OK`**.

## General Editor Options

From the General tab you can specify the following options that control how the Enhanced Editor works.

**`Allow cursor movement past end of line`**
specifies where the insertion point is positioned when you click the mouse pointer after the last text character on a line. If it is selected, the insertion point is positioned where you click the mouse pointer. If it is not selected, the insertion point is positioned after the last text character on the line.

**`Drag and drop text editing`**
specifies whether selected text can be moved by using drag-and-drop editing. If it is selected, selected text can be moved. If it is not selected, selected text cannot be moved.

**`Show line numbers`**
specifies whether to display line numbers in the margin. When line numbers are displayed, the current line number is red.

**`Strip Ctrl+Z characters`**
specifies to remove the Ctrl+Z end-of-file characters that might be included in files that were created in a DOS editor.

**`File type`**
specifies the type of file to which tabs, indention, and collapsible code sections apply. File types include HTML documents, SAS programs, SCL programs, and text documents.

**Tab size**
specifies the number of spaces to indent.

**Insert spaces for tabs**
specifies whether to insert the space character or the tab character when you press the TAB key. If it is selected, the space character is used. If it is not selected, the tab character is used.

**Replace tabs with spaces on file open**
specifies whether to replace all tab characters in a file with the space character when the file is opened.

**Indentation**
specifies the type of indention to use. When **None** is selected, no indention is used. When **Automatic** is selected, the next line is automatically indented by the same amount of space that the previous line is indented.

**Collapsible code sections**
specifies whether to enable the expansion and contraction of code sections. If it is selected, the collapsible code sections can be collapsed or expanded. If it is not selected, all code appears in the editor window. The following settings are active when the **Collapsible code sections** setting is selected:

- When **Show section lines in text** is selected, a line appears after each section of text.
- When **Show section brackets in margin** is selected, brackets are displayed around each section in the margin.

**Clear text on submit**
specifies whether to clear the contents of the Enhanced Editor window after you submit a program for processing. If it is selected, the Enhanced Editor window is cleared when you submit the program. If it is not selected, the program remains in the editor window. If this setting is selected, you can recall the last submitted program by using the F4 key.

**User Defined Keywords**
opens the User Defined Keywords dialog box that you use to create user-defined keywords.

**File Extensions**
opens the SAS Extensions dialog box. Use the SAS Extensions dialog box to define file extensions that are recognized by the Enhanced Editor.

## Appearance Options

The following appearance options enable you to specify foreground and background colors, and font styles for file elements. You can also create and save color schemes. For more information about using these appearance options, see "Setting Appearance Options" on page 105 and "Using Schemes" on page 106.

**File type**
specifies the type of file whose elements you want to color-code. You can color-code file elements for SAS programs, SCL programs, HTML and XML documents, and text documents. To color-code an XML document, select **HTML Document**. The default is the file type of the file that you are editing at the time that you invoke the Editor Options dialog box.

**Scheme**
is a name that represents a saved set of appearance options for the specified file type.

**`Name`**
specifies the name of the font for the scheme.

**`Size`**
specifies the font size for the scheme.

**`Script`**
lists the character sets available for the specified font. The character set that is used by the default script is determined by the Windows regional options.

**`File elements`**
lists the elements of the specified file type that can be color-coded.

**`Foreground`**
specifies the text color that is to be applied to the selected file element.

**`Background`**
specifies the background color that is to be applied to the selected file element.

**`Font Style`**
specifies whether **`Normal`**, **`Bold`**, **`Italic`**, or **`Bold Italic`** font is to be applied to the file element.

**`Underlined`**
specifies whether the file element is to be underlined.

**`Sample`**
displays a sample of the selected file element colors and font.

## Setting Appearance Options

When you set appearance options, you set them for the elements of the file type that you specified in the **`File type`** box. As you make your selections, the **`Sample`** box displays your selected formatting. The formatting options that you specify are applied to all opened Enhanced Editor windows of that file type. When you start SAS, the formatting options that are applied to the Enhanced Editor files are the formatting options that were in effect when the last SAS session ended.

To specify appearance options

1. Open the Editor Options window by selecting

   Tools ▶ Options ▶ Enhanced Editor ▶ Appearance

2. Select a file type from the **`File type`** box.
3. Optionally, you can select a saved formatting scheme from the **`Scheme`** box. For more information about using schemes, see "Using Schemes" on page 106.
4. From the **`Name`** box, select a font.
5. From the **`Size`** box, select a font size.
6. From the **`Script`** box, select a script that is appropriate for the language that your computer uses. The Default script is determined by the Windows regional options.
7. For each file element that you want to format
   a. Select a file element.
   b. Click in the **`Foreground`** box and select a color for the file element. To create a custom color, select **`Custom`** and create a color from the Color dialog box.
   c. Click the **`Background`** box and select a color for the background of the file element. To create a custom color, select **`Custom`** and create a color from the Color dialog box.

*Note:* Changing the background color for **`Normal text`** changes the Enhanced Editor window to the specified color. △

- d From the **`Font Style`** box, select **`Normal`**, **`Bold`**, **`Italic`**, or **`Bold Italic`**.
- e If you want the element to be underlined, select the **`Underlined`** box.

8 Review your selections in the **`Sample`** box. Click on a file element in the sample to see its color and font assignment. When you have finished formatting all file elements, click **`OK`**.

## Using Schemes

A *scheme* is a saved set of formatting options, such as font, fontsize, and script. You can set your appearance options by selecting a file type and a scheme instead of setting individual file elements. SAS provides several schemes which you can select from the Scheme box. Schemes provided by SAS use the Default script.

To create a scheme

1. Select a file type from the **`File type`** box.
2. Select a font, font size, and a script.
3. For each file element, select a color for the foreground and background, a font style, and the underlining option.
4. Click **`Save As`** and type a scheme name in the Save Scheme dialog box.
5. Click **`OK`**.

To modify a scheme

1. Click in the **`File type`** box and select a file type.
2. Click in the **`Scheme`** box and select a scheme.
3. Make the font and file element changes that you want.
4. Click **`Save As`**. The selected scheme name appears in the Scheme name entry box.
5. Click **`OK`**.

To delete a scheme

1. Click in the **`File type`** box and select a file type.
2. Click in the **`Scheme`** box and select the scheme name that you want to delete.
3. Click **`Delete`**.

## Using Keyboard Shortcuts to Customize the Enhanced Editor

You can customize Enhanced Editor commands and keyboard macros by using the Enhanced Editor Keys dialog box.

### Assigning Keyboard Shortcuts

When you open the Enhanced Editor Keys dialog box, you can choose to view only commands that have been assigned keyboard shortcuts, or you can view all commands. To see only the commands that have assigned keyboard shortcuts, ensure that the **`Show all commands`** check box is not selected. To see all commands, including those that have no key assignment, check the **`Show all commands`** check box.

To assign keyboard shortcuts

1. Select

   Tools ► Options ► Enhanced Editor Keys

2 Select a category from the **`Categories`** box. Macros are listed in the **`User Defined`** category.
3 Select a command from the **`Commands`** box. If a keyboard shortcut is already defined for the command, it is displayed in the **`Keys`** column.
4 Click **`Assign keys`**.
5 Place the insertion point in the **`Press new shortcut key`** field.
6 Press a key sequence for the selected command. The sequence displays in the **`Press new shortcut key`** field, and the assignment status for that key appears at the bottom of the dialog box. If the value in the **`Currently assigned to`** field is **`None`**, then no other command is assigned to this keyboard shortcut.
7 To assign the keyboard shortcut, click **`Assign`**.

*Note:* Assigning a keyboard shortcut to a key sequence that is assigned to another command deletes the shortcut for that command. For example, if you assign the Backspace key to the **`Add a new abbreviation`** command, pressing the Backspace key displays the Add Abbreviation dialog box, and you can no longer backspace by using the Backspace key. △

## Deleting Keyboard Shortcuts

To delete a keyboard shortcut

1 Select

Tools ▶ Options ▶ Enhanced Editor Keys

2 Click **`Assign keys`**.
3 Select the category in the **`Categories`** box. Macros are listed in the **`User Defined`** category.
4 Select the command in the **`Commands`** box.
5 Select the key sequence in the **`Keys currently assigned to command`** box.
6 Click **`Remove`**.

## Resetting Keyboard Shortcuts to the Enhanced Editor Defaults

Resetting keyboard shortcuts to the default keyboard shortcuts deletes all macro keyboard shortcuts. See "Keyboard Shortcuts within the Enhanced Editor" on page 618 for a list of the default keyboard shortcuts.

To reset keyboard shortcuts to the Enhanced Editor default keyboard shortcuts

1 Select

Tools ▶ Options ▶ Enhanced Editor Keys

2 Click **`Assign keys`**.
3 Click **`Reset All`**.

## Enabling and Disabling the Enhanced Editor

By default, the Enhanced Editor is the active editor when you start SAS.

To disable the Enhanced Editor when you start SAS, use the NOENHANCEDEDITOR system option. For more information, see "ENHANCEDEDITOR System Option" on page 501. You can also enable or disable the Enhanced Editor by using the **`Use Enhanced Editor`** setting in the Preferences dialog box Edit tab or by using the **`wedit`** command. For more information, see "WEDIT Command" on page 360.

When you disable the Enhanced Editor, the menu commands are not available. All opened Enhanced Editor windows remain open, and you can open new Enhanced Editor windows by using the **`View`** menu or the **`wedit`** command. If the Enhanced Editor is disabled when you start SAS, the Enhanced Editor window does not open.

When the Enhanced Editor is enabled, the **`Text Editor`** command in the **`Tools`** menu opens an Enhanced Editor window. When the Enhanced Editor is disabled, the **`Text Editor`** command opens SAS NOTEPAD.

## Using the Program Editor

The SAS text editor windows, Program Editor and NOTEPAD, work similarly to other Windows editors. Therefore, you can edit your SAS code without learning how to use a new text editor.

### Switching from the Enhanced Editor to the Program Editor

If the Enhanced Editor is enabled when SAS starts, the Program Editor is disabled. To start the Program Editor when SAS starts, disable the Enhanced Editor in one of the following ways:

- □ Start SAS using the NOENHANCEDEDITOR system option.
- □ Disable the Enhanced Editor in the Preferences dialgo box:
  1. Select

     Tools ► Options ► Preferences ► View
  2. Deselect the **`Use Enhanced Editor`** check box.
  3. Click **`OK`**.

For more information about the NOENHANCEDEDITOR system option, see "ENHANCEDEDITOR System Option" on page 501.

You can always access the Program Editor window from from the **`View`** menu.

### Opening Files

To open a file in the Program Editor:

1. With the editor window active, do one of the following:
   - □ Click the Open toolbar button (the opened file folder)
   - □ Type **`dlgopen`** in the command bar
   - □ Select **`File`** and click **`Open`**

   SAS displays the Open dialog box.
2. Use the Open dialog box to select the file you want to include. By default, SAS looks for files with the .SAS file extension (which contain SAS code, by convention). However, you can change this by adjusting the **`Files of type`** field. (If you change the selected file type, SAS will remember that selection and present it as the default the next time that you open a file for that window during the SAS session.)

   *Note:* To change the default directory for the Open dialog box, either start SAS using the SASINITIALFOLDER system option or change the current working directory. For more information, see "SASINITIALFOLDER System Option" on page 546 and "Changing the SAS Current Folder" on page 37. △

3 If the file that you are including contains SAS code that you want to submit, check the `Submit` box before clicking `OK`.

*Note:* If you select `Submit` , it remains selected each time you use the Open dialog box to open a file. You must deselect it if you do not want to submit the contents of the file you want to open. △

You can also drag and drop a file into the Program Editor from the Windows Explorer or the My Favorite Folders window. To do this,

1 Open the source window.
2 Position the source window and the Program Editor window so that both are visible.
3 In the source window, find and select the file you want to open; click and hold down the left mouse button
4 Drag the file over the Program Editor window and release the left mouse button.

If you open a file with lines longer than 256 characters in the Program Editor window, the lines are truncated unless you use the INCLUDE command with an LRECL= value equal to the number of characters in the longest line, and you set either the AUTOWRAP or AUTOFLOW command to ON. If you want to use the Open dialog box to open a file with lines longer than 256 characters, use the FILENAME statement to set up a fileref with the appropriate options and then use the fileref, enclosed in double quotes, in the `File Name` field in the Open dialog box.

If you recall a SAS program that has more than 256 characters on a line into the Program Editor, the Program Editor wraps the line on to the next line. A line that is greater than 256 characters and wraps onto the next line is considered one line of code.

## Using Line Numbers

If you are familiar with the SAS Program Editor window under other operating systems, such as OS/390, notice that line numbers are turned off by default under Windows. You can type **`numbers on`** in the command bar to display line numbers in the Program Editor window. You can also type **`nums`** to turn line numbers on and off.

You can also control line numbers using the Editor Options dialog box when the Program Editor or NOTEPAD is the active window. To open the Editor Options dialog box:

1 Type **`edop`** in the command bar or select

Tools ► Options ► Program Editor

2 Select the Editing tab.
3 Select `Display line numbers` and click `OK`.

## Moving the Insertion Point

The insertion point movement keys (arrow keys, PgUp, PgDn, and so on) function the same way in SAS text windows as they do in other Windows applications.

Pressing the CTRL key with the left arrow (word left) or right arrow (word right) causes the insertion point to move one word at a time. When advancing through text, the word-left and word-right commands stop at the end of the text on a line and at the beginning of the first word on a new line. You can move to the top of a file by pressing CTRL+PgUp or to the bottom of a file by pressing CTRL+PgDn.

Pressing the Home key causes the insertion point to go to the beginning of the current line unless the command line (not the command bar) is active in the active window. Pressing the Home key when the command line is active causes the insertion point to toggle between the current insertion point position in the text and the command line. The F11 key moves the insertion point to the command bar. You can toggle the command line on and off using the COMMAND command or by selecting **`Command line`** in the Preferences dialog box **`General`** tab.

## Using Tabs

Many text editors retain tab characters, while others expand tabs into space characters. The SAS Program Editor window expands tabs into space characters. Pressing the TAB key inserts spaces and moves any text to the right of the insertion point.

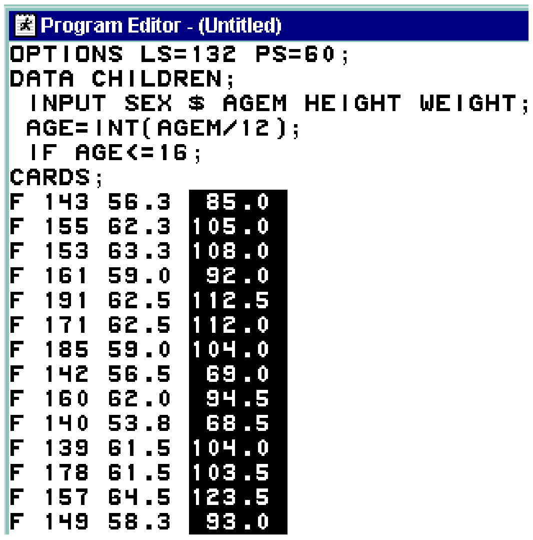
## Understanding Line Breaks

Conceptually, line breaks are at the end of the line rather than at the beginning. Pressing the ENTER key creates a line break. To delete a line break, press the Backspace key at the beginning of a line or press the Delete key at the end of the line.

## Selecting Text

You can use the mouse or the SHIFT key in combination with the insertion point movement keys to select text. The marking of an area of text continues until you release the mouse button or release the SHIFT key. To select all of the text in the active window, select the **`Edit`** menu and then select **`Select All`**. The following are some advanced text selection methods:

- □ Double-click a single word to select it. To select an entire line, hold down the CTRL key as you click on the line you want.
- □ Use the ALT key as you hold down the mouse button and drag the mouse to select a rectangular block (or column) of text (as illustrated below.)



```
OPTIONS LS=132 PS=60;
DATA CHILDREN;
 INPUT SEX $ AGEM HEIGHT WEIGHT;
 AGE=INT(AGEM/12);
 IF AGE<=16;
CARDS;
F 143 56.3  85.0
F 155 62.3 105.0
F 153 63.3 108.0
F 161 59.0  92.0
F 191 62.5 112.5
F 171 62.5 112.0
F 185 59.0 104.0
F 142 56.5  69.0
F 160 62.0  94.5
F 140 53.8  68.5
F 139 61.5 104.0
F 178 61.5 103.5
F 157 64.5 123.5
F 149 58.3  93.0
```

- □ Use the SHIFT key in combination with the mouse button to select the text between the current text insertion point and the position in the text where you click. You can also use this technique to extend a text selection. (You can use this feature only within the current page.)

If characters are selected and you start typing text, the marked area is replaced with the new text. This occurs even if you have moved the mouse pointer away from the marked area. For information about marking and copying text with a mouse, see "Using the Clipboard" on page 53.

To unmark text, click the left mouse button in the window. Alternatively, you can unmark text by selecting **`Deselect`** from the **`Edit`** menu or you can press the ESC key. Typing the WNAVKEYUNMARK ON in the command bar also enables unmarking with the arrow navigational keys.

## Deleting Text

The Delete key deletes the currently selected text, if there is any; otherwise, it deletes the character to the right of the insertion point. To delete from the insertion point to the end of the current line, press ALT+Delete. To delete from the insertion point to the end of the current word, press CTRL+Delete. To delete from the insertion point to the start of the current word, press CTRL+Backspace.

You can also use the Edit menu to delete text. To delete all text in the window, click **`Clear All`**. To delete only selected text, click **`Clear`**. To delete selected text and copy that text to the Windows clipboard, click **`Cut`**.

## Finding and Replacing Text

To find text

1 Open the Find dialog box by selecting

Edit ▶ Find

2 Supply the following information:

**`Find text`**
Type a text string to find. The initial value of this field is the last text string that was used in a search.

**`Direction`**
Select the **`Up`** or **`Down`** check box. **`Up`** specifies to search from the insertion point position toward the beginning of the file. **`Down`** specifies to search from the insertion point position toward the bottom of the file.

**`Match whole word only`**
Select the check box to specify that a match of the text must be a whole word and not part of a word.

**`Match case`**
Select the check box to specify that upper- and lowercase characters must match exactly.

3 Click **`Find Next`**.

To find and replace text

1 Open the Replace dialog box by selecting

Edit ▶ Replace

2 Supply the following information:

**`Find text`**

Type a text string to find and replace. The initial value of this field is the last text string that was used in a search.

**`Replace with`**
Type the replacement string.

**`Direction`**
Select either the **`Up`** or **`Down`** check box. **`Up`** specifies to search from the insertion point position toward the beginning of the file. **`Down`** specifies to search from the insertion point position toward the bottom of the file.

**`Match whole word only`**
Select this check box to specify that any match of the text must be a whole word and not part of a word.

**`Match case`**
Select this check box to specify that upper- and lowercase characters must match exactly.

3 Click **`Find Next`**.
4 If the text is found, click one of the following:
   - □ **`Replace`** to replace this single occurrence of the text with the replacement string.
   - □ **`Replace All`** to replace all occurrences of the text in the file with the replacement string.

## Dragging and Dropping Text

The following table lists the places from which you can drag text and to which you can drop the selected text.

**Table 3.11** Summary of Text Drag and Drop Possibilities

| Text Source | Text Destination |
|---|---|
| any SAS text window | another SAS window that supports text editing (such as the Program Editor window) |
| any SAS text window | another Windows application that supports text drag and drop |
| a Windows application that supports text drag and drop | any SAS window that supports editing |
| Windows Explorer (text file item) | any SAS window that supports editing |

To drag and drop text from one window to another:

1 Arrange your windows, if necessary, so that both the source and target windows are visible on the display.

   *Note:* Instead of arranging your windows so that the target window is visible, the target window will become the active window when you drag the selected text to the target window's button on the window bar. △

2 Select the desired text from the source window.
3 Click and hold the left mouse button with the pointer on the selected text.
4 With the mouse button still pressed, drag the text to the target window.

5 Move the insertion point to the position where you want to insert the text. (If you plan to just submit the text as SAS code for processing, position the insertion point anywhere in the window).

6 Release the mouse button. The text is either included at the point where you positioned the insertion point, or it is submitted to SAS for processing. (The default action depends on the type of the target window.)

You can override the default action of the drag and drop by initiating the drag and drop using the *right* mouse button. This is called *nondefault drag and drop*. When you drag the selection to the target SAS window and release the mouse button, SAS displays a popup menu to let you choose which action to perform.

Table 3.12 on page 113 is a summary of drag-and-drop actions available for the possible target windows in SAS.

**Table 3.12** Summary of Drag-and-Drop Actions

| Data | Target | Default Action | Nondefault Actions |
|---|---|---|---|
| text | SAS text editor | move | move, copy, cancel |
| text | PROGRAM EDITOR | copy | copy, submit, cancel |
| file | SAS text editor | not valid | not valid |
| file | PROGRAM EDITOR | move | copy, submit, cancel |
| file | LOG, OUTPUT | submit | submit, cancel |

The actions that occur when you drag text out of a SAS window into another Windows application depend on the target application. In most cases, dragging and dropping text between SAS and other applications actually moves the text from one window to another (that is, the text is cut from one window and placed in the other).

You can change that behavior by applying a *drag-modifier*—a key you press while you drag and drop. To copy text from one window to another (instead of moving it), press and hold the CTRL key before and during the drag and drop. When you release the mouse button to drop the text, release the CTRL key as well.

## Drag Scrolling

While dragging text to a SAS text editor window, you can cause the target window to scroll vertically or horizontally. This lets you drop text in a window area that is not currently visible.

Once you have selected the text and drag it to the SAS text editor window, pause near the border of the SAS text editor window. The window scrolls in the direction of that border. For example, to cause the target window to scroll down, drag the mouse pointer just above the bottom border of the window and pause.

Drag scrolling only happens when you pause near the drop area border; it does not occur if you drag quickly past the border.

## Using Rich Text Format Text

When you copy text out of a SAS window to the clipboard and paste it into the window of another application, the text retains all of the format information it had in

SAS (except for color) if the target window accepts RTF formatting. For example, the Windows Notepad application does not preserve formatting, but Microsoft Wordpad and many word processors do. The same is true when you drag text out of SAS and drop it in another application window.

If the display font is Sasfont, any text that you copy out of SAS is formatted with the SAS Monospace TrueType font. If your text has other highlighting attributes, such as underline, those attributes are also transferred to the target window in the other application (provided the target window supports rich text format (RTF)).

## Saving Files

To save the contents of the Program Editor window, click the Save toolbar button (the diskette icon). If the file is to be saved for the first time, the Save As dialog box will open for you to name the file.

To save a file with a new name:

1 Select

   File ▶ Save As

2 Select a folder in the **`Save in`** field.
3 Type a filename in the **`File name`** field.
4 Select a file type from the **`Save as type`** field.
5 Click **`OK`**.

*Note:* To change the default directory for the Save dialog box, either start SAS using the SASINITIALFOLDER system option or change the current working directory. For more information, see “SASINITIALFOLDER System Option” on page 546 and “Changing the SAS Current Folder” on page 37. △

## Saving Program Editor Files Using Autosave

To ensure that you do not lose any of your work in the Program Editor, SAS can automatically save your files at an interval you specify. The interval can range from 0 (Autosave off) to 480 minutes. The default interval is 10 minutes.

The autosave file is saved as pgm.asv in the current folder or in the location specified by the AUTOSAVELOC system option.

To enable or disable autosave and set the interval:

1 Select

   Tools ▶ Options ▶ Preferences ▶ Edit tab

2 Select or deselect **`Autosave every`**.
3 Set the interval by typing a number between 1 and 480 in the **`minutes`** box.

You can also use the WAUTOSAVE command to enable, disable, and set the interval. WAUTOSAVE INTERVAL=*minutes* will turn on autosave using *minutes* as the interval.

For more information on the Autosave feature, see “Edit Preferences” on page 59, “WAUTOSAVE Command” on page 355, and “AUTOSAVELOC= System Option” in *SAS Language Reference: Dictionary*.

## Understanding Unique Features of the Editor

The following features of a SAS text editor window are different from the standard features of other editors commonly used in the Windows environment:

- A SAS text editor window allows you to move the insertion point past the last character entered on a line or past the last line of text entered.
- You can mark an area of text, move the mouse pointer away from the marked area, and the marked text remains marked.
- You can unmark text by pressing the ESC key.
- You can use Shift + Tab to delete blank space characters back to the last tab stop.

CHAPTER
4

# Using SAS Files

*Introduction to SAS Files* **118**
- *What Is a SAS File?* **118**
- *File Extensions for SAS Files* **119**
- *SAS Data Sets (Member Type: Data or View)* **120**
- *SAS Catalogs (Member Type: Catalog)* **122**
- *SAS Stored Compiled Data Step Programs (Member Type: Program)* **122**
- *Access Descriptor Files (Member Type: Access)* **122**

*Multiple Engine Architecture* **122**
- *SAS Data Libraries* **122**
- *SAS Engines* **123**
  - *What Is an Engine?* **123**
  - *Types of Library Engines* **123**
  - *Native Library Engines* **123**
  - *Interface Library Engines* **124**
  - *Rules for Determining the Engine* **125**

*Using Data Libraries* **125**
- *Specifying a Libref* **125**
- *Assigning SAS Libraries Using the Graphical User Interface* **126**
- *Assigning SAS Libraries Using the LIBNAME Statement or Function* **126**
  - *LIBNAME Statement Syntax* **126**
  - *Assigning a Libref to a Single Folder* **127**
  - *Assigning a Libref to the Working Folder* **127**
  - *Assigning a Libref to Multiple Folders* **127**
  - *Assigning Engines* **127**
  - *Making Librefs Available When SAS Starts* **128**
  - *Assigning Multiple Librefs and Engines to a Folder* **128**
- *Assigning SAS Libraries Using Environment Variables* **128**
  - *Types of Environment Variables* **128**
  - *Using a SAS Environment Variable as a Libref* **129**
  - *Using Windows Environment Variables* **130**
- *Listing Libref Assignments* **130**
  - *Listing Librefs Using the Explorer Window* **130**
  - *Listing Librefs Using the LIBNAME Command* **130**
  - *Listing Librefs Using the LIBNAME Statement* **130**
- *Clearing Librefs* **130**
  - *SAS Explorer Window* **131**
  - *LIBNAME Window* **131**
  - *LIBNAME Statement* **131**
  - *LIBNAME Function* **131**
- *Understanding How Multi-Folder SAS Data Libraries Are Accessed* **131**
  - *Protocols for Accessing Folders* **131**

*Input and Update Access* **132**
*Output Access* **132**
*Accessing Data Sets with the Same Name* **132**
*Using the Sasuser Data Library* **133**
*Using the Work Data Library* **133**
*Using Temporary Files* **133**
*Using an Environment Variable* **134**
*Using the User Libref* **134**
*Using Large Data Sets with Windows and NTFS* **135**
*Accessing SAS Files from Multiple SAS Sessions* **135**
*Using SAS Files from Other Versions with SAS 9.1 for Windows* **136**
*Introduction to Using SAS Files from Other Versions with SAS 9.1 for Windows* **136**
*Using Release 6.08 through Release 8.2 Data Sets* **137**
*Using Release 6.03 and Release 6.04 SAS Data Sets* **137**
*Converting Release 6.08 through Release 6.12 SAS Data Sets* **137**
*Using Version 7 and 8 Catalogs in SAS 9.1* **138**
*Converting Version 6 SAS Catalogs in SAS 9.1* **138**
*Converting Release 6.08 SAS Catalogs to SAS 9.1* **138**
*Converting Release 6.03 and Release 6.04 SAS Catalogs to SAS 9.1* **139**
*Creating Release 6.08 through Release 6.12 Data Sets* **139**
*Using SAS 9.1 Files with Previous Releases* **140**
*Using Remote Host SAS Files in SAS 9.1* **140**
*Reading BMDP, OSIRIS and SPSS Files* **140**
*BMDP Engine* **140**
*BMDP Engine Examples* **141**
*OSIRIS Engine* **141**
*OSIRIS Engine Example* **142**
*SPSS Engine* **142**
*SPSS Engine Example* **143**
*Transferring SAS Files between Operating Environments* **143**
*Accessing Database Files with SAS/ACCESS Software* **143**
*Using the SAS ODBC Driver to Access SAS Data from Other Applications* **143**

## Introduction to SAS Files

### What Is a SAS File?

SAS creates and uses a variety of specially structured files called *SAS files*. Although Windows manages the file for SAS by storing it, the operating system cannot process it. For example, you can list SAS files with the Windows Explorer, but SAS files can only be processed by SAS. SAS files are different from external files. While external files can be processed by SAS statements and commands, they are not managed by SAS.

SAS files usually reside in SAS data libraries. Under Windows, a SAS library is simply a *named collection of SAS files within one or more Windows folders that SAS can access*. Each SAS data library has an access engine associated with it the first time that a file in the library is accessed. The engine name specifies the access method that SAS uses to process the files in the data library. SAS data libraries are described in detail in *SAS Language Reference: Concepts*.

The various engines enable SAS to access different formats or versions of SAS files and other vendors' files. For this reason, SAS is said to have Multiple Engine Architecture. Multiple Engine Architecture, combined with conversion utilities, provides access to SAS 9.1 files and SAS files created with previous releases of SAS (back to Version 5), whether they were created under Windows or other operating environments. Multiple Engine Architecture also provides access to files created by other vendors' products, including database files.

The following sections highlight information you need in order to create and use SAS files with the various engines under Windows.

## File Extensions for SAS Files

SAS files are stored in SAS data libraries and are referred to as *members* of a library. Each member has a *member type*. SAS distinguishes between SAS files and external Windows files in a folder by using unique file extensions. SAS assigns certain file extensions to a general set of SAS member types. The following table lists the Windows file extensions and their corresponding SAS member types for the V6, V7, V8, and V9 engines. For more information about engines, see "Multiple Engine Architecture" on page 122.

**Table 4.1** Windows File Extensions and Their Corresponding SAS Member Types

| V6 File Extension | V7 and Beyond File Extensions | SAS Member Type | Description |
|---|---|---|---|
| .sas | .sas | none | SAS program |
| .ss2 | .sas7bpgm | Program | stored program (DATA step) |
| | .cfg (Version 8 and beyond) | none | configuration file |
| .lst | .lst | none | output file |
| .log | .log | none | log file |
| none | .sas7baud | Audit | audit file |
| .sd2 | .sas7bdat | Data | data set |
| .sv2 | .sas7bvew | View | data set view |
| .si2 | .sas7bndx | Index | data set index. Indexes are stored as separate files but are treated by SAS as integral parts of the SAS data file. |
| .sc2 | .sas7bcat | Catalog | SAS catalog |
| .sa2 | .sas7bacs | Access | access descriptor file |
| .sf2 | .sas7bfdb | FDB | consolidation database file |
| .sm2 | .sas7bmdb | MDDB | multi-dimensional database file |
| none | .sas7bdmd | DMDB | data mining database file |
| none | .sas7bitm | Itemstor | item store file |
| .su2 | .sas7butl | Utility | utility file |
| .sp2 | .sas7bput | Utility | permanent utility |

| V6 File Extension | V7 and Beyond File Extensions | SAS Member Type | Description |
|---|---|---|---|
| .stx | none | none | transport file |
| none | .sas7bbak | none | backup file |

***CAUTION:***

**Do not change the file extension of a SAS file; doing so can cause unpredictable results.** The file extensions assigned by SAS to SAS files are an integral part of how SAS accesses these files. Also, you should not change the filename of a SAS file using operating system commands. If you want to change the name of a SAS file, use the DATASETS procedure or select the file in the SAS Explorer window or the My Favorite Folders window and select

Edit ► Rename

△

*Note:* Be sure not to delete files from your Work and Sasuser data libraries during your SAS session. SAS creates temporary utility files that you do not need to access directly but that are necessary for processing SAS data.

If your SAS session ends abnormally, you might need to delete files outside SAS in order to regain disk space. You can delete files in the Work library by using the WORKINIT and the WORKTERM system options when you start SAS. For more information, see “WORKINIT System Option” and “WORKTERM System Option” in *SAS Language Reference: Dictionary*. △

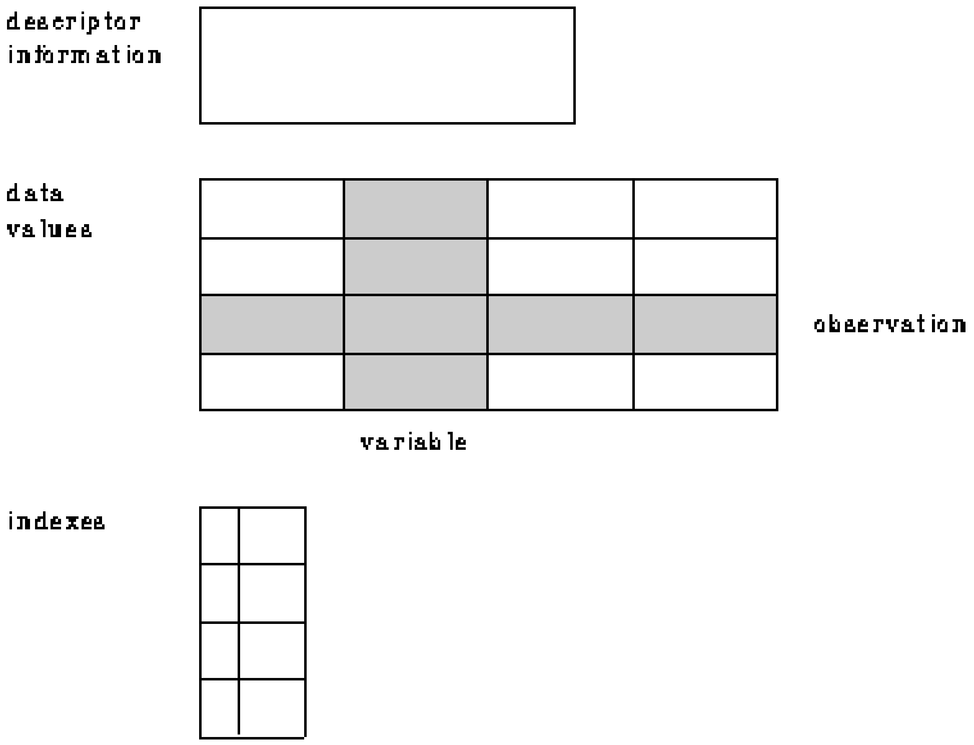
## SAS Data Sets (Member Type: Data or View)

*SAS data set* is an umbrella term for SAS data files and SAS data views, which are both discussed here. This section provides a brief overview of the concept of SAS data sets. For complete details, see the data sets section in *SAS Language Reference: Concepts*.

Logically, a SAS data set consists of two types of information: descriptor information and data values. The descriptor information includes such things as data set name, data set type, data set label, and number of variables, as well as the names and labels of the variables in the data set, their types (character or numeric), their length, their position within a record, and their formats. For more information, see “CONTENTS Procedure” on page 425.

The data values contain values for the variables. A SAS data set can be visualized as a table consisting of rows of observations and columns of variable values. The following table illustrates the SAS data set model.

**Figure 4.1** SAS Data Set Model



SAS data files (member type: Data)

The *SAS data file* is probably the most frequently used type of SAS file. SAS data files have a SAS member type of Data and are created in the DATA step and by certain SAS procedures such as the RANK procedure in Base SAS software. SAS data files have a file extension of .sas7bdat.

SAS defines two types of SAS data files, native and interface. Native data files store data values and descriptor information, as described earlier, in files formatted by SAS. These are the SAS data sets you may be familiar with from previous versions of SAS under other operating environments. In SAS under Windows, native SAS data files can be indexed. The *index* is an auxiliary file that you create to provide fast access to records within a SAS data file through a variable or key. Indexes are stored as separate files but are treated by SAS as integral parts of the SAS data file. To learn more about indexes, see *SAS Language Reference: Concepts*.

The second type of data file is the interface SAS data file. These files store data in a file formatted by other software. Examples of interface SAS data files are BMDP, OSIRIS and SPSS files, which SAS can access as read-only files. For more information, see "Reading BMDP, OSIRIS and SPSS Files" on page 140.

SAS data views (member type: View)

*SAS data views* have a member type of View. They describe data values and tell SAS where to find the values, but they do not contain the actual data values themselves. SAS data views have a file extension of .sas7bvew.

Views may be of two kinds, native or interface. A native SAS data view is created with the SQL procedure or with the DATA step and describes a subset or combination of the data in one or more SAS data files or SAS data views. For information on SQL views, see the *Base SAS Procedures Guide*. For information on DATA step views, see *SAS Language Reference: Concepts*.

Interface SAS data views contain descriptor information for data formatted by other software products, for example, a database management system. You access database views using the SAS/ACCESS LIBNAME statement. For more information, see *SAS/ACCESS for PC Files: Reference* and *SAS/ACCESS for Relational Databases: Reference*.

### SAS Catalogs (Member Type: Catalog)

A *SAS catalog* is a special type of SAS file that can contain multiple entries. You can keep different types of entries in the same SAS catalog. For example, the Sasuser.Profile catalog contains funtion key definitions, fonts for graphic applications, some of your selections from the Preferences dialog box, and other information from interactive windowing procedures. SAS catalogs have a file extension of .sas7bcat.

If you want to use SAS 9.1 to access catalogs created with earlier releases of SAS for Windows, you must first convert the catalogs from the earlier releases to the SAS 9.1 format before you can use them in a SAS 9.1 program.

For more information on how to convert SAS catalogs, see *Moving and Accessing SAS Files*.

### SAS Stored Compiled Data Step Programs (Member Type: Program)

A *stored compiled DATA step program* is a SAS file that contains a DATA step program that has been compiled and then stored in a SAS data library. You can execute compiled DATA step programs as needed, without having to recompile them. SAS stored compiled DATA step programs have a file extension of .sas7bpgm.

Stored compiled programs are available for DATA step applications only. Your stored programs can contain all SAS language elements except global statements. If you do include global statements in your source program, SAS stores the compiled program but not the global statements, and does not display a warning message in the SAS log.

For more information about this type of SAS file, see *SAS Language Reference: Concepts*.

### Access Descriptor Files (Member Type: Access)

*Descriptor files* created by the SAS/ACCESS LIBNAME statement have a member type of ACCESS and are used when creating interface SAS data views. Descriptor files describe the data formatted by other software products supported by SAS. For more information, see *SAS/ACCESS for Relational Databases: Reference* , *SAS/ACCESS for PC Files: Reference* and other available SAS/ACCESS documentation.

## Multiple Engine Architecture

### SAS Data Libraries

All permanent and temporary SAS files are stored in SAS data libraries. A *SAS data library* is a collection of SAS data files that are stored in a physical location under the operating system. Although the physical location in the operating system can contain files that are not managed by SAS, only SAS files are considered part of the SAS data library. Any Windows folder can be treated as a SAS data library.

To use a SAS data library in your SAS session, you must assign a *libref* (library reference) and an engine to the data library. The libref is the name you use to refer to the data library during a SAS session or job. You can create a libref from the Explorer window or you can programmatically define it with an environment variable or with the LIBNAME statement or function. For information on using librefs in the Windows environment, see "Using Data Libraries" on page 125. For a complete explanation of librefs, see *SAS Language Reference: Concepts*.

The Explorer window provides an easy way to manage all of your SAS files, including librefs. For information about working with SAS files in the Explorer window, see the SAS Help and Documentation.

## SAS Engines

### What Is an Engine?

Engines, also called access methods, provide access to many formats of data, giving SAS a *Multiple Engine Architecture*. Engines apply only to SAS data sets.

The engine identifies the set of routines that SAS uses to access the files in the data library. With this architecture, data can reside in different types of files, including SAS data files and data formatted by other software products, such as database management systems. By using the appropriate engine for the file type, SAS can write to or read from the file. For some types of files, you need to tell SAS what engine to use. For others, SAS automatically chooses the appropriate engine. For more details about engines and Multiple Engine Architecture, see *SAS Language Reference: Concepts*.

Engines are of two basic types, library and view. *Library engines* control access at the SAS data library level and can be specified in the LIBNAME statement or function. *View engines* enable SAS to read SAS data views described by the DATA step, SQL procedure, or SAS/ACCESS software. The use of SAS view engines is automatic because the name of the view engine is stored as part of the descriptor portion of the SAS data set.

### Types of Library Engines

SAS has two types of library engines: native and interface. These engines support the SAS data library model. Library engines perform several important functions, including determining fundamental processing characteristics. For a more detailed description of library engines, see *SAS Language Reference: Concepts*. For examples of using library engines, see "Using Data Libraries" on page 125.

### Native Library Engines

*Native library engines* are those engines that access forms of a SAS file created and maintained by SAS. Native library engines include the default engine, the compatibility engine, and the transport engine. The following table lists the acceptable names (and nicknames) for these engines.

**Table 4.2** Native Library Engines

| | Engine Names | Description |
|---|---|---|
| default | V9, BASE | accesses SAS System 9 and SAS 9.1 data files |
| Version 8 compatibility | V8 | accesses the Version 8 data files |
| Version 7 compatibility | V7 | accesses Version 7 data files |
| Release 6 compatibility | V6 | accesses any data file created by Release 6.08 through Release 6.12. In 64-bit environments, the V6 engine can only read data. |
| Release 6.12 compatibility | V612 | accesses Release 6.12 data files |
| Release 6.03 and Release 6.04 compatibility | V604 | read-only access to data files created by Release 6.03 and Release 6.04 |
| transport | XPORT | accesses transport files |

When using the default engine, choose which name, V9 or BASE, that you use in your SAS jobs with an eye to the future. If your application is intended for SAS 9.1 only, and you do not want to convert it to later releases, use the name V9. If, however, you plan to convert your application to new releases of SAS, use the name BASE because that refers to the latest default engine. Using the name BASE makes your programs easy to convert. The engine name BASE does not refer to Base SAS software; rather, it refers to the base, or primary, engine. The BASE engine can be used with more than the Base SAS software product.

This document uses the term *default engine* to refer to the V9 engine. The V9 engine is the default engine for accessing SAS files under SAS 9.1 unless the default engine is changed with the ENGINE system option. To see the value of the ENGINE system option, do one of the following:

- Submit

  ```
  proc options option=engine;
  run;
  ```

- select

  Tools ▶ Options ▶ System

  to open the SAS System Options window. Then select

  Files ▶ SAS Files

  The Engine system option displays the default engine for SAS data libraries.

## Interface Library Engines

Interface library engines support access to other vendors' files. These engines allow read-only access to BMDP, OSIRIS, and SPSS files. You must specify as part of the LIBNAME statement or function the name of the interface library engine that you want. The following table lists the interface engine names:

**Table 4.3** Interface Library Engines

| Name | Description |
|---|---|
| BMDP | allows read-only access to BMDP files in a 32-bit operating environment |
| OSIRIS | allows read-only access to OSIRIS files |
| SPSS | allows read-only access to SPSS files |

For more information about these engines, see "Reading BMDP, OSIRIS and SPSS Files" on page 140 and "ENGINE System Option" on page 499.

## Rules for Determining the Engine

If you do not specify an engine name in a LIBNAME statement or function, SAS attempts to determine the engine (either the default or a compatibility engine) that should be assigned to the specified data library libref. Under Windows, SAS looks at the file extensions that exist in the given folder and uses the following rules to determine which engine should be assigned to the libref:

- If the folder contains SAS data sets from only one of the supported native library engines (not including XPORT), the libref is assigned to that engine.
- If there are no SAS data sets in the given folder, the libref is assigned to the default engine.
- If the folder contains SAS data sets from more than one engine, this is called a mixed mode library. The libref is then assigned to the default engine. A message is printed in the SAS log informing you the libref is assigned to a mixed mode library.

*Note:* It is always more efficient to specify the engine name than to have SAS determine the correct engine. △

You can use the ENGINE system option to specify the default engine that SAS uses when it detects a mixed mode library or a library with no SAS files. By default, the ENGINE option is set to V9. For more information, see "ENGINE System Option" on page 499.

# Using Data Libraries

## Specifying a Libref

The libref is a label or alias that is assigned to a folder so that the storage location (the full path, including drive and folder) is in a form that is recognized by SAS. It is a logical concept describing a physical location, rather than something physically stored with the file.

If a libref is created from within a SAS program, it exists only during the session in which it is created. If a libref is created interactively, by using the New Library dialog box, you can select **`Enable at Startup`** to make it a permanent libref.

A libref follows the same rules of syntax as any SAS name. See the SAS language rules section in *SAS Language Reference: Concepts* for more information about SAS naming conventions.

There are several ways to specify a libref:

- Use the New Library dialog box that is described in SAS Help and Documentation.
- Use the LIBNAME statement or function as described in "Assigning SAS Libraries Using the LIBNAME Statement or Function" on page 126
- Define an environment variable as described in "Assigning SAS Libraries Using Environment Variables" on page 128.

*Note:* You can eliminate the LIBNAME statement by directly specifying the drive name and the dataset name within quotes. An example follows:

```
data "d:\mydata";
```

△

## Assigning SAS Libraries Using the Graphical User Interface

To assign librefs and specify engines using the graphical user interface (GUI), use either the New Library toolbar icon, the LIBASSIGN command, or Explorer to open the New Library dialog box.

- From the toolbar, click the New Library icon.
- In the command bar, type either **`libassign`** or **`libname`**.

  When the LIBNAME window opens, click the **`New`** toolbar button.
- Within Explorer
  1. Select the Library folder.
  2. Select

     File ▶ New

     or right-click the Library folder and select **`New`** from the menu.

  *Note:* When a second Explorer window is open on the right side of the SAS workspace, you can open the New Library dialog box if you right-click the **`Libraries`** folder and select **`New`**. △

For more information about the New Library window and Explorer, see the SAS Help and Documentation.

## Assigning SAS Libraries Using the LIBNAME Statement or Function

### LIBNAME Statement Syntax

You can use the LIBNAME statement or function to assign librefs and engines to one or more folders, including the working folder. The examples in this section use the LIBNAME statement. For information about the LIBNAME function, see *SAS Language Reference: Dictionary*.

The LIBNAME statement has the following basic syntax:

**LIBNAME** *libref <engine-name> 'SAS-data-library'*

An explanation of all the arguments in this statement can be found in *SAS Language Reference: Dictionary* and .

*Note:* The words AUX, CON, NUL, LPT1 - LPT9, COM1 - COM9, and PRN are reserved words under Windows. Do not use these reserved words as librefs. △

## Assigning a Libref to a Single Folder

If you have SAS 9.1 data sets stored in the C:\MYSASDIR folder, you can submit the following LIBNAME statement to assign the libref TEST to that folder:

```
libname test V9 'c:\mysasdir';
```

This statement indicates that the libref TEST accesses SAS 9.1 files stored in the folder C:\MYSASDIR. Remember that the engine specification is optional.

## Assigning a Libref to the Working Folder

The current working folder is shown in the status line of the main SAS window. If you want to assign the libref MYCURR to your current SAS working folder, use the following LIBNAME statement:

```
libname mycurr '.';
```

## Assigning a Libref to Multiple Folders

If you have SAS files located in multiple folders, you can treat these folders as a single SAS data library by specifying a single libref and concatenating the folder locations, as in the following example:

```
libname income ('c:\revenue' 'd:\costs');
```

This statement indicates that the two folders, C:\REVENUE and D:\COSTS, are to be treated as a single SAS data library. When you concatenate SAS data libraries, SAS uses a protocol (a set of rules) for accessing the libraries, depending on whether you are accessing the libraries for read, write, or update.

Furthermore, you may concatenate multiple libraries by specifying only their librefs, as in the following example:

```
libname sales (income revenue);
```

This statement indicates that two libraries that are identified by librefs INCOME and REVENUE are treated as a single SAS data library whose libref is SALES.

For more information, see "Understanding How Multi-Folder SAS Data Libraries Are Accessed" on page 131 and *SAS Language Reference: Dictionary*.

*Note:* The concept of library concatenation also applies when specifying system options, such as the SASHELP and SASMSG options. For information about how to specify multiple folders by using system options, see "Syntax for Concatenating Libraries in SAS System Options" on page 469. △

## Assigning Engines

If you want to use another access method, or engine, instead of the V9 engine, you can specify another engine name in the LIBNAME statement. For example, if you want to access only Version 6.12 SAS data sets from your SAS 9.1 session, you can specify the V612 engine in the LIBNAME statement, as in the following example:

```
libname oldlib V612 'c:\sas612';
```

As another example, if you plan to share SAS files between SAS 9.1 under Windows and Version 6 under Windows, you should use the V6 engine when assigning a libref to the SAS data library. Here is an example of specifying the V6 engine in a LIBNAME statement:

```
libname lib6 V6 'c:\sas6';
```

The V6 engine is particularly useful in your SAS 9.1 session if you are going to be accessing the same SAS files from a Version 6 SAS session. Remember that while SAS 9.1 can read Version 6 SAS data sets, Release 6 cannot read SAS 9.1 data sets.

For more information about using engine names in the LIBNAME statement, see "Using SAS Files from Other Versions with SAS 9.1 for Windows" on page 136 and "Reading BMDP, OSIRIS and SPSS Files" on page 140. You can also see the LIBNAME statement in *SAS Language Reference: Dictionary*.

### Making Librefs Available When SAS Starts

Instead of assigning the same librefs each time that you start SAS, you can specify a libref each time that SAS starts. In the New Library dialog box, select **`Enable at startup`**. The libref is available as soon as SAS initializes. Libraries that are enabled at startup are stored in the SAS Registry under the entry [CORE\OPTIONS\LIBNAMES].

### Assigning Multiple Librefs and Engines to a Folder

If a folder contains SAS files that were created by several different engines, only those SAS files that were created with the engine that is assigned to the given libref can be accessed by using that libref. You can assign multiple librefs with different engines to a folder. For example, the following statements are valid:

```
libname one V8 'c:\mydir';
libname two V9 'c:\mydir';
```

Data sets that are referenced by the libref ONE are created and accessed using the compatibility engine (V8), whereas data sets that are referenced by the libref TWO are created and accessed using the default engine (V9). You can also have multiple librefs (using the same engine) for the same SAS data library. For example, the following two LIBNAME statements assign the librefs MYLIB and INLIB (both using the V9 engine) to the same SAS data library:

```
libname mylib V9 'c:\mydir\datasets';
libname inlib V9 'c:\mydir\datasets';
```

Because the engine names and the Windows pathnames are the same, the librefs MYLIB and INLIB are identical and can be used interchangeably.

## Assigning SAS Libraries Using Environment Variables

### Types of Environment Variables

You can also assign a libref using environment variables instead of the LIBNAME statement or function. An *environment variable* equates one string to another within the Windows environment. SAS recognizes two kinds of environment variables:

- SAS environment variables
- Windows environment variables.

When you use a libref in a SAS statement, SAS resolves libref assignments in this order:

1. a libref assigned by a LIBNAME statement, a LIBNAME function, or by using the New Library dialog box, with the last assignment taking precedence
2. a libref assigned by a SAS environment variable
3. a libref assigned by a Windows environment variable.

For example, if the Windows environment variable TEMP is assigned to C:\WINNT\TEMP and you use the following libname statement:

```
libname temp c:\public
```

the libname resolves to **`c:\public`**.

There are two ways of defining an environment variable to SAS:

- □ Use the SET system option. This defines a SAS (internal) environment variable.
- □ Issue a Windows SET command. This defines a Windows (external) environment variable. Alternatively under Windows, you can define environment variables using the System Properties dialog box accessed from the Control Panel, or by right-clicking **`My Computer`** and selecting **`Properties`** from the menu.

***CAUTION:***

**You cannot assign engines to environment variables.** If you use environment variables as librefs, you must accept the default engine. △

The availability of environment variables makes it simple to assign resources to SAS prior to invocation.

## Using a SAS Environment Variable as a Libref

You can use the SET system option to define a SAS environment variable. For example, if you store your permanent SAS data sets in the C:\SAS\MYSASDATA folder, you can use the following SET option in the SAS command when you start SAS or in your SAS configuration file to assign the environment variable TEST to this SAS data library:

```
-set test c:\sas\mysasdata
```

When you assign an environment variable, SAS does not resolve the environment reference until the environment variable name is actually used. For example, if the TEST environment variable is defined in your SAS configuration file, the environment variable TEST is not resolved until it is referenced by SAS. Therefore, if you make a mistake in your SET option specification, such as misspelling a folder name, you do not receive an error message until you use the environment variable in a SAS statement.

Because Windows filenames can contain spaces or single quotation marks as part of their names, you should enclose the name of the physical path in double quotation marks when specifying the SET option. If you use the SET option in an OPTIONS statement, you must use quotation marks around the filename. For complete syntax of the SET system option, see "SET System Option" on page 549.

Any environment variable name that you use as a value for a system option in your SAS configuration file must be defined as an environment variable before it is used. For example, the following SET option must appear before the SASUSER option that uses the environment variable TEST:

```
-set test "d:\mysasdir"
-sasuser "!test"
```

In the following example, environment variables are used with concatenated libraries:

```
-set dir1 "c:\sas\base\sashelp"
-set dir2 "d:\sas\stat\sashelp"
-sashelp (!dir1 !dir2)
```

Note that when you reference environment variables in your SAS configuration file or in a LIBNAME statement in your SAS programs, you must precede the environment variable name with an exclamation point (!).

It is recommended that you use the SET system option in your SAS configuration file if you invoke SAS through a Windows shortcut.

### Using Windows Environment Variables

You can execute a Windows SET command prior to invoking SAS to create a Windows environment variable. You must define the environment variable prior to invoking SAS; you cannot define environment variables for SAS use from a Command Prompt window from within a SAS session.

SAS can recognize environment variables only if they have been assigned in the same context that invokes the SAS session. That is, you must either define the environment variable in the Windows AUTOEXEC.BAT file that runs when Windows starts (thus creating a global variable), or define the variable in either a Command Prompt window from which you then start SAS or from the System Properties dialog box.

If you define an environment variable in a Command Prompt window, and then start SAS from the Start menu (or with another shortcut), SAS will not recognize the environment variable.

The environment variables that you define with the SET command can be used later within SAS as librefs. In the following example, the Windows SET command is used to define the environment variables PERM and BUDGET:

```
SET PERM=C:\MYSASDIR
SET BUDGET=D:\SAS\BUDGET\DATA
```

## Listing Libref Assignments

### Listing Librefs Using the Explorer Window

If you are running SAS interactively, use the Explorer window to view the active librefs. The Explorer window lists all the librefs that are active for your current SAS session, along with the engine and the physical path for each libref. Any environment variables that you have defined as librefs are listed, provided you have used them in your SAS session. If you have defined an environment variable as a libref but have not used it yet in a SAS program, the SAS Explorer window does not list it.

### Listing Librefs Using the LIBNAME Command

In any SAS session, you can use the LIBNAME command to invoke the LIBNAME window. The LIBNAME window lists the active libraries. Using the LIBNAME window, you can view the contents of all your libraries.

### Listing Librefs Using the LIBNAME Statement

The following LIBNAME statement writes the active librefs to the SAS log:

```
libname _all_ list;
```

## Clearing Librefs

You can clear a libref by using one of the following methods:

- □ the SAS Explorer window
- □ the LIBNAME window

- the LIBNAME statement
- the LIBNAME function .

SAS automatically clears the association between librefs and their respective data libraries at the end of your job or session. If you want to associate the libref with a different SAS data library during the current session, you do not have to end the session or clear the libref. SAS automatically reassigns the libref when you use it to name a new library.

### SAS Explorer Window

To clear a libref by using the SAS Explorer window:

1. Right-click on the node of the libref that you want to clear.
2. Select `Delete`.

For more information about using the SAS Explorer window to manage libraries, see *The Little SAS Book* or the SAS Help and Documentation.

### LIBNAME Window

To clear a libref by using the LIBNAME window:

1. Issue the LIBNAME command in the command bar. The LIBNAME window opens.
2. Right-click on the node of the libref that you want to clear.
3. Select `Delete`.

### LIBNAME Statement

To clear a libref by using the LIBNAME statement, submit a LIBNAME statement using this syntax:

**LIBNAME** *libref* | _all_ <clear>;

If you specify a libref, only that libref is cleared. If you specify the keyword _all_, all the librefs you have assigned during your current SAS session are cleared. (Maps, Sasuser, Sashelp, and Work remain assigned.)

*Note:* When you clear a libref defined by an environment variable, the variable remains defined, but it is no longer considered a libref—that is, it is not listed in the Explorer window. You can use the variable in another LIBNAME statement to create a new libref. △

### LIBNAME Function

To clear a libref by using the LIBNAME function, the only argument to the function is the libref:

```
libname(libref);
```

## Understanding How Multi-Folder SAS Data Libraries Are Accessed

### Protocols for Accessing Folders

When you use the concatenation feature to specify more than one physical folder for a libref, SAS uses the following protocol for determining which folder is accessed:

- Input and update access

- □ Output access
- □ Accessing data sets with the same name

The protocol illustrated by the following examples applies to all SAS statements and procedures that access SAS files, such as the DATA, UPDATE, and MODIFY statements in the DATA step and the SQL and APPEND procedures.

## Input and Update Access

When a SAS file is accessed for input or update, the first SAS file found by that name is the one that is accessed. For example, if you submit the following statements and the file OLD.SPECIES exists in both folders, the one in the C:\MYSASDIR folder is printed:

```
libname old ('c:\mysasdir','d:\saslib');
proc print data=old.species;
run;
```

The same would be true if you opened OLD.SPECIES for update with the FSEDIT procedure.

## Output Access

If the data set is accessed for output, it is always written to the first folder, provided that the folder exists. If the folder does not exist, an error message is displayed. For example, if you submit the following statements, SAS writes the OLD.SPECIES data set to the first folder (C:\MYSASDIR), replacing any existing data set with the same name:

```
libname old ('c:\mysasdir','d:\saslib');
data old.species;
   x=1;
   y=2;
run;
```

If a copy of the OLD.SPECIES data set exists in the second folder, it is not replaced.

## Accessing Data Sets with the Same Name

One possibly confusing case involving the access protocols for SAS files occurs when you use the DATA and SET statements to access data sets with the same name. For example, suppose you submit the following statements and TEST.SPECIES originally exists only in the second folder, D:\MYSASDIR:

```
libname test ('c:\sas','d:\mysasdir');
data test.species;
   set test.species;
   if value1='y' then
      value2=3;
run;
```

In this case, the DATA statement opens TEST.SPECIES for output according to the output rules; that is, SAS opens a data set in the first of the concatenated libraries (C:\SAS). The SET statement opens the existing TEST.SPECIES data set in the second (D:\MYSASDIR) folder, according to the input rules. Therefore, the original TEST.SPECIES data set is not updated; rather, two TEST.SPECIES data sets exist, one in each folder.

## Using the Sasuser Data Library

SAS automatically creates a SAS data library with the libref Sasuser. This library contains, among other SAS files, your user profile catalog.

By default under Windows, the Sasuser libref points to the following folders:

**Table 4.4** Folders for the Sasuser Libref

| | |
|---|---|
| Windows NT | C:\WINNT\Profiles\*user-id*\Personal\My SAS Files\9.1 |
| Windows 2000<br>Windows XP<br>Windows Server 2003 | C:\Documents and Settings\*user-id*\My Documents\My SAS Files\9.1 |

You can use the SASUSER system option to make the Sasuser libref point to a different SAS data library. If a Sasuser folder does not exist, SAS creates one. If you use a folder other than the default folder, you can add the SASUSER system option to the sasv9.cfg configuration file.

SAS stores other files besides the profile catalog in the Sasuser folder. For example, sample data sets are stored in this folder.

The Sasuser data library is always associated with the V9 engine. You cannot change the engine associated with the Sasuser data library. If you try to assign another engine to this data library, you receive an error message. Therefore, even if you have set the ENGINE system option to another engine, any SAS files that are created in the Sasuser data library are SAS 9.1 files.

For more information about your profile catalog, see "Profile Catalog" on page 19. For more information about the SASUSER system option, see "SASUSER System Option" on page 547.

## Using the Work Data Library

### Using Temporary Files

The Work data library is the storage place for temporary SAS files. By default under Windows, the Work data library is created as a subfolder of !TEMP\SAS Temporary Files folder. This subfolder is named _TD*nnnnnnnnnn*, as discussed in "Work Data Library" on page 21. Temporary SAS files are available only for the duration of the SAS session in which they are created. At the end of that session, they are deleted automatically. If SAS terminates abnormally, you may need to delete the temporary files.

By default, any file that is not assigned a two-level name is automatically considered to be a temporary file. A special libref of Work is automatically assigned to any temporary SAS data sets created. For example, if you run the following SAS DATA step to create the data set Sports, a temporary data set named Work.Sports is created:

```
data sports;
   input @1 sport $10. @12 event $20.;
   datalines;
```

```
volleyball co-recreational
swimming   100-meter freestyle
soccer     team
;
```

If you display the SAS Explorer window now, you will see the Sports data set in the Work folder.

You can display all the temporary data sets that are created during this session from:

- □ the SAS Explorer window. Double-click the Libraries folder icon and then double-click the Work folder icon.
- □ the LIBNAME window. Type **libname** in the command bar and double-click the Work folder icon.

The Work data library is always associated with the V9 engine. You cannot change the engine associated with the Work data library. If you try to assign another engine to this data library, you receive an error message. Therefore, even if you have set the ENGINE system option to a different engine, any SAS files that are created in the Work data library are SAS 9.1 files.

## Using an Environment Variable

You can use an environment variable in your Work data library specification, similar to the method illustrated earlier with the SASUSER system option. Use this technique when you do not want to use the default location for your Work data library. You can put something similar to the following in your SAS configuration file to set up an environment variable to use for your Work data library:

```
-set myvar c:\ tempdir
-work !myvar
```

The SET option associates the MYVAR environment variable with the C:\TEMPDIR folder. Then the WORK option tells SAS to use that folder for the Work data library. When you exit your SAS session, the temporary folders and any files they contain are removed.

## Using the User Libref

Although by default SAS files with one-level names are temporary and are deleted at the end of your SAS session, you can use the User libref to cause SAS files with one-level names to be stored in a permanent SAS data library. For example, the following statement causes all SAS files with one-level names to be permanently stored in the C:\MYSASDIR folder:

```
libname user 'c:\mysasdir';
```

When you set the User libref to a folder as in the previous example and you want to create or access a temporary data set, you must specify a two-level name for the data set, with Work as the libref.

Alternatively, you can assign the User libref when you invoke SAS by using the USER system option or by creating a Windows environment variable named USER. If you have a Windows environment variable named USER, the USER libref is automatically assigned when you invoke SAS. For more information about the USER system option, see “USER System Option” on page 570 and *SAS Language Reference: Dictionary*.

*Note:* You can assign other engines to the User libref if you want the data sets that are saved with one-level names to be stored in a format for use with other releases of SAS. △

## Using Large Data Sets with Windows and NTFS

If you run SAS under Windows using the Windows NT file system (NTFS), SAS automatically takes advantage of the 64-bit file I/O features. The size limit for a SAS data set under Windows with NTFS is 4 gigabytes in 32–bit systems. In 64–bit systems, 2 terabytes is the practical limit for physical and logical volumes using NTFS.

If you run SAS on a Windows server that supports extended server memory (more than 4 GB of memory), the extended server memory can be used as a cache for processing large data sets as well as the Work data library. For more information, see "Memory-Based Libraries" on page 199.

## Accessing SAS Files from Multiple SAS Sessions

If you are running multiple SAS sessions, whether on a single machine or across a network, you can have multiple access to the same SAS file when you are reading from it.

If you have SAS/SHARE installed, the VIEWTABLE window and the FSEDIT or FSVIEW windows allow multiple users to edit the same SAS file. When you edit a data set using the VIEWTABLE window, you can set the editing mode to either Table Level Edit Access or Row Level Edit Access. When you select Table Level Edit Access, only you have access to the data set. Row Level Edit access allows multiple users to access the same SAS file, but only one user can access and make changes to a single record (observation) at a time.

To open a data set in the VIEWTABLE window, from the SAS Explorer window:

1 double-click the Libraries icon
2 double-click the library containing the data set
3 double-click the data set.

To edit the data set, select

Edit ► Edit Mode

and then select either **`Table Level Edit`** or **`Row Level Edit`**.

When you edit a data set using FSEDIT or FSVIEW, you can set the update mode to either MEMBER or RECORD. When you select MEMBER mode, only you have access to the data set. When you select RECORD mode, multiple users can write to the same SAS file but only one user can update a single record (observation) at a time.

To open a data set using FSEDIT or FSVIEW:

1 type FSEDIT or FSVIEW in the command bar
2 double-click the library name in the Select a Member dialog box
3 double-click the data set name.

To edit the data set, select

Edit ► Update

and then select either the **`MEMBER`** or **`RECORD`** radio button.

The RSASUSER system option, described in "RSASUSER System Option" on page 540 allows you to share the Sasuser data library. If multiple users need update access to common SAS data sets, use SAS/SHARE software.

For details about rules for multiple user access to the same data set and its members, see the SAS Help and Documentation and *SAS/SHARE User's Guide*.

## Using SAS Files from Other Versions with SAS 9.1 for Windows

### Introduction to Using SAS Files from Other Versions with SAS 9.1 for Windows

SAS files that were created in Versions 8, 7, and 6 can be processed, with some restrictions, without having to convert files to the SAS 9.1 format.

SAS 9.1 file formats are the same as Version 7 and 8 file formats with the exception that with SAS 9.1 you can use longer format and informat names. The following table summarizes the actions that you need to take in order to use SAS files from a previous release, if the files in the SAS library are for the same release of SAS.

**Table 4.5** Summary of Using Version 6, 7 and 8 Data Sets and Catalogs in SAS 9.1

| **Version or Release** | **Data Sets** | **Catalogs** |
|---|---|---|
| Version 7 and 8 | No action is necessary. SAS reads, updates, and writes to Version 7 and Version 8 data sets. | No action is necessary. SAS reads, updates, and writes to Version 7 and Version 8 catalogs. |
| Releases 6.08 - 6.12 | In 32-bit environments, no action is necessary. SAS reads, updates, and writes to Version 6 data sets.<br>In 64-bit environments, the V6 engine is automatically detected. SAS can read a V6 data set but not write to a V6 data set. | Convert using the CPORT and CIMPORT procedures |
| Releases 6.04 and 6.03 | Use the V604 engine to read data. You cannot write to Release 6.04 and 6.03 data sets. | not supported |

As the table shows, except for Release 6.04 and Release 6.03 data sets, Version 6 (32–bit environments) and Version 7 and 8 data sets do not need to be converted to SAS 9.1 data sets in order for SAS 9.1 to read, update, and write to the data sets.

Version 7 and 8 catalogs also do not need to be converted to V9 catalogs. Version 6 SAS catalogs can only be read. If a Version 6 catalog is to be updated, you must convert it to a SAS 9.1 catalog.

"SAS 9.1 Compatibility with SAS Files From Earlier Releases" in *SAS Language Reference: Concepts* discusses in detail how to use or convert SAS files that were created in Release 6.08 through Version 8. See the *SAS/CONNECT User's Guide* for information about accessing Version 6 SAS files if you use Remote Library Services to access SAS files on a server.

To use SAS files that were created on an operating environment other than Windows, you will need to transport those files to the Windows environment. A separate document, *Moving and Accessing SAS Files*, discusses transporting files from one operating environment to another operating environment.

## Using Release 6.08 through Release 8.2 Data Sets

If your SAS library contains SAS files from only a single release of SAS, such as Release 6.12 or Version 8, SAS automatically determines the appropriate engine to use for these SAS data sets. If your SAS files are in a mixed mode library that possibly contains SAS data sets from multiple releases, you must specify the *engine* parameter on the LIBNAME statement or the engine will default to V9.

For example, if you know that the 'c:\mydata' SAS library contains only Version 6 files, the following SAS statements print a Version 6 SAS data set that is named WINDATA.SALEFIGS created under Windows 95:

```
libname windata 'c:\mydata';
proc print data=windata.salefigs;
   title 'Sales Figures';
run;
```

Where all SAS files in the library are Version 6 SAS files, you can omit the engine parameter because SAS automatically detects the V6 engine.

Using the same example, if you are unsure or if you know that the SAS library is a mixed mode library, you must specify the engine name in the LIBNAME statement to access the V6 files:

```
libname windata v6 'c:\mydata';
proc print data=windata.salefigs;
   title 'Sales Figures';
run;
```

Release 6.03 and Release 6.04 SAS files require a specific engine. For more information, see "Using Release 6.03 and Release 6.04 SAS Data Sets" on page 137.

## Using Release 6.03 and Release 6.04 SAS Data Sets

The V604 engine enables you to read from Release 6.03 and Release 6.04 SAS data sets directly from your SAS 9.1 session. (Remember that there is no difference between Release 6.04 and Release 6.03 SAS data sets.) This feature is useful when you have SAS data sets that you want to share between Release 6.04 for PCs and SAS 9.1 under Windows. The V604 engine is supported only for SAS data sets (member type DATA). For example, if you have a Release 6.04 SAS data set that is named MYLIB.FRUIT that you want to print, you can submit the following statements from a SAS 9.1 session:

```
libname mylib v604 'c:\sas604';
proc print data=mylib.fruit;
run;
```

## Converting Release 6.08 through Release 6.12 SAS Data Sets

While access to Version 6 SAS data sets is quite easy when you use the V6 engine, you might want to consider converting your SAS data sets to the SAS 9.1 format if you access them often and do not need to read the files from Version 6 anymore. The data set format of SAS 9.1 is more efficient than the Version 6 format and there are new SAS 9.1 features that cannot be used unless the data sets are converted.

The following SAS statements use the COPY procedure to convert all the Release 6 SAS data sets in the V6DATA SAS data library to SAS 9.1 and to format and store the new data sets in the WINDATA library:

```
libname v6data v6 'c:\mydata';
libname windata V9 'd:\newdata';
proc copy in=v6data out=windata;
run;
```

Alternatively, you can use the DATA step to do the conversion, as in the following example. This technique works well if you want to convert only one or two data sets in a particular SAS data library.

```
libname v604data v604 'c:\mydata';
libname windata V9 'd:\newdata';
data windata.eggplant;
   set v604data.eggplant;
run;
```

*Note:* Do not convert your Version 6 files to SAS 9.1 if you need to access the files from both versions. △

## Using Version 7 and 8 Catalogs in SAS 9.1

Because SAS 9.1 file formats are basically the same as Version 7 and 8 file formats, SAS 9.1 can read, update, and write to Version 7 and 8 catalogs without having to convert them to SAS 9.1 catalogs.

SAS running in a 64–bit Windows operating environments cannot read 32–bit catalogs.

## Converting Version 6 SAS Catalogs in SAS 9.1

Because of the differences in the internal structures of the operating environments, you must use the CPORT and CIMPORT procedures to convert Version 6 SAS catalogs created under Windows to SAS 9.1 format before you can use the catalogs in your SAS 9.1 session under Windows. Follow these steps:

1 Using the CPORT procedure in your Version 6 SAS session, create a transport file that contains the SAS catalog to be converted.
2 Transfer the file (perhaps on a network or diskette) to a location that your SAS 9.1 session can read.
3 Use the CIMPORT procedure from your SAS 9.1 session to read the transport file and create a converted SAS catalog.

For information about using the CPORT and CIMPORT procedures, see *Moving and Accessing SAS Files* and *Base SAS Procedures Guide*.

## Converting Release 6.08 SAS Catalogs to SAS 9.1

If you are converting directly from Release 6.08 to SAS 9.1, you can use the CPORT procedure in Release 6.08 to create a transport file, and then use the SAS 9.1 CIMPORT procedure to convert the catalog to a SAS 9.1 catalog. However, the HSERVICE and TOOLBOX catalog entries are not portable if you use CPORT from a Release 6.08 session.

An alternative way to convert Release 6.08 catalogs is to use the C16PORT procedure that is provided in Release 6.10 through Release 6.12. SAS provided the C16PORT procedure to convert the 16-bit catalogs that were created with Release 6.08 under Windows to a 32-bit format that SAS can use. You can use the C16PORT procedure from within one of these earlier releases of SAS to create a catalog that can later be read by SAS 9.1. (The C16PORT procedure is not available in SAS 9.1.)

To follow to convert your SAS catalogs from Release 6.08 under Windows to SAS 9.1:

1. While in your Release 6.10, Release 6.11, or Release 6.12 session, use the C16PORT procedure (described in the documentation for those releases) to create a transport file that contains the SAS catalog from Release 6.08.
2. Transfer the file (perhaps on a network or by using binary FTP) to a location where SAS can read it.
3. Use the CIMPORT procedure to read the transport file and create a converted SAS catalog.

If you want to convert a catalog that currently exists on another machine running Release 6.08 for Windows, you must first transfer the file (perhaps on a network or by using binary FTP) to a place where your SAS 9.1 session can read it.

The following example uses the C16PORT procedure in Release 6.12 to create a transport file from the INLIB.CAT catalog, then creates a Release 6.12 catalog (OUTLIB.CAT) using the CIMPORT procedure.

```
   /* Folder where catalog    */
   /* 'cat.sc2' resides       */
libname inlib 'c:\cat608';
   /* Folder where catalog    */
   /* 'cat.sc8' will reside   */
libname outlib 'c:\cat612';
proc c16port file='transprt' c=inlib.cat;
run;

/* Move the transprt file to a location where SAS can read it */
/* Once the file is accessible, run the following procedure.  */

proc cimport infile='transprt' c=outlib.cat;
run;
```

The Release 6.12 SAS catalog can now be read by SAS 9.1. For information about the CPORT and CIMPORT procedures, see *Base SAS Procedures Guide* and *Moving and Accessing SAS Files*.

## Converting Release 6.03 and Release 6.04 SAS Catalogs to SAS 9.1

If you want to convert Release 6.04 SAS catalogs to their SAS 9.1 counterparts, see *Moving and Accessing SAS Files*.

## Creating Release 6.08 through Release 6.12 Data Sets

You may need to create Release 6.08 through Release 6.12 data sets from your Windows SAS session. This is similar to reading Version 6 data sets in that you use the V6 engine. For example, the following SAS statements use the V6 engine to create a SAS data set named QTR1. The raw data are read from the external file associated with the fileref MYFILE.

```
libname windata v6 'c:\mydata';
filename myfile 'c:\qtr1data.dat';
data windata.qtr1;
   infile myfile;
   input saledate amount;
run;
```

## Using SAS 9.1 Files with Previous Releases

It is possible to move SAS files between your SAS 9.1 session under Windows and Release 6.08 through Release 8.2 of SAS using the CPORT and CIMPORT procedures.

*Note:* When you transport SAS 9.1 files to previous releases, SAS 9.1 features will not be available from the transport file. △

For information about PROC CPORT, PROC CIMPORT, and about back-porting SAS/AF applications, see *Base SAS Procedures Guide*. For more information about back-porting SAS/EIS applications, see the SAS Help for SAS/EIS. For more information about transporting files in general, see *Moving and Accessing SAS Files*.

## Using Remote Host SAS Files in SAS 9.1

You can directly access in SAS 9.1 SAS data sets that were created on a remote host under any previous version of SAS. For example, you may have a Version 5 SAS data set on VSE or a Version 8 data set on UNIX. Alternatively, you can create a transport data set and transport your file from the host to Windows.

A complete explanation of using remote host SAS files in SAS 9.1 can be found in *Moving and Accessing SAS Files*.

## Reading BMDP, OSIRIS and SPSS Files

SAS 9.1 provides three interface library engines that enable you to access external data files directly from a SAS program: the BMDP, OSIRIS, and SPSS engines. These engines are all read-only. Because they are sequential engines (that is, they do not support random access of data), these engines cannot be used with the POINT= option in the SET statement or with the FSBROWSE, FSEDIT, or FSVIEW procedures. You can use PROC COPY or a DATA step to copy the system file to a SAS data set and then perform these functions on the SAS data set. Also, because they are sequential engines, some procedures (such as the PRINT procedure) give a warning message that the engine is sequential. With these engines, the physical filename that is associated with a libref is an actual filename, not a folder. This is an exception to the rules concerning librefs.

You can also use the CONVERT procedure to convert BMDP, OSIRIS, and SPSS files to SAS data files. For more information, see "CONVERT Procedure" on page 426.

### BMDP Engine

The BMDP interface library engine enables you to read BMDP DOS files from the BMDP statistical software package directly from a SAS program. The following sections assume that you are familiar with the BMDP save file terminology.

To read a BMDP save file, you must issue a LIBNAME statement that explicitly specifies that you want to use the BMDP engine:

LIBNAME *libref* BMDP <'filename'>;

In this form of the LIBNAME statement, *libref* is a SAS libref and *filename* is the BMDP physical filename. If the libref appears previously as a fileref, you can omit *filename* because the physical filename that is associated with the fileref is used. This engine can read only BMDP save files created under DOS.

Because there can be multiple save files in a single physical file, you reference the CODE= value as the member name of the data set within the SAS language. For example, if the save file contains CODE=ABC and CODE=DEF and the libref is MYLIB, you reference them as MYLIB.ABC and MYLIB.DEF. All CONTENT types are treated the same. Therefore, even if member DEF is CONTENT=CORR, it is treated as CONTENT=DATA.

If you know that you want to access the first save file in the physical file, or if there is only one save file, you can refer to the member name as _FIRST_. This is convenient if you do not know the CODE= value.

### BMDP Engine Examples

In the following example, the physical file MYBMDP.DAT contains the save file ABC. This example associates the libref MYLIB with the BMDP physical file, then runs the CONTENTS and PRINT procedures on the save file:

```
libname mylib bmdp 'mybmdp.dat';
proc contents data=mylib.abc;
run;
proc print data=mylib.abc;
run;
```

The following example uses the LIBNAME statement to associate the libref MYLIB2 with the BMDP physical file. Then it prints the data for the first save file in the physical file:

```
libname mylib2 bmdp 'mybmdp.dat';
proc print dat=mylib2._first_;
run;
```

## OSIRIS Engine

Because the Inter-University Consortium on Policy and Social Research (ICPSR) uses the OSIRIS file format for distribution of its data files, SAS provides the OSIRIS interface library engine to support ICPSR data users and to be compatible with PROC CONVERT, which is described in "CONVERT Procedure" on page 426.

The read-only OSIRIS engine enables you to read OSIRIS data and dictionary files directly from a SAS program. These files must be stored in EBCDIC format. This means that you must have downloaded the OSIRIS files from your host computer in binary format. The following section assumes that you are familiar with the OSIRIS file terminology. *

To read an OSIRIS file, you must issue a LIBNAME statement that explicitly specifies you want to use the OSIRIS engine. In this case, the LIBNAME statement takes the following form:

* See documentation provided by the Institute for Social Research for more information.

**LIBNAME** *libref* OSIRIS '*data-filename*' DICT='*dictionary-filename*';

In this form of the LIBNAME statement, *libref* is a SAS libref, *data-filename* is the physical filename of the OSIRIS data file, and *dictionary-filename* is the physical filename of the OSIRIS dictionary file. The *dictionary-filename* argument can also be an environment variable name or a fileref. (Do not use quotation marks if it is an environment variable name or fileref.) The DICT= option must appear because the engine requires both files.

OSIRIS data files do not have member names. Therefore, you can use whatever member name you like. You can use the same OSIRIS dictionary file with different OSIRIS data files. Write a separate LIBNAME statement for each one.

The layout of an OSIRIS data dictionary is consistent across operating environments. The reason is that the OSIRIS software does not run outside the z/OS environment, but the engine is designed to accept an z/OS data dictionary on any other operating environment under which SAS runs. It is important that the OSIRIS dictionary and data files not be converted from EBCDIC to ASCII; the engine expects EBCDIC data. There is no specific file layout for the OSIRIS data file. The file layout is dictated by the contents of the OSIRIS dictionary file.

### OSIRIS Engine Example

In the following example, the data file is MYOSIRIS.DAT, and the dictionary file is MYOSIRIS.DIC. The example associates the libref MYLIB with the OSIRIS files and then runs PROC CONTENTS and PROC PRINT on the data:

```
libname mylib osiris 'myosiris.dat'
        dict='myosiris.dic';
proc contents data=mylib._first_;
run;
proc print data=mylib._first_;
run;
```

## SPSS Engine

The SPSS interface library engine enables you to read SPSS export files directly from a SAS program. The SPSS export file must be created by using the SPSS EXPORT command. **The SPSS engine is a read-only engine.

To read an SPSS export file you must issue a LIBNAME statement that explicitly specifies that you want to use the SPSS engine. In this case, the LIBNAME statement takes the following form:

**LIBNAME** *libref* SPSS <'*filename*'>;

In this form of the LIBNAME statement, the *libref* argument is a SAS libref, and *filename* is the SPSS physical filename, including the file extension. If the libref appears also as a fileref, you can omit *filename* because the physical filename that is associated with the fileref is used. The SPSS native file format is not supported. Export files can originate from any operating environment.

Because SPSS files do not have internal names, you can refer to them by any member name that you like. (The example in this discussion uses _FIRST_ .)

** See documentation provided by SPSS Inc. for more information.

### SPSS Engine Example

The following example associates the libref MYLIB with the physical file MYSPSS.POR in order to run PROC CONTENTS and PROC PRINT on the save file:

```
libname mylib spss 'myspss..por';
proc contents data=mylib._first_;
run;
proc print data=mylib._first_;
run;
```

## Transferring SAS Files between Operating Environments

For complete information on transferring SAS files between operating environments, see *Moving and Accessing SAS Files*.

## Accessing Database Files with SAS/ACCESS Software

SAS/ACCESS software provides an interface between SAS and several database management systems (DBMS) that run under Windows. The interface consists of three procedures and an interface view engine, which can perform the following tasks:

LIBNAME statement
: by assigning the engine to a specific database engine, the LIBNAME statement lets you reference a DBMS object directly in a DATA step or SAS procedure, enabling you to read from and write to a DBMS object a though it were a SAS data set.

SQL Procedure Pass-Through Facility
: accesses data from several different relational DBMSs, including Oracle and SQLServer.

interface view engine
: enables you to use descriptor files in SAS programs to access DBMS data directly and enables you to specify descriptor files in SAS programs to update, insert, or delete DBMS data directly.

For more information about using SAS/ACCESS software under Windows, consult *SAS/ACCESS for PC Files: Reference* and other available SAS/ACCESS documentation.

## Using the SAS ODBC Driver to Access SAS Data from Other Applications

The SAS ODBC driver is an implementation of the open database connectivity (ODBC) standard that enables you to access, manipulate, and update SAS data sources. These data sources can include SAS data sets, flat files, VSAM files, as well as data from any database management system (DBMS) for which you have licensed SAS/ACCESS software. For information about how to access data from other Windows applications that comply with the ODBC standard, see the SAS Help and Documentation.

The SAS ODBC Driver accesses data by communicating with either a local or remote (SAS/SHARE) SAS server session using the TCP/IP protocol. The TCP/IP protocol enables users to access remote SAS servers on a variety of host platforms. A SAS server is a SAS procedure (either PROC SERVER or PROC ODBCSERV) that runs in its own SAS session; it accepts input and output requests from other SAS sessions and from the SAS ODBC driver on behalf of the ODBC-compliant application. For remote access to SAS data, a SAS server must be installed on the server machine, but not on the client machine.

The SAS ODBC Driver is included with Base SAS. Remote server configurations that use the SAS ODBC driver require that these SAS products be installed:

- Base SAS
- SAS/SHARE.

For details about installing and configuring the SAS ODBC Driver, see the installation documentation for SAS under Windows. For more information on configuring and using the SAS ODBC Driver, see *SAS ODBC Driver: User's Guide and Programmer's Reference*.

CHAPTER

5

# Using External Files

*About External Files* **146**
*Referencing External Files* **146**
- *Accessing External Files* **146**
- *Using a Fileref* **147**
  - *Assigning File Shortcuts* **147**
  - *Using the FILENAME Statement* **147**
  - *Using Environment Variables* **148**
  - *Using the SET system option* **149**
  - *Using the SET command* **149**
  - *Assigning a Fileref to a Directory* **149**
  - *Assigning a Fileref to Concatenated Directories* **151**
  - *Summary of Rules for Resolving Member-Name Syntax* **151**
  - *Assigning a Fileref to Concatenated Files* **152**
  - *Referencing External Files with Long Filenames* **152**
  - *Referencing Files Using UNC Paths* **152**
  - *Listing Fileref Assignments* **153**
  - *Clearing Filerefs* **153**
  - *Understanding How Concatenated Directories Are Accessed* **153**
  - *Input* **154**
  - *Output* **154**
  - *Understanding How Concatenated Files Are Accessed* **154**
  - *Input* **154**
  - *Output* **154**
- *Using a Quoted Windows Filename* **155**
  - *Using Reserved Operating System Physical Names* **155**
- *Using a File in Your Working Directory* **156**

*Accessing External Files with SAS Statements* **156**
- *Using the FILE Statement* **156**
- *Using the INFILE Statement* **157**
- *Using the %INCLUDE Statement* **158**

*Accessing External Files with SAS Commands* **158**
- *Using the FILE Command* **159**
- *Using the INCLUDE Command* **159**
- *Using the GSUBMIT Command* **160**

*Advanced External I/O Techniques* **160**
- *Altering the Record Format* **160**
- *Appending Data to an External File* **161**
- *Determining Your Drive Mapping* **161**
- *Reading External Files with National Characters* **162**

*Reading Data from the Communications Port* **162**
- *Communications Port Timeouts* **163**

*Options that Relate to Communications Port Timeouts* 163

## About External Files

*External files* are files that contain data or text, such as SAS programming statements, records of raw data, or procedure output. SAS can use these files, but they are not managed by SAS.

*SAS Language Reference: Concepts* contains basic, platform-independent information on external files.

For information on how to access external files containing transport data libraries, see the SAS Customer Support Center Web page, http://support.sas.com.

## Referencing External Files

### Accessing External Files

To access external files, you must tell SAS how to find the files. Use the following statements to access external files:

FILENAME
: associates a fileref with an external file that is used for input or output.

FILE
: opens an external file for writing data lines. Use the PUT statement to write lines.

INFILE
: opens an external file for reading data lines. Use the INPUT statement to read lines.

%INCLUDE
: opens an external file and reads SAS statements from that file. (No other statements are necessary.)

These statements are discussed in the section “SAS Statements under Windows” on page 441, and in the SAS statements section in *SAS Language Reference: Dictionary*.

You can also specify external files in various SAS dialog entry fields (for example, as a file destination in the Save As dialog), the FILENAME function, and in SAS commands, such as FILE and INCLUDE.

Depending on the context, SAS can reference an external file by using:

- a fileref assigned with the FILENAME statement or function
- an environment variable defined with either the SET system option or the Windows SET command
- a Windows filename enclosed in quotes
- member-name syntax (also called aggregate syntax)
- a single filename within quotation marks (a file in the working directory).

The following sections discuss these methods of specifying external files.

Because there are several ways to specify external files in SAS, SAS uses a set of rules to resolve an external file reference and uses this order of precedence:

1. Check for a standard Windows file specification enclosed in quotes.
2. Check for a fileref defined by a FILENAME statement or function.

3 Check for an environment variable fileref.
4 Assume the file is in the working directory.

In other words, SAS assumes an external file reference is a standard Windows file specification. If it is not, SAS checks to see if the file reference is a fileref (defined by either a FILENAME statement, FILENAME function, or an environment variable). If the file reference is none of these, SAS assumes it is a filename in the working directory. If the external file reference is not valid for one of these choices, SAS issues an error message indicating that it cannot access the external file.

## Using a Fileref

One way to reference external files is with a *fileref*. A fileref is a logical name associated with an external file. You can assign a fileref with a File Shortcut in the SAS Explorer window, the My Favorite Folders window, the FILENAME statement, the FILENAME function, or you can use a Windows environment variable to point to the file. This section discusses the different ways to assign filerefs and also shows you how to obtain a listing of the active filerefs and clear filerefs during your SAS session.

### Assigning File Shortcuts

In an interactive SAS session, you can use the SAS Explorer window or the My Favorite Folders window to create filerefs. The SAS Explorer File Shortcuts folder contains a listing of active filerefs. To create a new fileref from SAS Explorer:

1 Select the File Shortcuts folder and then select

   File ▶ New

2 In the File Shortcut Assignment window, enter the name of the shortcut (fileref) and the path to the SAS file that the shortcut represents.
3 Optionally, check **`Enable at Startup`** to reassign the shortcut for all subsequent SAS sessions.

To assign a file shortcut using the My Favorite Folders window:

1 Open the folder that contains the file.
2 Position the cursor over the file, right mouse click and select **`Create File Shortcut`**.
3 In the Create File Shortcut dialog box, type the name of the file shortcut and press Enter or click **`OK`**.

You can then use these file shortcuts in your SAS programs.

*Note:* File Shortcuts are active only during the current SAS session. △

### Using the FILENAME Statement

The FILENAME statement provides a means to associate a logical name with an external file or directory.

*Note:* The syntax of the FILENAME function is similar to the FILENAME statement. For information about the FILENAME function, see *SAS Language Reference: Dictionary*. △

The simplest syntax of the FILENAME statement is as follows:

FILENAME *fileref* "*external-file*";

For example, if you want to read the file C:\MYDATA\SCORES.DAT, you can issue the following statement to associate the fileref MYDATA with the file C:\MYDATA\SCORES.DAT:

```
filename mydata "c:\mydata\scores.dat";
```

Then you can use this fileref in your SAS programs. For example, the following statements create a SAS data set named TEST, using the data stored in the external file referenced by the fileref MYDATA:

```
data test;
   infile mydata;
   input name $ score;
run;
```

*Note:* The words AUX, CON, NUL, PRN, LPT1 - LPT9, and COM1 - COM9 are reserved words under Windows. Do not use these words as filerefs. △

You can also use the FILENAME statement to concatenate directories of external files and to concatenate multiple individual external files into one logical external file. These topics are discussed in “Assigning a Fileref to Concatenated Directories” on page 151 and “Assigning a Fileref to Concatenated Files” on page 152 .

The * and ? wildcards can be used in either the external file name or file extension for matching input file names. Use * to match one or more characters and the ? to match a single character. Wildcards are supported for input only in the FILENAME and INFILE statements, and in member-name syntax (aggregate syntax). Wildcards are not valid in the FILE statement. The following filename statement reads input from every file in the current directory that begins with the string **wild** and ends with **.dat**:

```
filename wild 'wild*.dat';
data;
   infile wild;
   input;
run;
```

The following example reads all files in the current working directory:

```
filename allfiles '*.*';
data;
   infile allfiles;
   input;
run;
```

The FILENAME statement accepts various options that enable you to associate device names, such as printers, with external files and to control file characteristics, such as record format and length. Some of these options are illustrated in “Advanced External I/O Techniques” on page 160. For the complete syntax of the FILENAME statement, refer to .

## Using Environment Variables

Just as you can define an environment variable to serve as a logical name for a SAS data library (see “Assigning SAS Libraries Using Environment Variables” on page 128), you can also use an environment variable to refer to an external file. You can choose either to define a SAS environment variable using the SET system option or to define a Windows environment variable using the Windows SET command. Alternatively, you can define environment variables using the System dialog box, accessed from the Control Panel.

*Note:* The words AUX, CON, NUL, PRN, LPT1 - LPT9, and COM1 - COM9 are reserved words under Windows. Do not use these words as environment variables. △

The availability of environment variables makes it simple to assign resources to SAS prior to invocation. However, the environment variables you define (using the SET system option) for a particular SAS session are not available to other applications.

### Using the SET system option

For example, to define a SAS environment variable that points to the external file C:\MYDATA\TEST.DAT, you can use the following SET option in your SAS configuration file:

```
-set myvar c:\mydata\test.dat
```

Then, in your SAS programs, you can use the environment variable MYVAR to refer to the external file:

```
data mytest;
   infile myvar;
   input name $ score;
run;
```

It is recommended that you use the SET system option in your SAS configuration file if you invoke SAS using the Windows Start menu.

### Using the SET command

An alternative to using the SET system option to define an environment variable is to use the Windows SET command. For example, the Windows SET command that equates to the previous example is

```
SET MYVAR=C:\MYDATA\TEST.BAT
```

You can also define SET commands by using System Properties dialog box that you access from the Control Panel.

You must issue all the SET commands that define your environment variables before you invoke SAS. If you define an environment variable in an MS-DOS window, and then start SAS from the Start menu, SAS will not recognize the environment variable.

## Assigning a Fileref to a Directory

You can assign a fileref to a directory and then access individual files within that directory using member-name syntax (also called aggregate syntax).

For example, if all your regional sales data for January are stored in the directory C:\SAS\MYDATA, you can issue the following FILENAME statement to assign the fileref JAN to this directory:

```
filename jan "c:\sas\mydata";
```

Now you can use this fileref with a member name in your SAS programs. In the following example, you reference two files stored in the JAN directory:

```
data westsale;
   infile jan(west);
   input name $ 1-16 sales 18-25
         comiss 27-34;
run;
data eastsale;
   infile jan(east);
   input name $ 1-16 sales 18-25
         comiss 27-34;
run;
```

When you use member-name syntax, you do not have to specify the file extension for the file you are referencing, as long as the file extension is the expected one. For instance, in the previous example, the INFILE statement expects a file extension of .DAT. The following table lists the expected file extensions for the various SAS statements and commands:

**Table 5.1** Default File Extensions for Referencing External Files with Member-Name Syntax

| SAS Command or Statement | SAS Window | File Extension |
|---|---|---|
| FILE statement | EDITOR | .DAT |
| %INCLUDE statement | EDITOR | .SAS |
| INFILE statement | EDITOR | .DAT |
| FILE command | EDITOR | .SAS |
| FILE command | LOG | .LOG |
| FILE command | OUTPUT | .LST |
| FILE command | NOTEPAD | none |
| INCLUDE command | EDITOR | .SAS |
| INCLUDE command | NOTEPAD | none |

For example, the following program submits the file C:\PROGRAMS\TESTPGM.SAS to SAS:

```
filename test "c:\programs";
%include test(testpgm);
```

SAS searches for a file named TESTPGM.SAS in the directory C:\PROGRAMS.

If your file has a file extension different from the default file extension, you can use the file extension in the filename, as in the following example:

```
filename test "c:\programs";
%include test(testpgm.xyz);
```

If your file has no file extension, you must enclose the filename in quotes, as in the following example:

```
filename test "c:\programs";
%include test("testpgm");
```

To further illustrate the default file extensions SAS uses, here are some more examples using member-name syntax. Assume the following FILENAME statement has been submitted:

```
filename test "c:\mysasdir";
```

The following example opens the file C:\MYSASDIR\PGM1.DAT for output:

```
file test(pgm1);
```

The following example opens the file C:\MYSASDIR\PGM1.DAT for input:

```
infile test(pgm1);
```

The following example reads and submits the file C:\MYSASDIR\PGM1:

```
%include test("pgm1");
```

These examples use SAS statements. SAS commands, such as the FILE and INCLUDE commands, also accept member-name syntax and have the same default file extensions as shown in Table 5.1 on page 150.

Another feature of member-name syntax is that it enables you to reference a subdirectory in the working directory without using a fileref. As an example, suppose you have a subdirectory named PROGRAMS that is located beneath the working directory. You can use the subdirectory name PROGRAMS when referencing files within this directory. For example, the following statement submits the program stored in *working-directory*\PROGRAMS\PGM1.SAS:

```
%include programs(pgm1);
```

The next example uses the FILE command to save the contents of the active window to *working-directory*\PROGRAMS\TESTPGM.DAT:

```
file programs(testpgm);
```

*Note:* If a directory name is the same as a previously defined fileref, the fileref takes precedence over the directory name. △

## Assigning a Fileref to Concatenated Directories

Member-name syntax is also handy when you use the FILENAME statement to concatenate directories of external files. For example, suppose you issue the following FILENAME statement:

```
filename progs ("c:\sas\programs",
                "d:\myprogs");
```

This statement tells SAS that the fileref PROGS refers to all files stored in both the C:\SAS\PROGRAMS and the D:\MYPROGS directories. When you use the fileref PROGS in your SAS program, SAS looks in these directories for the member you specify. When you use this concatenation feature, you should be aware of the protocol SAS uses, which depends on whether you are accessing the files for read, write, or update. For more information, see "Understanding How Concatenated Directories Are Accessed" on page 153.

## Summary of Rules for Resolving Member-Name Syntax

SAS resolves an external file reference that uses member-name syntax by using a set of rules. For example, suppose your external file reference in a SAS statement or command is the following:

```
progs(member1)
```

SAS uses the following set of rules to resolve this external file reference. This list represents the order of precedence:

1 Check for a fileref named PROGS defined by a FILENAME statement.

2 Check for a SAS or Windows environment variable named PROGS.

3 Check for a directory named PROGS beneath the working directory.

The member name must be a valid physical filename. If no extension is given (as in the previous example), SAS uses the appropriate default extension, as given in Table 5.1 on page 150. If the extension is given or the member name is quoted, SAS does not assign an extension, and it looks for the filename exactly as it is given.

## Assigning a Fileref to Concatenated Files

You can specify concatenations of files when reading external files from within SAS. Concatenated files consist of two or more file specifications (which may contain wildcard characters) separated by blanks or commas. Here are some examples of valid concatenation specifications:

- **`filename allsas ("one.sas", "two.sas", "three.sas");`**
- **`filename alldata ("test1.dat" "test2.dat" "test3.dat");`**
- **`filename allinc "test*.sas";`**
- **`%include allsas;`**
- **`infile alldata;`**
- **`include allinc`**

When you use this concatenation feature, you should be aware of the protocol SAS uses, which depends on whether you are accessing the files for read, write, or update. For more information, see "Understanding How Concatenated Files Are Accessed" on page 154.

*Note:* Do not confuse concatenated file specifications with concatenated directory specifications, which are also valid and are illustrated in "Assigning a Fileref to Concatenated Directories" on page 151. △

## Referencing External Files with Long Filenames

SAS supports the use of long filenames. (For more information about valid long filenames, see your Windows operating environment documentation.) You can use long filenames whenever you specify a filename as an argument to a dialog, command, or any aspect of the SAS language.

When specifying external filenames with the SAS language, such as in a statement or function, you should enclose the filename in double quotes to reduce ambiguity (since a single quote is a valid character in a long filename). When you need to specify multiple filenames, enclose each filename in double quotes and delimit the names with a blank space.

Here are some examples of valid uses of long filenames within SAS:

- **`libname abc "My data file";`**
- **`filename myfile "Bernie's file";`**
- **`filename summer ("June sales" "July sales" "August sales");`**
- **`include "A really, really big SAS program";`**

## Referencing Files Using UNC Paths

SAS supports the use of the Universal Naming Convention (UNC) paths. UNC paths let you connect your computer to network devices without having to refer to a network drive letter. SAS supports UNC paths to the extent that Windows and your network software support them. In general, you can refer to a UNC path anywhere in SAS where you would normally refer to a network drive.

UNC paths have the following syntax:

\\SERVER\SHARE\FOLDER\FILEPATH

where

SERVER
: is the network file server name.

SHARE

is the shared volume on the server.

FOLDER
: is one of the directories on the shared volume.

FILEPATH
: is a continuation of the file path, which might reference one or more subdirectories.

For example, the following command includes a file from the network file server ZAPHOD:

```
include "\\zaphod\universe\galaxy\stars.sas"
```

## Listing Fileref Assignments

If you have assigned several filerefs during a SAS session and need to refresh your memory as to which fileref points where, you can use either the SAS Explorer window or the FILENAME statement to list all the assigned filerefs.

To use the SAS Explorer window to list the active filerefs, double-click on File Shortcuts. The Explorer window lists all the filerefs active for your current SAS session. Any environment variables you have defined as filerefs are listed, provided you have used them in your SAS session. If you have defined an environment variable as a fileref but have not used it yet in a SAS program, the fileref is not listed in the Explorer window.

You can use the following FILENAME statement to write the active filerefs to the SAS log:

```
filename _all_ list;
```

## Clearing Filerefs

You can clear a fileref by using the following syntax of the FILENAME statement:

FILENAME *fileref*|_ALL_ <CLEAR>;

If you specify a fileref, only that fileref is cleared. If you specify the keyword _ALL_, all the filerefs you have assigned during your current SAS session are cleared.

To clear filerefs using the SAS Explorer File Shortcuts:

1. select the File Shortcuts you want to delete. To select all File Shortcuts, select

   Edit ▶ Select All

2. press the Delete key or select

   Edit ▶ Delete

3. Click **OK** in the message box to confirm deletion of the File shortcuts.

*Note:* You cannot clear a fileref that is defined by an environment variable. Filerefs that are defined by an environment variable are assigned for the entire SAS session. △

SAS automatically clears the association between filerefs and their respective files at the end of your job or session. If you want to associate the fileref with a different file during the current session, you do not have to end the session or clear the fileref. SAS automatically reassigns the fileref when you issue a FILENAME statement for the new file.

## Understanding How Concatenated Directories Are Accessed

When you associate a fileref with more than one physical directory, which file is accessed depends upon whether it is being accessed for input or output.

### Input

If the file is opened for input or update, the first file found that matches the member name is accessed. For example, if you submit the following statements, and the file PHONE.DAT exists in both the C:\SAMPLES and C:\TESTPGMS directories, the one in C:\SAMPLES is read:

```
filename test ("c:\samples","c:\testpgms");
data sampdat;
   infile test(phone.dat);
   input name $ phonenum $ city $ state $;
run;
```

### Output

When you open a file for output, SAS writes to the file in the first directory listed in the FILENAME statement, even if a file by the same name exists in a later directory. For example, suppose you input the following FILENAME statement:

```
filename test ("c:\sas","d:\mysasdir");
```

Then, when you issue the following FILE command, the file SOURCE.PGM is written to the C:\SAS directory, even if a file by the same name exists in the D:\MYSASDIR directory:

```
file test(source.pgm)
```

## Understanding How Concatenated Files Are Accessed

When you associate a fileref with more than one physical file, the behavior of SAS statements and commands depends on whether you are accessing the files for input or output.

### Input

If the file is opened for input, data from all files are input. For example, if you issue the following statements, the %INCLUDE statement submits 4 programs for execution:

```
filename mydata ("qtr1.sas","qtr2.sas",
                 "qtr3.sas","qtr4.sas");
%include mydata;
```

### Output

If the file is opened for output, data are written to the first file in the concatenation. For example, if you issue the following statements, the PUT statement writes to MYDAT1.DAT:

```
filename indata "dogdat.dat";
filename outdata ("mydat1.dat","mydat2.dat",
                  "mydat3.dat","mydat4.dat");
data _null_;
   infile indata;
   input name breed color;
   file outdata;
   put name= breed= color=;
run;
```

## Using a Quoted Windows Filename

Instead of using a fileref to refer to external files, you can use a quoted Windows filename. For example, if the file C:\MYDIR\ORANGES.SAS contains a SAS program you want to invoke, you can issue the following statement:

```
%include "c:\mydir\oranges.sas";
```

When you use a quoted Windows filename in a SAS statement, you can omit the drive and directory specifications if the file you want to reference is located in the working directory. For instance, if in the previous example the working directory is C:\MYDIR, you can submit this statement:

```
%include "oranges.sas";
```

### Using Reserved Operating System Physical Names

You can use several reserved names as quoted physical filenames. Reserved operating system physical names enable you to do a variety of things, such as read data directly from the communications port (such as COM1).The following table lists these physical names and their corresponding device-type keywords:

**Table 5.2** Reserved Windows Physical Names

| Physical Name | Device Type | Use |
|---|---|---|
| COM1–COM9 | COMMPORT | Read/write from the communications port. |
| NUL | DUMMY | Discard data. This name is useful in testing situations. |

You can specify operating system physical names with or without a colon. For example, you can specify either COM1: or COM1. For additional information, see your Windows documentation.

The following example demonstrates how to capture data from an external device or application that is transmitting data via a serial (RS-232C port).

```
options noxwait xsync;
data _null_;
   if symget("sysscpl") = "WIN_NT" then
          rc = system("mode COM1:9600,n,8,1,xon=on");
      stop;
run;

filename commdata commport "COM1:";

data fruit;
      keep num type;
      infile commdata unbuffered;
      file commdata;
      put "ready";
      input totrecs records $;
      if totrecs = . or records ne "RECORDS" then
        do;
          file log;
          put "ERROR: Unable to determine
                number of records to read.";
```

```
         stop;
       end;
     do i = 1 to totrecs;
       input num type $;
       output;
       put "NEXT";
     end;
     stop;
run;
```

Note the use of the device-type keyword COMMPORT in the FILENAME statement in this example. Because the access protocols for devices are slightly different from the access protocols for files, you should always use the appropriate device-type keyword in combination with the reserved physical name in the FILENAME statement. If you do not use a device-type keyword, SAS defaults to using the access protocols for files, not for devices.

For more information about available device-type keywords in the FILENAME statement, see "SAS Statements under Windows" on page 441. "Reading Data from the Communications Port" on page 162 discusses the access protocols for using a communications port device.

## Using a File in Your Working Directory

If you store the external files you need to access in your working directory and they have the expected file extensions (see Table 5.1 on page 150), you can simply refer to the filename, without quotes or file extensions, in a SAS statement. For example, if you have a file named ORANGES.SAS stored in your working directory and ORANGES is not defined as a fileref, you can submit the file with the following statement:

```
%include oranges;
```

Remember, though, that using this type of file reference requires that

- □ the file is stored in the working directory
- □ the file has the correct file extension
- □ the filename is not also defined as a fileref.

For more information about how to determine and change the SAS working directory, see "Determining the Current Folder When SAS Starts" on page 9 and "Changing the SAS Current Folder" on page 37.

## Accessing External Files with SAS Statements

This section presents simple examples of using the FILE, INFILE, and %INCLUDE statements to access external files. For more complex examples of using these statements under Windows, see "Advanced External I/O Techniques" on page 160.

### Using the FILE Statement

The FILE statement enables you to direct lines written by a PUT statement to an external file.*

Here is a simple example using the FILE statement. This example reads the data in the SAS data set MYLIB.TEST and writes only those scores greater than 95 to the external file C:\MYDIR\TEST.DAT:

```
filename test "c:\mydir\test.dat";
libname mylib "c:\mydata";
data _null_;
   set mylib.test;
   file test;
   if score ge 95 then
      put score;
run;
```

The previous example illustrates writing the value of only one variable of each observation to an external file. The following example uses the _ALL_ option in the PUT statement to copy all variables in the current observation to the external file if the variable REGION contains the value **west**.

```
libname us "c:\mydata";
data west;
   set us.pop;
   file "c:\sas\pop.dat";
   where region="west";
   put _all_;
run;
```

This technique of writing out entire observations is particularly useful if you need to write variable values in a SAS data set to an external file so that you can use your data with another application that cannot read data in a SAS data set format.

*Note:* This example uses the _ALL_ keyword in the PUT statement. This code generates *named output*, which means that the variable name, an equals sign (=), and the variable value are all written to the file. Consider this when reading the data later. For more information about named output, see the description of the PUT statement in *SAS Language Reference: Dictionary*. △

The FILE statement also accepts several options. These options enable you to control the record format and length. Some of these options are illustrated in "Advanced External I/O Techniques" on page 160. For the complete syntax of the FILE statement, see .

*Note:* The default record length used by the FILE statement is 256 characters. If the data you are saving contains records that are longer than this, you must use the FILENAME statement to define a fileref and either use the LRECL= option in the FILENAME statement to specify the correct logical record length or specify the LRECL= option in the FILE statement. For details about the LRECL= option, see LRECL= in . △

## Using the INFILE Statement

Use the INFILE statement to specify the source of data read by the INPUT statement in a SAS DATA step. The INFILE statement is always used in conjunction with an INPUT statement, which defines the location and type of data being read.

Here is a simple example of the INFILE statement. This DATA step reads the specified data from the external file and creates a SAS data set named SURVEY:

* You can also use the FILE statement to direct PUT statement output to the SAS log or to the same destination as procedure output. For more information, see *SAS Language Reference: Dictionary*.

```
filename mydata "c:\mysasdir\survey.dat";
data survey;
   infile mydata;
   input fruit $ taste looks;
run;
```

Of course, you can use a quoted Windows filename instead of a fileref:

```
data survey;
   infile "c:\mysasdir\survey.dat";
   input fruit $ taste looks;
run;
```

The INFILE statement also accepts other options. These options enable you to control the record format and length. Some of these options are illustrated in "Advanced External I/O Techniques" on page 160. For the complete syntax of the INFILE statement, see .

*Note:* The default record length used by the INFILE statement is 256 characters. If the data you are reading have records longer than this, you must use the FILENAME statement to define a fileref and either use the LRECL= option in the FILENAME statement to specify the correct logical record length or specify the LRECL= option in the INFILE statement. For details about the LRECL= option, see LRECL= in . △

## Using the %INCLUDE Statement

When you submit an %INCLUDE statement, it reads an entire file into the current SAS program you are running and submits that file to SAS immediately. A single SAS program can have as many individual %INCLUDE statements as necessary, and you can nest up to ten levels of %INCLUDE statements. Using the %INCLUDE statement makes it easier for you to write modular SAS programs.

Here is an example that submits the statements stored in C:\SAS\MYJOBS\PROGRAM1.SAS using the %INCLUDE statement and member-name syntax:

```
filename job "c:\sas\myjobs";
%include job(program1);
```

The %INCLUDE statement also accepts several options. These options enable you to control the record format and length. Some of these options are illustrated in "Advanced External I/O Techniques" on page 160. For the complete syntax of the %INCLUDE statement, see .

*Note:* The default record length used by the %INCLUDE statement is 256 characters. If the program you are reading has records longer than this, you must use the FILENAME statement to define a fileref and either use the LRECL= option in the FILENAME statement to specify the correct logical record length or specify the LRECL= option in the %INCLUDE statement. For details about the LRECL= option, see LRECL= in . △

# Accessing External Files with SAS Commands

This section illustrates how to use the FILE and INCLUDE commands to access external files. Commands provide the same service as the Save As and Open dialog boxes. The method that you use to access external files depends on the needs of your SAS application and your personal preference.

## Using the FILE Command

The FILE command has a different use than the FILE statement; the FILE command writes the current contents of a window to an external file rather than merely specifying, for example, a destination for PUT statement output in a DATA step.

For example, if you want to save the contents of the LOG window to an external file named C:\SASLOGS\TODAY.LOG, you can issue the following FILE command from the Command dialog box; however, the LOG window must be active:

```
file "c:\saslogs\today.log"
```

If you have already defined the fileref LOGS to point to the SASLOGS directory, you can use the following FILE command:

```
file logs(today)
```

In this case, the file extension defaults to .log, as shown in Table 5.1 on page 150.

If you use the FILE command to attempt to write to an already existing file, a dialog box enables you to replace the existing file, append the contents of the window to the existing file, or cancel your request.

If you issue the FILE command with no arguments, the contents of the window are written to the file that is referenced in the last FILE command. This is useful if you are editing a program and want to save it often. However, the dialog box that prompts you about replacing or appending appears only the first time that you issue the FILE command. Thereafter, unless you specify the filename in the FILE command, it uses the parameters that you specified earlier (replace or append) without prompting you.

Choosing **`Save As`** from the SAS main window **`File`** menu displays the Save As dialog box. This dialog box performs the same function as the FILE command, but it is more flexible in that it gives you more choices and is more interactive than the FILE command. For more information, see “Saving Files” in “Using the Enhanced Editor Window” on page 82 and in “Using the Program Editor” on page 108.

The FILE command also accepts several options. These options enable you to control the record format and length. Some of these options are illustrated in “Advanced External I/O Techniques” on page 160. For the complete syntax of the FILE command, see “FILE Command” on page 342.

## Using the INCLUDE Command

The INCLUDE command, like the %INCLUDE statement, can be used to copy an entire external file into the Editor window, the NOTEPAD window, or whatever window is active. In the case of the INCLUDE command, however, the file is simply copied to the window and is not submitted.

For example, suppose you want to copy the file C:\SAS\PROG1.SAS into the Editor window. If you have defined a fileref SAMPLE to point to the correct directory, you can use the following INCLUDE command from the Command dialog box (if the Editor is the active window) to copy the member PROG1 into the Editor window:

```
include sample(prog1)
```

Another way to copy files into your SAS session is to use the Open dialog box. In addition to copying files, the Open dialog box gives you other choices, such as invoking the program that you are copying. The Open dialog box is the most flexible way for you to copy files into the Editor window. For more information, see the Opening Files in “Using the Enhanced Editor Window” on page 82 and in “Using the Program Editor” on page 108.

The INCLUDE command also accepts several arguments. These arguments enable you to control the record format and length. Some of these arguments are illustrated in "Advanced External I/O Techniques" on page 160. For the complete syntax of the INCLUDE command, see "INCLUDE Command" on page 347.

Issuing the INCLUDE command with no arguments includes the that is file referenced in the last INCLUDE command. If no previous INCLUDE command exists, you receive an error message.

## Using the GSUBMIT Command

The GSUBMIT command can be used to submit SAS statements that are stored in the Windows clipboard. To submit SAS statements from the clipboard, use the following command:

```
gsubmit buffer=default
```

You can also use the GSUBMIT command to submit SAS statements that are specified as part of the command. For more information about the GSUBMIT command, see the SAS Help and Documentation.

*Note:* SAS statements in the Windows clipboard will not be submitted using the GSUBMIT command if a procedure that you submitted using the Enhanced Editor is still running.

You can copy the SAS statements to a new Enhanced Editor window and then submit them. △

## Advanced External I/O Techniques

This section illustrates how to use the FILENAME, FILE, and INFILE statements to perform more advanced I/O tasks, such as altering the record format and length, appending data to a file, using the DRIVEMAP device-type keyword to determine which drives are available.

### Altering the Record Format

Using the RECFM= option in the FILENAME, FILE, %INCLUDE, and INFILE statements enables you to specify the record format of your external files. The following example shows you how to use this option.

Usually, SAS reads a line of data until a carriage return and line feed combination ('0D0A'x) are encountered or until just a line feed ('0A'x) is encountered. However, sometimes data do not contain these carriage control characters but do have fixed-length records. In this case, you can specify RECFM=F to read your data.

To read such a file, you need to use the LRECL= option to specify the record length and the RECFM= option to tell SAS that the records have fixed-length record format. Here are the required statements:

```
data test;
   infile "test.dat" lrecl=60 recfm=f;
   input x y z;
run;
```

In this example, SAS expects fixed-length records that are 60 bytes long, and it reads in the three numeric variables X, Y, and Z.

You can also specify RECFM=F when your data do contain carriage returns and line feeds, but you want to read these values as part of your data instead of treating them as carriage control characters. When you specify RECFM=F, SAS ignores any carriage controls and line feeds and simply reads the record length you specify.

## Appending Data to an External File

Occasionally, you may not want to create a new output file, but rather append data to the end of an existing file. In this case, you can use the MOD option in the FILE statement as in the following example:

```
filename myfile "c:\sas\data";
data _null_;
   infile myfile(newdata);
   input sales expenses;
   file myfile(jandata) mod;
   put sales expenses;
run;
```

This example reads the variables SALES and EXPENSES from the external data file C:\SAS\DATA\NEWDATA.DAT and appends records to the existing data file C:\SAS\DATA\JANDATA.DAT.

If you are going to append data to several files in a single directory, you can use the MOD option in the FILENAME statement instead of in the FILE statement. You can also use the FAPPEND function or the PRINTTO procedure to append data to a file. For more information, see the SAS functions section in *SAS Language Reference: Dictionary* and the PRINTTO procedure in *Base SAS Procedures Guide*.

## Determining Your Drive Mapping

You can use the DRIVEMAP device-type keyword in the FILENAME statement to determine which drives are available for use.

You might use this technique in SAS/AF applications, where you could build selection lists to let a user choose a hard drive. You could also use the DRIVEMAP keyword to enable you to assign macro variables to the various available hard drives.

Using the DRIVEMAP device-type keyword in the FILENAME statement implies you are using the fileref for read-only purposes. If you try to use the fileref associated with the DRIVEMAP device-type keyword in a write or update situation, you receive an error message indicating you do not have sufficient authority to write to the file.

Here is an example using this keyword:

```
filename myfile drivemap;
data mymap;
   infile myfile;
   input drive $;
   put drive;
run;
```

The information written to the SAS log looks similar to that shown in Output 5.1.

**Output 5.1** Drive Mapping Information

```
50   filename myfile drivemap;
51
52   data mymap;
53      infile myfile;
54      input drive $;
55      put drive;
56   run;
NOTE: The infile MYFILE is:
      FILENAME=DRIVEMAP,
      RECFM=V,LRECL=256
A:
C:
D:
J:
K:
L:
M:
N:
R:
S:
T:
U:
NOTE: 12 records were read from the infile MYFILE.
      The minimum record length was 2.
      The maximum record length was 2.
NOTE: The data set WORK.MYMAP has 12 observations
      and 1 variables.
NOTE: The DATA statement used 2.04 seconds.
```

### Reading External Files with National Characters

SAS under Windows, like most Windows applications, reads and writes character data using ANSI character codes. In SAS 9.1, SAS does not provide the option to read or write files using OEM character sets.

Characters such as the Â are considered national characters. Windows represents each character with a hexadecimal number. If your external file was created with a Windows editor (including applications such as WordPerfect) or in SAS, you do not need to do anything special. Simply read the file using the FILENAME or FILE statements, as you would normally do.

## Reading Data from the Communications Port

You can read data directly from the communications (serial) port on your machine. To set the serial communications parameters, use the port configuration tools in the Windows Control Panel to set up the communications port. The communications parameters you specify are specific to each data collection device.

After you invoke SAS, submit a FILENAME statement to associate a fileref with the communications port, as in the following example:

```
filename test commport "com1:";
```

This FILENAME statement defines the fileref TEST, uses the COMMPORT device-type keyword that specifies you are going to use a communications port, and specifies the COM1: reserved physical name.

Next, read the data from COM1: into a SAS data set using the TEST fileref. The following DATA step reads in the data, 1 byte at a time, until SAS encounters an end-of-file (the hex value of end-of-file is **'1a'x**):

```
data acquire;
   infile test lrecl=1 recfm=f unbuffered;
   input i $;
   /* Read until you find an end-of-file. */
   if i='1a'x then stop;
run;
```

The communications port can be accessed multiple times. However, while multiple reads are allowed, only one user at a time can write to the port.

Two useful functions in data acquisition applications are SLEEP and WAKEUP. These functions enable you to control when your program is invoked. For example, you can use the WAKEUP function to start your program at exactly 2:00 a.m. For more information about these two functions, see "SLEEP Function" on page 409 and "WAKEUP Function" on page 411.

## Communications Port Timeouts

By default, if you are reading from a communications port and a timeout occurs, an end-of-file (EOF) is returned to the program. You can specify how communications port timeouts are handled by using the COMTIMEOUT= option. The COMTIMEOUT= option is valid in the FILENAME statement and must be used in conjunction with the COMMPORT device-type keyword in the FILENAME statement.

The COMTIMEOUT= option accepts the following values:

| | |
|---|---|
| EOF | returns an end-of-file when a timeout occurs. This is the default behavior. This causes the current DATA step to terminate. |
| WAIT | instructs the communications port to wait forever for data. In other words, this value overrides the timeout. In this case, no record is returned to the DATA step until data are available. This can cause your program to go into a loop, so use this value with caution. |
| ZERO | does not wait if there is no incoming data. |

Here is an example of a FILENAME statement specifying that a record length of 0 bytes be returned to the program when a timeout occurs:

```
filename test commport "com1" comtimeout=eof;
data test;
   infile test length=linelen recfm=F eof=eof;
   input @;
eof:  if linelen ne 0 then input value;
   else put 'Timeout reading from COM1:';
run;
```

## Options that Relate to Communications Port Timeouts

These options relate to the communications port timeouts.

| | |
|---|---|
| RMULTI | specifies the multiplier, in milliseconds, that is used to calculate the total timeout period for read operations. For each read operation, this value is multiplied by the requested number of bytes to be read. |

| | |
|---|---|
| RCONST | specifies the constant, in milliseconds, that is used to calculate the total timeout period for read operations. For each read operation, this value is added to the product of RMULTI and the requested number of bytes. |
| WMULTI | specifies the multiplier, in milliseconds, that is used to calculate the total timeout period for write operations. For each write operation, this value is multiplied by the number of bytes to be written. |
| WCONST | specifies the constant, in milliseconds, that is used to calculate the total timeout period for write operations. For each write operation, this value is added to the product of the WMULTI member and the number of bytes to be written. |
| RINT | specifies the maximum time, in milliseconds, that is allowed to elapse between the arrival of two characters on the communications line. |

CHAPTER
6

# Managing SAS Output

*Printing* **166**
*Introduction to Printing in SAS within the Windows Environment* **166**
*Printing from within a SAS Window* **166**
*Overview of Printing From Within a SAS Window* **166**
*Setting Print Options* **166**
*Windows That Can Be Printed* **168**
*Printing a Window* **168**
*Changing the Printer* **169**
*Changing the Print Font* **170**
*Setting Up the Printed Page* **170**
*Print Options That Affect the Line Size and Page Size* **173**
*Printing Line Numbers, Page Numbers, and in Color* **173**
*Printing a Window as a Bitmap* **173**
*Setting the Number of Copies to Print* **174**
*Setting the Page Range to Print* **174**
*Previewing Your Output Before You Print* **174**
*How to Preview a Window* **174**
*Features of the Print Preview Window* **175**
*Print Preview Shortcut Keys* **175**
*Using SAS Print Forms* **175**
*Setting Print Options to Use Forms* **175**
*Specifying the Current Print Form* **176**
*Creating a Print Form* **176**
*Printing with SAS Commands* **177**
*Sending DATA Step Output to a Printer* **177**
*Sending Printed Output to a File* **177**
*Using the Print Dialog Box to Print to a File* **177**
*Using the FILENAME Statement to Print to a File* **178**
*Using the FILE Printer Option in Windows* **178**
*Printing in Batch Mode* **178**
*Default Printer Details* **178**
*Canceling a Print Job* **179**
*Routing Procedure Output to a Web Browser* **179**
*Introduction to Routing Procedure Output to a Web Browser* **179**
*Configuring Preferences for HTML Output* **180**
*Using the Results Viewer Window* **180**
*Configuring the Internal Browser* **180**
*Using the Toolbar* **181**
*Viewing HTML Files* **181**
*Routing Procedure Output and the SAS Log to a File* **182**
*Introduction to Routing Procedure Output and the SAS Log to a File* **182**

*Using the Save As Dialog Box* **182**
*Using the PRINTTO Procedure* **182**
*Using SAS System Options* **183**
*Producing Graphics* **184**
*Producing Graphics on Your Display* **184**
*Printing Graphics* **185**
*Using the SAS/GRAPH Generic (WINPxxx) Drivers* **185**
*Using the SAS/GRAPH Native Printer Drivers* **186**
*Printing and Previewing from the GRAPH Window* **187**
*Choosing between a SAS/GRAPH Native Driver and the WINPxxx Driver* **187**
*Importing Graphics from Other Applications* **188**
*Pasting Graphics from the Windows Clipboard* **188**
*Importing a Graphics File from within a SAS/GRAPH Window* **188**
*Exporting Graphics for Use with Other Applications* **189**
*Exporting a Graphic to a File from a SAS/GRAPH Window* **190**
*Pasting Graphics from SAS/GRAPH into other Windows Applications* **190**
*Creating CGM Files for Export to Other Applications* **191**
*Creating WMF (Windows Metafile) Files for Export to Other Applications* **193**
*Additional Resources* **193**

# Printing

## Introduction to Printing in SAS within the Windows Environment

By default, SAS under Windows uses Microsoft Windows print settings so that you can manage your output in the same manner as you manage output from other Windows applications. When you use Windows print settings, you use Windows TrueType fonts and fonts that are supported by your printers.

You can also use Universal Printing with the Output Delivery System. In the Windows environment, you enable Universal Printing by specifying the UNIVERSALPRINT option and the UPRINTMENUSWITCH option. The information in this section focuses on using Windows printing. For information about using Universal Printing, see *SAS Language Reference: Concepts* and “UPRINTMENUSWITCH System Option” on page 570.

For details about using Windows printing, see your Windows documentation. “Producing Graphics” on page 184 discusses how to route graphics from your SAS session to printers.

## Printing from within a SAS Window

### Overview of Printing From Within a SAS Window

Printing from SAS for Windows is much like printing in other Windows applications where you print using a toolbar button or a dialog box. You specify printing options using the Print, Print Setup, and Page Setup dialog boxes. As in other Windows applications, you can preview a printed page using the preview facility.

### Setting Print Options

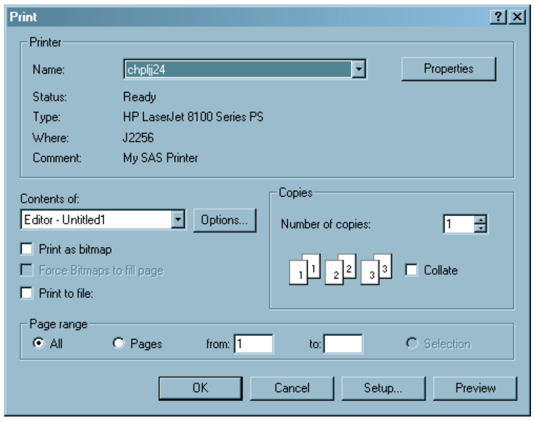
Use the Print dialog box to set the following print options:

- ☐ Change a printer destination
- ☐ Specify the window to print
- ☐ Print line numbers, page numbers, and in color
- ☐ Print as a bitmap
- ☐ Print to a file
- ☐ Print the contents of the clipboard
- ☐ Print multiple copies
- ☐ Print a range of pages or selected text
- ☐ Collate copies.

To access the Print dialog box, select

File ▶ Print

**Display 6.1** The Print Dialog Box



Print
Printer
Name: chplj24
Status: Ready
Type: HP LaserJet 8100 Series PS
Where: J2256
Comment: My SAS Printer
Contents of: Editor - Untitled1
Print as bitmap
Force Bitmaps to fill page
Print to file:
Copies
Number of copies: 1
Collate
Page range
All
Pages from: 1 to:
Selection

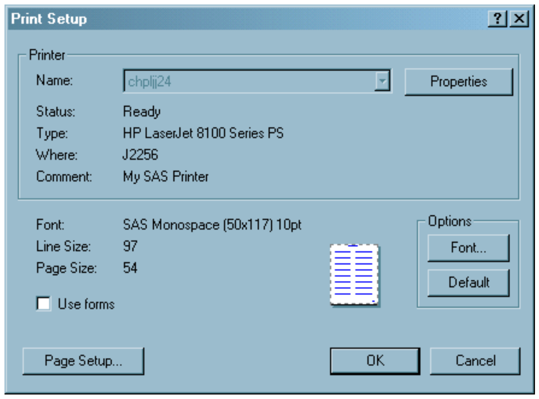
Use the Print Setup dialog box to set the following print options:

- ☐ Change fonts
- ☐ Use Forms.

To access the Print Setup dialog box, select

File ▶ Page Setup

**Display 6.2** The Print Setup Dialog Box



Print Setup
Printer
Name: chplj24
Status: Ready
Type: HP LaserJet 8100 Series PS
Where: J2256
Comment: My SAS Printer
Font: SAS Monospace (50x117) 10pt
Line Size: 97
Page Size: 54
Use forms

Use the Page Setup dialog box to set the following print options:

□ Change the paper size

□ Change the paper source

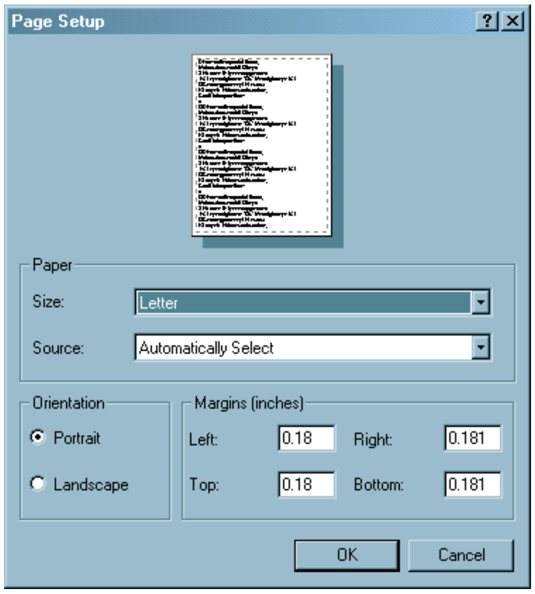
□ Specify the orientation (portrait or landscape)

□ Set the margin sizes.

To access the Page Setup dialog box, select

File ▶ Page Setup

**Display 6.3** Page Setup Dialog Box



Size: Letter

Source: Automatically Select

Orientation: (●) Portrait ( ) Landscape

Left: 0.18 Right: 0.181

Top: 0.18 Bottom: 0.181

You can specify document properties for the selected printer by selecting

File ▶ Print ▶ Properties

## Windows That Can Be Printed

Not all SAS windows can be printed. To determine if a SAS window can be printed, make the window the active window. If the Print toolbar button or the Print command in the File menu is active, the window can be printed.

The print output is a bitmap of the window if the Print toolbar button is active and the Print command in the File menu is not. An example of a window that would be printed as a bitmap is the SAS System Options window.

## Printing a Window

To print the contents of a window, make the window the active window and do one of the following:

□ To print using the current print settings, click the Print toolbar button.

□ To change the print options and then print, select

  File ▶ Print

  and select your printing options.

The Print dialog box might differ somewhat from what you see on your system, depending on which Windows operating environment you use to run SAS, and on the active SAS window.

## Changing the Printer

SAS consults these sources for default printer settings, in order of precedence:

1 the value of the SYSPRINT system option

2 the Windows default printer.

The destination printer is determined by the value of the SYSPRINT system option, which is displayed in the **Name** box of the Print dialog box.
To change the printer:

1 Select

File ▶ Print

2 Click in the **Name** list box and select a printer.

Using the SYSPRINT and PRTPERSISTDEFAULT system options, you can specify a printer when you start SAS as shown in the following table:

**Table 6.1** Specifying a Printer When You Start SAS

| Printer | Action |
|---|---|
| the Windows default printer | Do not specify the SYSPRINT system option when you start SAS. |
| a specific printer | Start SAS with the SYSPRINT system option. |
| the printer specified in the previous SAS session | Start SAS with the PRTPERSISTDEFAULT system option each time you start SAS. |

If both SYSPRINT and PRTPERSISTDEFAULT system options are specified when SAS starts, the destination printer is determined by the value of the SYSPRINT system option. For more information about these system options, see “SYSPRINT System Option” on page 566 and “PRTPERSISTDEFAULT System Option” on page 536.
Alternatively, you can change the destination printer by using an OPTIONS statement or by using the SAS System Options window. To change the printer using the SAS System Options window:

1 Select

Tools ▶ Options ▶ System

2 Select the **Log and procedure output control** folder and then select the **Procedure output** folder.

3 Double-click **Sysprint**.

4 Type a printer name as it appears in the Windows Printer folder in the **New Value** box and click **OK**. The printer name is case-sensitive.

The information about the printer in the Print dialog box, the **Status**, **Type**, **Where**, and **Comment** fields, displays information that is obtained from the Windows operating environment.

## Changing the Print Font

The print font options enable you to change the font, the font style, the point-size, and the script. When you change the font size, SAS recalculates the maximum LINESIZE and PAGESIZE values that are displayed in the Print Setup dialog box.

To specify a print font:

1 Select

   File ▶ Print Setup ▶ Font

2 Select **`Font`**, **`Font Style`**, and **`Size`**.

3 Click **`OK`**.

*Note:* SAS formats tabular and columnar reports assuming the use of a monospace font. Use of a proportionally spaced font might produce improperly formatted reports. △

Alternatively, you can change the font using the SYSPRINTFONT system option when you start SAS, using an OPTIONS statement, or using the SAS System Option window. Using the SYSPRINTFONT system options requires an exact match of the font face-name and printer names.

To modify SYSPRINTFONT using the SAS System Option window,

1 Select

   Tools ▶ Options ▶ System

2 Select the **`Log and procedure output control`** folder and then select the **`Procedure output`** folder.

3 Right- click **`Sysprintfont`** and select **`Modify Value`** from the pop-up menu.

4 Type the font value in the **`New Value`** text box. Enclose the value in parentheses.

5 Click **`OK`**.

The following SYSPRINTFONT system option sets the font to Arial, bold, and italic for the printer named "second-floor":

```
sas -sysprintfont="Arial" bold italic named "second-floor";
```

For more information, see “SYSPRINTFONT System Option” on page 567.

## Setting Up the Printed Page

Setting up a page involves specifying the paper, the orientation of the paper, and the margins. You set up the page by using the Page Setup dialog box or by using system options.

To open the Page Setup dialog box select

File ▶ Page Setup

The following table describes the Page Setup dialog box options and their related system options.

**Table 6.2** Options for Setting Up a Printed Page

| Page Setup Option | Description | Setting the Option | Related System Option |
|---|---|---|---|
| Orientation | Specifies to print the page vertically or horizontally. | To print the page vertically, select **`Portrait`**. To print the page horizontally, select **`Landscape`**. | ORIENTATION |
| Margins | Specifies the amount of space to leave blank from the top, bottom, left, and right edges of the paper | Type the number of inches in the **`Left`**, **`Right`**, **`Top`**, and **`Bottom`** fields. | LEFTMARGIN, RIGHTMARGIN, TOPMARGIN, BOTTOMMARGIN |
| Paper size | Specifies the size of paper to print on. See the table below for a list of some paper types. | Click the **`Size`** box and select a paper size. | PAPERSIZE |
| Paper source | Specifies the source of the paper, such as a printer tray, envelope or manual feeder, or specifies to automatically select the paper source. | Click the **`Source`** box and select a paper source. | PAPERSOURCE |

Alternatively, you can set the page setup options using system options in the OPTIONS statement or from the SAS System Options window. To set page setup options from the SAS System Options window:

1 Select

Tools ▶ Options ▶ System ▶ Log and procedure output control ▶ ODS printing

2 Place the cursor over the appropriate system option and double- click the right mouse button.

3 Type a new value and click **OK**.

***CAUTION:***

**Modifying print options by using the Windows printing dialog boxes might change the values of SAS printing system options, which might cause unpredictable output.** If you set printing options using SAS system options such as LINESIZE and PAGESIZE, and then use the Windows printing dialogs to set printing options, the SAS system options are set to the values specified in the Windows print dialog boxes. △

Support for a particular paper size is printer dependent. The following is a list of some paper size names:

| | |
|---|---|
| LETTER | Letter, 8-1/2-by-11-inch paper |
| LEGAL | Legal, 8-1/2-by-14-inch paper |

| | |
|---|---|
| A4 | A4 Sheet, 210-by-297-millimeter paper |
| CSHEET | C Sheet, 17-by-22-inch paper |
| DSHEET | D Sheet, 22-by-34-inch paper |
| ESHEET | E Sheet, 34-by-44-inch paper |
| LETTERSMALL | Letter Small, 8-1/2-by-11-inch paper |
| TABLOID | Tabloid, 11-by-17-inch paper |
| LEDGER | Ledger, 17-by-11-inch paper |
| STATEMENT | Statement, 5-1/2-by-8-1/2-inch paper |
| EXECUTIVE | Executive, 7-1/4-by-10-1/2-inch paper |
| A3 | A3 sheet, 297-by-420-millimeter paper |
| A4SMALL | A4 small sheet, 210-by-297-millimeter paper |
| A5 | A5 sheet, 148-by-210-millimeter paper |
| B4 | B4 sheet, 250-by-354-millimeter paper |
| B5 | B5 sheet, 182-by-257-millimeter paper |
| FOLIO | Folio, 8-1/2-by-13-inch paper |
| QUARTO | Quarto, 215-by-275-millimeter paper |
| 10X14 | 10-by-14-inch paper |
| 11X17 | 11-by-17-inch paper |
| NOTE | Note, 8-1/2-by-11-inch paper |
| ENV_9 | #9 Envelope, 3-7/8 by 8-7/8 inches |
| ENV_10 | #10 Envelope, 4-1/8 by 9-1/2 inches |
| ENV_11 | #11 Envelope, 4-1/2 by 10-3/8 inches |
| ENV_12 | #12 Envelope, 4-3/4 by 11 inches |
| ENV_14 | #14 Envelope, 5 by 11-1/2 inches |
| ENV_DL | DL Envelope, 110 by 220 millimeters |
| ENV_C5 | C5 Envelope, 162 by 229 millimeters |
| ENV_C3 | C3 Envelope, 324 by 458 millimeters |
| ENV_C4 | C4 Envelope, 229 by 324 millimeters |
| ENV_C6 | C6 Envelope, 114 by 162 millimeters |
| ENV_C65 | C65 Envelope, 114 by 229 millimeters |
| ENV_B4 | B4 Envelope, 250 by 353 millimeters |
| ENV_B5 | B5 Envelope, 176 by 250 millimeters |
| ENV_B6 | B6 Envelope, 176 by 125 millimeters |
| ENV_ITALY | Italy Envelope, 110 by 230 millimeters |
| ENV_MONARCH | Monarch Envelope, 3-7/8 by 7-1/2 inches |
| ENV_PERSONAL | 6-3/4 Envelope, 3-5/8 by 6-1/2 inches |
| FANFOLD_US | U.S. Standard Fanfold, 14-7/8-by-11-inch paper |

| | |
|---|---|
| FANFOLD_STD-_GERMAN | German Standard Fanfold, 8-1/2-by-12-inch paper |
| FANFOLD_LGL-_GERMAN | German Legal Fanfold, 8-1/2-by-13-inch paper |

## Print Options That Affect the Line Size and Page Size

The line size is the number of characters that can fit on one line. The page size is the number of lines on a page. The line size and the page size that appear in the Print Setup dialog box are automatically calculated based on these print options:

**Table 6.3** Print Options That Affect The Line Size and Page Size

| From the Print Setup Dialog Box | From the Page Setup Dialog Box | From the Font Dialog Box |
|---|---|---|
| □ the printer | □ paper size | □ font |
| | □ paper source | □ font style |
| | □ orientation | □ size |
| | □ margin settings | |

Although you cannot set the linesize and pagesize from the dialog box, you can adjust them by changing these print settings. The LINESIZE and PAGESIZE system options also change when you modify these print options.

***CAUTION:***
**Modifying print options by using the Windows printing dialog boxes might change the values of SAS printing system options, which might cause unpredictable output.** If you set printing options using SAS system options such as LINESIZE and PAGESIZE, and then use the Windows printing dialogs to set printing options, the SAS system options are set to the values specified in the Windows print dialog boxes. △

## Printing Line Numbers, Page Numbers, and in Color

Options for printing line numbers, page numbers, and in color are available in the Additional Printing Options dialog box. To open this dialog box, click **`Options`** in the Print dialog box.

The **`Options`** button is enabled only for windows that allow the printing of these options.

It is not necessary for you to turn on line numbers in your window or specify the NUMBER system option in order to print line numbers and page numbers. Color printing is available when you print to a color printer and the window that you are printing supports color printing.

## Printing a Window as a Bitmap

The following table lists the bitmap forms that are available and how to print them:

**Table 6.4** Printing Bitmap Forms

| Bitmap Form | Print Dialog Box Selection |
|---|---|
| Print the active SAS window | Select **Print as bitmap** |
| Print the SAS window | 1 Select **Print as bitmap**<br>2 In the **Contents of** box, select **AWS windows (bitmap)**. |
| Print the entire screen | 1 Select **Print as bitmap**<br>2 In the **Contents of** box, select **Entire screen(bitmap)**. |
| Fill an entire page with the bitmap | Select **Force Bitmaps to fill page** |

## Setting the Number of Copies to Print

In the Print dialog box **Number of copies** box, you can either type in the number of copies that you want or you can use the up and down arrows to select the number of copies that you want. If your printer supports collation, the Collate box is active. Under Windows NT, SAS does not collate its output pages.

You can also set the number of copies to print by setting the COPIES system option either in an OPTIONS statement or in the SAS System Options window.

To set the number of copies by using the SAS System Options window:

1 Select

 Log and procedure output control ▶ ODS Printing

2 Double-click **Copies**.
3 In the **New Value** box, type the number of copies and click **OK**.

## Setting the Page Range to Print

Use these settings in the Print dialog box to select the pages that you want to print:

- To print all pages, select **All**.
- To print a range of pages:
  - Select **Pages**.
  - Type the beginning page in the **from** box.
  - Type the ending page in the **to** box.
- To print only what you have selected in the window, select **Selection**. The **Selection** option is available only when you have made a selection.

## Previewing Your Output Before You Print

### How to Preview a Window

To see how the contents of a window will appear as printed output:

1 Select the window you want to preview
2 Select

 File ▶ Print Preview

Alternatively, you can click the Print Preview toolbar button or type **dlgprtpreview** in the command bar.

## Features of the Print Preview Window

The following table lists the features of the Print Preview window.

**Table 6.5** Features of the Print Preview Window

| Task | Action |
|---|---|
| Navigate through the pages | Use **`Next`** or **`Previous`** |
| Zoom in or out | Click **`Zoom`** or the page. |
| Determine the current page | The status bar displays the current page and total number of pages in the document. |
| Get help | Click **`Help`** |
| Print the window | Click **`Print`** |
| Close the Print Preview window | Click **`Close`** |

For a list of keyboard shortcuts that you can use in the Print Preview windows, see "Keyboard Shortcuts within Print Preview" on page 621.

## Print Preview Shortcut Keys

In the Print Preview window, you can navigate by using the following shortcut keys:

| Key | Action in Full Page Mode | Action in Zoom Mode |
|---|---|---|
| PgDn | Advance to next page | Scroll down on current page |
| PgUp | Go back to previous page | Scroll up on current page |
| Ctrl+PgDn | none | Scroll right on current page |
| Ctrl+PgUp | none | Scroll left on current page |
| Ctrl+Home | Go to first page | Go to first page |
| Ctrl+End | Go to last page | Go to last page |

Not all SAS windows support the Print Preview feature.

## Using SAS Print Forms

### Setting Print Options to Use Forms

To use a form to print from SAS:

1 Select

File ► Print Setup

2 Select the **`Use Forms`** check box.

If the **`Use Forms`** check box does not appear in the Print Setup dialog box, use the PRTSETFORMS system option to enable it. For more information, see "PRTSETFORMS System Option" on page 537.

When you print, SAS prints your output with the current print form.

To use forms in a batch SAS session, use the NOHOSTPRINT system option. When NOHOSTPRINT is specified, the **`Use Forms`** check box is selected and SAS uses the linesize, pagesize, and font settings that are specified in your SAS form.

## Specifying the Current Print Form

To specify a print form as the current print form, do one of the following:

- Type FORMNAME CLEAR in the command bar to use the default form.
- Type FORMNAME *form-name* in the command bar to use a specific form.
- Specify the FORMS system option in your SAS configuration file or in the SAS System Options window.

To learn the name of the current form, issue the FORMNAME command with no parameters. The form name is displayed in the message area of the status line.

The FORMNAME command is not supported for all windows. If it is not supported, SAS displays the following message in the status bar:

```
ERROR: Unrecognized command FORMNAME
```

## Creating a Print Form

The FSFORM command opens the FORM window, in which you can define print forms to use when you print SAS output. You can specify printer, page formats, margins, fonts, and printer control language in a FORM entry.

SAS print forms are especially useful when you use the PRINT command from within an interactive SAS session and when you print from SAS/AF windows.

To invoke the FORM window, issue the following command:

FSFORM *catalog-name.form-name*

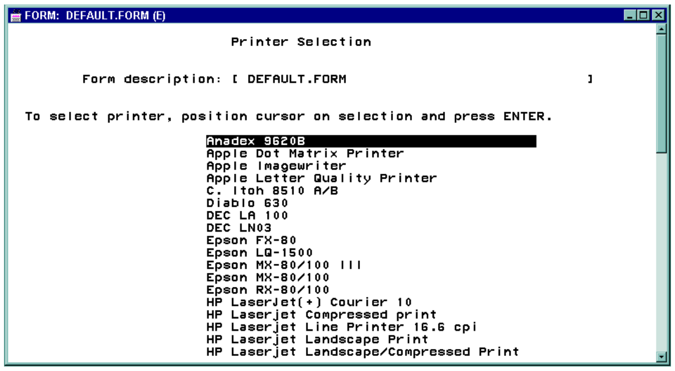
See the SAS Help and Documentation for more information about the FSFORM command.

Although the majority of the frames in the FORM window are the same across all operating environments, the first frame that you see after issuing the FSFORM command is the Printer Selection frame, which lists the printers that you are able to use under Windows. Display 6.4 on page 176 shows the default information for this frame.

To navigate though the FORM window frames for a printer:

1. Click a printer name or select a printer and press ENTER.
2. From the **`Tools`** menu, select **`Next Screen`** and **`Previous Screen`** to move through the frames.

**Display 6.4** Printer Selection Frame



FORM: DEFAULT.FORM (E)

```
                         Printer Selection

      Form description: [ DEFAULT.FORM                              ]

 To select printer, position cursor on selection and press ENTER.

                         Anadex 9620B
                         Apple Dot Matrix Printer
                         Apple Imagewriter
                         Apple Letter Quality Printer
                         C. Itoh 8510 A/B
                         Diablo 630
                         DEC LA 100
                         DEC LN03
                         Epson FX-80
                         Epson LQ-1500
                         Epson MX-80/100 III
                         Epson MX-80/100
                         Epson RX-80/100
                         HP LaserJet(+) Courier 10
                         HP Laserjet Compressed print
                         HP Laserjet Line Printer 16.6 cpi
                         HP Laserjet Landscape Print
                         HP Laserjet Landscape/Compressed Print
```

The information in the Printer Selection frame is also site-dependent, so the printer list at your site will be different from the one shown in Display 6.4 on page 176.

The Printer Selection frame appears only when you create a new print form. After you create a form, it is stored in your user profile catalog or in the catalog that was specified with the FSFORM command (entry type FORM). The next time you modify this form, the Printer Selection frame is skipped. You cannot return to the Printer Selection frame from the second FORM window frame.

## Printing with SAS Commands

If you prefer typing commands to using menus, you can use the PRINT or SPRINT command to print the contents of the active window. The SPRINT command is not available for all windows. For more information on these commands, refer to the SAS Help and Documentation.

These SAS commands open the print dialog boxes:

**Table 6.6** SAS Commands That Open Print Dialog Boxes

| **Command** | **Dialog Box** |
|---|---|
| DLGPRT | Print |
| DLGPAGESETUP | Page Setup |
| DLGPRTSETUP | Print Setup |
| DLGPRTPREVIEW | Print Preview |

## Sending DATA Step Output to a Printer

You may want to spool your DATA step output to a printer instead of to a file. Use the FILENAME statement and the PRINTER device-type keyword to accomplish this, as in the following example:

```
filename myfile printer;
data _null_;
   set sashelp.shoes;
   file myfile;
   where stores ge 25;';
   put _all_;
run;
```

In this example, the PRINTER device-type keyword specifies to print the output to the printer that is specified in the SYSPRINT system option. For more information, see “SYSPRINT System Option” on page 566.

## Sending Printed Output to a File

### Using the Print Dialog Box to Print to a File

You can send your printed output to a file by selecting the **`Print to File`** check box in the Print dialog box, and then specifying the name of a file to print to. This is *not* the same as a save operation; the resulting printer file contains all the printer control language that is necessary to support whatever options you have chosen with the

Printer Setup dialog box, such as fonts and page orientation. In most cases, this printer file is not readable with a text editor; it is meant only to be sent to the printer.

### Using the FILENAME Statement to Print to a File

You can route printed output to a file by using the FILENAME statement, which is useful for routing DATA step output. Here is an example:

```
filename myfile printer altdest='c:\results.dat';
data _null_;
   set sashelp.shoes;
   file myfile;
   where stores ge 25;
   put _all_;
run;
```

In this example, the output from the DATA step is routed to a file, yet still contains all the printer control information that is necessary for you to use your printer to produce formatted output.

### Using the FILE Printer Option in Windows

Another method of sending printed output to a file is to direct the output to the **`FILE:`** device instead of to a printer in the Windows printer Properties dialog box **`Ports`** page. If you assign the **`FILE`**: device to a printer, Windows prompts you for a filename each time that you print. When you send output to a file, the contents of the file are overwritten if the file already exists. For more information about changing printer properties, see your Windows documentation.

## Printing in Batch Mode

When you run SAS jobs in batch mode, you do not have access to the Print and Printer Setup dialog boxes, but you can still use the Windows printer. Use the SYSPRINT system option to specify your default printer (and the SYSPRINTFONT system option to specify your printer font, if you want) as described in "SYSPRINT System Option" on page 566. For example, suppose your SAS configuration file contains the following option:

```
-sysprint "f2hp5"
```

Then, your SAS program might contain the following statements:

```
filename myfile printer;
data _null_;
   set sashelp.shoes;
   file myfile;
   where stores ge 25;
   put _all_;
run;
```

When you submit your job, SAS uses the SYSPRINT printer specification to spool your output from the DATA step to the Windows printer.

## Default Printer Details

SAS looks for a default printer as follows (in order of precedence, first to last):

- the SYSPRINT system option value
- the Windows system default printer.

To see the value of the SYSPRINT system, open the Print Setup dialog box either by

- selecting

  File ▶ Print Setup

- entering DLGPRTSETUP in the command bar.

If you start SAS with the PRTPERSISTDEFAULT system option and not with the SYSPRINT system option, SAS sets the SYSPRINT system option to the name of the destination printer from the previous SAS session.

For information about changing the printer, see "Changing the Printer" on page 169.

## Canceling a Print Job

You can cancel a print job while SAS is spooling a print file to a folder or to the Windows Printer by clicking **Cancel** in the Print Abort dialog box. The Print Abort dialog box appears only while SAS is spooling a print file to its destination. Small files spool to a destination quickly and the Print Abort dialog box dismisses before you have a chance to cancel the print job.

You can specify when you want to enable or suppress the Print Abort dialog box by using the PRTABORTDLGS system option. See the following table for valid PRTABORTDLGS values.

| When to Display the Print Abort Dialog Box | PRTABORTDLGS Value |
|---|---|
| Printing either to a file or to a printer | BOTH |
| Never | NEITHER |
| Only when printing to a file | FILE |
| Only when printing to a printer | PRINTER |

For more information, see "PRTABORTDLGS System Option" on page 536.

## Routing Procedure Output to a Web Browser

### Introduction to Routing Procedure Output to a Web Browser

The Output Delivery System (ODS) can create your procedure output in HTML to be viewed in any Web browser. The ODS system also has the ability to create RTF procedure output. If you have Microsoft Internet Explorer 4.0 (IE) or later installed, you can view HTML and RTF procedure output within the SAS main window by using the Results Viewer. Viewing RTF procedure output also requires an RTF viewer, such as Microsoft Word.

If you are not using IE, HTML procedure output is displayed in the preferred browser that is specified in the Preferences dialog box **Web** sheet. Having IE installed enables you to view the HTML output from within the main SAS window .

For more information about using the Output Delivery System, see *SAS Output Delivery System: User's Guide*.

## Configuring Preferences for HTML Output

To create and view HTML output, you configure the **`Results`** and **`Web`** tabs of the Preferences dialog box.

To open the Preferences dialog box, select

Tools ▶ Options ▶ Preferences

To configure the **`Results`** page of the Preferences dialog box:

1 Click on the Results tab.

2 Select the **`Create HTML`** check box.

3 Select where to save your HTML file. To save HTML files only while the current SAS session is active, select the **`Use WORK folder`** check box. Saving HTML files to a temporary folder is useful when you are testing a procedure and creating a multitude of HTML files. For example, when you exit SAS, all temporary files are deleted.

   To save HTML files to a folder other than the Work folder, ensure that the **`Use WORK folder`** check box is not selected. Then type a path in the **`Folder`** text box or click on **`Browse`** to search for a folder.

4 Click in the **`Style`** box and highlight a style. The style defines colors and fonts to display your output. You define styles by using the TEMPLATE Procedure in *SAS Output Delivery System: User's Guide*.

5 Select **`View results as they are generated`** to view the results immediately..

6 Under **`View results using`**, select **`Internal browser`** to view HTML output in the Results Viewer or select **`Preferred browser`** to view the HTML output in the Web browser specified on the **`Web`** tab. If you do not have Microsoft Internet Explorer installed, you must select **`Preferred web browser`**.

To configure the **`Web`** page of the Preferences dialog box:

- □ Click on the Web tab.
- □ Choose which browser you want to use:
  - □ Select the **`Use default`** radio button to view the HTML output in the default browser that is configured in the Windows Registry. For information about how to view the Windows Registry, see your Windows documentation.
  - □ Select the **`Other`** radio button to view HTML from a Web browser other than the default browser. Then type the browser's path and filename, or click **`Browse`** to find the browser filename.

## Using the Results Viewer Window

### Configuring the Internal Browser

When you select **`Internal browser`** from the **`Results`** tab of the Preferences dialog box, SAS uses the Results Viewer window to display HTML output that is created by the Output Delivery System. You can open any HTML output that is listed in the Results window or any HTML file that is located on your system.

## Using the Toolbar

The following toolbar buttons are available in the Results Viewer for HTML output:

Go Back
: Select the left arrow button to cycle back through files that you opened in the Results Viewer.

Go Forward
: Select the right arrow button to move forward through files that you opened in the Results Viewer.

Stop download
: Select the traffic light button to stop loading a file.

Refresh content
: Select the arrow circle button to reload the current file.

Cycle font
: Select the double-A button to change the font size. Five font sizes are available.

## Viewing HTML Files

You can view HTML procedure output by opening the file from the Results window. You may prefer to see your procedure output as it is generated. To do this, you set an option in the **`Results`** tab of the Preferences dialog box:

1 Select

   Tools ▶ Options ▶ Preferences

2 Select the **`Results`** tab.

3 Select **`View results as they are generated`**.

4 Click **`OK`**.

To open additional Results Viewer windows from the Results window:

1 Select the **`Results`** window.

2 Click on the output data set name with the right mouse button.

3 Select **`Open in New Window`**.

To open an HTML file that is on your system:

1 Make the Results Viewer window the active window.

2 Select the Open toolbar button.

3 Specify the HTML file in the Open dialog box.

4 Click **`OK`**.

To edit your HTML output:

1 From the Results window, click with the right mouse button on the output data set name.

2 Select **`Edit Source`**.

3 Make changes to the HTML file.

4 Select the Save toolbar button.

5 Select the Refresh content toolbar button to view the updated HTML file in the Results Viewer.

## Routing Procedure Output and the SAS Log to a File

### Introduction to Routing Procedure Output and the SAS Log to a File

This section provides examples of the most common methods of routing SAS procedure output and the SAS log to a file. Generally, this task is the same across operating environments and is discussed in the SAS Help and Documentation. However, the specification of external filenames and devices is system dependent. For complete information about the various ways to reference external files, see "Referencing External Files" on page 146.

You can route your SAS procedure output or the SAS log to a file in one of several ways. The method that you choose depends on the method you use to run SAS, the moment at which you make your decision to route the output or SAS log, and your personal preference.

Some methods of sending SAS procedure output or the SAS log to a file include using

- the Save As dialog box, invoked either from the **`File`** menu, the pop-up menu for the window that you want to save, or by typing DLGSAVE in the command bar.
- the FILE command.
- the PRINTTO procedure.
- various system options, including PRINT and ALTLOG.

### Using the Save As Dialog Box

The easiest way to save the contents of the active window to a file is to select

File ► Save As

from the main SAS window, making sure that the active window (for example, Log or Output) contains the output that you want to save. For more information about the Save As dialog box, see "Saving Files" on page 85 within the Enhanced Editor or "Saving Files" on page 114 within the Program Editor.

### Using the PRINTTO Procedure

In batch SAS sessions, the SAS procedure output and the SAS log are written by default to files named *filename*.LST and *filename*.LOG, respectively, where *filename* is the name of your SAS job. For example, if your SYSIN file is MYPROG.SAS, the procedure output file is named MYPROG.LST, and the log file is named MYPROG.LOG. However, you can override these default filenames and send your output to any file that you choose. For example, suppose that your job contains the following statements, which assign the fileref MYOUTPUT to the file C:\SAS\FIRST.TXT. Then the PROC PRINTTO statement tells SAS to send any upcoming SAS procedure output to the file that is associated with MYOUTPUT.

```
filename myoutput 'c:\sas\first.txt';
proc printto print=myoutput;
run;
data uspres;
   input pres $ party $ number;
   datalines;
Adams F 2
```

```
;
run;
proc print;
run
```

Any PROC or DATA statements that follow these statements and that generate output send their output to the C:\SAS\FIRST.TXT file, not to the default procedure output file. If you want to return to the default file, issue an empty PROC PRINTTO statement like the following example:

```
proc printto;
run;
data uspres2;
   input pres $ party $ number;
   datalines;
Lincoln R 16
Grant R 18
;
run;
proc print;
run;
```

Issuing these statements redirects the SAS procedure output to the default destination (*filename*.LST). In this way, you can send the output and log from different parts of the same SAS job to different files.

*Note:* If you route procedure output to a file, the resulting file may contain carriage control characters. To suppress these control characters when you include the file in the Program Editor, set the RECFM= option to P in the FILENAME statement. Note that this affects the way the file is read into the Program Editor, not the file itself. △

If you want to send the SAS log to a specific file, use the LOG= option instead of the PRINT=option in the PROC PRINTTO statement. For more information about the PRINTTO procedure, see “PRINTTO Procedure” on page 433 and *Base SAS Procedures Guide*.

*Note:* When you use the PRINTTO procedure to route SAS procedure output or the SAS log, the Status window does not reflect any rerouting of batch output but indicates that it is routing the procedure output file and log to *filename*.LST and *filename*.LOG. △

## Using SAS System Options

You can use SAS system options to route your SAS output or SAS log to a file. For example, if you want to override the default behavior and send your procedure output from a batch SAS job to the file C:\SASOUTPUT\PROG1.TXT, you can invoke SAS with the following command:

```
SAS -SYSIN C:\SASPROGS\PROG1
    -PRINT C:\SASOUTPUT\PROG1.TXT
```

This SAS command executes the SAS program PROG1.SAS and sends the procedure output to the file C:\SASOUTPUT\PROG1.TXT. You can treat the SAS log similarly by using the LOG system option instead of the PRINT system option. Two other related system options, the ALTPRINT and ALTLOG options, are explained in “ALTPRINT System Option” on page 484 and “ALTLOG System Option” on page 483.

*Note:* The Status window does reflect the PRINT and LOG system options values when recording where the procedure output and log are being sent. △

# Producing Graphics

## Producing Graphics on Your Display

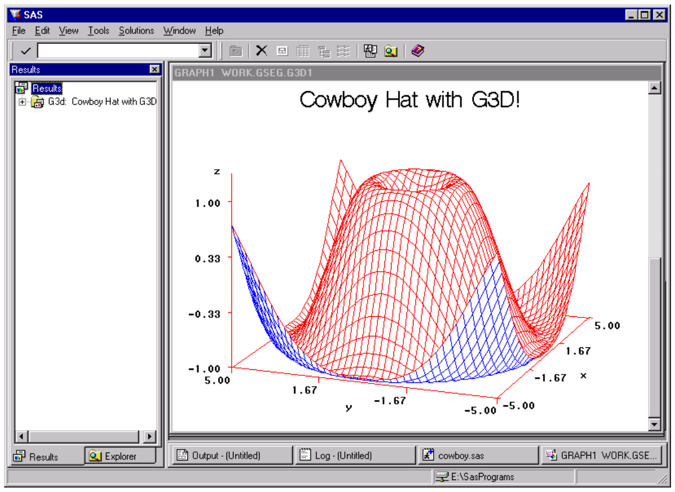
In most cases, output is automatically displayed on your monitor when you run a SAS/GRAPH procedure; it is not necessary to specify a SAS/GRAPH device driver. Information about your graphics display is stored in a Windows information file and is automatically used by SAS during an interactive SAS session.

Here is a simple example of how to produce a graphic:

```
data hat;
   do x=-5 to 5 by .25;
      do y=-5 to 5 by .25;
         z=sin(sqrt(x*x+y*y));
         output;
      end;
   end;
proc g3d data=hat;
   plot y*x=z/ctop=red;
   title 'Cowboy Hat with G3D';
run;
quit;
```

The following display shows the output for this program:

**Display 6.5** Cowboy Hat Program Output



If you use the DEVICE= option in the GOPTIONS statement to route your graphics to a hardcopy device, and then you want to return to using your monitor to display graphics, you must specify a driver. Submit the following statement to display graphics output on your monitor:

```
goptions device=win;
```

You should also use the WIN device driver to produce graphics on your display when you run your SAS job in batch mode.

If you specify that your program output is to be displayed in HTML, your graphic is converted to a .GIF file and stored in the same folder as your SAS data set. For more information, see *SAS Output Delivery System: User's Guide*.

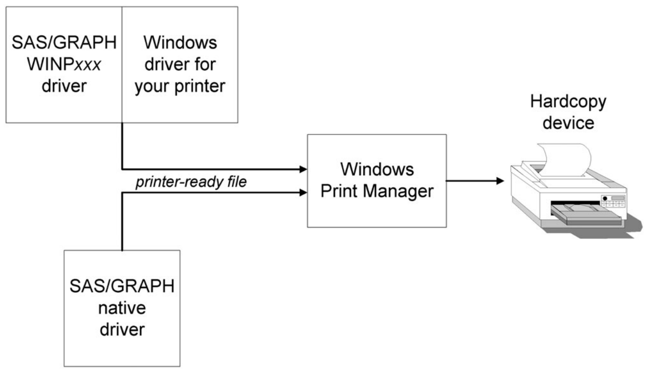
## Printing Graphics

You can use two methods to print output from SAS/GRAPH:

- Use the SAS/GRAPH generic driver with the Windows printer driver for your device (supplied with Windows or with your output device). This method allows SAS/GRAPH to send generic graphics commands to the Windows printer driver, which then translates the commands to a format that the printer can use.
- Use a SAS/GRAPH printer driver to create a printer-specific graphics stream that can be sent directly to the device. When you use a SAS/GRAPH printer driver, SAS bypasses the Windows printer drivers and creates printer-ready output for your device. The SAS/GRAPH printer driver is also called a SAS/GRAPH native print driver, which means that the driver creates output by using the printer language that is native to the target device. SAS/GRAPH printer drivers under Windows are similar to those used by SAS on mainframe and UNIX operating environments.

After SAS prepares output for a printer (by using either Windows printer drivers or a SAS/GRAPH printer driver), the output is sent to the Windows printer, which then queues it for printing on the device of your choice. Figure 6.1 on page 185 illustrates how you can use the two sets of printer drivers within SAS/GRAPH to produce output for a given device.

**Figure 6.1** SAS/GRAPH Generic Printer Drivers Compared with SAS/GRAPH Native Printer Drivers



The method that you choose depends on the output device that you are using. For more information, see "Choosing between a SAS/GRAPH Native Driver and the WINP*xxx* Driver" on page 187. You can control both graphics printing methods by using either the Print and Print Setup dialog boxes or the SYSPRINT= option and the GOPTIONS DEVICE= statement.

### Using the SAS/GRAPH Generic (WINP*xxx*) Drivers

To print a graphic by using the SAS/GRAPH generic device drivers with the Windows printer drivers:

1 From the `File` pull-down menu, select `Print Setup` and verify that the `Printer` field in the Print Setup dialog box lists the correct Windows printer driver and

port. You can use the Print Setup dialog box to select any printer driver/port combinations that you have installed. (To install new drivers and port combinations, you can use the Add Printer Wizard in Windows.)

Alternatively, you can use the SYSPRINT system option to assign the destination printer. For example,

```
options sysprint='HP LaserJet III';
```

Note that you may assign only printer driver names that have been previously configured in Windows.

2 Run your SAS/GRAPH program with the following graphic options:

```
goptions device=winp xxx;
```

The value of WINP*xxx* you specify depends on the type of output device that you use to print your graph:

WINPRTM
for black and white (monochrome) printers

WINPRTG
for grayscale printers

WINPRTC
for color printers

WINPLOT
for plotters.

The orientation of graphics output is determined by the following:

- If you specify the ROTATE= graphic option, the output is oriented according to the value that you specify for ROTATE. For example, suppose you specify

  ```
  goptions rotate=landscape
  ```

  Then the output is oriented as landscape, regardless of the settings in the Print Setup dialog box.
- If you do not use the ROTATE= graphic option, the output is oriented according to the settings in the Print Setup dialog box.

*Note:* Graphics printing is affected by the margins that are specified in the Page Setup dialog box. If you modify the margins when printing graphics and your intention is to keep the graphic proportional, be sure to change the top and bottom margins by the same amount you change the left and right margins. △

## Using the SAS/GRAPH Native Printer Drivers

SAS/GRAPH native drivers produce output in the native language of the target device. Examples of SAS/GRAPH native drivers include:

| | |
|---|---|
| PS | produces Postscript output. |
| HPLJS3 | produces output in the PCL5 language that is used by Hewlett Packard LaserJet III printers. |
| HP7550 | produces HPGL output that is used by Hewlett–Packard 7550 plotters. |

After the SAS/GRAPH native printer driver has produced output in the native language of the target device, SAS then routes the output to the device using the Windows printer. SAS bypasses the Windows driver that is currently associated with the target device, but it does respect the destination that is specified in the Print Setup dialog box when deciding where to send the output.

To print a graph by using a SAS/GRAPH printer driver, run your SAS/GRAPH program with the following graphic options:

```
goptions device=driver-name;
```

where *driver-name* is the name of a valid SAS/GRAPH device driver. Consider this example,

```
goptions device=hplj5p3;
```

This statement formats the graph for the Hewlett Packard LaserJet Series V printer. You can view the complete list of SAS/GRAPH drivers by submitting the PROC GDEVICE statement.

To print a graph to a printer file (also called a graphics stream file, or GSF) instead of directly to a printer, use the GSFNAME option on the GOPTIONS statement and use a filename or a fileref to specify where you want the output. For example:

```
filename graphout "graphpic.prn";
goptions gsfname=graphout gsfmode=replace
         device=hpljs2;
```

## Printing and Previewing from the GRAPH Window

You can preview a graph that you create and, at the same time, format it for optimal display on the device of your choice. To preview the graph before you print it, run your SAS/GRAPH program with the following GOPTIONS statement:

```
goptions targetdevice=driver-name;
```

where *driver-name* is either one of the WINP*xxx* drivers or a SAS/GRAPH native driver.

By specifying a target device, SAS/GRAPH can format the graph with colors and attributes that are appropriate for the target printer. To print the graph after it is displayed, select the **`File`** pull-down menu and then select **`Print`**.

*Note:* If you do not specify a target device before you create the graph, SAS/GRAPH will prompt you (in the Print dialog box) for a device driver name when you attempt to print the graph that you are previewing. (In most cases the WINPRTM or WINPRTC driver is specified by default. The graph colors, orientation, and sizing might not be optimal for the output device you specify. △

## Choosing between a SAS/GRAPH Native Driver and the WINP*xxx* Driver

When deciding whether to use SAS/GRAPH native drivers or the WINP*xxx* series of drivers, consider such factors as the device that you are using and the type of output that you want to produce. Note the following specific considerations:

- If no Windows printer driver is available for your device, use a SAS/GRAPH native driver.
- If you have a device for which there is no SAS/GRAPH native driver, use the WINP*xxx* driver, if there is a Windows printer driver available for the device. In cases where a new model of hardcopy device becomes available between releases of SAS and the hardware vendor provides a new Windows driver that uses new features of the device, you can use a WINP*xxx* driver to take advantage of those features.
- If you want to use options such as HSIZE= or VSIZE= to customize the size specifications used in your graph, using SAS/GRAPH native drivers usually produces more reliable results.

- □ To use TrueType fonts in your SAS/GRAPH output, use one of the WINP*xxx* drivers and specify the font just as you would specify one of the installed hardware fonts for your printer. For more information about TrueType fonts, see “Using TrueType Fonts with SAS/GRAPH Software” on page 613

## Importing Graphics from Other Applications

SAS/GRAPH lets you import bitmap and vector graphics that were created by other software packages. This provides these benefits:

- □ You can create your graphic using another graphics editor, then import the graphic into SAS/GRAPH to produce output on a specialized device.
- □ You can merge clip art and graphs from other applications with the graphs that you create in SAS/GRAPH.

You can import bitmap graphics into these SAS windows:

- □ GRAPH window. The imported graphic becomes a new GRSEG entry in the current catalog.
- □ Graphics Editor. The imported graphic becomes part of the current graph.
- □ Image editor window. The imported graphic becomes a new image.

SAS provides two ways to import bitmap graphics into SAS/GRAPH:

- □ From the application that you used to create the graphic, copy the graphic to the Windows clipboard; then switch to your SAS Session and paste the graphic into the SAS/GRAPH window, as described in “Pasting Graphics from the Windows Clipboard” on page 188.
- □ From the SAS/GRAPH window (or the Graphics Editor or Image Editor) import the graphics file by using the Import Image dialog box, as described in “Importing a Graphics File from within a SAS/GRAPH Window” on page 188.

To import vector graphics, use the GIMPORT procedure to import computer graphics metafile (CGM) files. The imported files are stored as GRSEG catalog entries. This method preserves the individual graphic objects in the imported graph, whereas the other methods treat the imported graphic as a single (uneditable) bitmap. For more information about PROC GIMPORT, see *SAS/GRAPH Reference, Volumes 1 and 2*.

### Pasting Graphics from the Windows Clipboard

If the tool that you use to create the source graphics is a Windows application, then you can use the Windows clipboard to copy the graphics to your SAS session as follows:

1. From the application that you used to create the graphic, select the graphic and copy it to the clipboard using the copy procedures for your graphics tool.
2. Switch to your SAS session (or start your SAS session, if it is not already running).
3. With the SAS/GRAPH window active, select **`Paste`** from the **`Edit`** menu. The graphic is pasted into the SAS/GRAPH window.

### Importing a Graphics File from within a SAS/GRAPH Window

SAS/GRAPH provides import filters to translate graphics files that were created in other applications to a format that you can use with SAS.

You can import graphics from other applications that produce files in any of the formats that are shown in the following table:

**Table 6.7** Graphics Import File Formats

| Graphics File Format | File Extension |
|---|---|
| Microsoft Windows bitmap | BMP |
| Microsoft Windows metafile | WMF |
| enhanced metafile | EMF |
| Device independent bitmap | DIB |
| JPEG format | JPG |
| graphic interchange format (GIF) | GIF |
| tag image file format (TIFF) | TIF |
| PC Paintbrush | PCX |
| Truevision Targa | TGA |
| Encapsulated PostScript Interchange (EPSI) | PS |
| Portable Network Graphics | PNG |
| Photo CD image | PCD |
| Portable Pixmap | PBM |
| X Window bitmap | XBM |
| X Window dump | XWD |

To import bitmap graphics into SAS/GRAPH:

1 Make the GRAPH window the active window and then select **`Import Image`** from the **`File`** menu.

2 Use the Import Image dialog box to select the source directory and graphics file. Be sure that the **`Format`** field shows the correct source format; this indicates which import filter SAS/GRAPH will use. You can have SAS automatically detect the file format of the file to import by selecting AUTO as the format. Click **`OK`**.

   *Note:* Automatic file format detection using AUTO does not detect the DIB, EMF, and WMF file formats △

You can also include IMAGE catalog entries in your graphs. For information about including IMAGE catalog entries, see SAS Help and Documentation.

## Exporting Graphics for Use with Other Applications

SAS provides the following methods of exporting graphics created in SAS/GRAPH for use with other word processing or desktop publishing packages, or for display on the Internet or intranet:

- Export the graphics to a file from the GRAPH window, Graphics Editor, or Image Editor, as described in "Exporting a Graphic to a File from a SAS/GRAPH Window" on page 190.
- Pasting the contents of the Windows clipboard into the target application (as a bitmap), as described in "Pasting Graphics from SAS/GRAPH into other Windows Applications" on page 190.
- Create a computer graphics metafile (CGM) file for use with a specific graphics package, using drivers that are included with SAS, as described in "Creating CGM Files for Export to Other Applications" on page 191.

- Create a Windows metafile for use with another Windows application, as described in “Creating WMF (Windows Metafile) Files for Export to Other Applications” on page 193.

You can also use SAS/GRAPH to create GIF and VRML files for use with Web browsers, PDF files for use with the Adobe Acrobat reader, and many other useful types of graphics files. For more information about how to create these types of files, see *SAS/GRAPH Reference, Volumes 1 and 2* and SAS/GRAPH in SAS Help and Documentation.

## Exporting a Graphic to a File from a SAS/GRAPH Window

SAS/GRAPH provides export filters to translate graphics that were generated in SAS/GRAPH into formats that you can use with other applications, such as spreadsheet and desktop publishing programs.

You can export graphics from SAS/GRAPH in any of the formats that are shown in the following table:

**Table 6.8** Graphics Export File Formats

| Graphics File Format | File Extension |
|---|---|
| Microsoft Windows bitmap | BMP |
| Microsoft Windows metafile | WMF |
| enhanced metafile | EMF |
| Device independent bitmap | DIB |
| JPEG format | JPG |
| graphic interchange format (GIF) | GIF |
| tag image file format (TIFF) | TIF |
| Adobe PostScript | PS |
| Encapsulated PostScript Interchange (EPSI) | PS |
| Portable Network Graphics | PNG |
| Portable Pixmap | PBM |

To export a graph from the GRAPH window:

1 Make the GRAPH window the active window and select **`Export Image`** from the **`File`** pull-down menu.

2 In the Export Image dialog box, select the target file format.

3 Specify the directory and filename for the exported graphic. Click **`OK`**.

For more information about exporting graphics to a SAS IMAGE catalog entry from the Image editor, see SAS Help and Documentation for SAS/GRAPH.

## Pasting Graphics from SAS/GRAPH into other Windows Applications

A quick way to export graphics from SAS to another Windows application is to use the Windows clipboard. When you copy information from SAS/GRAPH to the clipboard, you can then paste that information into any application that accepts DIB, BMP or WMF input.

To copy information from SAS/GRAPH to the clipboard:

1 From the GRAPH window, hold down the left mouse button and drag the mouse over the portion of the graph that you want to copy. A selection box marks off the selected area as you move the mouse. When you are finished, release the mouse button.

   If you do not select an area of the graph to copy, the next step will copy the entire graph to the clipboard.

2 With the GRAPH window still active, press CTRL+C (or select **`Copy to Paste Buffer`** from the **`Edit`** menu).

This copies the graph to the clipboard. You can then return to the target application and paste the graph (typically by using the **`Paste`** or **`Paste Special`** options in the target Windows application). For more information about how to paste information from the clipboard, see the documentation for the other Windows application.

## Creating CGM Files for Export to Other Applications

You can export graphs from SAS/GRAPH to other graphics packages by using drivers that were developed specifically for those packages. When you use computer graphics metafiles (CGMs) as the medium of transport between packages, your graph retains its separate components so that you can independently edit and size it. The editing capabilities you can use depend on the target graphics package.

To create a CGM from SAS/GRAPH, set GOPTIONS as follows:

```
filename fileref 'filename.cgm';
goptions device=cgmxxxx gsfname=fileref
         gsfmode=replace;
```

where CGM*xxxx* is the appropriate CGM driver for your target application, and *filename*.CGM is the name of the file that you want to create. Table 6.9 on page 191 lists the graphics packages to which you can export CGMs, the appropriate drivers to use, and the SAS documents that describe how to export files and how to use them with the target application. You access these documents from SAS Institute's Technical Support Division on the SAS Institute Web site.

The driver names that are marked with an asterisk (*) are already provided with SAS 9.1. For each driver that is not provided, the corresponding document describes how to build the driver.

**Table 6.9** CGM Drivers and Documentation for Popular Graphics Packages

| Package | Suggested Driver | Document |
|---|---|---|
| Aldus PageMaker | CGMAPMA* | TS-252F |
| Aldus Persuasion | CGMAPSA* | TS-252D |
| BPS 35 MM Express | CGM35 | TS-252 |
| Borland Quattro Pro (Windows) | CGMBQWC | TS-252J |
| Borland Quattro Pro (DOS) | CGMBQA* | TS-252 |
| CorelDRAW 5 | CGMCOR5L | TS-252R |
| Frame Tech FrameMaker | CGMFRCA* CGMFRGA* CGMFRMA* PSCFRAME PSGFRAME PSMFRAME | TS-252H |
| Harvard Graphics 2.12 for DOS | CGHHG | TS-252 |
| Harvard Graphics 3.0 for DOS | CGMHG3A* | TS-252A |

| Package | Suggested Driver | Document |
|---|---|---|
| Harvard Graphics for Windows | CGMHGWA* | TS-252C |
| Harvard Graphics 3.0 for Windows | CGMHG3L | TS-252T |
| ImageBuilder | CGMIMG | TS-252 |
| Interleaf 5 | CGMCILFC* CGMGILFG* CGMMILFM* PSILEAFC PSILEAFG PSILEAFM | TS-252I |
| Lotus Ami Pro 3.0 | CGMAM3C* CGMAM3G CGMAM3M | TS-252M |
| Lotus Ami Pro 2.0 | CGMAMIA | TS-252 |
| Lotus Freelance for DOS | CGMFLALJ* CGMFLAPL* CGMFLAPT* | TS-252 |
| Lotus Freelance for Windows 2.0 | CGMFL2C* CGMFL2G CGMFL2M | TS-252K |
| Lotus Freelance for Windows 1.0 | CGMFLWA* | TS-252 |
| Lotus 1-2-3 4.0 | CGM123C* | TS-252N |
| Lotus 1-2-3 | CGM123 | TS-252 |
| Lotus Office 97 | CGMLOT97 CGMLT97P | TS-252Y |
| Microsoft Word for Windows 6.0 | CGMMW6C* CGMMW6G CGMMW6M | TS-252L |
| Microsoft Word for Windows 2.0 | CGMMWWC* | TS-252B |
| Microsoft PowerPoint | CGMMPPA | TS-252E |
| Microsoft Office 97 | CGMOFF97 CGMOF97P | TS-252X |
| Polaroid CI3000 | CI3000 | TS-252 |
| WordPerfect 5.1 for DOS | CGMWPCA CGMWPCAP* CGMWPGA CGMWPGAP* CGMWPMA CGMWPMAP* | TS-252G |
| WordPerfect 5.2 for Windows | CGMWPWA* | TS-252 |
| WordPerfect 6.0 for Windows (Beta) | CGMWP6C CGMWP6G CGMWP6M | TS-252O |
| WordPerfect 6.0A for Windows | CGMWPGCA | TS-252P |
| WordPerfect 6.1 Windows | CGMWPGIL | TS-252S |
| WordPerfect Suite 8 | CGMWP80P | TS-252Z |
| WordPerfect 6.0 for UNIX | CGMWPUXL | TS-252U |

| Package | Suggested Driver | Document |
|---|---|---|
| WordPerfect Presents for DOS | CGMWPCA* CGMWPGA* CGMWPMA* | TS-252G |
| Zenographics Pixie | CGMPIX | TS-252 |

## Creating WMF (Windows Metafile) Files for Export to Other Applications

To learn how to export WMF files from SAS/GRAPH software, contact SAS Institute's Technical Support Division and ask for document TS-352C, which explains this process for several target applications. For information about creating graphics for use with Microsoft Office 97, ask for document TS-252X. For information about creating graphics for use with Lotus Office 97, ask for document TS-252Y. For information about creating graphics for use with Corel Word Perfect Suite 8, ask for document TS-252Z.

## Additional Resources

For full details about using SAS/GRAPH software, see *SAS/GRAPH Reference, Volumes 1 and 2*. For further details about using graphics and fonts with SAS under Windows, see "Using TrueType Fonts with SAS/GRAPH Software" on page 613.

CHAPTER
7

# Performance Considerations

*Hardware Considerations* **195**
- *Processor Speed* **196**
- *Memory* **196**
- *Disk Space for I/O* **197**
- *Graphics Adapter* **197**

*Windows Features That Optimize Performance* **198**
- *Controlling SAS Responsiveness* **198**
  - *Overview of Controlling SAS Responsiveness* **198**
  - *Optimizing Application Performance under Windows NT* **198**
  - *Optimizing Application Performance under Windows 2000* **198**
  - *Optimizing Application Performance under Windows XP* **198**
  - *Optimizing Application Performance under Windows Server 2003* **199**
- *I/O Enhancements for Multiple Processors* **199**
- *Memory-Based Libraries* **199**
  - *Memory-Based Libraries Overview* **199**
  - *Setup Requirements for Using Memory-Based Libraries under Windows NT Server Enterprise Edition 4.0* **200**
  - *Setup Requirements for Using Memory-Based Libraries under Windows 2000 Servers and Later* **201**
  - *Setup Requirements for Using Memory-Based Libraries under Windows 2000 Professional and Windows XP Professional* **201**
  - *Specifying the MEMLIB and MEMCACHE Options in 64-Bit Windows Environments* **202**
  - *Specifying the Local Security Settings* **202**
  - *Processing SAS Libraries as Memory-Based Libraries* **203**
  - *Using a Memory-Based Library as a SAS File Cache* **204**

*SAS Features That Optimize Performance* **205**

*Network Performance Considerations* **206**

*Advanced Performance Tuning Methods* **206**
- *Improving Performance of the SORT Procedure* **206**
  - *SORTSIZE Option* **206**
  - *TAGSORT Option* **207**
  - *Choosing a Location for the Sorted File* **207**
- *Calculating Data Set Size* **208**
- *Increasing the Efficiency of Interactive Processing* **209**

## Hardware Considerations

The following hardware factors might affect SAS performance:

- □ the processor speed
- □ the amount of physical memory that is available

- the amount of disk space that is available for I/O
- the graphics adapter.

Not all of these factors will apply to your particular configuration or to the way in which you run SAS. Consult your system administrator for details.

## Processor Speed

In general, a faster processor enables the computer to perform more operations per second. The more operations that can be performed, the more performance will improve.

The amount of processor cache that is available also influences performance. More processing cache will result in better performance.

The processor required to run SAS depends on the operating environment.

- In 32–bit environments, you must have a PC that contains an Intel or Intel-compatible Pentium class processor.
- In 64–bit environments, you must have an Intel Itanium Processor.

## Memory

In general, more physical memory will result in better performance. Systems that have large amounts of available memory are capable of handling large amounts of data without swapping. Swapping uses the temporary space on the hard drive to store the data that could not be loaded into memory. However, memory is faster than the hard drive in manipulating temporary files and other system operations. Consequently, the more memory that is available, the less the hard drive will need to be accessed for these types of operations.

The minimum amount of memory required depends on the operating environment.

**Table 7.1** Memory Requirements for 32–Bit Environments

| **Operating Environment** | **Memory Requirements** |
|---|---|
| Windows, Windows NT, Windows 2000 Professional, Windows XP Professional | 128 MB minimum (More memory is recommended for improved performance.) |
| | 128 MB minimum of swap file space |
| Windows NT Server and Windows 2000 Server | 256 MB minimum (More memory is recommended for improved performance.) |
| | 256 MB minimum of swap file space |

**Table 7.2** Memory Requirements for 64–Bit Environments

| Operating Environment | Memory Requirements |
|---|---|
| Windows Server 2003 | 1 GB minimum (More memory is recommended for improved performance.) |
| | 1 GB minimu of swap file space |

## Disk Space for I/O

An application uses I/O for data storage and data access. Therefore, faster I/O results in faster overall performance. The three factors that influence I/O performance are the disk controller and bus, the hard drive, and the hard drive configuration.

*disk controller and bus*

In general, hard drive disk controllers that use their memory buffers to cache data have better throughput than conventional controllers. This can improve I/O performance.

The type of I/O controller can affect I/O performance. SCSI and EIDE controllers generally offer higher bus speeds. A fast SCSI or EIDE bus used with the appropriate SCSI or EIDE drive can offer improved performance over other hard drives and controller types. Currently, SCSI controllers offer the best I/O performance.

*hard drive*

Since SAS is heavily I/O oriented, access time and transfer rate are important to system performance. SCSI and EIDE drives generally have faster access times than MFM or IDE drives.

Low disk space or a heavily fragmented disk can hinder I/O performance. It is recommended that you defragment your hard drive regularly to keep I/O performance from degrading.

*hard drive configuration*

The hard drive configuration can have the greatest impact on I/O performance, especially on large server systems. Generally, a RAID configuration has better I/O performance than a non-RAID system. Consult your system administrator to determine the appropriate configuration for your computer.

## Graphics Adapter

Since some SAS features use a significant amount of graphics memory, the type of graphics adapter that you use can make a difference in performance. Generally, if the same amount of graphics memory is available, AGP adapters are faster than PCI adapters. However, the type of adapter that you can use depends on your motherboard.

The amount of memory that is available on the adapter can impact the speed at which graphics are rendered. More memory usually results in better performance.

# Windows Features That Optimize Performance

## Controlling SAS Responsiveness

### Overview of Controlling SAS Responsiveness

You can control the relative responsiveness of your SAS session by altering the application performance level. Using Windows performance options, you can specify which type of programs, interactive or background, receive more processor time. Use these guidelines to determine the application performance level:

- If you are running SAS interactively and you want your session to have the best response time, set the performance options for programs or applications.
- If you are running SAS in batch mode and you want your batch jobs to execute more quickly, set the performance options for background services.

To analyze the performance of your SAS applications, you can specify SAS performance counters within the Windows performance monitor. For more information, see “Performance Tools” on page 226.

### Optimizing Application Performance under Windows NT

Under Windows NT, follow this procedure to optimize application performance:

1. Open the Control Panel.
2. Click **System**.
3. Select the **Performance** tab within the System Properties window.
4. To optimize performance of an interactive SAS session, drag the **Boost** arrow to **Maximum**.
5. To optimize performance of a batch SAS session, drag the Boost arrow halfway between **None** and **Maximum**.
6. Click **OK**.

### Optimizing Application Performance under Windows 2000

Under Windows 2000, follow this procedure to optimize application performance:

1. Open the Control Panel.
2. Click **System**.
3. Select the **Advanced** tab and then click **Performance Options**.
4. To optimize performance of an interactive SAS session, select **Applications**.
5. To optimize performance of a batch SAS session, select **Background services**.
6. Click **OK**.

### Optimizing Application Performance under Windows XP

Under Windows XP, follow this procedure to optimize application performance:

1. Open the Control Panel.
2. Click **System**.
3. Select the **Advanced** tab.

4 In the Performance box, click **`Settings`** and then select the **`Advanced`** tab.
5 To optimize performance of an interactive SAS session, select **`Programs`**.
6 To optimize performance of a batch SAS session, select **`Background services`**.
7 Click **`OK`**.

### Optimizing Application Performance under Windows Server 2003

Under Windows Server 2003, follow this procedure to optimize application performance:

1 Open the Control Panel.
2 Click **`System`**.
3 Select the **`Advanced`** tab.
4 In the Performance box, click **`Settings`**.
5 To optimize performance of an interactive SAS session, select **`Programs`** in the Processor Scheduling box.
6 To optimize performance of a batch SAS session, select **`Background services`** in the Processor Scheduling box.
7 Click **`OK`**.

## I/O Enhancements for Multiple Processors

If your PC has multiple processors, SAS uses symmetric multiprocessing (SMP) using I/O enhancements. More read-ahead processing is done for procedures that have large amounts of sequential data access on data that is stored on a Windows server. This occurs more on systems that have extra processing power to serve Windows and its disk cache.

The following is generally true in multiprocessing SMP environments:

- Machines that are used as servers for multiple applications perform best if they are SMP-based.
- SAS/CONNECT remote computing environments perform best if they are SMP-based.
- The supporting hardware in SMP boxes (RAID, RAM) generally help any application to perform better.

## Memory-Based Libraries

### Memory-Based Libraries Overview

Using the MEMLIB and MEMCACHE options, you can create memory-based SAS libraries. Depending on your operating environment, extended memory or conventional memory is used support these memory-based libraries.

Thirty-two bit processing in Windows operating environments uses 2 gigabytes (GB) of physical memory for the operating environment, leaving 2 GB of physical memory available for use by applications. When a PC or server has more than 4 GB of memory, extended memory is defined as the memory that is above 4 GB, all of which is available for use by applications. Extended memory can be used to support memory-based libraries.

Some Windows operating environments do not support extended memory. In operating environments where extended memory is not supported or installed, SAS uses conventional memory to support the MEMLIB and MEMCACHE options. Conventional memory is defined as the memory that is below 4 GB in 32-bit environments and all of the memory in 64-bit environments.

Using memory-based libraries reduces I/O to and from disk, therefore improving SAS performance. Memory-based libraries can be used

- as storage for the Work library
- for processing SAS libraries that have a high volume of input and output
- as a cache for very large SAS libraries.

Extended memory can be used to support memory-based libraries in

- Windows NT Server Enterprise Edition 4.0
- Windows 2000 Advanced Server
- Windows 2000 Datacenter Server
- 32-bit versions of the Windows Server 2003 Family.

Conventional memory is used to support the MEMLIB and MEMCACHE system options in

- Windows 2000 Professional
- 32-bit and 64-bit versions of Windows XP Professional
- 64-bit versions of the Windows Server 2003 Family.

Windows NT Server Enterprise Edition 4.0 uses Intel's Extended Server Memory Architecture, which allows only one process at a time to access the extended memory. The other servers allow multiple processes to access memory-based libraries simultaneously.

See the following topics for the setup requirements for memory-based libraries:

- "Setup Requirements for Using Memory-Based Libraries under Windows NT Server Enterprise Edition 4.0" on page 200.
- "Setup Requirements for Using Memory-Based Libraries under Windows 2000 Servers and Later" on page 201.
- "Setup Requirements for Using Memory-Based Libraries under Windows 2000 Professional and Windows XP Professional" on page 201.

After you have completed the setup for your operating environment, you use the MEMLIB and MEMCACHE system options and the MEMLIB option in the LIBNAME statement to access memory-based libraries.

## Setup Requirements for Using Memory-Based Libraries under Windows NT Server Enterprise Edition 4.0

You need these system components to use extended memory under Windows NT Server Enterprise Edition 4.0:

- Intel Pentium II Xeon or Pentium III Xeon processor.
- Intel PSE36 device driver. To see if the PSE36 device driver is installed, contact your system administrator.
- Sufficient system memory. More than 4 GB is recommended; you might have better performance if you use 8 to 16 GB. The amount of the extended memory can be increased by changing the boot settings. The boot.ini MAXMEM option reduces the amount of memory that is claimed by Windows NT at system startup. The memory that is not claimed at system startup is used as extended memory.

  *Note:* Reducing the amount of memory that is available to Windows NT can adversely affect system performance. △

- Microsoft Windows NT Server Enterprise Edition 4.0 with Service Pack 3 or later
- SAS 9.1 for Windows.

Only one SAS session can use the PSE36 device driver because only one application can open the PSE36 device driver at a time.

## Setup Requirements for Using Memory-Based Libraries under Windows 2000 Servers and Later

The Windows 2000 memory manager can access up to 8 GB of physical RAM in the Windows 2000 Advanced Server and up to 32 GB of physical RAM in the Windows 2000 Datacenter Server.

To use extended memory under Windows 2000 Servers or above, ensure that your operating environment meets the following requirements:

- Intel or Intel-compatible 32-bit processor (Pentium Pro or later)
- Windows 2000 Advanced Server, Windows 2000 Datacenter Server, Windows Server 2003 Family
- More than 4 GB of RAM; you might have better performance if you use 8 GB or more.
- The parameter **`/PAE`** has been added to the server start-up line in the boot.ini file.
- SAS 9.1 for Windows.

Then follow these steps:

- Set the total amount of memory to be used for memory-based libraries by using the MEMMAXSZ system option. See "MEMMAXSZ System Option" on page 522.
- Set the memory block size for memory-based libraries by using the MEMBLKSZ system option. See "MEMBLKSZ System Option" on page 519.

SAS 9.1 for Windows uses the available memory to support the MEMLIB and MEMCACHE options.

***CAUTION:***

**It is possible to exhaust system memory, causing system failure.** You can use the MEMMAXSZ option to limit the amount of system memory that SAS allocates to the MEMLIB and MEMCACHE options. △

## Setup Requirements for Using Memory-Based Libraries under Windows 2000 Professional and Windows XP Professional

Windows 2000 Professional and Windows XP Professional are limited to 4 GB of physical RAM. Memory-based libraries are designed to optimize performance in server environments where more than 4 GB of physical memory is present. In these operating environments, conventional memory is used to the support the MEMLIB and MEMCACHE options. Specifying these options in Windows 2000 Professional or Windows XP Professional environments might or might not improve performance.

To use memory-based libraries under Windows 2000 Professional and Windows XP Professional, ensure that your operating environment meets the following requirements:

- Intel or Intel-compatible 32-bit processor (Pentium Pro or later)
- Windows 2000 Professional and Windows XP Professional
- 4 GB of RAM
- SAS 9.1 for Windows.

Then follow these steps:

- Set the total amount of memory for memory-based libraries by using the MEMMAXSZ system option. See "MEMMAXSZ System Option" on page 522.

- □ Set the memory block size for memory-based libraries by using the MEMBLKSZ system option. See “MEMBLKSZ System Option” on page 519.

SAS 9.1 for Windows uses the available memory to support the MEMLIB and MEMCACHE options.

***CAUTION:***
**It is possible to exhaust system memory causing system failure.** You can use the MEMMAXSZ option to limit the amount of system memory that SAS allocates to the MEMLIB and MEMCACHE options. △

## Specifying the MEMLIB and MEMCACHE Options in 64-Bit Windows Environments

Sixty-four bit processing in Windows operating environments uses 16 terabytes (TB) of virtual address space, so in these environments, extended memory is not needed. SAS uses the conventional memory that is available to support the MEMLIB and MEMCACHE options.

***CAUTION:***
**It is possible to exhaust system memory causing system failure.** You can use the MEMMAXSZ option to limit the amount of system memory that SAS allocates for the MEMLIB and MEMCACHE options. △

To use the MEMLIB and MEMCACHE options, ensure that the operating environment meets the following requirements:

- □ Intel or Intel-compatible 64-bit processor (Itanium or later)
- □ Any 64-bit version of Windows
- □ 4 GB of RAM or more; you might have better performance if you use more than 8 GB.
- □ SAS 9.1 for Windows.

Then follow these steps:

- □ Set the total amount of memory for memory-based libraries by using the MEMMAXSZ system option. See “MEMMAXSZ System Option” on page 522.
- □ Set the memory block size for memory-based libraries by using the MEMBLKSZ system option. See “MEMBLKSZ System Option” on page 519.

## Specifying the Local Security Settings

In 32-bit environments starting with Windows 2000 Professional and above, the **`Lock pages in memory security`** setting must be set, so that each user who is running SAS has access to the extended memory. If a user does not have the correct permissions, then SAS will issue a warning message to the log.

System administrators can set the local security settings through the Start Menu. To set this value

1. Open the Control Panel.
2. Double click **`Administrative Tools`**.
3. Double click **`Local Security Policy`**. The Local Security Settings window will open.
4. Double click **`Local Policies`**.
5. Double click **`User Rights Assignment`**.
6. Double click **`Lock pages in memory`**. The Local Security Policy Setting window will open.

**7** Click Add. The Select User or Groups window will open.

**8** Select the user IDs or name of the group of users who need to run SAS with access to the extended memory.

**9** Click OK in the Select User or Groups window.

**10** Click OK in the Local Security Policy window.

*Note:* In the Local Security Policy window, you might see two check boxes: the Local Security Setting check box and the Effective Policy Setting check box. If the Effective Policy Setting check box is not selected for the users that you added, you will need to reboot your computer so that the new security settings will take effect. △

## Processing SAS Libraries as Memory-Based Libraries

SAS libraries that are well suited for memory-based processing have data that is referenced or updated multiple times within a SAS session.

Using a Work library that is memory-based is beneficial for procedures such as PROC SORT that write multiple times to large temporary files. To designate the Work library as memory-based, specify the MEMLIB system option when you start SAS.

You designate a library as memory-based by using the MEMLIB option in the LIBNAME statement. All librefs, including a libref to the Work directory, must have a valid disk directory.

After the library is designated as memory-based, your SAS program needs to copy the library from disk to memory. After processing the library in memory, the library must be copied back to disk.

***CAUTION:***

**Copy the library that is in memory to disk after processing is complete.** If you do not, you will lose any changes that were made to the library. The changes are lost when the SAS session ends △

The following example shows how to use the LIBNAME statement and the PROC COPY statement to copy a library to and from memory.

```
/* Set up two librefs, one to the library in memory
and the other to the SAS library on disk.  The library on
disk contains dataset1, dataset2, dataset3 and dataset4. */

libname inmemory ‘‘g:\memlib’’ memlib;
libname ondisk ‘‘g:\disk’’;

/* Copy dataset1, dataset2, dataset3, and dataset4 to memory   */

proc copy in=ondisk out=inmemory;
run;

/* ...Assume dataset1 and dataset4 are updated   */

/* Save the updated datasets back to disk         */

proc copy in=inmemory out=ondisk;
   select dataset1 dataset4;
run;
```

You can also copy a data set to memory by using a DATA statement, as shown in the following example:

```
data ondisk.dataset1;
   set inmemory.dataset1;
run;
```

For more information, see “MEMLIB System Option” on page 521 and the LIBNAME Statement MEMLIB option“LIBNAME Statement” on page 456.

## Using a Memory-Based Library as a SAS File Cache

A SAS file cache is most useful in multiple references of data. For example, a SAS file cache improves performance in SAS programs that make multiple passes of the data. SAS file caching improves performance in the following situations:

- Repeated read operations of a file while other files are being written. Writing to a file clears the Windows NT file system (NTFS) cache.
- Repeated read operations of a file when Scatter Gather I/O is active. Scatter Gather I/O operates outside the NTFS cache. Without the SAS file cache, there is no data cache and all read operations access the disk.
- Repeated read operations when Windows is low on memory. In systems that have high memory usage, the performance of the NTFS cache is degraded.

To use memory as a SAS file cache, specify the MEMCACHE system option when you start SAS or when you submit an OPTIONS statement. If you set MEMCACHE to 4, SAS uses the memory to cache all files. If you set MEMCACHE to 1, only files that are currently in memory are cached to memory. When you use the MEMCACHE system option in the OPTIONS statement, you can control which data sets use the SAS file cache, as shown in the following example.

```
/* Example of controlling cached files with the options statement   */

/* Assume cachelib contains 2 data sets, ds1 and ds2.               */
/* Also assume ds1 and ds2 are large enough that they cannot exist */
/* in the cache together. ds1 is read many times, so caching is     */
/* desired. ds2 is accessed only once, so caching is of no         */
/* benefit.  When you use the memcache option, ds1 is cached, and ds2   */
/* is not cached.                                                     */

libname cachelib "e:\tmp";

/* Turn on full caching */

options memcache = 4;

/* Read ds1 and place the data in the cache.  This read operation could be a  */
/* more useful read operation of the file in a real case.               */

data _null_;
  set cachelib.ds1;
run;

/* Change memcache setting to use the cache only for files that   */
/* already exist in the cache.                                    */

options memcache = 1;

/* Data from ds1 will come from the cache and ds2 will not be     */
/* cached.                                                        */
```

```
proc sort data=cachelib.ds1 out=cachelib.ds2;
  by j;
run;

/* Other access of ds1...                                  */

/* All use of the cache can be ended with a memcache system      */
/* option value of 0.                                      */

options memcache = 0;

/* Neither ds1 or ds2 will access the cache.               */

proc sort data=cachelib.ds1 out=cachelib.ds2;
  by j;
run;
```

For more information about the MEMCACHE system option, see “MEMCACHE System Option” on page 520. For more information about Scatter Gather I/O, see “SAS Features That Optimize Performance” on page 205 and “SGIO System Option” on page 550.

## SAS Features That Optimize Performance

The following are some additional features of SAS that you can control to improve system performance and make efficient use of your computer’s resources. For additional information about optimizing SAS performance, see the chapter on optimizing system performance in *SAS Language Reference: Concepts*.

- Create SAS data sets instead of accessing flat ASCII files. SAS can access a SAS data set more efficiently than it can read flat files.

  Also, you should convert existing data sets that you use frequently to SAS 9.1 format.
- In your SAS code, use IF-THEN-ELSE conditional structures instead of multiple IF-THEN structures. When one condition in the IF-THEN-ELSE structure is met, control returns to the top of the structure (skipping the ELSE clause, which might contain subsequent IF-THEN structures). With multiple IF-THEN structures, each condition must be checked.
- When using arrays, make them _TEMPORARY_ if possible. This requires less memory and less time for memory allocation.
- Use programming structures that reduce file I/O, the most time-intensive aspect of SAS processing. Some ideas for reducing file I/O are:
  - Use the WHERE statement in a procedure to reduce extra data processing.
  - Use indexed data sets to speed access to the desired observations.
  - Use the SQL procedure to subset and group your data.
- Experiment with the value of the CATCACHE system option, which specifies the number of SAS catalogs to keep open at one time. By default, no catalogs are cached in memory (and CATCACHE is set to 0). Caching catalogs is an advantage if one SAS application uses catalogs that are subsequently needed by another SAS application. The second SAS application can access the cached catalog more efficiently.

*Note:* Storing catalogs in memory can consume considerable resources. Use this technique only if memory issues are not a concern. △

- □ Store your data sets in a compressed format (using the COMPRESS data set option). This can improve the performance of your SAS application, though it might require more CPU time to decompress observations as SAS needs them. The COMPRESS data set option is described in the data set options section of *SAS Language Reference: Dictionary*.
- □ If you specify the Scatter-read/Gather-write system option, SGIO, SAS bypasses intermediate buffer transfers when reading or writing data. SAS will read ahead the number of pages specified by the BUFNO system option and place the data in memory before it is needed. When the data is needed it is already in memory, and is in effect a direct memory access. Different values for the BUFNO system option should be tried for each SAS job to find the maximum performance benefit.

  Scatter–read / gather–write is active only for SAS I/O opened in INPUT or OUTPUT mode. If any SAS I/O files are opened in UPDATE or RANDOM mode, SGIO is inactive for that process. Compressed and encrypted files can also be read ahead using scatter-read/gather-write. For more information on the SGIO system option, see “SGIO System Option” on page 550.

  *Note:* If you are using Windows NT 4, Service Pack 4 is required to use the SGIO system option. △

## Network Performance Considerations

Under Windows, loading application DLL (dynamic link library) files from a network drive can result in slower performance than loading the DLL files from a local drive.

## Advanced Performance Tuning Methods

This section presents some advanced performance topics, such as improving the performance of the SORT procedure and calculating data set size. Use these methods only if you are an experienced SAS user and you are familiar with the way SAS is configured on your machine.

### Improving Performance of the SORT Procedure

Two options for the PROC SORT statement are available under Windows, the SORTSIZE= and TAGSORT options. These two options control the amount of memory the SORT procedure uses during a sort and are discussed in the next two sections. Also included is a discussion of determining where the sorting process occurs for a given data set and determining how much disk space you need for the sort. For more information about the SORT procedure, see .

#### SORTSIZE Option

The PROC SORT statement supports the SORTSIZE= option, which limits the amount of memory available for PROC SORT to use.

If you do not use the SORTSIZE option in the PROC SORT statement, PROC SORT uses the value of the SORTSIZE system option. If the SORTSIZE system option is not set, PROC SORT uses the amount of memory specified by the REALMEMSIZE system option. If PROC SORT needs more memory than you specify, it creates a temporary utility file in your Saswork directory to complete the sort.

The default value of this option is 64 megabytes (MB).

## TAGSORT Option

The TAGSORT option is useful in single-threaded situations where there may not be enough disk space to sort a large SAS data set. The TAGSORT option is not supported for multi-threaded sorts.

When you specify the TAGSORT option, only sort keys (that is, the variables specified in the BY statement) and the observation number for each observation are stored in the temporary files. The sort keys, together with the observation number, are referred to as tags. At the completion of the sorting process, the tags are used to retrieve the records from the input data set in sorted order. Thus, in cases where the total number of bytes of the sort keys is small compared with the length of the record, temporary disk use is reduced considerably. However, you should have enough disk space to hold another copy of the data (the output data set) or two copies of the tags, whichever is greater. Note that although using the TAGSORT option can reduce temporary disk use, the processing time may be much higher.

## Choosing a Location for the Sorted File

Where the physical sort occurs for a given data set depends on how you reference the data set name and whether you use the OUT= option in the PROC SORT statement. You might want to know where the sort occurs if you think there might not be enough disk space available for the sort.

When you sort a SAS data set, SAS creates a temporary utility file. If the sort uses multiple threads, you can specify the location of the utility file by using the UTILLOC system option. The default location for utility files is the Work data library. If two or more locations are specified for the UTILLOC option, the second location is used as the location for the utility file. For sorts that use a single thread, the temporary utility file is opened in the Work data library if there is not enough memory to hold the data set during the sort. The utility file has a .sas7butl file extension. Before you sort, be sure that your Work data library has room for this temporary utility file.

If you specify the OVERWRITE option on the PROC SORT statement, SAS replaces the input data set with the sorted data set.

If you do not specify the OVERWRITE option on the PROC SORT statement, a second file that has a .sas7butl file extension is created. If the sort completes successfully, this file is renamed to the data set name of the file being sorted (with a .sas7bdat file extension). The original data set is deleted after the sort is complete. Before you sort a data set, be sure that you have space for this .sas7butl file.

Use the following rules to determine where the .sas7butl file and the resulting sorted data set are created:

□ If you omit the OUT= option in the PROC SORT statement, the data set is sorted on the drive and in the directory or subdirectory where it is located. For example, if you submit the following statements (note the two-level data set name), the .sas7butl file is created on the C: drive in the MYDATA subdirectory:

```
libname mylib 'c:\sas\mydata';
proc sort data=mylib.report;
   by name;
run;
```

Similarly, if you specify a one-level data set name, the .sas7butl file is created in your Work data library.

□ If you use the OUT= option in the PROC SORT statement, the .sas7butl file is created in the directory associated with the libref used in the OUT= option. If you use a one-level name (that is, no libref), the .sas7butl file is created in the Work data library. For example, in the following SAS program, the first sort occurs in the Saswork subdirectory, while the second occurs on the F: drive in the JANDATA directory:

```
proc sort data=report out=newrpt;
   by name;
run;
libname january 'f:\jandata';
proc sort data=report out=january.newrpt;
   by name;
run;
```

## Calculating Data Set Size

In single-threaded environments, you always need free disk space that equals three to four times the data set size. For example, if your data set takes up 1MB of disk space, you need 3 to 4MB of disk space to complete the sort.

In multi-threaded environments, if you use the OVERWRITE option on the PROC SORT statement, you need space equal to the data set size. If you do not specify the OVERWRITE option, the space you need is equal to two times the data set size. For more information about the OVERWRITE option, see the SORT procedure in *Base SAS Procedures Guide*.

To estimate the amount of disk space that is needed for a SAS data set:

1 create a dummy SAS data set that contains one observation and the variables you need
2 run the CONTENTS procedure using the dummy data set
3 determine the data set size by performing simple math using information from the CONTENTS procedure output.

For example, for a data set that has one character variable and four numeric variables, you would submit the following statements:

```
data oranges;
  input variety $ flavor texture looks;
  total=flavor+texture+looks;
  datalines;
navel 9 8 6
;
proc contents data=oranges;
  title 'Example for Calculating Data Set Size';
run;
```

These statements generate the output shown in the following output:

**Output 7.1** Example for Calculating Data Set Size with PROC CONTENTS

```
                      Example for Calculating Data Set Size                    1
                                            19:39 Wednesday, February 12, 2003

                              The CONTENTS Procedure

Data Set Name        WORK.ORANGES                      Observations          1
Member Type          DATA                              Variables             5
Engine               V9                                Indexes               0
Created              Wednesday, February               Observation Length    40
                     12, 2003 07:41:04
Last Modified        Wednesday, February               Deleted Observations  0
                     12, 2003 07:41:04
Protection                                             Compressed            NO
Data Set Type                                          Sorted                NO
Label
Data Representation  WINDOWS_32
Encoding             wlatin1  Western (Windows)


                        Engine/Host Dependent Information

Data Set Page Size          4096
Number of Data Set Pages    1
First Data Page             1
Max Obs per Page            101
Obs in First Data Page      1
Number of Data Set Repairs  0
File Name                   C:\TEMP\SAS Temporary Files\_TD246\oranges.sas7bdat
Release Created             9.0101B0
Host Created                WIN_NT


                    Alphabetic List of Variables and Attributes

                           #    Variable    Type    Len

                           2    flavor      Num       8
                           4    looks       Num       8
                           3    texture     Num       8
                           5    total       Num       8
                           1    variety     Char      8
```

The size of the resulting data set depends on the data set page size and the number of observations. The following formula can be used to estimate the data set size:

number of data pages = 1 + (floor(number of obs / **`Max Obs per Page`**))

size = 1024 + (**`Data Set Page Size`** * number of data pages)

(*floor* represents a function that rounds the value down to the nearest integer.)

Taking the information shown in Output 7.1, you can calculate the size of the example data set:

number of data pages = 1 + (floor(1/101))

size = 1024 + (4096 * 1) = 5120

Thus, the example data set uses 5,120 bytes of storage space.

## Increasing the Efficiency of Interactive Processing

If you are running a SAS job using SAS interactively and the job generates numerous log messages or extensive output, consider using the AUTOSCROLL command to suppress the scrolling of windows. This makes your job run faster because

SAS does not have to use resources to update the display of the LOG and OUTPUT windows during the job. For example, issuing **`autoscroll 0`** in the LOG window causes the LOG window not to scroll until your job is finished. (For the OUTPUT window, AUTOSCROLL is set to 0 by default.)

Minimizing the LOG window also might make your job run faster, especially if SAS is generating numerous log messages.

PART 2

# Using SAS with Other Windows Applications

*Chapter* ***8***. . . . . . . . . . **Using Lotus Notes to Distribute SAS Data** *213*

*Chapter* ***9***. . . . . . . . . . **Using Windows System Tools with SAS** *221*

*Chapter* ***10***. . . . . . . . . **Using OLE in SAS/AF Software** *241*

*Chapter* ***11***. . . . . . . . . **Controlling SAS from Another Application Using OLE** *265*

*Chapter* ***12***. . . . . . . . . **Using Dynamic Data Exchange** *273*

*Chapter* ***13***. . . . . . . . . **Using Unnamed and Named Pipes** *283*

*Chapter* ***14***. . . . . . . . . **Accessing External DLLs from SAS** *295*

*Chapter* ***15***. . . . . . . . . **Considerations for SAS/AF Programmers** *315*

CHAPTER

8

# Using Lotus Notes to Distribute SAS Data

*Introduction to Using Lotus Notes with SAS* **213**
*The NOTESDB Engine* **213**
*Client Requirements* **213**
*Populating a Lotus Notes Database Using the DATA Step and SCL Code* **214**
*Creating New Notes Documents* **214**
*SAS Statements to Interact with Notes* **214**
*Syntax for Populating a Lotus Notes Database* **214**
*Examples of Populating Lotus Notes Databases* **216**
*Preparing SAS/GRAPH Output for a Notes Document* **218**
*Using SAS with the NotesSQL ODBC Driver* **219**
*Retrieving Information from Preexisting Notes Documents* **219**

## Introduction to Using Lotus Notes with SAS

### The NOTESDB Engine

SAS provides an access engine, NOTESDB, that enables client users to add new Notes documents to an existing Notes database. This engine is compatible with Lotus Notes R5.

The NOTESDB access engine cannot create a new database and does not provide a way to retrieve information from existing Notes documents. However, the NotesSQL ODBC driver allows for retrieval of data from Notes databases. See "Retrieving Information from Preexisting Notes Documents" on page 219 for these details.

Do not schedule a program using the NOTESDB engine as a batch job because the server prompts for a password.

### Client Requirements

The following are requirements for using Lotus Notes with SAS:

□ A client version of Lotus Notes and a valid Notes user ID certification must be installed on the machine that will be using the NOTESDB engine. However, Lotus Notes does not have to be running in order for SAS to access it. You will be prompted for a password to access the Notes server through SAS.

□ The Lotus Notes directory must be in the system path. You update the system path by adding to the PATH system environment variable in the `Advanced` tabbed page of the System Properties dialog box.

## Populating a Lotus Notes Database Using the DATA Step and SCL Code

### Creating New Notes Documents

#### SAS Statements to Interact with Notes

DATA step and SCL code that interacts with a Notes database has the following components:

- □ a FILENAME statement that includes the NOTESDB device-type keyword
- □ PUT statements that contain data directives and the data to place in the Notes database
- □ PUT statements that contain action directives to control when to send the data to the Notes database.

#### Syntax for Populating a Lotus Notes Database

**FILENAME** *fileref* NOTESDB;

where:

*fileref*
is a valid fileref.

NOTESDB
is the device-type keyword that indicates that you want to use a Lotus Notes database.

In your DATA step, use PUT statements that have data directives to define which database you want to use and the data you want to send.

*Note:* Although the directives that you specify to access a Notes database are not case-sensitive, the fields that you specify using those directives are. Also, only one directive per PUT statement is permitted. Each directive should be delimited with an exclamation point and surrounded with single quotes. △

Use these data directives to specify the database location and the data to add to the database:

!NSF_SERVER! *server-name*
indicates the Notes server to access, where *server-name* represents a Lotus Notes server. If you do not specify this directive, SAS uses your local system as the source for the databases. If you specify this directive more than once, the server that was specified in the most recent PUT statement is used.

*Note:* If you attempt to access a Notes server through SAS, you will be prompted for your password to the server. △

!NSF_DB! *database-filename*
indicates the Notes database file to access. When accessing a database locally, SAS looks for the database in the Notes data directory. If it is not found there, SAS searches the system path. Alternatively, you can specify the fully qualified path for the database. You must specify a Notes database file that has this directive before

you can access a Notes database from SAS. If you specify this directive more than once, the database that was specified in the most recent PUT statement is used.

!NSF_FORM! *form-name*
: specifies the form that Notes should use when displaying the added note. If this directive is not specified, Notes uses the default database form. If you specify this directive more than once, the form that was specified in the most recent PUT statement that has the !NSF_FORM! directive is used.

!NSF_ATTACH! *filename*
: attaches a file to the added note. SAS looks for the file in the Notes data directory. If it is not found there, SAS searches the system path. Alternatively, you can specify the fully qualified path for the file. You can attach only one file in a single PUT statement that has the !NSF_ATTACH! directive. To attach multiple files, use separate PUT statements that have !NSF_ATTACH! directives for each file.

!NSF_FIELD! *field-name!field value*
: adds the value to the field name specified. SAS detects the correct format for the field and formats the data accordingly. Note that SAS extracts all line feeds or carriage returns; you should not insert any of these control characters as they affect the proper display of the document in Notes. Multiple PUT statements that have the !NSF_FIELD! directive and the same field name will concatenate the information in that field. Also, PUT statements that have no directives are concatenated to the last field name submitted, or they are ignored if no PUT statements that have !NSF_FIELD! directives have previously been submitted.

You can populate fields, which can be edited, of the following types:

- text
- numeric
- keywords

You may add text which will be formatted by Lotus Notes. You may also add a bitmap (in Windows bitmap format) using the following form:

**!NSF_FIELD!** *field-name <bitmap-filename>*

Use these action directives to perform actions on the Notes database:

!NSF_ADD!
: immediately adds a document to the Notes database within the DATA step program.

!NSF_ABORT!
: indicates not to add the note when closing the data stream. By default, the driver attempted to add a note at the end of a SAS program for every FILE statement used.

!NSF_CLR_FIELDS!
: clears all the field values that were specified by the !NSF_FIELD! directive. This directive in conjunction with !NSF_ADD! facilitates writing DATA step programs with loops that add multiple notes to multiple databases.

!NSF_CLR_ATTACHES!
: clears all the field values that were specified by the !NSF_ATTACH! directive. This directive in conjunction with !NSF_ADD! facilitates writing DATA step programs with loops that add multiple notes to multiple databases.

*Note:* The contents of PUT statements that do not contain directives are concatenated to the data that is associated with the most recent field value. △

## Examples of Populating Lotus Notes Databases

The following example uses the Business Card Request database that is supplied by Lotus Notes. This DATA step creates a new document in the database and supplies values for all of its fields.

**Example Code 8.1** Using the Business Card Request Database

```
01 filename reqcard NOTESDB;
02 data _null_;
03  file reqcard;
04  put '!NSF_DB! examples\buscard.nsf';
05  put '!NSF_FIELD!Status! Order';
06  put '!NSF_FIELD!Quantity! 500';
07  put '!NSF_FIELD!RequestedBy! Systems';
08  put '!NSF_FIELD!RequestedBy_CN! Jane Doe';
09  put '!NSF_FIELD!NameLine_1! Jane Doe';
10  put '!NSF_FIELD!NameLine_2! Developer';
11  put '!NSF_FIELD!AddressLine_1! Software R Us';
12  put '!NSF_FIELD!AddressLine_2! 123 Silicon Lane';
13  put '!NSF_FIELD!AddressLine_3! Garner, NC 27123';
14  put '!NSF_FIELD!AddressLine_4! USA';
15  put '!NSF_FIELD!PhoneLine_1! (910) 777-3232';
16 run;
```

Line 1 assigns a fileref by using the FILENAME statement to point to Notes instead of to an ordinary file. NOTESDB is the device type for Lotus Notes. Line 3 uses the assigned fileref to direct output from the PUT statement. Line 4 indicates which Notes database to open. Lines 5 to 15 specify the field and the value for that field for the new Notes document that is being created. Status is the field name and Order is the value that is placed in the Status field for the particular document. Line 16 executes these SAS statements. A new Notes document is created in the Business Card Request database.

The next example uses each observation in the SALES data set to create a new document in the **`qrtsales.nsf`** database and fills in the **`Sales`**, **`Change`**, and **`Comments`** fields for the documents.

**Example Code 8.2** Creating a New Document from a Data Set

```
01 data sasuser.sales;
02    length comment $20;
03    format comment $char20.;
04    input sales change comment $ 12-31;
05 datalines;
06 123472 342 Strong Increase
07 423257 33  Just enough
09 218649 4   Not high enough
09 ;
10 run;
11 filename sales NOTESDB;
12 data _null_;
13    file sales;
14    set sasuser.sales;
15    put '!NSF_DB! qrtsales.nsf';
16    put '!NSF_FORM! Jansales';
17    put '!NSF_ADD!';
18    put '!NSF_FIELD!Sales !' sales;
```

```
19    put '!NSF_FIELD!Change!' change;
20    put '!NSF_FIELD!Comments!' comment;
21    put '!NSF_CLR_FIELDS!';
22 run;
```

Line 11 assigns a fileref by using the FILENAME statement to point to Notes instead of to an ordinary file. NOTESDB is the device type for Lotus Notes. Line 13 uses the assigned fileref to direct the output from the PUT statement. In line 15, the NSF_DB data directive indicates which Notes database to open. Lines 18, 19, and 20 specify the field and its value for the new Notes document that is being created. **`Sales`** is the field name and **`sales`** is the value that is placed in the Status field for the particular document. Line 22 executes these SAS statements. A new Notes document is created in the Sales database.

Expanding on the Business Card Request database example, you can create multiple Notes documents within a single DATA step or within SCL code by using action directives as well as data directives. The following example shows how to create multiple Notes documents within a single DATA step.

**Example Code 8.3** Creating Multiple Notes Documents within a Single DATA Step

```
01 filename reqcard NOTESDB;
02 data _null_;
03   file reqcard;
04   put '!NSF_DB!Examples\buscard.nsf';
05   put '!NSF_FIELD!Status! Order';
06   put '!NSF_FIELD!Quantity! 500';
07   put '!NSF_FIELD!RequestedBy!Systems';
08   put '!NSF_FIELD!RequestedBy_CN! Jane Doe';
09   put '!NSF_FIELD!NameLine_1! Jane Doe';
10   put '!NSF_FIELD!NameLine_2! Developer';
11   put '!NSF_FIELD!AddressLine_1! Software R Us';
12   put '!NSF_FIELD!AddressLine_2! 123 Silicon Lane';
13   put '!NSF_FIELD!AddressLine_3! Garner, NC 27123';
14   put '!NSF_FIELD!AddressLine_4! USA';
15   put '!NSF_FIELD!PhoneLine_1! (910) 555-3232';
16   put '!NSF_ADD!';
17   put '!NSF_CLR_FIELDS!';
18   put '!NSF_FIELD!Status! Order';
19   put '!NSF_FIELD!Quantity! 10';
20   put '!NSF_FIELD!RequestedBy! Research and Development';
21   put '!NSF_FIELD!RequestedBy_CN! John Doe';
22   put '!NSF_FIELD!NameLine_1! John Doe';
23   put '!NSF_FIELD!NameLine_2! Analyst';
24 put '!NSF_FIELD!AddressLine_1! Games Inc';
25 put '!NSF_FIELD!AddressLine_2! 123 Software Drive';
26 put '!NSF_FIELD!AddressLine_3! Cary, NC 27511';
27 put '!NSF_FIELD!AddressLine_4! USA';
28 put '!NSF_FIELD!PhoneLine_1! (910) 555-3000';
29run;
```

Line 1 assigns a fileref by using the FILENAME statement to point to Notes instead of to an ordinary file. NOTESDB is the device type for Lotus Notes. Line 3 uses the assigned fileref to direct the output from the PUT statement. Line 4 indicates which Notes database to open. Lines 5 to 15 specify the field and the value for that field for the new Notes document that is being created. **`Status`** is the field name and **`Order`** is the value placed in the **`Status`** field for this particular document. Line 16 forces the creation of a new Notes document. Line 17 clears the values for the fields that are used with the !NSF_FIELD! data directives in the previous lines. Lines 18 to 28 specify the field and the value for that field for the second Notes document that is being created. `Status` is the field name and `Order` is the value placed in the `Status` field for the second document. Line 29 executes these SAS statements. A second Notes document is created in the Business Card Request database.

Only one !NSF_DB! data directive is issued in the preceding example. By default, the second Notes document is created in the same database as that referenced in the !NSF_DB! data directive on line 4. In order to create the second Notes document in another database, you would have to issue another !NSF_DB! data directive with the new database filename prior to the execution of line 18. The key additions to this example are the action directives on lines 16 and 17.

*Note:* All directives are not case sensitive. However, the values following the data directives, such as form name and field name, are case sensitive. △

## Preparing SAS/GRAPH Output for a Notes Document

SAS/GRAPH output can be passed to a Notes document through the NOTESDB access engine. A slight variation of the syntax for the !NSF_FIELD! data directive enables SAS/GRAPH output to be directed to a rich text format field in a Notes document. The procedure is:

- □ Export the SAS/GRAPH output to a bitmap file format.
- □ Use the modified !NSF_FIELD! data directive syntax to assign the value of the bitmap filename to an RTF field. Syntax is:

  **!NSF_FIELD!** *RTF-field-name < bitmap-filename*

The following example uses the modified syntax.

*Note:* This example uses the Electronic Library sample database. △

**Example Code 8.4** Exporting SAS/GRAPH Output into a Notes Document

```
01 filename myfile 'test1.bmp';
02  goptions device=bmp gsfname=myfile gsfmode=replace;
03  title1 'US Energy Consumption for 1955-1988';
04  proc gplot data=3Dsampsio.energy1;
05  plot consumed*year / des=3D'D0319U01-1';
06 run;
07 quit;

08 filename newdoc NOTESDB;
09 data _null_;
10 file newdoc;
11  put '!NSF_DB!Examples\hrdocs.nsf';
12  put '!NSF_FIELD!Subject! US Energy Consumption';
13  put '!NSF_FIELD!Categories! Office Services';
14  put '!NSF_FIELD!Body! US Energy Consumption for 1955-1988';
15  put '!NSF_FIELD!Body<c:\usenergy.bmp';
16 run;
```

Lines 1 to 6 contain code that is taken from the SAS/GRAPH samples by using a sample data set to generate SAS/GRAPH output. Line 8 assigns a fileref by using the FILENAME statement to point to Notes instead of to an ordinary file. NOTESDB is the device type for Lotus Notes. Line 10 uses the assigned fileref to direct the output from the PUT statement. Line 11 indicates which Notes database to open. Lines 12 to 14 specify the field and the value for that field for the new Notes document that is being created. **`Subject`** is the field name and **`US Energy Consumption`** is the value that is placed in the **`Subject`** field for this particular document. Line 15 indicates a display of **`usenergy.bmp`** bitmap file in the **`Body`** field because the less than symbol (<) rather than exclamation point (!) is used to separate the field value from the field name. Line 16 executes these SAS statements. A new Notes document is created in the Electronic Library database.

In the preceding example, the **`Detailed`** field is an RTF field. When using RTF fields, you can intersperse data and bitmaps.

## Using SAS with the NotesSQL ODBC Driver

SAS also provides a SAS/ACCESS to ODBC pass-through engine that enables you to retrieve information about existing Notes documents in a Notes database. You can retrieve text fields only. Graphical data cannot be retrieved.

*Note:* You must have configured a Notes ODBC driver and data source. △

The following software is required:

- □ Lotus Notes Client Version 5 (32-bit only)
- □ ODBC Driver NotesSQL 3.01 (32–bit).

  NotesSQL is an ODBC driver that is provided by Lotus. It can be downloaded free of charge from **`http://www.lotus.com`**.

After you have the software, you must

1. Set up the NotesSQL ODBC driver.

   Lotus provides a file (.nfs) that explains how to set up the driver.
2. Configure the ODBC data source.

   You must complete the Lotus Notes ODBC 2.0 Setup screen. Add the appropriate information to these fields:

   *Note:* Examples are in parentheses. △

   - □ Data Source name (**`buscard`**)
   - □ Description (**`Test Notes Access`**)
   - □ Server or Database name (**`c:\notes\data\buscard.nsf`**).
   - □ NotesSQL Options Setup, which contains these fields:
     - □ Max Length of Text Fields (254)
     - □ Max Number of Tables (20)
     - □ Max Number of Subqueries (20)
     - □ Max Length of Rich Text Fields (512)
     - □ SQL Statement Length (4096)
     - □ Thread Timeout in seconds (60).

       Click **`OK`** after you have completed the Setup screen.

## Retrieving Information from Preexisting Notes Documents

The SAS/ACCESS to ODBC pass-through engine enables you to retrieve information about existing Notes documents in a Notes database. Example Code 8.5 on page 220 shows an example of how to use the DATA step to retrieve information from the Business Card Request database.

**Example Code 8.5** Using ODBC to Retrieve Information from Preexisting Notes Documents

```
proc sql;
   connect to ODBC ("dsn=3Dbuscard");
   create table sasuser.buscard as
   select * from connection to
   ODBC (select * from All_Requests_By_Organization);
   disconnect from ODBC;
run;
```

Line 1 processes SQL statements to manipulate SQL views and tables. Line 2 connects to ODBC, which establishes a connection to Notes through the SAS/ACCESS to ODBC driver and the NotesSQL ODBC driver by using the 3Dbuscard data source. Lines 3, 4, and 5 create a table and sasuser.buscard from the data that is retrieved from the Notes Business Card Request database table that is called **`All_Requests_By_Organization`**. This is the default view that is assigned to the Business Card Request database. Line 6 disconnects from ODBC and closes the connection to the Notes database. Line 7 executes these SAS statements. A new data set named **`buscard`** is created in the Sasuser library.

As another alternative, you may view the available tables within Notes databases by using the SQL Query Window. The SQL Query Window, a component of SAS, is an interactive interface that enables you to easily build queries without writing programming statements. You can invoke it by issuing the QUERY command from the command line.

For more information about PROC SQL, see *Base SAS Procedures Guide*.

CHAPTER

9

# Using Windows System Tools with SAS

*Introduction to Using Windows System Tools with SAS* **221**
*Event Viewer Application Log* **222**
  *Accessing the Application Log Using Windows NT* **222**
  *Accessing the Application Log Using Windows 2000 and Windows Server 2003* **222**
  *Accessing the Application Log Using Windows XP* **222**
  *Viewing a SAS Event* **223**
  *Sending Messages to the Application Log Using a User-Written Function* **223**
    *Specifying the Function's First Parameter* **223**
    *Examples of Using the User-Written Function to Write to the Event Log* **223**
  *Sending Messages to the Application Log Using LOGEVENT.EXE* **224**
*Performance Tools* **226**
  *Why Use a Performance Monitor?* **226**
  *Starting the Windows Performance Monitors* **226**
  *Performance Counters and Objects* **227**
  *SAS Counters in the Performance and System Monitors* **227**
  *Selecting SAS Counters to Monitor* **227**
  *Examples of Monitoring the DATA Step, PROC SORT, and PROC SQL* **228**
    *Configuring the Performance Monitors* **228**
  *Examining the Performance between the DATA and PROC SORT Steps* **228**
  *Examining a PROC SQL Query* **230**
*Starting SAS as a Windows Service* **231**
  *Overview of Starting SAS as a Windows Service* **231**
  *Starting the SAS Service Configuration Utility* **232**
  *Creating an Initialization File* **232**
    *Overview of the Initialization File* **232**
    *Creating an Initialization File Using the SAS Service Configuration Utility* **233**
    *Creating the Initialization (INI) File Using an ASCII Editor* **235**
  *Installing a SAS Service* **238**
  *Starting a SAS Service* **239**
  *Removing a SAS Service* **240**

## Introduction to Using Windows System Tools with SAS

Advanced users and system administrators can start SAS by using Windows services and monitor SAS by using Windows event logging and performance tools.

SAS supports logging of error messages to the Windows Event Viewer's Application Log. Abnormal termination of SAS tasks (such as an access violation) can be viewed in the Application Log in addition to the SAS Log. Also, informational messages from SAS/CONNECT software may be viewed in the Application Log. For more information, see "Event Viewer Application Log" on page 222.

Using either the Windows Performance Monitor under Windows NT or the Windows System Monitor under Windows 2000, Windows XP, and Window Server 2003, you can monitor your SAS sessions to obtain the information that you need to diagnose problems and tune your session. For more information, see "Performance Tools" on page 226.

You can start SAS as a Windows service, which enables you to start SAS automatically and to specify recovery procedures if SAS fails. For more information, see "Starting SAS as a Windows Service" on page 231.

## Event Viewer Application Log

### Accessing the Application Log Using Windows NT

To open the Application Log:

**1** Open the Event Viewer by selecting

Start ▶ Programs ▶ Administrative Tools ▶ Event Viewer

**2** Select

Log ▶ Application

An alternate method of starting the Event Viewer is to type **`eventvwr`** in the Run dialog box and click **`OK`**.

### Accessing the Application Log Using Windows 2000 and Windows Server 2003

Using Windows 2000, you access the Application Log from the Event Viewer tree view:

**1** Select

Start ▶ Settings ▶ Control Panel

**2** Double-click **`Administrative Tools`**.

**3** Double-click **`Event Viewer`**.

**4** In the Tree view, select **`Application Log`**.

An alternate method of starting the Event Viewer is to type **`eventvwr`** in the Run dialog box and click **`OK`**.

### Accessing the Application Log Using Windows XP

Using Windows XP, you access the Application Log from the Event Viewer tree view:

**1** Select

Start ▶ Control Panel

**2** Double-click **`Administrative Tools`**.

**3** Double-click **`Event Viewer`**.

**4** In the Tree view, select **`Application Log`**.

An alternate method of starting the Event Viewer is to type **`eventvwr`** in the Run dialog box and click **`OK`**.

## Viewing a SAS Event

If a SAS task ends abnormally, information regarding the task is placed in the Application Log. The **`Source`** column shows "SAS" as the event source. Messages from SAS/CONNECT display "SAS Job Spawner" as the event source. Double-clicking a SAS event opens the Event Properties window (the Event Detail window under Windows NT) that contains information about the event.

## Sending Messages to the Application Log Using a User-Written Function

### Specifying the Function's First Parameter

SAS events can be sent to the Application Log by using a user-written function in either SAS code or SAS Component Language (SCL). Input to the function is a specific text string. This text string corresponds to the type of event and to the text string that will appear in the Event Viewer:

```
yourfunction("type_of_event", "text_string");
```

The following table lists the types of events that are available for the first parameter.

**Table 9.1** Types of SAS Events

| **Type of Event** | **First Parameter Value** |
|---|---|
| Error | "ERROR" |
| Warning | "WARNING" |
| Information | "INFORMATION" |
| Success Audit | "SUCCESSAUDIT" |
| Failure Audit | "FAILUREAUDIT" |

Although the first parameter values that are displayed in the table are shown in uppercase, mixed case is also allowed. The second parameter of the function is a string that will appear in the Windows Event Viewer.

### Examples of Using the User-Written Function to Write to the Event Log

In the following example, the existence of a semaphore file is checked before SAS performs lengthy processing:

```
%macro pdata(file);
    %let cmdstr = "dir &file";
    options noxwait;
    data _null_;
       call system(&cmdstr);
    run;
    %put &sysrc = sysrc;
    %put &file;
    %if &sysrc=0 %then %do;
       filename indata "&file";
       /* Your data step code for this file.  */
       DATA a;
          infile indata length=linelen;
          length line $ 200;
          input @1 line $ varying200. linelen;
       PROC print;
       run;
    %end;
    %else %do;
       /*  Log an Event of type Error.  */
       %let cmdstr = %str("The file &file did not exist
                              so no data step ran.");
       %put &cmdstr;
       DATA _null_;
          x=ntlog("INFORMATION",&cmdstr);
       run;
    %end;
 %mend;

 %pdata(c:\config.syss)
```

The following is SCL code to write to the Application Log:

```
/* Build a frame and add a pushbutton.  Change the Attribute
Name ''name'' to ''object1''.  In the Source window, add the
following code. */
object1:

x=ntlog("INFORMATION", "This is an INFORMATION event.");
x=ntlog("WARNING", "This is a WARNING event.");
x=ntlog("ERROR", "This is an ERROR event.");
x=ntlog("SUCCESSAUDIT", "This is a SUCCESSAUDIT event.");
x=ntlog("FAILUREAUDIT", "This is a FAILUREAUDIT event.");

return;
```

## Sending Messages to the Application Log Using LOGEVENT.EXE

Using the Windows LOGEVENT.EXE utility that is provided by the Windows Resource Kit, you can send your own information messages to the Application Log from within SAS code.

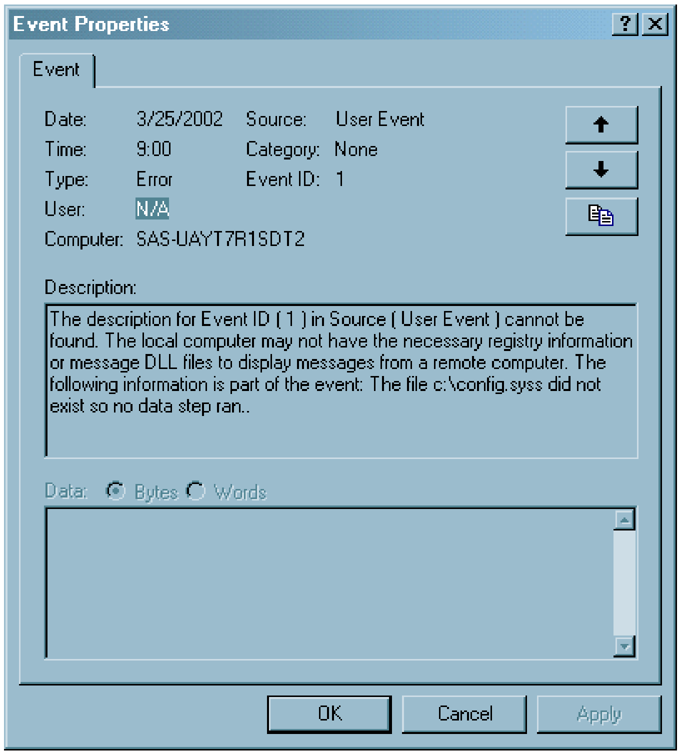
In the following example, the existence of a semaphore file is checked before SAS performs some lengthy processing.

```
%macro pdata(file);
   %local cmdstr;
   %let cmdstr = "dir &file";
   options noxwait;
   DATA _null_;
      call system(&cmdstr);
   run;
   %if &sysrc=0 %then %do;
      filename indata "&file";
      /* Your data step code for this file.  */
      DATA a;
         infile indata length=linelen;
         length line $ 200;
         input @1 line $ varying200. linelen;
      PROC print;
      run;
   %end;
   %else %do;
      /*  Log an Event of type Error.  */
      %let cmdstr = %bquote(c:\support\sasset2\logevent.exe -s E
      "The file &file did not exist so no data step ran.");
      DATA _null_;
         %sysexec &cmdstr;
      run;
   %end;
%mend;

%pdata(c:\config.syss)
```

When you double-click the event in the Application Log, the Event Properties window (the Event Details window under Windows NT) will display the message in the **Description** box.

**Display 9.1** Displaying a User Message in The Event Detail Dialog Box



Date: 3/25/2002 Source: User Event
Time: 9:00 Category: None
Type: Error Event ID: 1
User: N/A
Computer: SAS-UAYT7R1SDT2

Description:

The description for Event ID ( 1 ) in Source ( User Event ) cannot be found. The local computer may not have the necessary registry information or message DLL files to display messages from a remote computer. The following information is part of the event: The file c:\config.syss did not exist so no data step ran..

For information about LOGEVENT.EXE, see the documentation for the Windows Resource Kit.

## Performance Tools

### Why Use a Performance Monitor?

The Windows performance monitors are useful for tuning and diagnosing problems in your application or computer system. These include Performance Monitor under Windows NT and System Monitor under Windows 2000, Windows XP, and Windows Server 2003. By correlating the information from SAS counters with other operating environment counters, you can more easily troubleshoot performance problems.

For example, suppose that your SAS job appears not to be running. Perhaps the job is performing a long and complicated DATA step that generates a very large data set on a network drive. You can be certain that the job is still running by monitoring the Disk WriteFile Bytes Written/Sec and Disk WriteFile Bytes Written Total counters.

### Starting the Windows Performance Monitors

When you type **perfmon** in the Run dialog box, you open the Performance Monitor when you use Windows NT, and the Performance window when you use Windows 2000 and Windows XP.

You can also access the Performance Monitor and the Performance window from the Administrative Tools folder.

## Performance Counters and Objects

A counter is a piece of information that the system monitors. Performance objects represent individual processes, sections of shared memory, and physical devices, such as Memory and LogicalDisk. Counters are grouped by objects. For example, the Memory object contains counters such as Available Bytes, Committed Bytes, and Page Faults/sec. The Processor object has counters such as %Processor Time and %User Time.

By observing various system counters and application-defined counters, you can determine performance problems. You can search for problems in your system and isolate them to areas such as hardware, system software, or your application. For more information about the Performance Monitor, see the Windows NT Resource Kit. For more information about the System Monitor, see the Windows 2000 Resource Kit and the Windows XP Resource Kit.

## SAS Counters in the Performance and System Monitors

SAS includes the following application-defined counters in the SAS object:

Virtual Alloc'ed Memory
: specifies the amount of committed virtual memory that SAS allocates through the VirtualAlloc() API.

Disk ReadFile Bytes Read Total
: specifies the total number of bytes that SAS reads from disk files through the ReadFile() API.

Disk ReadFile Bytes Read/Sec
: specifies the number of bytes that SAS reads per second from disk files through the ReadFile() API.

Disk WriteFile Bytes Written Total
: specifies the total number of bytes that SAS writes to disk files through the WriteFile() API.

Disk WriteFile Bytes Written/Sec
: specifies the number of bytes that SAS writes per second to disk files through the WriteFile() API.

Disk SetFilePointer/Sec
: specifies the number of times per second that SAS successfully calls the SetFilePointer() API on disk files.

Memlib/Memcache Current Usage K
: specifies in bytes the amount of Extended Server Memory that is currently in use.

Memlib/Memcache Peak Usage K
: specifies in bytes the maximum amount of Extended Server Memory that is used in the current SAS session.

## Selecting SAS Counters to Monitor

Use the following procedures to monitor SAS counters in your respective operating environment:

**Table 9.2** Procedure for Selecting SAS Counters

| Using Windows NT | Using Windows 2000 and Windows XP |
|---|---|
| 1 Start SAS | 1 Start SAS |
| 2 Open the Performance Monitor | 2 Open the Performance window. |
| 3 Click the **Add Counter** (+) button. | 3 Click the **Add** (+) button. |
| 4 From the **Objects** list, select **SAS**. | 4 From the **Performance object** list, select **Process**. |
| 5 For each counter that you want to monitor, select the counter from the list and click **Add**. | 5 From the Instances list, select **SAS**. |
| 6 Click **Done** when you have completed selecting counters. | 6 For each counter that you want to monitor, select the counter from the list and click **Add**. |
|  | 7 Click **Close** when you have completed selecting counters. |

The performance monitor immediately collects and displays information about the counters that you selected.

Multiple SAS counters may be monitored. You may see multiple instances monitored, where each instance is a separate SAS process. SAS instances are listed in the form SAS PID *number*. The PID number is the process identifier of the SAS session. You can see a list of all processes by using the Task Manager.

## Examples of Monitoring the DATA Step, PROC SORT, and PROC SQL

### Configuring the Performance Monitors

Configure the Performance Monitor and the System Monitor for all examples as follows:

1. Invoke SAS and the Performance Monitor or the System Monitor.
2. Open the Add Chart window or the Add Counters window and select the SAS object.
3. Add these SAS counters:
   - Disk ReadFile Bytes Read/Sec
   - Disk WriteFile Bytes Written/Sec
   - Disk SetFilePointer/Sec
4. Select the Process object.
5. Add these Process counters:
   - %Processor Time
   - %User Time
   - %Privileged Time
6. Click **Done** or **Close**.

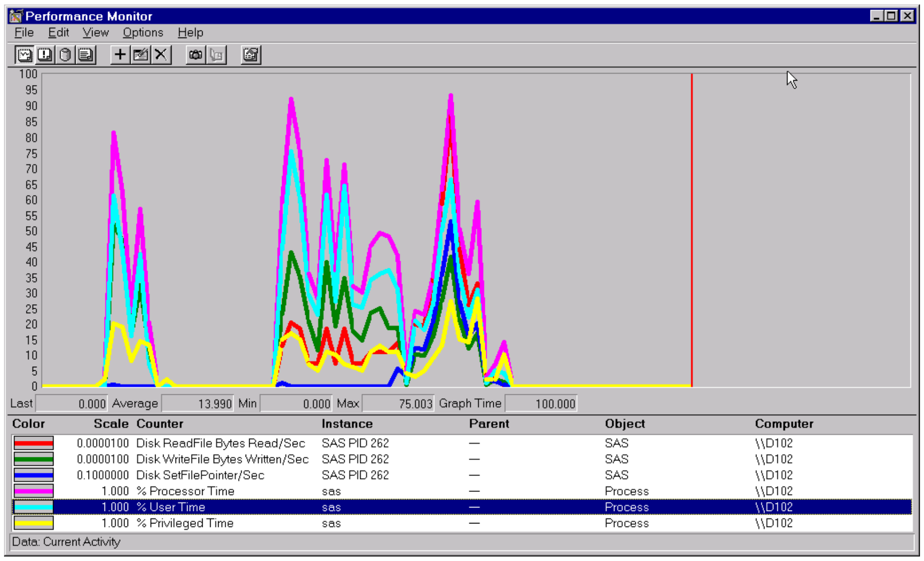
## Examining the Performance between the DATA and PROC SORT Steps

To see the difference in performance between the DATA step and the PROC step, submit this code:

```
options fullstimer;
  /* Create a test data set with some random data. */
DATA a (drop=s);
   do i = 1 to 500000;
      x = ranuni(i);
      y = x*2;
      z = exp(x*y);
      output;
   end;
   /* The sleep helps to delineate the subsequent  */
   /* sort in the Performance Monitor graph        */
   s = sleep(15);
run;
PROC sort data = a noduplicates;
   by z y x i;
run;
```

After you submit this code, the Performance Monitor or System Monitor will generate results similar to those in Display 9.2 on page 229. You might have to adjust the scale factor of the different counters.

**Display 9.2** Performance of the DATA Step and the PROC SORT Step



Last 0.000 Average 13.990 Min 0.000 Max 75.003 Graph Time 100.000

| Color | Scale | Counter | Instance | Parent | Object | Computer |
|---|---|---|---|---|---|---|
| 🟥 | 0.0000100 | Disk ReadFile Bytes Read/Sec | SAS PID 262 | — | SAS | \\D102 |
| 🟩 | 0.0000100 | Disk WriteFile Bytes Written/Sec | SAS PID 262 | — | SAS | \\D102 |
| 🟦 | 0.1000000 | Disk SetFilePointer/Sec | SAS PID 262 | — | SAS | \\D102 |
| 🟪 | 1.000 | % Processor Time | sas | — | Process | \\D102 |
| 🟦 | 1.000 | % User Time | sas | — | Process | \\D102 |
| 🟨 | 1.000 | % Privileged Time | sas | — | Process | \\D102 |

Data: Current Activity

The DATA step in the display shows that there is very little activity from Disk ReadFile Bytes Read/Sec or Disk SetFilePointer/Sec. Notice that in the subsequent PROC SORT output there is much more activity from these two counters. This indicates that the data set is being read (Disk Readfile Bytes Read/Sec) in order to be sorted, and that a certain amount of random I/O is performed by the sort (Disk SetFilePointer/Sec).

The pause in the activity is caused by the SLEEP function that follows the DATA step. The Disk WriteFile Bytes Written/Sec counter is active in both the DATA step and in the PROC SORT step.

Finally, you can correlate the counters from the Process object with the user and system CPU times in your SAS log.

## Examining a PROC SQL Query

To examine the performance of a PROC SQL query with an index, submit this following code:

1 Submit the code in Step 1 and Step 2. Step 2 creates an index.

```
/* Step 1                                         */
   /* Create a test data set with some random data. */
   /* Do this twice - once with Step 2 and once     */
   /* without Step 2.                               */

libname sample 'c:\';
DATA sample.a;
   do i = 1 to 500000;
      x = ranuni(i);
      y = x*ranuni(i);
      z = exp(y);
      output;
   end;
run;

   /* Step 2                                        */
   /* Create a simple index on variable x.          */
   /* Submit this step once.                        */

PROC DATASETS library = sample;
   modify a;
   index create x;
   quit;
```

2 Clear the graph by selecting **`Clear Display`** from the **`Edit`** menu or the **`Clear Display`** toolbar button.

3 Submit the code in Step 3 to see a graph such as Display 9.3 on page 231.

```
      /* Step 3                                        */
   /* Perform a query on the data.  Do this twice - */
   /* once with an index and once without an index  */
   /* The query should select about 50% of the      */
   /* observations in the data set.                 */

PROC SQL;
   create table sample.yz as
   select y,z
      from sample.a
      where x > 0.5;
   quit;
```

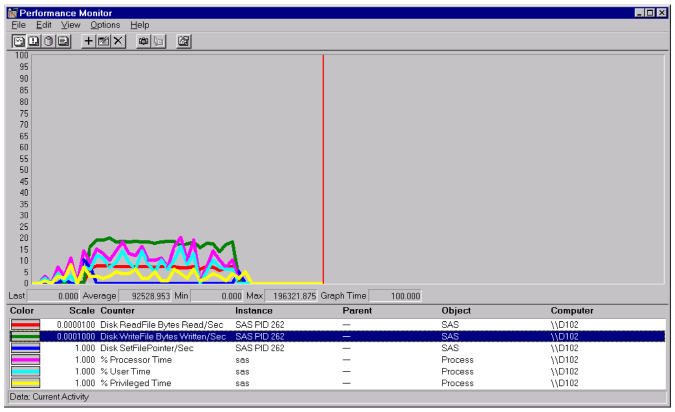
To perform a PROC SQL query without an index:

1 Resubmit Step 1.

2 Clear the graph.

3 Resubmit Step 3 to see a graph such as Display 9.4 on page 231.

**Display 9.3** Performance of PROC SQL Query with an Index

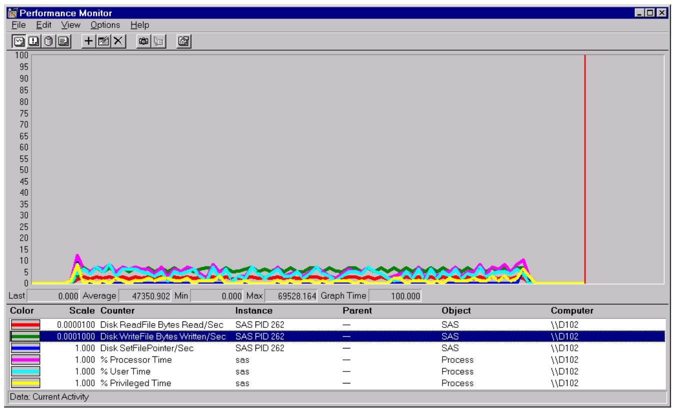


Last 0.000 Average 92528.953 Min 0.000 Max 196321.875 Graph Time 100.000

| Color | Scale | Counter | Instance | Parent | Object | Computer |
|---|---|---|---|---|---|---|
| 🟥 | 0.0000100 | Disk ReadFile Bytes Read/Sec | SAS PID 262 | — | SAS | \\D102 |
| 🟩 | 0.0001000 | Disk WriteFile Bytes Written/Sec | SAS PID 262 | — | SAS | \\D102 |
| 🟦 | 1.000 | Disk SetFilePointer/Sec | SAS PID 262 | — | SAS | \\D102 |
| 🟪 | 1.000 | % Processor Time | sas | — | Process | \\D102 |
| 🟦 | 1.000 | % User Time | sas | — | Process | \\D102 |
| 🟨 | 1.000 | % Privileged Time | sas | — | Process | \\D102 |

Data: Current Activity

**Display 9.4** Performance of PROC SQL Query without an Index



Last 0.000 Average 47350.902 Min 0.000 Max 69528.164 Graph Time 100.000

| Color | Scale | Counter | Instance | Parent | Object | Computer |
|---|---|---|---|---|---|---|
| 🟥 | 0.0000100 | Disk ReadFile Bytes Read/Sec | SAS PID 262 | — | SAS | \\D102 |
| 🟩 | 0.0001000 | Disk WriteFile Bytes Written/Sec | SAS PID 262 | — | SAS | \\D102 |
| 🟦 | 1.000 | Disk SetFilePointer/Sec | SAS PID 262 | — | SAS | \\D102 |
| 🟪 | 1.000 | % Processor Time | sas | — | Process | \\D102 |
| 🟦 | 1.000 | % User Time | sas | — | Process | \\D102 |
| 🟨 | 1.000 | % Privileged Time | sas | — | Process | \\D102 |

Data: Current Activity

In Display 9.4 on page 231, the counters averaged under 10% on the scale, whereas in Display 9.3 on page 231, several of the counters averaged more than 10%, and the Disk WriteFile Bytes Written/Sec counter rose more than 25%. A higher value for these counters implies good overall throughput for the operation.

Note that to make a valid comparison like this with the Performance Monitor graph or with the System Monitor graph, you must ensure that the counters are using the same scale. You can confirm this by observing the absolute values. The Average value for Disk WriteFile Bytes Written/Sec in Display 9.3 on page 231 was 92528.953. Contrast this with the same counter in Display 9.4 on page 231, in which the Average value was 47350.902. For this operation, bytes were written almost twice as fast when the data set was indexed.

## Starting SAS as a Windows Service

### Overview of Starting SAS as a Windows Service

Starting SAS as a Windows service enables you to start SAS automatically or manually, define recovery procedures if SAS fails, and enable users to log on and log off a PC without interrupting SAS. When SAS is defined to start manually, you can start

SAS either from an application by using the **net start** command or by using the Windows Services dialog box.

The general process for configuring SAS as a Windows service is as follows:

1 Create an initialization (.INI) file.
2 Install the .INI file to register SAS as a Windows service.
3 If SAS is configured to start automatically, restart your machine. If SAS is configured to start manually, you can start SAS either from an application or from the Services dialog box in Windows NT or the Services window in all other Windows operating environments. For more information, see “Starting a SAS Service” on page 239.

If you have multiple SAS configurations, you can create an initialization file for each configuration.

SAS provides the SAS Service Configuration Utility (SSCU) to configure the service and install the .INI file.

## Starting the SAS Service Configuration Utility

To start the SSCU, do one of the following:

□ Select

  Start ▶ Programs ▶ *your- SAS- System- folder* ▶ SAS 9.1 Utilities ▶ SAS Service Configuration Utility

□ From the SSCU directory, type **sasservicemngr.exe**. The default SSCU directory is c:\Program Files\SAS\SAS 9.1\core\sscu.

## Creating an Initialization File

### Overview of the Initialization File

Before you can start SAS as a Windows Service, you need to configure and install an initialization (.INI) file. The. INI file is a text file that

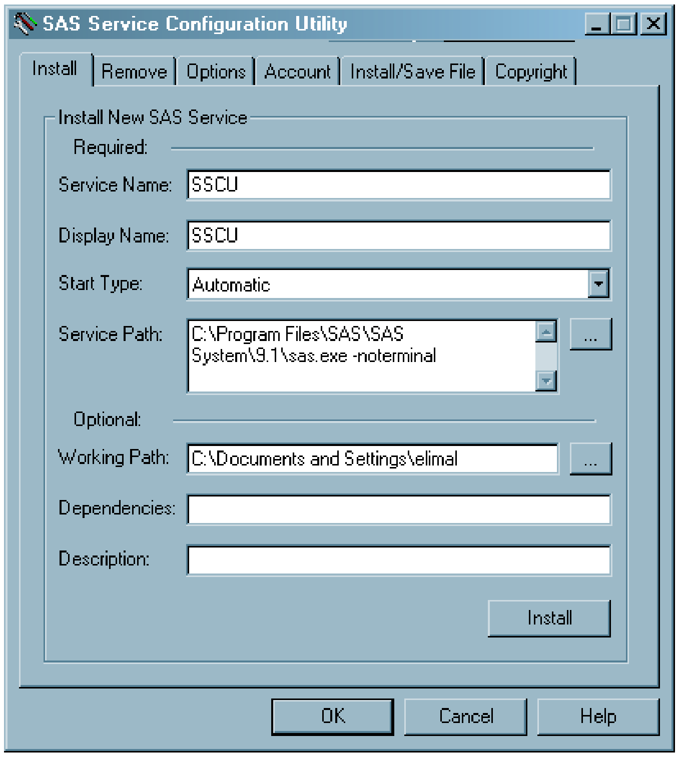
□ names the SAS service
□ specifies if the service is to start automatically or manually
□ defines paths to SAS and SAS working paths
□ specifies the level of access that an application has to the SAS service
□ names other Windows services that must be started before this service can be started
□ defines the actions that Windows is to complete if SAS fails to start as a service
□ specifies whether the SAS service is a system account or a local account
□ specify whether the user can interact with the SAS desktop.

You create the initialization file either by using the SSCU graphical user interface (GUI) or by using a text editor. If you use the SSCU GUI, you specify only the required values and the SSCU creates the .INI file for you. If use a text editor to create the .INI file, you must specify the SAS service settings and their values. Table 9.4 on page 236 explains the settings that you specify to create a SAS service .INI file with a text editor.

## Creating an Initialization File Using the SAS Service Configuration Utility

Use the SAS Service Configuration Utility (SSCU) that is shown in the following display to create the .INI file. After configuring the settings, click the **`Install/Save File`** tabbed page to save and install the .INI file.

**Display 9.5** SAS Service Configuration Utility



Service Name: SSCU
Display Name: SSCU
Start Type: Automatic
Service Path: C:\Program Files\SAS\SAS System\9.1\sas.exe -noterminal
Working Path: C:\Documents and Settings\elimal
Dependencies:
Description:

To configure the INI file, select the following tabbed pages and modify the appropriate settings:

**`Install`** tabbed page

**`Service Name`**
: is the service name that is registered to Windows when the service is installed. The service name is also the name that is used when a net start or a net stop command is issued. This is a required field. The default is **`SASService`**.

**`Display Name`**
: is the name of the service that is displayed to user-interface applications. This is a required field. The default is **`A SAS Service`**.

**`Start Type`**
: specifies whether the SAS service is started manually, automatically, or is disabled. This is a required field. The default is **`Manual`**. **`Manual`** specifies that the service can be started by another process. **`Automatic`** specifies that the service is started automatically during system startup. **`Disabled`** specifies that the service cannot be started.

**`Service Path`**
: contains both the directory path in which SAS is installed and also the SAS command that is used to start the service. This is a required field. For a new

installation, the default path is the SAS installation path, followed by the command **`sas.exe -noterminal`**. If the SSCU has been installed previously, the default path is c:\Program Files\SAS Institute\SAS\V9\ followed by **`sas.exe -noterminal`**. The NOTERMINAL system option is required. To start a SAS program as a service, add the SYSIN system option followed by the program pathname and filename to the Service Path. To select a service path, click **`...`** (ellipse button).

**`Working Path`**
is the working path that is used by applications that use the SAS Service to create directories, store files, and log information. This field is optional. The default is the user's profile directory, c:\WINNT\Profiles\*username* in Windows NT or c:\Documents and Settings\*username* in Windows 2000, Windows XP, or Windows Server 2003. To select a working path, click **`...`** (ellipse button).

**`Dependencies`**
specifies one or more Windows services that must be started before this service is started. If a dependent service is installed and enabled, the service is started before this service is started. If a service is installed but disabled, this service will not be started.

To specify dependencies, type one or more service names separated by the pipe ( | )character. For example, NetDDE|NetDDEdsdm.

**`Description`**
Type a description of the service. The description appears in the Windows Services window.

**`Remove`** tabbed page

**`Remove Existing Service`**
specifies the name of the installed SAS service that you want to remove.

**`Options`** tabbed page

**`Error Control`**
determines the error severity if the SAS Service fails to start. Select one of the following error controls:

**Table 9.3**

| Error Control | Description |
|---|---|
| Ignore | The error is logged. Startup operations continue. |
| Normal | The error is logged and a message is displayed. Startup operations continue. |
| Severe | The error is logged and startup operations continue by using the last successfully installed INI file. |
| Critical | An attempt to log the error is made. If the startup operation is using the last known successful INI file, startup operations fail. If the startup operation is not using the last known successful INI file, it will attempt to restart the service by using the last successful INI file. |

**`Access`**
is the level of access that an application has to the SAS Service. When you select an access level, such as Read, Write, or Execute, certain access type

settings are set to **TRUE** in the INI file. To further configure all access types settings, click **Custom**. For a description of access type settings, see Table 9.4 on page 236. The access levels are

**Read**
enables an application to set the **Interrogate**, **Query Configuration** and **Query Status** access type settings. Selecting this access level sets the AccessInterrogate=, AccessQryCfg=, and AccessQryStatus= settings in the .INI file to **TRUE**.

**Write**
enables an application to set the **Change Configuration** access type. Selecting this access level sets the AccessChgCfg= setting in the .INI file to **TRUE**.

**Execute**
enables an application to set the **Interrogate**, **Pause/Continue**, **Start Service**, **Stop Service**, and **Define Control** access types. Selecting this access level sets the AccessInterrogate=, AccessPauseCont=, AccessStart=, AccessStop, and AccessUserDefCtrl= settings in the .INI file to **TRUE**.

**Account** tabbed page

**System Account**
specifies that the service is shared for all users that log on to the this machine. To enable the service to interact with the user from the desktop, select **Allow this Service to interact with the Desktop**. When you select **System Account**, the ServiceStartName= setting in the INI file is set to **LocalSystem**.

**This Account**
specifies that the service is for a specific user only. When you select **This Account**, type the account name in the box. Then type the password in the **Password** and **Confirm Password** boxes.

**Install/Save File** tabbed page

**Install from file**
Click the **Install from file** button to specify an initialization (INI) file to install.

**Save settings to file**
Click the **Save settings to file** button to save the settings that you have specified in the SSCU Configuration Utility GUI to a file.

**Show file contents**
Select the **Show file contents** box if you want to display the initialization file that you want to install or save in the **File Content**s box.

**Copyright** tabbed page
display the copyright information for the SAS Services Configuration utility.

## Creating the Initialization (INI) File Using an ASCII Editor

To create a SAS Service INI file by using any text editor, create a new file in the editor and assign a valid value to each of the settings in the following table. Type only one setting per line:

**Table 9.4** SAS Service INI File Settings and Default Values

| Setting Name | Required | Explanation | Valid Values | Defaults | Related SSCU Field |
|---|---|---|---|---|---|
| Service Name= | Yes | The SAS Service name registered to Windows. | Can contain up to 32 characters (/ and \ are not valid). The name is not case sensitive and it must be contained in quotation marks. | "SASService" | Service Name |
| Display Name= | Yes | The name of the service that is displayed to user-interface applications. | Can contain up to 256 characters, is not case sensitive, and must be contained in quotation marks. | "A SAS Service" | Display Name |
| Binary Path Name= | Yes | Contains the directory path in which the SAS Service INI file is installed, followed by the SAS command to start the service. | The path name must be contained in both brackets and quotation marks. | ["*SAS installation path*\sas.exe -noterminal"] | Service Path |
| Start Type= | Yes | Specifies whether the SAS Service is to start manually or automatically. | SERVICE_AUTO_ START<br>SERVICE_ DEMAND_START<br>SERVICE_DISABLED | SERVICE_ DEMAND_ START | Start Type |
| Dependencies= | No | Specifies Windows services that must be started before this service is started. | One or more Windows service names, separated by the pipe (\|) character. Enclose dependences in quotation marks. | none | Dependencies |
| Description | No | A description of the service | The description can contain alphanumeric characters and must be enclosed in quotation marks. | none | Description |
| WorkDir= | No | The directory used by applications to store files created and used by the SAS Service. | The path to the working directory must be contained in quotation marks. | "c:\WINNT \Profiles\ *username*" or "c:\Documents and Settings\ *username*" | Working Path |
| ErrorControl= | Yes | Determines the error severity if the SAS Service fails to start. | SERVICE_ ERROR_ IGNORE<br>SERVICE_ ERROR_ NORMAL<br>SERVICE_ ERROR_ SEVERE<br>SERVICE_ ERROR_ CRITICAL | SERVICE_ ERROR_ NORMAL | Error Control |

| Setting Name | Required | Explanation | Valid Values | Defaults | Related SSCU Field |
|---|---|---|---|---|---|
| Interactive= | Yes | Specifies whether the service allows a user to interact with the SAS desktop. | TRUE<br>FALSE | FALSE | Interactive Process |
| AccessChgCfg= | Yes | Modifies the SAS Service configuration. | TRUE<br>FALSE | TRUE | Change Configuration |
| Access Interrogate= | Yes | Requests that the SAS Service immediately update its current status. | TRUE<br>FALSE | TRUE | Interrogate |
| Access PauseCont= | Yes | Pauses and resumes the SAS Service. | TRUE<br>FALSE | TRUE | Pause/ Continue |
| AccessQryCfg= | Yes | Makes queries about the SAS Service configuration. | TRUE<br>FALSE | TRUE | Query Configuration |
| AccessQry Status= | Yes | Queries Windows NT about the status of the SAS Service. | TRUE<br>FALSE | TRUE | Query Status |
| AccessStart= | Yes | Starts the SAS Service. | TRUE<br>FALSE | TRUE | Start Service |
| AccessStop | Yes | Stops the SAS Service. | TRUE<br>FALSE | TRUE | Stop Service |
| AccessUser DefCtrl= | Yes | Specifies a user-defined control code. | TRUE<br>FALSE | TRUE | Define Control |
| ServiceStart Name= | No | The Windows user account with proper user rights to run the SAS Service. | LocalSystem or Windows account name | LocalSystem | This Account |
| Password= | No | The Windows account password. | an encrypted password | none | Password |

When you create an .INI file by using an ASCII editor and you want to specify ServiceStartName for a specific user, the Windows account name must be of the format *domainname\username* and you must include an encrypted password in the PASSWORD setting name. You can use the PWENCODE procedure to create an encrypted password. For example, the following PWENCODE procedure specifies **mypw** as the input password:

```
proc pwencode in='mypw';
run;
```

The SAS log displays the encrypted password **`[sas001}bXlwdw==`**. You then specify **`[sas001}bXlwdw==`** as the value for the Password= setting in your ASCII file. An encrypted password is necessary only if you specify Password= in an ASCII file. In comparison, when you create an .INI file by using the SSCU, you specify a text password. The SSCU encrypts the password for you.

For more information about the PWENCODE procedure, see *Base SAS Procedures Guide*.

## Installing a SAS Service

When you have created the initialization file, you use the initialization file to install SAS as a service. A SAS service can be installed either from the SSCU, from the command prompt, or from within an application.

To install a SAS Service by using the SSCU:

1 Select the Install/Save File tab.
2 Select **`Install from file`**.
3 From the Open dialog box, select an INI file.
4 Click **`Open`**.

To install a SAS Service from the command prompt, ensure that both the SAS Service Configuration Utility directory and the directory that contains the INI file are accessible from your system path. From the command prompt type **`sscu.exe`** ***`path/filename`*****`.ini`**. When you install a SAS Service from the command prompt, user messages are disabled.

When a SAS Service is installed from an application, the command to install the service is **`sscu.exe`** ***`path/filename`*****`.ini`**. The following table lists the return codes that can be passed back to the calling application:

**Table 9.5** Return Codes from Installing or Running a SAS Service

| Numeric Code | Error Code | Description |
|---|---|---|
| 0 | SUCCESS | The service has successfully been installed. |
| 5 | ERROR_ACCESS_DENIED | Access to the Service Control Manager is denied. |
| 6 | ERROR_INVALID_HANDLE | Error loading the Service Control Manager. |
| 25 | ERROR_NOT_FULL_PATH_CREATED | The full path could not be created. |
| 26 | USER_CANCELLED_INSTALL | The user cancelled the installation. |
| 30 | SUCCESS_NO_REG_DIR | The service was installed but failed to register the working directory. |
| 35 | ERROR_BINPATH_NOTFOUND | The service file was not found, no installation |
| 40 | ERROR_USER_CANCEL_NOSRVC | The user cancelled the installation because an INI file was not found. |

| Numeric Code | Error Code | Description |
|---|---|---|
| 50 | ERROR_MISSING_FILE_ARGUMENT | A required argument in the INI file is missing. |
| 51 | ERROR_INVLAID_FILE_ARGUMENT | An INI file argument contains an incorrect value. |
| 55 | ERROR_OPENFILE | The INI file could not be opened. |
| 60 | ERROR_ITEMTOOLARGE | A string value exceeds the maximum character limit. |
| 65 | ERROR_PASSED_DECRYPT_FAILED | The password could not be decrypted. |
| 87 | ERROR_INVALID_PARAMETER | A service parameter is incorrect. |
| 123 | ERROR_INVALID_NAME | The specified service name is not valid. |
| 1057 | ERROR_INVALID_SERVICE_ACCOUNT | The account name is incorrect or does not exist. |
| 1060 | ERROR_SERVICE_DOES_NOT_EXIST | The specified service does not exist as an installed service. |
| 1065 | ERROR_DATABASE_DOES_NOT_EXIST | The specified database does not exist. |
| 1072 | ERROR_SERVICE_MARKED_FOR_DELETE | The specified service has been marked for deletion. |
| 1073 | ERROR_SERVICE_EXISTS | A duplicate service name exists on the network. |
| 1078 | ERROR_DUPLICATE_SERVICE_NAME | A duplicate display name exists on the network. |

## Starting a SAS Service

A SAS Service can be started automatically or manually. If the SAS Service is configured to start automatically, the service starts when the system starts. If the SAS Service is configured to start manually, the service can be started either from an application by using the **`net start`** command or by using the Services dialog box.

To start a SAS Service using the Services dialog box:

- □ Under Windows NT, select

  Start ▶ Settings ▶ Control Panel ▶ Services

  Under all other Windows operating environments, select

  Start ▶ Settings ▶ Control Panel ▶ Administrative Tools ▶ Services

- □ From the **`Services`** list box, select the SAS service.
- □ Click **`Start`**.

## Removing a SAS Service

A SAS Service can be removed as a Windows service from the SSCU or from the command prompt.

To remove a SAS Service by using the SSCU:

1 Open the SSCU and click the **`Remove`** tab.

2 Select the SAS Service from the **`Remove Existing Services`** box.

3 Click **`Remove`**.

To remove a SAS Service from the command prompt, type **`sasservicemngr.exe /remove <`*`servicename`*`>`**.

CHAPTER

10

# Using OLE in SAS/AF Software

*About OLE* **242**
*SAS/AF Catalog Compatibility* **242**
*Inserting an OLE Object in a FRAME Entry* **242**
- *Introduction to Inserting an OLE Object in a FRAME Entry* **242**
- *Inserting an OLE Object* **243**
- *Pasting an OLE Object from the Clipboard* **243**
- *Reading an OLE Object from an HSERVICE Entry* **244**
- *Inserting an OLE Object by Dragging It* **244**
  - *Dragging OLE Objects During Run Time* **245**
  - *Changing the Drag Action* **245**

*Editing an OLE Object within a FRAME Entry* **245**
*Invoking OLE Verbs* **247**
*Using Linked OLE Objects* **247**
- *Updating a Linked Object with the Links Dialog Box* **248**
- *Updating a Linked Object Programmatically* **248**

*Converting OLE Objects* **249**
*Automating OLE Objects and Applications* **250**
- *Accessing Array Values Returned by the OLE Automation Server* **251**
- *Using Value Properties* **252**
- *Specifying Optional Parameters in OLE Server Methods* **253**
- *Creating an External OLE Automation Instance* **253**
- *Example: Populating a Microsoft Excel Spreadsheet with SAS Data* **255**

*Using OLE Custom Controls (OCXs) in Your SAS/AF Application* **257**
- *Inserting an OLE Control in a FRAME Entry* **257**
- *Registering OLE Controls* **258**
- *Accessing OLE Control Properties* **258**
  - *Accessing the OLE Control Properties Page* **258**
  - *Accessing Properties Using SCL Code* **259**
- *Interacting with the OLE Control Using SCL Methods* **259**
- *Responding to OLE Control Events* **260**
  - *Assigning SCL Code to an OLE Control Event* **260**
  - *Retrieving Argument Values from Events* **261**
  - *Example: Mapping OLE Control Events to SCL Code* **261**
  - *Example: Subclassing an OLE Custom Control* **262**
  - *Adding an Item to a Combo Box List* **262**
  - *Finding an Item in a Combo Box* **263**
  - *Retrieving the Text Value of the Control* **263**

## About OLE

OLE is a means of integrating multiple sources of information from different applications into a unified document. These objects can include text, graphics, charts, sound, video clips, and much more.

OLE 1.0, which the SAS has supported since Release 6.08, allowed you to link and embed OLE objects into SAS/AF FRAME entries and SAS/EIS applications. OLE 2.0, which SAS 9.1 supports, provides many new features that you can use to enhance your SAS/AF frames and SAS/EIS applications.

*Note:* SAS under Windows (and OLE 2.0 in general) still supports all the features from OLE 1.0. △

SAS can function as an object container or *client*. The applications that create (and update) the objects you place in a FRAME entry are known as *servers*. You can also use SAS as a server from within other applications through OLE automation. For more information, see Chapter 11, “Controlling SAS from Another Application Using OLE,” on page 265 .

For more information about OLE in general, see the documentation for the Windows operating environment. For descriptions of the error messages you might receive while using OLE features in SAS/AF software, see “Using OLE” on page 600.

## SAS/AF Catalog Compatibility

SAS/AF catalogs that contain OLE HSERVICE entries can be ported from Release 6.09 for Windows NT and Release 6.10 or later for Windows transparently, just by assigning libnames to those catalogs in your SAS 9.1 session.

*Note:* SAS/AF catalogs created in SAS 9.1 that contain HSERVICE entries can be ported back to Release 6.08 using the V608 option of the CPORT procedure, but the features that are ported are limited to those available in Release 6.08. △

HSERVICE entries are only usable on the platform in which they were created. That is, any OLE features that you include in your SAS/AF applications using SAS under Windows cannot be ported to another operating environment. (For portability purposes, all variations of Microsoft Windows are considered a single platform.)

## Inserting an OLE Object in a FRAME Entry

### Introduction to Inserting an OLE Object in a FRAME Entry

SAS provides three items on the object Selection List to facilitate OLE:

**`OLE - Insert Object`**
inserts an OLE object as a new object of the type associated with a registered server application, as an object created from an existing file, or as an OLE control.

**`OLE - Paste Special`**
pastes an OLE object to the FRAME entry from the Windows clipboard.

**`OLE - Read Object`**
creates an object that references an existing HSERVICE entry in a SAS catalog.

These three items correspond to the three OLE classes in SAS/AF software: INSERT, PASTE, and READOLE.

In addition to using the Selection List to insert objects, you can select and drag objects from other Windows applications and drop them onto an open FRAME entry (in BUILD mode, or during run time if the frame or work area object is registered as a drop site for the SAS_DND_OLEOBJ representation).

## Inserting an OLE Object

To insert an OLE object in a FRAME entry:

1 From the COMPONENTS window, select the **`V6 objects`** item to expand the object tree.

2 Scroll through the list of objects in the Selection List and select and hold down the left mouse button on **`OLE - Insert Object`**.

3 Drag **`OLE - Insert Object`** to a position for the object in the FRAME entry. Release the mouse button to place the object. The Insert Object dialog box appears.

4 Select the type of object that you want to insert. The list of objects that are available to you depends on which OLE-capable applications are registered on your system. Selecting a type of object will insert an object of that type into the FRAME entry.

   Alternatively, you can create an object from a file by clicking on **`Create from File`**. The file you specify must have been created by one of the applications you have available to supply OLE objects. For example, if you have Microsoft Excel installed on your system, you can create an object from an Excel spreadsheet file. You also have the option of making it a linked object (instead of embedded). For more information about linked objects, see “Using Linked OLE Objects” on page 247.

   When you have selected the type of object or filename to insert, click **`OK`**. SAS inserts the object into the FRAME entry.

5 With the BUILD window active, select

   View ► Properties Window

   In the Properties windows, select the object and select **`Object Attributes`**.

   Enter a name for the object entry in the **`Entry`** field. Two-level HSERVICE names are allowed, defaulting to the current catalog. Optionally, you can also change the **`Name`** of the object. Note that the HSERVICE entry is not created until you **`Save`** or **`End`** the FRAME editing session.

   Click **`OK`**.

## Pasting an OLE Object from the Clipboard

To paste an OLE object from the Windows clipboard:

1 From another Windows application, copy or cut to the Windows clipboard the object or data you want to include in your FRAME entry.

2 From the COMPONENTS window, select the **`V6 objects`** item to expand the object tree.

3 Scroll through the list of objects in the Selection List and select and hold down with the left mouse key **`OLE - Paste Special`**. Drag **`OLE - Paste Special`** to the frame. The Paste Special dialog box appears.

4 Select the type of OLE object you would like to insert based on the clipboard contents. This is determined by the application from which you copied the data. (For example, you would typically paste Microsoft Word data as a Microsoft Word object.)

5 If you want the OLE object to link to the data instead of embed the actual data in the FRAME entry, choose **`Paste Link`** on the Paste Special dialog box. For more information about linked objects, see “Using Linked OLE Objects” on page 247.

*Note:* If you paste data from a temporary source (such as a document that you did not save), SAS will be unable to locate the data source when it attempts to link to it later when it no longer exists.You should save your data file before copying it to the Windows clipboard. △

6 After you select the type of object to paste, click **`OK`**. SAS pastes the object into the FRAME entry.

7 Select

View ▶ Properties Window

Select the object from the **`Properties`** box and click on **`Object Attributes`**.

8 Enter a name for the object entry in the **`Entry`** field. Two-level HSERVICE names are allowed, defaulting to the current catalog. Optionally, you can also change the **`Name`** of the object. Note that the HSERVICE entry is not created until you **`Save`** or **`End`** the FRAME editing session.

Click **`OK`**.

## Reading an OLE Object from an HSERVICE Entry

To read an existing OLE object stored as an HSERVICE entry in a SAS catalog:

1 From the COMPONENTS window, select the **`V6 objects`** item to expand the object tree.

2 Scroll through the list of objects in the selection list and select and drag **`OLE - Read Object`** to the BUILD window.

3 With the cursor over the blank object, right mouse click and select **`Object Attributes`**.

4 In the OLE-Read Object Attributes window, enter the name of the HSERVICE entry in the **`Entry`** field. Two-level HSERVICE names are allowed, defaulting to the current catalog. To use the Select window to find the entry, click on the arrow next to the **`Entry`** field.

Click **`OK`**. SAS inserts the object in the FRAME entry, displaying a representation of the object at the position you selected.

*Note:* You cannot change the name of an HSERVICE entry that you read in. If you want to assign a different name to the HSERVICE entry, copy the HSERVICE entry to a new name before you read the object. △

## Inserting an OLE Object by Dragging It

To insert an OLE object into a FRAME entry by dragging and dropping it:

1 Create the object using the server application. For example, if you want to embed a Microsoft Excel chart object into your FRAME entry, use Microsoft Excel to create the object. Or, you can select an OLE object that is embedded in another application.

**2** With both SAS and the server application running, arrange the application windows so that both the server application (with the object) and the SAS BUILD: DISPLAY window (with the FRAME entry) are visible on the screen.

**3** Select the object in the server application. With the mouse button depressed, drag the object from the server application to the position in the FRAME entry where you want to place the object. The cursor changes to a box with an arrow, indicating that the FRAME entry is a valid place to drop the object. Note that you do not need to draw a region in the FRAME to insert the object. You can also use drag modifier keys, as discussed in "Changing the Drag Action" on page 245 to control the drag and drop behavior.

   When you release the mouse button ("dropping" the object), SAS inserts the object into the FRAME, automatically creating a name and an HSERVICE entry for the OLE object. SAS displays a representation of the object at the position you selected.

### Dragging OLE Objects During Run Time

You can allow the dragging and dropping of OLE objects while your SAS/AF application is running. To enable this, you must register the OLE object type with a valid drag and drop representation.

OLE objects must be registered with the SAS_DND_OLEOBJ representation. For more information about registering objects for drag and drop, see the SAS/AF online documentation for information about working with the FRAME application development environment and for information about the Widget class.

### Changing the Drag Action

By default, dragging an OLE object from another application into SAS moves the object (unless the object is of a type that can only be read and not removed). You can override this default action by using a *drag modifier*; that is, a key press that indicates that you want to perform a different drop action:

- To copy an object from the server application, hold down the Ctrl key when you drop the object on the target window. When you press the Ctrl key, the cursor changes to an arrow with a box and a plus (+) sign.
- To create a link to the data in a SAS/AF FRAME entry, hold down the Ctrl and Shift keys when you drop the object on the BUILD window. When you press the Ctrl and Shift keys, the cursor changes to an arrow with a box and a plus (+) sign. (This feature might vary based on the other application.) Remember not to paste a linked object from a temporary source, because SAS cannot locate a data source when it no longer exists.

Alternatively, you can initiate a *nondefault* drag and drop action (if the server application supports it). Use the *right* mouse button to select the object and drag and drop it into the FRAME entry. When you release the mouse button, SAS displays a pop-up menu allowing you to select whether to move, copy, or link to the object. The choices in the pop-up menu might vary among different types of OLE objects.

## Editing an OLE Object within a FRAME Entry

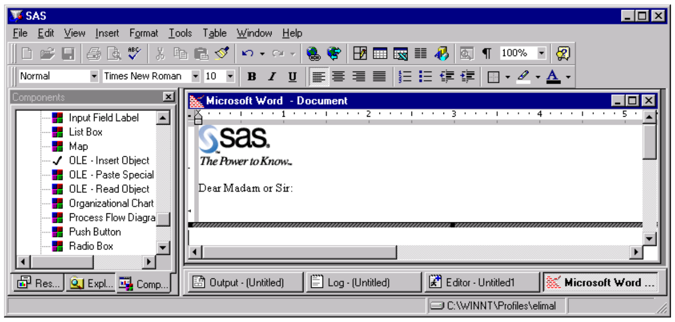
One of the most impressive features of OLE 2.0 is visual editing–the ability to edit an embedded object in-place, without explicitly changing to another application.

To activate visual editing for an OLE object in your FRAME entry at build time, click the right mouse button and select Edit. To activate visual editing at run time, simply double-click on the object. If the object's application supports visual editing as a server application, then the following occurs:

- □ The object's representation in the FRAME entry changes to an editing session of the actual object. The object's borders might change to accommodate the tools supplied by the server application.
- □ The SAS menu bar changes to accommodate the menu bar of the server application. The **`File`** and **`Window`** menu remains the same, but the remainder of the menu bar changes to that of the server application.
- □ If the server application normally provides any tools, such as toolbar icons or a floating toolbox, those items also become available.

For example, Display 10.1 on page 246 shows a SAS/AF FRAME entry with a Microsoft Word object activated.

**Display 10.1** SAS/AF FRAME Entry with Word Object Activated



After this transformation, you can edit the object using all of the tools and menus provided by the server application.

To end your visual editing session, click elsewhere inside the FRAME entry and outside the object. SAS resumes control of the session, and returns to the default SAS menus and tools.

*Note:*

1. The HSERVICE entry is automatically updated at the end of a visual editing session *only* if the object has been saved previously (that is, an HSERVICE entry has been created for it). Otherwise, you must select **`Save`** (or **`End`**) from the **`File`** menu in SAS/AF software to create the HSERVICE entry.

   Also, if you modify the object during TESTAF mode and you want to save the modifications in the HSERVICE entry, you must update the object's contents by selecting **`Update`** from the **`Locals`** menu before returning to BUILD mode.
2. If you move the OLE object within the FRAME entry during visual editing (in BUILD mode), the object returns to its original position when you click outside of it (ending the visual editing session). If you want to move the object to another position in the FRAME entry, end the visual editing session and then move the object region.
3. Most OLE objects require that you double-click on them to activate them. However, a few types of objects require only a single-click to activate them.
4. If you attempt to edit a linked object or an OLE object whose server application does not support visual editing, the server application launches as a separate instance and allows you to edit the object. This is known as *open editing* and is consistent with the behavior of linked objects and all OLE 1.0 objects.

△

## Invoking OLE Verbs

Each OLE object (except OLE controls) has a default action that it performs when you double-click on it. For many objects, the default action is **`Edit`** (invoking a visual editing session for OLE 2.0 or an open editing session for linked objects and all OLE 1.0 objects). However, there are some objects for which the **`Edit`** action is secondary (for example, a Media Clip object, where **`Play`** is the primary action). Also, many objects have more than one action that they can perform, so they understand more than one OLE verb. (Note that double-clicking on an OLE object in BUILD: DISPLAY mode does not perform the default action, but double-clicking on the object in TESTAF mode does.)

To access the menu of OLE verbs for an OLE object in BUILD mode, click on the object with the *right* mouse button. The name of the OLE object is located at the bottom of the pop-up menu. In the cascading menu off that item, there is a list of valid OLE verbs for the object. Select a verb from this menu to perform that action. The default verb appears first in the list of verbs.

For example, a Microsoft Excel object understands **`Edit`** (for visual editing) and **`Open`** (for open editing). A Media Clip object understands **`Play`** and **`Edit`**.

You can also access the list of valid verbs by clicking on the **`Associated Verbs`** item in the Object Attributes dialog box for the object. This list just contains the names of the verbs; you cannot initiate the verbs from here. Again, the verb at the top of the list is the default verb.

Using SCL, you can invoke any verb that a particular OLE object understands by using the _EXECUTE_ method with the verb as an argument. For example, this code would invoke the verb **`Play`** on the OLE object **`mediaobj`**:

```
call notify('mediaobj', '_EXECUTE_', 'Play');
```

You can specify multiple verbs in a single call to _EXECUTE_. For more information about the _EXECUTE_ method, see “_EXECUTE_” on page 590.

## Using Linked OLE Objects

A linked OLE object contains information about the object’s server application and points to the data file that resides on disk, but does not contain data for the object itself. The object contains a static picture that represents the contents of the linked source.

Using the Links dialog box, you can specify to update a linked object:

- automatically, whenever you update the source file that the object points to. (You must reload the FRAME entry before it reflects the change.)
- manually, by choosing **`Update Now`** in the Links dialog box or by using the _UPDATE_ method in SCL.
- manually, by pointing the object to a different source file using either the Links dialog box or the _UPDATE_ method in SCL.

Linked OLE objects that you include in a FRAME entry:

- support open editing only (as opposed to visual editing, described in “Editing an OLE Object within a FRAME Entry” on page 245). When you double-click on the object’s representation in the FRAME, the server application is invoked in a separate window with the object’s data file open.

  You can also update the data of a linked object by using the server application to open the data file the object points to.
- must point to existing data files. If you change the location of a data file to which an object is linked, you must update the links information for the object.

If you create a linked object using **`OLE - Paste Special`**, the data source that you paste from must be permanent (that is, you must have saved it to disk). If you create a linked object from a temporary data source, SAS will be unable to locate the data to update the object when the data source no longer exists.

## Updating a Linked Object with the Links Dialog Box

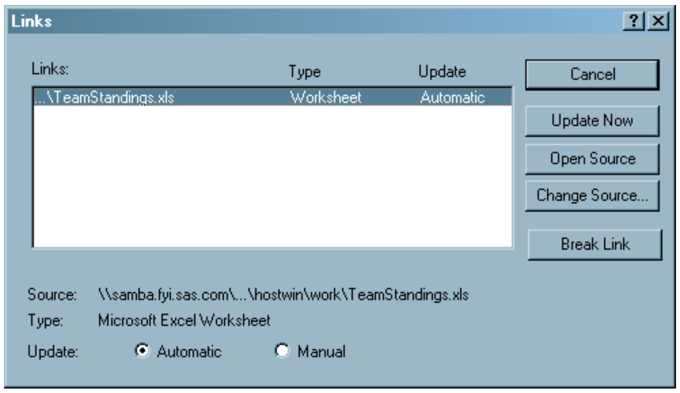
To update the links information with the Links dialog box (shown in Display 10.2 on page 248):

1. Click on the object with the right mouse button. A pop-up menu appears, with the object type listed as the bottom menu item.
2. Click on the bottom menu item. A cascading menu containing valid OLE verbs for the object appears.
3. Click **`Links`**. The Links dialog box appears, containing link information for all of the linked objects in the FRAME entry. (If there are no linked objects in the FRAME, then the **`Links`** item is disabled.)
4. Use the Links dialog box to change information about the object as necessary. For example, if the data file resides in a different location, you can change the source for the object link.

An alternate way to open the Links dialog box for a linked OLE object is to use the DLGLINKS command from the command line. You can also use the _EXECUTE_ method in SCL to invoke the DLGLINKS command. For example:

```
call notify('linkobj','_execute_','dlglinks');
```

**Display 10.2** Links Dialog Box



| Links: | Type | Update |
|---|---|---|
| ...\TeamStandings.xls | Worksheet | Automatic |

Source: \\samba.fyi.sas.com\...\hostwin\work\TeamStandings.xls
Type: Microsoft Excel Worksheet
Update: [✓ Automatic] Manual

## Updating a Linked Object Programmatically

To change the source of a linked object programmatically with SCL, use the _UPDATE_ method to specify a new HSERVICE entry to associate with the object. The _UPDATE_ method for OLE objects accepts the name of an HSERVICE entry as a third argument. (This method overrides the Widget class _UPDATE_ method.) For the syntax of the OLE _UPDATE_ method, see "_UPDATE_" on page 595.

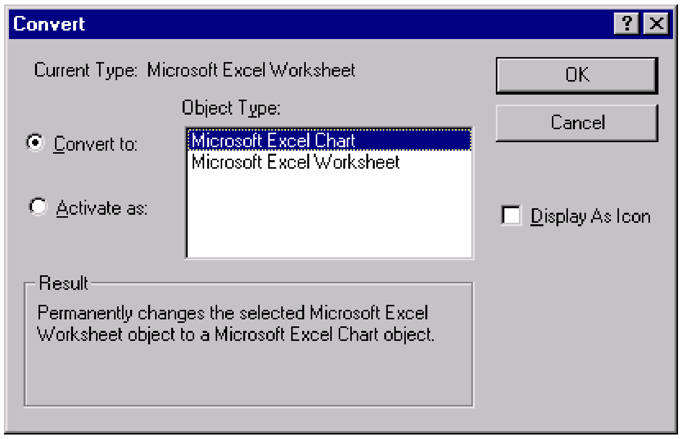
## Converting OLE Objects

An OLE object can be associated with only one server application, but some OLE objects can be converted for use with a different server application than the one that created them.

You can convert an object by using the Convert dialog box (shown in Display 10.3 on page 249). This dialog box lets you:

- change the object's view from an icon to object content and vice-versa.
- change the object's type from one server application to another. For example, you can convert a Microsoft Excel object to a Lotus 1-2-3 object, provided that you have the server application that can convert the object on your system. This type of conversion is permanent.
- activate the object with a different server application than originally created it, without altering the object type. For example, you can choose to activate a Lotus 1-2-3 object using Microsoft Excel as a server. This allows you to edit the object as if it were an Excel object. The object continues to be a Lotus 1-2-3 object. All subsequent Lotus 1-2-3 OLE objects you create will use Excel as an OLE server, unless you change the conversion settings again.

**Display 10.3** Convert Dialog Box



Current Type: Microsoft Excel Worksheet

[● Convert to:] ○ Activate as:

Object Type: [✓ Microsoft Excel Chart] Microsoft Excel Worksheet

☐ Display As Icon

Result
Permanently changes the selected Microsoft Excel Worksheet object to a Microsoft Excel Chart object.

To convert an OLE object within a SAS/AF FRAME entry:

1. Click on the object with the *right* mouse button.
2. At the bottom of the pop-up menu, select the object's name (thus revealing the cascading menu).
3. In the cascading menu, select **`Convert`**. The Convert dialog box appears, listing the valid object types to which you can convert the selected object.
4. If you want to actually convert the object to another type, select the desired target object type and click **`OK`**.

   If you want to toggle the object between icon view and content view, check **`Display As Icon`**.

   If you want to activate the object using another server, click on **`Activate as`** and then select the server application to use.
5. Click **`OK`**.

An alternate way to open the Convert dialog box for an OLE object is to select the object and issue the DLGCONVERT command on the command line. Also, you can use the _EXECUTE_ method in SCL to invoke the DLGCONVERT command. For example:

```
call notify('sheetobj','_execute_',
            'dlgconvert');
```

## Automating OLE Objects and Applications

Some Windows applications provide a scripting language that allows you to control and update objects and external applications through automation. In SAS/AF software, you can use SAS Component Language (SCL) for OLE automation. Using SCL code to send instructions to the OLE object, you can update the object's data based on a user's actions in your SAS/AF application.

In SAS/AF software, you can automate:

- □ OLE objects embedded in a FRAME entry, using the OLE class
- □ OLE objects linked to a FRAME entry, using the OLE class
- □ OLE applications not associated with a FRAME entry, using the OLE Automation class.

Using SCL, you can communicate with any OLE object or application that supports OLE automation as a server. In this communication, SAS acts as a client while the automation application acts as a server. The server provides OLE automation objects, which you can control with SCL code. Using SCL methods, you can send OLE methods to the server for execution. You can also get and set the properties of the objects you control. OLE automation servers can support multiple types of objects, each of which can have a unique set of methods and properties. The SCL methods you can use are listed in Table 10.1 on page 250 and described in detail in "Summary of OLE Class Methods" on page 587.

*Note:* Do not confuse the SCL OLE automation methods (listed in the table) with the methods provided by the OLE automation server. In SAS/AF software, the _COMPUTE_ and _DO_ SCL methods provide access to the methods supported by the OLE automation server. Each OLE automation server supports different methods, but you must always use the _COMPUTE_ or _DO_ method in SCL to invoke them. (You can use subclassing to create new methods that encapsulate these, such as the methods listed in this table.) △

**Table 10.1** OLE Automation Class Methods

| OLE Automation Method | Description |
|---|---|
| _COMPUTE_ | invokes a method supported by the OLE automation server and returns a value |
| _DO_ | invokes a method supported by the OLE automation server (with no return value) |
| _GET_PROPERTY_ | retrieves the value of a property exposed by the OLE automation server |
| _GET_REFERENCE_ID_ | returns the reference identifier of an object provided by the OLE automation server |
| _IN_ERROR_ | returns an object's ERROR status |

| OLE Automation Method | Description |
|---|---|
| _NEW_ | assigns an SCL identifier to an external instance of an OLE automation server |
| _SET_PROPERTY_ | sets the value of a property exposed by the OLE automation server |

*Note:* The return values and arguments passed between the automation server and SAS using the OLE automation methods are passed by value, not by reference–including those arguments that the server defines as pass-by-reference. That is, the arguments contain actual static values, not pointers to values that you can modify. △

## Accessing Array Values Returned by the OLE Automation Server

Using SCL methods and the OLE automation server, SAS lets you

receive a single-dimensional array that is passed to SAS as an SCL list

send or receive multi-dimensional SCL arrays.

In this first example, the SCL code creates and populates a listbox in a Microsoft Excel worksheet and stores the contents of the listbox in an SCL list:

```
list=makelist();  /* create the SCL list */
   /* Add a Listbox in a worksheet */
call send(worksht, '_COMPUTE_', 'Listboxes',
          listbox);
call send(listbox, '_DO_', 'Add', 20, 50,
          40, 100);
call send(worksht, '_COMPUTE_', 'Listboxes',
          1, listone);
   /* Fill the Listbox with a range of */
   /* values from the worksheet        */
call send(listone, '_SET_PROPERTY_',
          'ListFillRange', 'A1:A3');
   /* Get the contents of the Listbox */
call send(listone, '_GET_PROPERTY_',
          'List', list);
```

Using several SCL arrays, the following SCL code creates and populates another Microsoft Excel worksheet:

```
Init:
/* Initialization */
   HostClass = loadclass('sashelp.fsp.hauto');

/* Instantiate the Excel object and make it visible */
   call send (Hostclass, '_NEW_', ExcelObj, 0, 'Excel.Application');
   call send (ExcelObj, '_SET_PROPERTY_', 'Visible', -1);

/* Get the Workbook Object, add a new Sheet and get the Sheet object */
   call send (ExcelObj, '_GET_PROPERTY_', 'Workbooks', WkBkObj);
   call send (WkBkObj, '_DO_', 'Add');
   call send (ExcelObj, '_GET_PROPERTY_', 'ActiveSheet', WkShtObj);

   dcl char names{3,2) =  ('Lucy', 'Ricky',
                         'Julliette', 'Romeo',
                         'Elizabeth', 'Richard');

/* Set the range to be A1:A4 and fill that range with names */
   call send(WkShtObj, '_COMPUTE_', 'Range', 'A1', 'B3', RangeObj);
   call send(RangeObj, '_SET_PROPERTY_', 'Value', names);

   dcl num primes{2,4} = ( 1, 3, 5, 7,
                         11, 13, 17, 23);

/* Set the range to be A5:D6 and fill that range with ints values */
   call send(WkShtObj, '_COMPUTE_', 'Range', A5', 'D6', RangeObj);
   call send(RangeObj, '_SET_PROPERTY_', 'Value', ints);

   dcl char totals{1,4} = ('=SUM(A5,A6)',
                          '=SUM(B5,B6)',
                          '=SUM(C5,C6)',
                          '=SUM(D5,D6)');

/* Set the range to be A1:A4 and fill that range with totals */
   call send(WkShtObj, '_COMPUTE_', 'Range', 'A7', 'D7', RangeObj);
   call send(RangeObj, '_SET_PROPERTY_', 'Value', totals);

   dcl char vals{7,4};

   call send(WkShtObj, '_COMPUTE_', 'Range', 'A5', 'D7', RangeObj);
   call send(RangeObj, '_GET_PROPERTY_', 'Value', vals);
return
```

## Using Value Properties

OLE automation servers (including OLE custom controls) can designate one of their properties or methods as a *value property*, which is used as the default property or method when the automation code accesses an object provided by the server without explicitly specifying a property or method name.

In SCL, you can access the value property of a server by specifying an empty string in place of the property name when invoking _GET_PROPERTY_ or _SET_PROPERTY_, or in place of the method name when using _DO_ or _COMPUTE_. For example, if the Text property is the value property, then the following code:

```
call notify('sascombo', '_set_property_', '',
            'An excellent choice');
```

is equivalent to:

```
call notify ('sascombo', '_set_property_',
             'Text', 'An excellent choice');
```

Both the SAS ComboBox and SAS Edit controls (supplied with SAS) designate Text as their value property.

## Specifying Optional Parameters in OLE Server Methods

Some OLE server applications expose methods that have optional parameters; that is, if you do not specify a value for one or more of the parameters that a method supports, the OLE server uses a default value for those parameters. Refer to the documentation for the OLE server application you are using for information about which parameters are optional.

SAS supports the use of optional parameters by letting you specify a SAS missing value in place of the parameter you want to omit. The default missing value character is a period (but that can be changed by using the MISSING system option).

For example, Microsoft Excel supports a ChartWizard method that accepts 11 arguments, most of which are optional. This SCL code invokes this method with all of its arguments:

```
call send(chart, '_DO_', 'ChartWizard', hcell,
                 -4098, 6, 1, 0, 0, 1,
                 "Automation at work!",
                 'Column', 'Value', 'Row');
```

Here is the equivalent SCL code that omits the optional parameters (substituting the missing value character):

```
call send(chart, '_DO_', 'ChartWizard', hcell,
                ., ., ., ., ., .,
                "Automation at work!",
                 ., ., .);
```

*Note:* Your SCL code must still respect the position of the optional parameters when invoking methods. When you specify a missing value character as an argument, it must be in place of a parameter that is optional to the OLE server's method. △

## Creating an External OLE Automation Instance

External OLE Automation Instances can be for an application on your local machine or an application on a remote machine. Before you can automate an external OLE application, you must create an instance of the OLE Automation class. (Note that this is not necessary when you automate objects that you embed or link in your FRAME entry, because placing them in the FRAME entry creates the instance for you.) Unlike the OLE class, the OLE Automation class is not derived from the Widget class and, therefore, has no visual component to include in a FRAME entry. Instead, you must load an instance of the HAUTO class (using the LOADCLASS function) in the SCL code that drives the automation. For example:

```
hostcl=loadclass('sashelp.fsp.hauto');
```

After you create an instance of the OLE Automation class, you must associate the new instance with an SCL object identifier (which you need to use when calling methods with CALL SEND) and an OLE server application. To obtain the identifier, use the _NEW_ method on the newly created instance of the OLE Automation class. This example stores the object identifier in **`oleauto`** and associates the object with Microsoft Excel (which has the identifier **`Excel.Application`** in the Windows registry) on the local machine.

```
call send(hostcl, '_NEW_', oleauto, 0,
          'Excel.Application');
```

To create an instance of the OLE Automation class for a remote machine, the remote machine must be configured to permit the user to start remote instances using Distributed COM Configuration Properties (DCOMCNFG.EXE). The DCOMCNFG.EXE is located in the \WINNT\SYSTEM32 folder. For more information on Distributed COM Configuration Properties, see your Windows documentation. The following example creates an instance of Microsoft Excel on a remote machine. Once created, the method and property calls to that instance work as if it were on a local machine.

```
Init:
  HostClass = loadclass('sashelp.fsp.hauto');
  ExcelObj = 0;

  /* Define the machine name and put it in a list  */

  machineName = '\\Aladdin';
  inslist = makelist();
  attrlist = makelist ();

  rc = insertc (attrlist, machineName, -1, 'remoteServer');
  rc = insertl (inslist, attrlist, -1, '_ATTRS_');

  /* Instantiate the Excel object and make it visible */

  call send (HostClass, '_NEW_',ExcelObj, inslist,
  'Excel.Application');
  call send (ExcelObj, '_SET_PROPERTY_', 'Visible', -1);
return;
```

For more information about the _NEW_ method, see "_NEW_" on page 594.

After you create an instance of an OLE Automation object, you can automate that object in much the same way you would automate an object that you have embedded or linked in your frame. The following table notes some key differences between the types of objects.

| SAS OLE objects... | SAS OLE Automation objects... |
|---|---|
| are derived from the Widget class. | are derived from the Object class. |
| have a visual component (the object you place in the FRAME entry). | have no visual component within the FRAME entry. |
| are created by placing the object in a region in the FRAME entry (using drag and drop). | are created by using the LOADCLASS statement and the _NEW_ method in SCL. |
| represent the specific type of data object (which you choose) supported by the OLE server. | represent the top-level application object supported by the OLE server, which you then might use to open objects of specific data types. |
| allow you to call methods with CALL NOTIFY by passing in the object name from the FRAME entry. | require you to call methods with CALL SEND, passing in the object identifier returned by the _NEW_, _GET_PROPERTY_, or _COMPUTE_ methods. |

## Example: Populating a Microsoft Excel Spreadsheet with SAS Data

The following table contains SCL code to populate a Microsoft Excel spreadsheet with data from a SAS data set.

**Table 10.2** SCL Code for Populating a Microsoft Excel Spreadsheet

| | |
|---|---|
| Load an instance of the OLE Automation class and invoke Excel. Set the object to Visible so you can see the automation in progress. | `LAUNCHXL:`<br><br>`hostcl = loadclass('sashelp.fsp.hauto');`<br>`call send(hostcl, '_NEW_', excelobj, 0,`<br>`'Excel.Application');`<br>`call send(excelobj, '_SET_PROPERTY_', 'Visible', 'True');`<br>`return;` |
| Get the identifier for the current Workbooks property and add a worksheet. Then get the identifier for the new worksheet. | `CREATEWS:`<br>`call send(excelobj, '_GET_PROPERTY_', 'Workbooks',`<br>`wbsobj);`<br>`call send(wbsobj, '_DO_', 'Add' );`<br>`call send(excelobj, '_GET_PROPERTY_', 'ActiveSheet',`<br>`wsobj);` |
| Open a SAS data set. | `dsid=open('sasuser.class','i');`<br><br>`call set(dsid);`<br>`rc=fetch(dsid);`<br>`nvar=attrn(dsid, 'NVARS');`<br>`nobs=attrn(dsid, 'NOBS');` |

Traverse the data set and populate the cells of the Excel worksheet with its data, row by row.

```
do
col=1 to nvar;
  call send(wsobj, '_COMPUTE_', 'Cells',1,col,retcell);
  var=varname(dsid,col);
  call send(retcell,'_SET_PROPERTY_', 'Value',var);
end;
do while (rc ne -1);
   do row = 1 to nobs;
      do col = 1 to nvar;
         r=row+1;
         call send (wsobj, '_COMPUTE_', 'Cells', r ,col,retcell);
         if vartype(dsid,col) eq 'N' then do;
            varn=getvarn(dsid,col);
            call send(retcell, '_SET_PROPERTY_', 'Value' ,varn);
         end;
         else do;
            varc=getvarc(dsid,col);
            call send(retcell, '_SET_PROPERTY_', 'Value' ,varc);
         end;
      end;
      rc=fetch(dsid);
   end;
end;
dsid=close(dsid);
return;
```

Close the worksheet and end the Excel session. The _TERM_ method deletes the OLE automation instance.

```
QUITXL:
call send(excelobj,'_GET_PROPERTY_', 'ActiveWorkbook',
 awbobj);
call send(awbobj,  '_DO_', 'Close', 'False');
call send(excelobj, '_DO_', 'Quit');
call send(excelobj, '_TERM_');
return;
```

As you can see from this example, automating an application object requires some knowledge of the object's properties and methods. To help you decide which SCL commands to use for an Excel automation object, you can use the Macro Recorder in Excel to perform the task you want to automate, then look at the Visual Basic code that is generated. It is then relatively simple to map the Visual Basic code to comparable SCL statements and functions.

The following table shows some excerpts of Visual Basic code and their SCL equivalents.

**Table 10.3** Visual Basic Code Samples and Their SCL Equivalents

| Visual Basic Code | OLE Automation in SCL |
|---|---|
| *Launch Excel and make it visible*<br>`Set excelobj = CreateObject("Excel.Application")`<br>`excelobj.Visible = True` | `hostcl = loadclass('sashelp.fsp.hauto');`<br><br>`call send ( hostcl,  '_NEW_', excelobj, 0,`<br>`   'Excel.Application');`<br>`call send (excelobj,'_SET_PROPERTY_',`<br>`   'Visible','True');` |
| *Create a new worksheet*<br>`Dim wbsobj, wsobj As Object`<br>`Set wbsobj = excelobj.Workbooks`<br>`wbsobj.Add`<br>`Set wsobj = excelobj.ActiveSheet` | `call send(excelobj,'_GET_PROPERTY_',`<br>`   'Workbooks', wbsobj);`<br>`call send(wbsobj,  '_DO_', 'Add');`<br>`call send(excelobj,'_GET_PROPERTY_',`<br>`   'ActiveSheet', wsobj );` |
| *Set the value of a cell*<br>`wsobj.Cells(row + 1, col).Value`<br>`=var` | `r=row+1;`<br>`call send(wsobj,'_COMPUTE_', 'Cells', r, col,`<br>`   retcell);`<br>`call send(retcell,'_SET_PROPERTY_',`<br>`   'Value' ,var);` |
| *Close the Excel application object*<br>`excelobj.ActiveWorkbook.Close`<br>`("False")`<br>`excelobj.Quit` | `call send(excelobj,'_GET_PROPERTY_',`<br>`'ActiveWorkbook', awbobj);`<br>`call send(awbobj, '_DO_', 'Close', 'False');`<br>`call send(excelobj,'_DO_', 'Quit');`<br>`call send(excelobj,'_TERM_');` |

## Using OLE Custom Controls (OCXs) in Your SAS/AF Application

An OLE custom control is a special type of OLE object or collection of OLE objects that has an interface to expose its own properties and methods. You can control these objects through its graphical interface and with SCL code.

OLE custom controls differ from other OLE objects in these ways:

- □ They generate events based on user actions, which you can respond to in your FRAME entry. Note that the object's SCL label is not run by default when you activate an OLE control.
- □ They assume ambient properties (such as color and font) based on the environment in which they are used.

OLE controls are packaged in their own dynamic linked library (with a file extension of OCX). Using SCL code, your FRAME entry can respond to events generated by the OLE control (mouse clicks, key presses, and so on). The events exposed by OLE controls vary among controls. For a list of events, see the documentation for the control you are using. After inserting the control into the FRAME entry, you can view the event map by selecting **`Object Attributes`** for the OLE control object and then **`Event Map`**.

*Note:* The OLE controls that SAS provides require 32-bit containers, which makes them unusable with Windows applications that offer only 16-bit container support. Also, because SAS is a 32-bit container, you cannot use 16-bit controls with it. △

### Inserting an OLE Control in a FRAME Entry

To insert an OLE control in a FRAME entry:

1 From the COMPONENTS window, select the **`V6 objects`** item to expand the object tree.
2 Double-click on **`OLE - Insert Object`** from the Selection List. The Insert Object dialog box opens. You can also drag **`OLE - Insert Object`** from the Selection List to the BUILD window. When you release the mouse button the Insert Object dialog box opens.
3 Select the **`Create Control`** radio button to display a list of registered OLE custom controls. If the OLE control you want to use is not listed here and you have it on your system, you need to register the control (see "Registering OLE Controls" on page 258).
4 Select the name of the OCX control you want to insert.

## Registering OLE Controls

Before you can use any OLE control in Windows, the control must be registered with Windows. SAS ComboBox and SAS Edit, the two OLE controls provided with SAS, are automatically registered when you install SAS.

If you want to install other controls for use with SAS or other applications, you must register the control with Windows (unless the control was installed by a process that performed the registration for you). The OLE control will not be available from the Insert Object dialog box until it is registered.

To register an OLE control:

1 Complete steps 1-4 as described in "Inserting an OLE Control in a FRAME Entry" on page 257 to invoke the Insert Object dialog box with the list of registered controls.
2 Click on **`Add Control`** to invoke the Browse file selection dialog box.
3 Use the dialog box to select the control (which usually has a file extension of OCX) that you want to register.

   When you click **`OK`**, the control is added to the list of registered controls in the Insert Object dialog box.

## Accessing OLE Control Properties

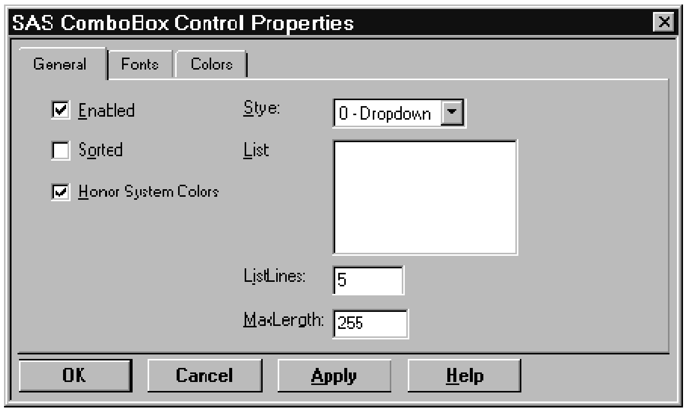
OLE controls have properties that you can set or retrieve using SCL methods. Some controls make some of their properties available through a properties page, which lets you set or retrieve the data interactively.

### Accessing the OLE Control Properties Page

To invoke the properties page for a control, click on the right mouse button within the control's region in the BUILD: DISPLAY window and then select **`Properties`** from the pop-up menu. The properties page for the control appears. Display 10.4 on page 259 shows an example of a properties page for an OLE control.

OLE controls provide a Properties verb, which you can use with the _EXECUTE_ method in SCL to bring up the Properties page for the control. Or, you can access the pop-up menu for the control, then choose the cascading menu with the control's name. The Properties verb is available off that cascading menu.

**Display 10.4** An Example Properties Page


SAS ComboBox Control Properties

General | Fonts | Colors

☑ Enabled
☐ Sorted
☑ Honor System Colors

Stye: 0 - Dropdown
List:
ListLines: 5
MaxLength: 255

OK | Cancel | Apply | Help

You can use the properties page to view or change settings for some of the exposed properties.

Note that the control is not active (that is, you cannot interact with its interface) while you are in DISPLAY mode. The control becomes active in TESTAF mode.

## Accessing Properties Using SCL Code

When you use OLE controls in a SAS/AF application, you may want to access the properties of the control programmatically. Also, an OLE control might not expose all of its properties in a properties page. You can access the properties of a control by using the _SET_PROPERTY_ and _GET_PROPERTY_ methods.

Before you can access a property, you must know:

- □ the object label of the OLE control in your SAS/AF FRAME entry
- □ the name of the property you want to access
- □ the type of data that the property holds.

For example, suppose you have a combo box control named **sascombo** in your FRAME entry, and you want to set the list style to **simple** (represented by the integer 1):

```
call notify ('sascombo', '_set_property_',
             'Style', 1);
```

If you want to retrieve data from a property, you must use a variable that is of the same type as the data you want to read. For example, if you want to learn what text the user specified in the edit portion of a combo box, include the following code:

```
length text $ 200;
call notify ('sascombo', '_get_property_',
             'Text', text);
```

## Interacting with the OLE Control Using SCL Methods

OLE controls support methods that control their content and behavior. You use either the _DO_ or _COMPUTE_ SCL methods to send a message to an OLE control telling it to implement one of its methods.

- □ Use the _DO_ method in SCL when the OLE control method performs some action but does not return a value. For example, the SAS ComboBox OLE control has a method that clears all items from the list:

  ```
  call notify('sascombo', '_DO_', 'Clear');
  ```

- Use the _COMPUTE_ method in SCL when the OLE control method returns a value. You specify a variable in the SCL code that will contain the return value when the method ends. For example, the SAS ComboBox OLE control has a method that returns an item at a specified position in the list:

  ```
  length item $ 80;
  call notify('sascombo', '_COMPUTE_',
              'GetItem', 2, item);
  ```

  When this call returns, *item* contains the text of the item at position 2 (the third item in the list).

## Responding to OLE Control Events

OLE controls generate events that you can respond to in your SCL code. You can create a label in your SCL code for OLE events just like you do for SAS/AF events.

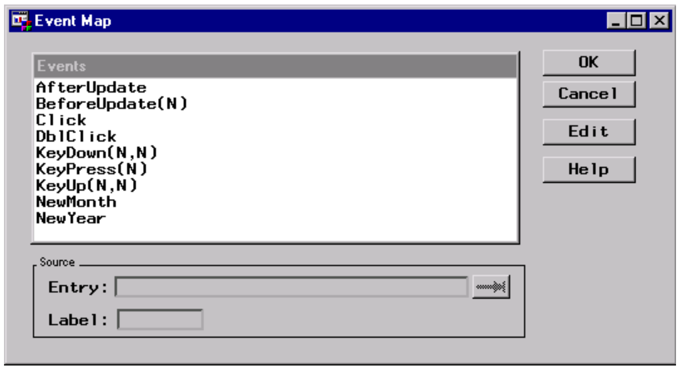
### Assigning SCL Code to an OLE Control Event

To assign SCL code to run when an OLE control event occurs:

1. Select the OLE control object in the BUILD window.
2. Select **Object Attributes** from the pop-up menu for the object.
3. Select **Event Map** from the Object Attributes dialog box. The Event Map dialog box appears (shown in Display 10.5 on page 260).
4. In the Event Map dialog box, select the event that you want to respond to using SCL code.
5. Specify the SCL, FRAME, or PROGRAM source entry and (if applicable) the SCL label where the event-handling code resides.

   *Note:* You can specify the same SCL source entry that is stored with the FRAME entry; however, in addition to compiling the code with the FRAME entry, you must also compile the SCL entry outside of the FRAME context (that is, outside of the BUILD: SOURCE and BUILD: DISPLAY windows) in order for the event handler to recognize the SCL label. Thus, it is more efficient to store event-handling code for OLE controls in an SCL source entry that is not associated with a FRAME entry. △

**Display 10.5** Event Map Dialog Box



*Note:* Many OLE controls include a LostFocus event, which they generate when the control loses window focus. Because of the way that SAS/AF software communicates with the control, mapping the LostFocus event sometimes has the effect of placing focus back on the control that just lost it. Although you can still respond to the LostFocus event in your FRAME entry, be aware that this might cause unusual focus behavior. △

## Retrieving Argument Values from Events

Some OLE control events also include parameters you might find useful. For example, the SAS ComboBox control generates a KeyPress event that also reports the ASCII value of the key that was pressed. If a particular event passes an argument back to the FRAME entry, the type of value returned is indicated in the Event Map dialog box. A numeric value is indicated with an **`N`**; a character value is indicated with a **`C`**.

To retrieve the value returned by an OLE control event, you must define a method (using the METHOD statement in SCL) in the event-handling code. In the argument list for the METHOD statement, specify a variable of the type that you expect the OLE control to return. This variable contains the value returned by the event. You can then use that variable as you wish inside your event-handler.

For example, suppose you want to retrieve the value of the key that triggered the KeyPress event in the SAS ComboBox control and then report it as an ASCII character. The KeyPress event returns an integer that represents the ASCII value of the key pressed. Your event-handling code would look like the following:

```
   /* Label specified in Event Map dialog box */
KEYPRESS:
   /* Define a method with an
      integer argument */
method keyval 8;
      /* Convert the integer to an
         ASCII character */
   keychar=byte(keyval);
  put keychar=; /* Output the character */
endmethod;
```

## Example: Mapping OLE Control Events to SCL Code

When mapping OLE control events, you can do one of the following:

- □ Map each event in the Event Map window to a different labeled section of SCL code, with each piece of code performing different actions.
- □ Map all of the events in the Event Map window to a single labeled section of SCL code, use the _GET_EVENT_ method to detect which event was triggered, and act accordingly.
- □ Use a combination of these strategies by assigning one event (such as the Click event) to SCL code that runs the object's label (using the _OBJECT_LABEL_ method), and map the remaining events to a single label that uses the _GET_EVENT_ method to determine the event and appropriate action. The object's label is not run by default for OLE controls.

The following example shows how to structure the SCL code when all events for an OLE control are mapped to a single label, which in turn runs the object's label to determine the event and act accordingly:

```
length event $ 80;
  /* All OLE control events are mapped to
     this label */
RUNLABEL:
     /* Call the object's label */
  call send(_self_, '_OBJECT_LABEL_');
```

```
return;
  /* This is the label of the OLE control */
OBJ1:
     /* Determine the last event */
  call notify('obj1', '_GET_EVENT_', event);
  select (event);
     when('Click') put 'Click received';
     when('DblClick') put 'DblClick received';
     otherwise put event=;
  end;
return;
```

## Example: Subclassing an OLE Custom Control

If you create SAS/AF applications that make frequent use of one or more OLE custom controls, you might want to write your own methods to abstract the methods that the control recognizes without having to specify the intermediate _DO_ and _COMPUTE_ methods in SCL.

You can achieve this by creating a subclass of the OLE class and adding methods to your derived class. When you insert the OLE control into your FRAME entry, be sure to insert it as an instance of the new class that you define (instead of **`OLE - Insert Object`**). The examples provided here contain sample code you can use to abstract the methods of a control. They do not include details about how to create subclasses. For information about creating subclasses of a SAS/AF classes, see the online documentation for SAS/AF software.

## Adding an Item to a Combo Box List

You can use this method to add a new item to the list portion of the SAS ComboBox control. The SAS ComboBox control uses zero-based numbering to indicate the positions of the list items (that is, the first item is at position 0, the second is at position 1, and so on). The following method lets you specify the position numbers such that position 1 holds the first item.

```
/* Add a new item to a ComboBox list.  */
ADDITEM:
method text $200 row 8 rc 8;
     /* adjust for zero-based index */
  ocxrow = row-1;
  call send(_self_, '_COMPUTE_', 'AddItem',
             text, ocxrow, rc);
  if ( rc = 0 ) then
    _MSG_="ERROR: Could not add item to list.";
endmethod;
```

Assuming you mapped this code to a new method called ADD_ITEM, you would use this syntax to add a new item to the control:

```
   /* Adds 'Item 1' at the first position */
   /* in the control                      */
length success 8;
call notify('sascombo', 'ADD_ITEM',
            'Item 1', 1, success);
```

## Finding an Item in a Combo Box

The following method finds the specified item and returns its position in the list. As in the previous example, this method adjusts the position number to be one-based instead of zero-based.

```
FINDITEM:
method text $200 row 8;
  call send(_self_, '_COMPUTE_', 'FindItem',
             text, row);
  row = row + 1; /* adjust for zero-based */
endmethod;       /* index                */
```

Assuming you mapped this code to the FIND_ITEM method, you would then use it as in this example:

```
length position 8;
call notify('sascombo','FIND_ITEM',
            'Lost Item', position);
```

## Retrieving the Text Value of the Control

Both the SAS ComboBox and SAS Edit controls have Text properties, which you can access using the _GET_PROPERTY_ method with the property name. For easier and more intuitive access from your OLE subclass, you can override the _GET_TEXT_ method and map it to this code:

```
GETTEXT:
method text $200;
   call send(_self_, '_GET_PROPERTY_',
             'Text', text);
endmethod;
```

You would then access the Text property of a control the same way you access the text of other SAS/AF widget objects:

```
length text $ 200;
call notify('sasedit', '_GET_TEXT_', text);
```

# CHAPTER 11 Controlling SAS from Another Application Using OLE

*Introduction to Automating SAS* **265**
*Creating an Instance of SAS* **265**
*Getting Feedback from the SAS Session* **266**
*Examples of Automating SAS with OLE* **267**
  *Creating a SAS Automation Object* **267**
  *Determine Whether the SAS Session is Busy* **267**
  *Toggle the SAS Session between Visible and Invisible* **267**
  *Set the Main SAS Window Title of the SAS Session* **267**
  *Assign a SAS Data Library and Run a SAS Procedure* **267**
  *End the SAS Session* **268**
*Methods and Properties for Use with a SAS OLE Automation Object* **268**
  *Command Method* **268**
  *QueryWindow Method* **269**
  *Quit Method* **269**
  *Submit Method* **270**
  *Top Method* **270**

## Introduction to Automating SAS

SAS can perform as an OLE automation server. This means that you can use an application that can act as an OLE automation controller (such as Visual Basic) to create a SAS session and control it using the methods and properties that SAS makes available.

Many Windows applications use Visual Basic or Visual Basic for Applications as the scripting language for automation. All examples that are provided in this document use Visual Basic, but you can achieve the same results with any application that can act as an OLE automation controller.

## Creating an Instance of SAS

To create an instance of SAS (that is, invoke a SAS session), you must create an OLE object by using the SAS program identifier as it is listed in the Windows registry. The SAS program identifier is **`SAS.Application`**. Here is a Visual Basic example that instantiates (creates an instance of) a SAS session:

```
Dim OleSAS as Object
Set OleSAS = CreateObject("SAS.Application")
```

This example sets the identifier **OleSAS** to the new SAS session. You can then use this identifier to access the methods and properties that SAS makes available.

If you want to control an existing SAS automation object by using OLE automation, you can use your automation controlling language. In Visual Basic, you can use the following:

```
Dim OleSAS as Object
Set OleSAS = GetObject(,"SAS.Automation")
```

Note that this code does not create an instance of SAS if one does not already exist. Also, the existing SAS session must have been created as an OLE automation object (for example, using **CreateObject** in Visual Basic). You cannot use OLE automation to control a SAS session you invoked by using another method (for example, by using the **Start** menu).

## Getting Feedback from the SAS Session

SAS provides two properties, RC and ResultString, that make it possible to pass information from the SAS session that you are automating back to the application that is controlling it. RC can contain a number; ResultString can contain a text string.

To set the values of these properties from within the SAS session, use the SETRC function with this syntax:

```
error=SETRC("result-string", rc-number);
```

where *result-string* is the value to be assigned the ResultString property, and *rc-number* is the value to be assigned to the RC property.

For example, you can use the Submit method to submit DATA step code that returns an error code as part of its processing. You can then check the value of that error using the RC or ResultString property. Here is a Visual Basic example of this:

```
Private Delcare Sub Sleep Lib ‘‘kernel32’’ (ByVal dwMilliseconds As Long)

Private Sub Command1_Click()
Dim OleSAS as Object
Set OleSAS = CreateObject("SAS.Application")
OleSAS.Submit("data _null_;
       error=setrc(’Error string’, 2.0);
       put error; run;")
Sleep 500
Do while x=0
  If (OleSAS.Busy) then
     Sleep 500
  else
     If (OleSAS.RC <> 0) Then
         Response = MsgBox(OleSAS.ResultString,
               vbOKOnly, "Error message is",
               0, 0)
   x=1
EndIf
Loop
End Sub
```

## Examples of Automating SAS with OLE

The following examples use Visual Basic as the scripting language to control SAS with OLE automation. You can use any scripting language from any Windows application that can act as an OLE automation controller.

### Creating a SAS Automation Object

This Visual Basic code defines an object and creates an instance of SAS to associate with that object.

```
Dim OleSAS As Object
Set OleSAS = CreateObject("SAS.Application")
```

### Determine Whether the SAS Session is Busy

This Visual Basic code queries the SAS session (using the Busy property) to test whether the session is busy processing code.

```
If (OleSAS.Busy) Then
   Response = MsgBox("SAS Session is Busy",
              vbOKOnly, "SAS Session", 0, 0)
Else
   Response = MsgBox("SAS Session is Idle",
              vbOKOnly, "SAS Session", 0, 0)
End If
```

### Toggle the SAS Session between Visible and Invisible

This Visual Basic code hides or unhides the SAS session based on its current state.

```
OleSAS.Visible = false
```

### Set the Main SAS Window Title of the SAS Session

This Visual Basic code assigns a title to the main SAS window of the SAS session and then displays the title in a message box.

```
OleSAS.Title = "Automation Server"
Response = MsgBox(OleSAS.Title, vbOKOnly,
           "Title Is", 0, 0)
```

### Assign a SAS Data Library and Run a SAS Procedure

This Visual Basic code submits SAS code to the SAS session, assigning a SAS data library and running the INSIGHT procedure on sample data.

```
OleSAS.Submit("libname insamp
                'c:\sas\insight\sample';
               proc insight data=insamp.drug;
               run;")
```

## End the SAS Session

This Visual Basic code ends the SAS session provided that there are no other OLE automation controllers making use of it.

```
OleSAS.Quit
Set OleSAS = Nothing
```

## Methods and Properties for Use with a SAS OLE Automation Object

Once instantiated, the SAS OLE automation object supports these methods as well as several properties:

- "Command Method" on page 268
- "QueryWindow Method" on page 269
- "Quit Method" on page 269
- "Submit Method" on page 270
- "Top Method" on page 270
- "Properties for Controlling a SAS Automation Object" on page 271

## Command Method

**Invokes a command as if it was entered from the SAS command line**

### Syntax

Command("*sas-command*")

### Details

By default, the active window receives the command. You can change which window receives the command by changing the CommandWindow property.

### Example

This Visual Basic code invokes a SAS session and opens the BUILD window:

```
Set OleSAS = CreateObject("SAS.Application")
OleSAS.Command("build")
```

## QueryWindow Method

**Queries whether a specified window exists within the SAS session**

### Syntax

QueryWindow("*window-name*")

### Details

QueryWindow returns either True or False based on whether the specified window is open in the automated SAS session. If the window exists but is not visible, QueryWindow still returns True.

The window name that you specify must match the exact spelling of the window name in SAS. The *window-name* argument is not case sensitive.

### Example

This Visual Basic code gets an existing SAS session and checks whether the BUILD window is open. If the window is not open, this code invokes BUILD:

```
Dim OleSAS as Object
Set OleSAS = GetObject(,"SAS.Application")
If (Not OleSAS.QueryWindow("build")) Then
   OleSAS.Command("build")
EndIf
```

## Quit Method

**Ends the SAS session**

### Syntax

Quit

### Details

If the automation controller that issues the Quit method is the only controller that is using that particular SAS session, then the SAS session ends. If at least one other automation process is still using the SAS session, then the session remains running.

## Submit Method

**Submits DATA step or procedure code for processing**

### Syntax

Submit(“*SAS-program-code*”)

### Details

The string of text that you specify as *SAS-program-code* can contain multiple SAS statements separated by semicolons. The contents of the string are submitted to SAS for processing.

### Example

The following example references a data library and invokes a SAS/AF application:

```
Dim OleSAS as Object
Set OleSAS = CreateObject("SAS.Application")
OleSAS.Visible = True
OleSAS.Submit("libname afapp 'f:\sas\afapp';")
OleSAS.Command("af c=afapp.bigapp.main.frame")
```

## Top Method

**Brings the SAS session to the foreground**

### Syntax

Top

### Details

The Top method works only if the Visible property is set to True.

### Example

The following example invokes a SAS session, makes it a visible window, and then brings it to the foreground.

```
Dim OleSAS as Object
Set OleSAS = CreateObject("SAS.Application")
OleSAS.Visible = True
OleSAS.Top
```

## Properties for Controlling a SAS Automation Object

**Specify various properties of the SAS automation object**

### Properties and Descriptions

Busy
: indicates whether SAS is idle or working (for example, running a procedure, DATA step, and so on). This property is read only.

CommandWindow
: sets the window (based on the window title) to receive commands you specify using the Command method. The name you specify must match the spelling of the window name exactly (though this property is not case sensitive). Once set, the window receives subsequent commands you specify with the Command method until CommandWindow is changed or set to Null (by specifying ""). If Null, which is the default, the currently active window receives the command. This property is read/write.

CommandWindowVisible
: controls whether the window specified by the CommandWindow property is visible. If set to False, the window specified by the CommandWindow property is set to invisible. If the CommandWindow property is Null, this property has no effect. This property is read/write.

ConfirmExit
: controls the behavior of how SAS exits. A value of 0 means that no confirmation box is displayed before SAS exits. A value of 1 means that a confirmation box is displayed before SAS exits. A value of 2 selects the default action, which is controlled by an alternative method that defines how SAS exits; for example, the Preferences dialog.

Height
: sets the height, in pixels, of the SAS application window. This property is read/write.

Parent
: sets the name of the parent window that contains the SAS application window. If you change this value to another window, the SAS application window resizes to fit in the new frame. This property is read/write.

RC
: returns the return code passed by a user function. You can set this property from within the SAS session by using the SETRC function. This property is read-only from the automation controller.

ResultString
: returns a string passed by a user function. You can set this property from within the SAS session by using the SETRC function. This property is read-only from the automation controller.

Title

sets the main SAS window title. This property is read/write.

Visible
: controls whether SAS is visible. This property is read/write.

Width
: sets the width, in pixels, of the SAS application window. This property is read/write.

X
: sets the horizontal coordinate, in pixels, for the top left corner of the SAS application window. This property is read/write.

Y
: sets the vertical coordinate, in pixels, for the top left corner of the SAS application window. This property is read/write.

CHAPTER

# 12 Using Dynamic Data Exchange

*Overview of Dynamic Data Exchange (DDE)* **273**
*DDE Syntax within SAS* **273**
*Referencing the DDE External File* **274**
*Using the DDE Triplet* **275**
*Controlling Another Application Using DDE* **276**
*DDE Examples* **276**
*Using the X Command to Open a DDE Server* **276**
*Using DDE to Write Data to Microsoft Excel* **276**
*Using DDE to Write Data to Microsoft Word* **277**
*Using DDE to Read Data from Microsoft Excel* **277**
*Using DDE to Read Data from Microsoft Word* **277**
*Using DDE and the SYSTEM Topic to Invoke Commands in an Application Using Excel* **278**
*Using the NOTAB Option with DDE* **279**
*Using the DDE HOTLINK* **280**
*Using the !DDE_FLUSH String to Transfer Data Dynamically* **281**
*Reading Missing Data* **281**

## Overview of Dynamic Data Exchange (DDE)

*Dynamic Data Exchange (DDE)* is a method of dynamically exchanging information between Windows applications. DDE uses a client/server relationship to enable a client application to request information from a server application. SAS is always the client. In this role, SAS requests data from server applications, sends data to server applications, or sends commands to server applications.

You can use DDE with the DATA step, the SAS macro facility, SAS/AF applications, or any other portion of SAS that requests and generates data. DDE has many potential uses, one of which is to acquire data from a Windows spreadsheet or database application.

*Note:* Many Windows programs, including SAS, now support OLE to facilitate communication between applications. If you need to share data with an application that supports OLE, you might prefer to use the OLE support that is built into SAS. For more information, see “About OLE” on page 242. △

## DDE Syntax within SAS

To use DDE in SAS, issue a FILENAME statement with the following syntax:

**FILENAME** *fileref* DDE ’*DDE-triplet*’ <DDE-options>;

where:

*fileref*
: is a valid fileref (as described in “Referencing External Files” on page 146).

DDE
: is the device-type keyword that tells SAS you want to use Dynamic Data Exchange.

’*DDE-triplet*’
: is the name of the DDE external file.

*DDE-options*
: can be any of the following:

  COMMAND
  : allows remote commands to be issued to DDE server applications. For more information, see “Controlling Another Application Using DDE” on page 276.

  HOTLINK
  : instructs SAS to use the DDE HOTLINK. For an example of using this option, see “Using the DDE HOTLINK” on page 280.

  LRECL=*record-length*
  : specifies the record length (in bytes). Under Windows, the default is 256. The value of *record-length* can range from 1 to 1,048,576 (1 megabyte).

  NOTAB
  : instructs SAS to ignore tab characters between variables. For an example of using this option, see “Using the NOTAB Option with DDE” on page 279.

  RECFM=*record-format*
  : controls the record format. The following are valid values under Windows:

| | |
|---|---|
| F | indicates fixed format. |
| N | indicates binary format and causes the file to be treated as a byte stream. |
| P | indicates print format. |
| S370V | indicates the variable S370 record format (V). |
| S370VB | indicates the variable block S370 record format (VB). |
| S370VBS | indicates the variable block with spanned records S370 record format (VBS). |
| V \| D | indicates variable format. This is the default. |

The S370 values are valid with z/OS types of files only. That is, they are valid in files that are binary, have variable-length records, and are in EBCDIC format. If you want to use a fixed-format z/OS file, first copy it to a variable-length, binary z/OS file.

***CAUTION:***
**Use caution when using DDE with data values that are blank or missing.** For sample code, see “Reading Missing Data” on page 281. △

## Referencing the DDE External File

When you define a fileref to use with DDE, the *DDE-triplet* argument refers to the DDE external file.

## Using the DDE Triplet

The DDE triplet is application-dependent and is different for every application you run. For information on an application's DDE triplet, see the application's documentation.

The triplet takes the following form:

'*application-name*|*topic*!*item*'

where:

*application-name*
: is the executable filename of the server application. For example, the *application-name* for Microsoft Word is **winword**, and for Microsoft Excel it is **excel**.

*topic*
: is the topic of conversation (between SAS and the DDE server application). This is typically the full path filename of the document or spreadsheet with which you want to share data.

*item*
: is the range of conversation specified between the client and server applications. In spreadsheet applications, this is usually a range of cells. For document-based applications (for example, Microsoft Word), the item is something that defines a location in the document, such as a bookmark.

Valid values for all of these arguments vary depending on the server application. A software package supporting DDE as a server should list acceptable values for the triplet information in documentation supplied with the application.

*Note:* The server application must be started before trying to communicate with it using DDE. Also, the DDE triplet format might differ among different applications and among different versions of the same application. △

For example, in order to place text into a Microsoft Word document TESTDDE.DOC located at C:\TEMP with a bookmark named NUMBER, you could use this code:

```
filename test dde 'winword|"c:\temp\testdde.doc"
                  !NUMBER' notab;
```

The application-name is **winword**, the topic is **"c:\temp\testdde.doc"**, and the range is **!NUMBER**.

This following example assumes you are using Microsoft Excel 5.0 or greater.

Suppose you want to use SAS to populate the first 4 rows and 2 columns of the Microsoft Excel spreadsheet named Sales Data stored in C:\EXCEL\SALES.XLS. You would use the following code:

```
filename test dde 'Excel|c:\excel\
                  [Sales.xls]Sales Data!R1C1:R4C2'
```

The application-name is **Excel**, the topic is **c:\excel\[Sales.xls] Sales Data**, and the range is **R1C1:R4C2**.

If your server application is able to copy the DDE-triplet to the Windows clipboard, you can display the DDE-triplet in SAS. To do this, select the information in the server application and copy it to the Windows Clipboard. Return to SAS and select

Solutions ▶ Accessories ▶ DDE triplet

### Controlling Another Application Using DDE

DDE server applications support certain commands that you can issue by using a DDE link to control the application. To use these commands, use the special topic name SYSTEM in the DDE triplet and leave the item name blank. You can then use the INPUT statement for input from an application and the PUT statement to issue commands to the server application.

For those DDE server applications that do not recognize the SYSTEM topic name, you can specify the COMMAND option in the FILENAME statement you use to define the DDE link. When you specify the COMMAND option, you do not specify the item name in the DDE triplet.

*Note:* With SAS/AF software and OLE automation, you can automate any Windows application that supports OLE 2.0 as a server. For more information about using SAS and OLE, see “Automating OLE Objects and Applications” on page 250. △

## DDE Examples

This section provides several examples of using DDE with SAS under Windows. These examples use Microsoft Excel and Microsoft Word as DDE servers, but any application that supports DDE as a server can communicate with SAS.

Before you run these examples, you must first invoke Microsoft Excel and Microsoft Word, and open the spreadsheet or document used in the example.

*Note:* DDE examples are included in the host-specific sample programs that you access from the **Help** menu. △

### Using the X Command to Open a DDE Server

A DDE server application can be opened using the X command within SAS code. The XWAIT and XSYNC options must be turned off.

```
options noxwait noxsync;
x '"c:\microsoft office\office\excel.exe"';
```

Double quotation marks are required around the path if the path contains a space. The single quotation marks are for the X command.

### Using DDE to Write Data to Microsoft Excel

The first example sends data from a SAS session to an Excel spreadsheet. The target cells are rows 1 through 100 and columns 1 through 3. To do this, submit the following program:

```
/* The DDE link is established using   */
/* Microsoft Excel SHEET1, rows 1      */
/* through 100 and columns 1 through 3 */
filename random dde
   'excel|sheet1!r1c1:r100c3';
data random;
   file random;
   do i=1 to 100;
      x=ranuni(i);
      y=10+x;
      z=x-10;
      put x y z;
   end;
run;
```

## Using DDE to Write Data to Microsoft Word

This example sends a text string to a Microsoft Word document at a given bookmark. Note the difference between using DDE with Microsoft Word and Microsoft Excel.

```
filename testit dde 'winword|"c:\temp\testing.doc"
                     !MARK' notab;

data _null_;
   file testit;
   put 'This is a test.';
run;
```

*Note:* If you are writing to Microsoft Word97, use Visual Basic commands such as FileOpen.Name, FileSave, FileClose, and Insert. If the PUT statement contains a macro that Word97 does not understand, you will see this message:

```
Ambiguous name detected:  TmpDDE
```

△

## Using DDE to Read Data from Microsoft Excel

You can also use DDE to read data from an Excel application into SAS, as in the following example:

```
/* The DDE link is established using   */
/* Microsoft Excel SHEET1, rows 1      */
/* through 10 and columns 1 through 3  */
filename monthly
  dde 'excel|sheet1!r1c1:r10c3';
data monthly;
   infile monthly;
   input var1 var2 var3;
run;
proc print;
run;
```

## Using DDE to Read Data from Microsoft Word

This example reads data from a Microsoft Word document at a given bookmark.

```
filename testit dde 'winword|"c:\temp\testing.doc"
                     !MARK' notab;

libname workdir 'c:\temp';

/* Get ready to read the first bookmark.  */
```

```
data workdir.worddata;
   length wordnum $5;
   infile testit;
   input wordnum $;
run;

proc print;
run;
```

## Using DDE and the SYSTEM Topic to Invoke Commands in an Application Using Excel

You can issue commands to Excel or other DDE-compatible programs directly from SAS using DDE. In the following example, the Excel application is invoked using the X command; a spreadsheet called SHEET1 is loaded; data are sent from SAS to Excel for row 1, column 1 to row 20, column 3; and the commands required to select a data range and sort the data are issued. The spreadsheet is then saved and the Excel application is terminated.

```
/* This code assumes that Excel       */
/* is installed on the current        */
/* drive in a directory called EXCEL. */

options noxwait noxsync;
x 'c:\microsoft office\office\excel.exe';

/* Sleep for 60 seconds to give */
/* Excel time to come up.       */

data _null_;
   x=sleep(60);
run;

/* The DDE link is established using   */
/* Microsoft Excel SHEET1, rows 1      */
/* through 20 and columns 1 through 3  */

filename data
   dde 'excel|sheet1!r1c1:r20c3';
data one;
   file data;
   do i=1 to 20;
      x=ranuni(i);
      y=x+10;
      z=x/2;
      put x y z;
   end;
run;

/* Microsoft defines the DDE topic */
/* SYSTEM to enable commands to be */
/* invoked within Excel.           */

filename cmds dde 'excel|system';
```

```
/* These PUT statements are       */
/* executing Excel macro commands */

data _null_;
   file cmds;
   put '[SELECT("R1C1:R20C3")]';
   put '[SORT(1,"R1C1",1)]';
   put '[SAVE()]';
   put '[QUIT()]';
run;
```

## Using the NOTAB Option with DDE

SAS expects to see a TAB character placed between each variable that is communicated across the DDE link. Similarly, SAS places a TAB character between variables when data are transmitted across the link. When the NOTAB option is placed in a FILENAME statement that uses the DDE device-type keyword, SAS accepts character delimiters other than tabs between variables.

The NOTAB option also can be used to store full character strings, including embedded blanks, in a single spreadsheet cell. For example, if a link is established between SAS and the Excel application, and a SAS variable contains a character string with embedded blanks, each word of the character string is normally stored in a single cell. To store the entire string, including embedded blanks in a single cell, use the NOTAB option as in the following example:

```
/* Without the NOTAB option, column1  */
/* contains 'test' and column2        */
/* contains 'one'.                    */

filename test
  dde 'excel|sheet1!r1c1:r1c2';
data string;
   file test;
   a='test one';
   b='test two';
   put a $15. b $15.;
run;

/* You can use the NOTAB option to store   */
/* each variable in a separate cell. To    */
/* do this, you must force a tab           */
/* ('09'x) between each variable, as in    */
/* the PUT statement.                      */
/* After performing this DATA step, column1*/
/* contains 'test one' and column2         */
/* contains 'test two'.                    */

filename test
  dde 'excel|sheet1!r2c1:r2c2' notab;
data string;
   file test;
   a='test one';
   b='test two';
   put a $15. '09'x b $15.;
run;
```

## Using the DDE HOTLINK

If the HOTLINK option is specified, the DDE link is activated every time the data in the specified spreadsheet range are updated. In addition, DDE enables you to poll the data when the HOTLINK option is specified to determine whether data within the range specified have been changed. If no data have changed, the HOTLINK option returns a record of 0 bytes. In the following example, row 1, column 1 of the spreadsheet SHEET1 contains the daily production total. Every time the value in this cell changes, SAS reads in the new value and outputs the observation to a data set. In this example, a second cell in row 5, column 1 is defined as a status field. Once the user completes data entry, typing any character in this field terminates the DDE link:

```
/* Enter data into Excel SHEET1 in   */
/* row 1 column 1. When you          */
/* are through entering data, place  */
/* any character in row 5            */
/* column 1, and the DDE link is     */
/* terminated.                       */

filename daily
  dde 'excel|sheet1!r1c1' hotlink;
filename status
  dde 'excel|sheet1!r5c1' hotlink;
data daily;
   infile status length=flag;
   input @;
   if flag ne 0 then stop;
   infile daily length=b;
   input @;

   /* If data have changed, then the */
   /* incoming record length         */
   /* is not equal to 0.             */

   if b ne 0 then
      do;
         input total $;
         put total=;
         output;
      end;
run;
```

It is possible to establish multiple DDE sessions. The previous example uses two separate DDE links. When the HOTLINK option is used and there are multiple cells referenced in the *item* specification, if any one of the cells changes, then all cells are transmitted.

Unless the HOTLINK option is specified, DDE is performed as a single one–time data transfer. That is, the values currently stored in the spreadsheet cells at the time that the DDE is processed are values that are transferred.

## Using the !DDE_FLUSH String to Transfer Data Dynamically

DDE also enables you to program when the DDE buffer is dumped during a DDE link. Normally, the data in the DDE buffer are transmitted when the DDE link is closed at the end of the DATA step. However, the special string **'!DDE_FLUSH'** issued in a PUT statement instructs SAS to dump the contents of the DDE buffer. This function allows you considerable flexibility in the way DDE is used, including the capacity to transfer data dynamically through the DATA step, as in the following example:

```
/* A DATA step window is displayed.   */
/* Enter data as prompted.            */
/* When you are finished, enter STOP  */
/* on the command line.               */

filename entry
   dde 'excel|sheet1!r1c1:r1c3';
dm 'pmenu off';
data entry;
   if _n_=1 then
      do;
         window ENTRY color=black
         #3 'This is data for Row 1 Column 1'
            c=cyan +2 var1 $10. c=orange
         #5 'This is data for Row 1 Column 2'
            c=cyan +2 var2 $10. c=orange
         #7 'This is data for Row 1 Column 3'
            c=cyan +2 var3 $10. c=orange;
      end;
   flsh='!DDE_FLUSH';
   file entry;
   do while (upcase(_cmd_) ne 'STOP');
      display entry;
      put var1 var2 var3 flsh;
      output;
      VAR1='';
      VAR2='';
      VAR3='';
   end;
   stop;
run;
dm 'pmenu on';
```

## Reading Missing Data

This example illustrates reading missing data from an Excel spreadsheet called SHEET1. This example reads the data in columns 1 through 3 and rows 10 through 20. Some of the data cells may be blank. Here is an example of what some of the data look like:

```
...
10   John    Raleigh      Cardinals
11   Jose    North Bend   Orioles
12   Kurt    Yelm         Red Sox
13   Brent                Dodgers
```

...

Here’s the code that can read these data correctly into a SAS data set:

```
filename mydata
  dde 'excel|sheet1!r10c1:r20c3';
data in;
   infile mydata dlm='09'x notab
          dsd missover;
   informat name $10. town $char20.
            team $char20.;
   input name town team;
run;
proc print data=in;
run;
```

In this example, the NOTAB option tells SAS not to convert tabs that are sent from the Excel application into blanks. Therefore, the tab character can be used as the delimiter between data values. The DLM= option specifies the delimiter character, and **`'09'x`** is the hexadecimal representation of the tab character. The DSD option specifies that two consecutive delimiters represent a missing value. The default delimiter is a comma. For more information about the DSD option, see *SAS Language Reference: Dictionary*. The MISSOVER option prevents a SAS program from going to a new input line if it does not find values in the current line for all the INPUT statement variables. With the MISSOVER option, when an INPUT statement reaches the end of the current record, values that are expected but not found are set to missing.

The INFORMAT statement forces the DATA step to use modified list input, which is crucial to this example. If you do not use modified list input, you receive incorrect results. The necessity of using modified list input is not DDE specific. You would need it even if you were using data in a CARDS statement, whether your data were blank- or comma-delimited.

CHAPTER
13

# Using Unnamed and Named Pipes

*Overview of Pipes* **283**
*Using Unnamed Pipes* **284**
*Introduction to Unnamed Pipes* **284**
*Unnamed Pipe Syntax* **284**
*Using Redirection Sequences* **285**
*Unnamed Pipe Example* **285**
*Using Named Pipes* **286**
*Introduction to Named Pipes* **286**
*Named Pipe Syntax* **286**
*Using the CALL RECONNECT Routine* **288**
*Using Named Pipes in SCL* **288**
*Named Pipe Examples* **288**
*Simple Named Pipes: One Client Connected to One Server* **288**
*One Server Connected to Several Clients* **289**
*The NOBLOCK Option* **291**
*The CALL RECONNECT Routine* **293**

## Overview of Pipes

A pipe is a channel of communication between two processes. A process with a handle to one end can communicate with another process that has a handle to the other end. This means that you can use a specialized Windows application to provide information to your SAS session or vice versa.

Pipes can be one-way or two-way. With a one-way pipe, one application can only write data to the pipe while the other application reads from it. With a two-way pipe, both applications can read and write data. There are two types of pipes:

*unnamed pipe*
handles one way communication. Also called an anonymous pipe (or simply pipe), it is typically used to communicate between a parent process and a child process. Within SAS, SAS is the parent process that invokes (and reads data from) a child process.

*named pipe*
handles one-way or two-way communication between two unrelated processes. That is, one process is not started by the other. In fact, it is possible to have two applications communicating over a pipe on a network. You can use named pipes within SAS to communicate with other applications or even with another SAS session.

# Using Unnamed Pipes

## Introduction to Unnamed Pipes

Unnamed pipes enable you to invoke a program outside of SAS and redirect the program's input, output, and error messages to SAS. This capability enables you to capture data from a program external to SAS without creating an intermediate data file.

For unnamed pipes to work with Windows applications external to SAS, the application program must read data from standard input (STDIN), write output to standard output (STDOUT), and write errors to standard error (STDERR). These files have numeric file handles associated with them, as follows:

| File | File Handle |
|---|---|
| STDIN | 0 |
| STDOUT | 1 |
| STDERR | 2 |

When SAS captures STDERR from another application, the error messages are routed by default to the SAS log. If you want to write to STDIN in another application, you can use a PUT statement in a SAS DATA step. Because SAS can write to STDIN and capture from STDOUT in the same application, unnamed pipes can be used to send data to an external program, as well as to capture the output and error messages of the same program. You can use redirection sequences to redirect STDIN, STDOUT, and STDERR.

When you start SAS from the Windows desktop, STDIN and STDOUT are not available to your programs.

For more information, see "Using Redirection Sequences" on page 285 or your Windows documentation.

## Unnamed Pipe Syntax

To use an unnamed pipe, issue a FILENAME statement with the following syntax:

**FILENAME** *fileref* PIPE '*program-name*' *option-list*;

You can use the following arguments with this syntax of the FILENAME statement:

*fileref*
: is any valid fileref, as described in "Referencing External Files" on page 146.

PIPE
: is the device-type keyword that tells SAS you want to use an unnamed pipe.

*program-name*
: specifies the external Windows application program. This argument must fully specify the pathname to the program, or the path to the directory containing the program must be contained in the Windows PATH environment variable. This argument can also contain program options. For example, you can specify the

following argument to indicate you want to invoke the STOCKMKT program on all stocks:

```
'stockmkt.exe -all'
```

*option-list*
can be any of the options valid in the FILENAME statement, such as the LRECL= or RECFM= options. For a complete list of options available for the FILENAME statement under Windows, see .

## Using Redirection Sequences

Any Windows application that accommodates standard input, output, and error commands can use the unnamed pipe feature. Because many Windows system commands use standard input, output, and error commands, you can use these commands with unnamed pipes within SAS. Unless you specify otherwise, an unnamed pipe directs STDOUT and STDERR to two different files. To combine the STDOUT and STDERR into the same file, use redirection sequences. The following is an example that redirects STDERR to STDOUT for the Windows DIR command:

```
filename listing pipe 'dir *.sas 2>&1';
```

In this example, if any errors occur in performing this command, STDERR (2) is redirected to the same file as STDOUT (1). This is an example of SAS's ability to capitalize on operating environment capabilities. This feature of redirecting file handles is a function of the Windows operating system rather than of SAS.

## Unnamed Pipe Example

In the following example, you use the unnamed pipes feature of SAS under Windows to produce some financial reports. The example assumes you have a stand-alone program that updates stock market information from a financial news bureau. You need SAS to invoke a stock market report with the most recently created data from the stock market program. The following is how you create and use the pipe within your SAS session:

```
filename stocks pipe 'stockmkt.exe -all' console=min;
data report;
   infile stocks;
   input stock $ open close change;
run;
proc print;
   var stock open close change;
   sum change;
   title 'Stock Market Report';
run;
```

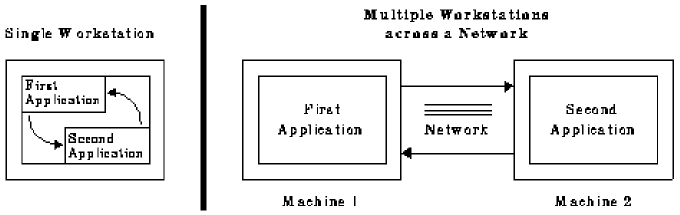
In this example, the PIPE device-type keyword in the FILENAME statement indicates that the fileref STOCKS is an unnamed pipe. The STOCKMKT.EXE reference is the name of the stand-alone program that generates the stock market data. The host-option CONSOLE=MIN indicates that the command prompt window that is opened to run the STOCKMKT.EXE program is opened minimized. The INFILE statement causes SAS to invoke the STOCKMKT.EXE program and read the data in the pipe from it. The STOCKMKT.EXE program completes without you being aware that it has been implemented (except for the command prompt window button on the Windows task bar). Because the fileref STOCKS has already been defined as an unnamed pipe, the standard output from STOCKMKT.EXE is redirected to SAS and captured through the INFILE statement. The SAS program reads in the variables and uses the PRINT procedure to generate a printed report. Any error messages generated by STOCKMKT.EXE appear in the SAS log.

## Using Named Pipes

### Introduction to Named Pipes

The *named pipes* capability is one of the most powerful tools available in SAS under Windows for communicating with other applications. The named pipes feature enables bidirectional data or message exchange between applications on the same machine or applications on separate machines across a network. The following figure illustrates these two methods of communication.

**Figure 13.1** Communication Using Named Pipes



The applications can be SAS sessions or other Windows applications. For example, you can use the PRINTTO procedure to direct the results from SAS procedures to another Windows application, using a named pipe. Therefore, you have the choice of having multiple SAS sessions that communicate with each other or one SAS session communicating with another Windows application.

Whether you are communicating between multiple SAS sessions or between a SAS session and another Windows application that supports named pipes, the pipes are defined in a client/server relationship. One process is defined as the *server*, while one or more other processes are defined as *clients*. In this configuration, you can have multiple clients send data to the server or the server send data to the various clients. Named pipes enable you to coordinate processing between the server and clients using various options.

### Named Pipe Syntax

You can use a named pipe anywhere you use a fileref in SAS. To use a named pipe, issue a FILENAME statement with the following syntax:

**FILENAME** *fileref* NAMEPIPE '*pipe-specification*' <*named-pipe-options*>;

You can use the following arguments with this syntax of the FILENAME statement:

*fileref*
is any valid fileref as described in "Referencing External Files" on page 146.

NAMEPIPE
: is the device-type keyword that tells SAS you want to use a named pipe.

*pipe-specification*
: is the name of the pipe.

This argument has two mutually exclusive syntaxes:

\\.\PIPE*pipe-name*
: indicates you are establishing a pipe on a single PC or defining a server pipe across a network. The *pipe-name* argument specifies the name of the pipe.

\\*server-name*\PIPE*pipe-name*
: indicates you are establishing a client pipe over a network named-pipe server. Remember to include the double backslash (\\) in this situation. The *pipe-name* argument specifies the name of the client pipe. The *server-name* argument specifies the name of the named-pipe server.

*named-pipe-options*
: can be any of the following. The default value is listed first:

SERVER | CLIENT
: indicates the mode of the pipe. SERVER is the default.

BLOCK | NOBLOCK
: indicates whether the client or server is to wait for data to be read if no data are currently available. BLOCK indicates to wait and is the default. NOBLOCK indicates not to wait. Control is returned immediately to the program if no data are available in the pipe. Writing to the pipe always implies BLOCK.

BYTE | MESSAGE
: indicates the type of pipe. BYTE is the default. The difference between a BYTE pipe and a MESSAGE pipe is that a MESSAGE pipe includes an encoded record length, whereas a BYTE pipe does not.

RETRY=*seconds*
: indicates the amount of time the client or server is to wait to establish the pipe. The minimum value for *seconds* is 10. This option allows time for synchronization of the client and server. The default waiting period is 10 seconds.

There are two values for the *seconds* argument that indicate special cases:

| | |
|---|---|
| -2 | indicates that the client is to wait the amount of time defined by the server's RETRY= option. If this option is used, the SERVER must always be active or the pipe connection fails. |
| -1 | indicates that the client or the server is to wait indefinitely for the pipe connection. |

EOFCONNECT
: is valid only when defining the server and indicates that if an end-of-file (EOF) is received from a client, the server is to try to connect to the next client.

All of these options are consistent with terminology used in Windows programmers' reference guides such as those provided with the Microsoft Win32 SDK.

## Using the CALL RECONNECT Routine

A special SAS CALL routine, CALL RECONNECT, enables the server to disconnect the current client and try to connect to the next available client. Normally, a pipe terminates when the client side of the pipe sends an end-of-file to the server. To break the pipe connection at any time, the server SAS session can issue a CALL RECONNECT statement. For an illustration of this routine, see “The CALL RECONNECT Routine” on page 293.

## Using Named Pipes in SCL

To establish named pipes using SCL code, you must use the FOPEN function to open a file (or pipe) before you can access it. In doing so, you must specify the appropriate open mode for both the client and server applications so that the two can communicate over the pipe. Here is a summary of the different nodes you can use:

| If the server accesses the pipe as... | then the client must access it as... |
|---|---|
| I (input) | O (output) |
| O (output) | S (sequential) |
| U (update) | O (output) or S (sequential) |

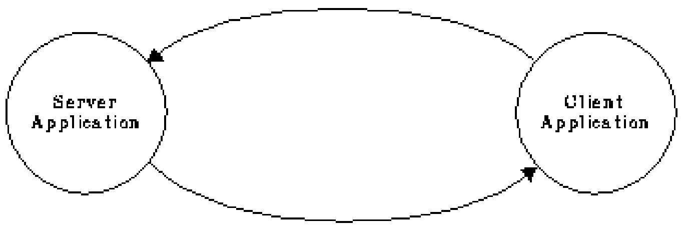
## Named Pipe Examples

The best way to understand named pipes is to examine several different examples illustrating their use. In most of the examples in this section, the named pipe is established between two SAS sessions. However, named pipes work between SAS and other applications that support named pipes.

### Simple Named Pipes: One Client Connected to One Server

The simplest named pipe configuration is one server connected to one client, as shown in Figure 13.2 on page 288.

**Figure 13.2** One Server Connected to One Client



In the following example, a named pipe called WOMEN is established between two SAS sessions. The server SAS session selectively sends data to the client SAS session. You can start the server or the client first; one waits 30 seconds for the other to connect.

In the first SAS session, create a named pipe as a server:

```
   /* Creates a pipe called WOMEN, acting */
   /* as a server. The server waits 30    */
   /* seconds for a client to connect.    */
filename women namepipe '\\.\pipe\women'
         server retry=30;
   /* This code writes three records into */
   /* the named pipe called WOMEN.        */
data class;
   input name $ sex $ age;
   file women;
   if upcase(sex)='F' then
      put name age;
   datalines;
MOORE M 15
JOHNSON F 16
DALY F 14
ROBERTS M 14
PARKER F 13
;
```

In the second SAS session, you can use SAS statements to exchange data between the two SAS sessions. For example, you can submit the following program from the client session:

```
   /* Creates a pipe called WOMEN, acting */
   /* as a client.  The client waits 30   */
   /* seconds for a server to connect.    */
filename in namepipe '\\.\pipe\women' client
         retry=30;
data female;
   infile in;
   input name $ age;
proc print;
run;
```

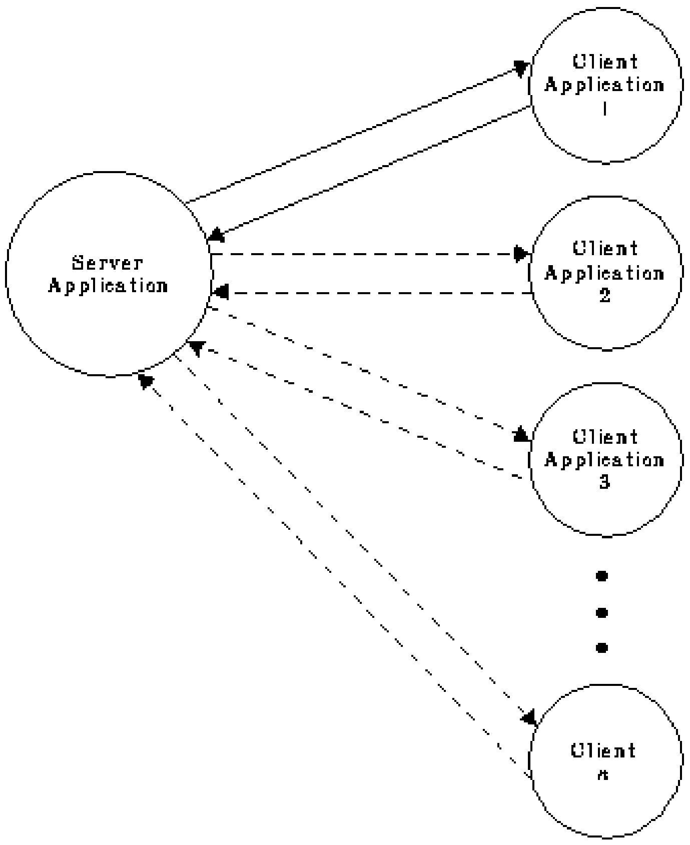
The following program is another example of a single client and server. This example illustrates using the PRINTTO procedure to direct results from the SUMMARY procedure to another Windows application, using a named pipe called RESULTS:

```
filename results namepipe '\\.\pipe\results'
     server retry=60;
proc printto print=results new;
run;
proc summary data=monthly;
run;
```

## One Server Connected to Several Clients

You can choose to have one server connected to several clients. In this case, the named pipe configuration looks like that shown in Figure 13.3 on page 290.

**Figure 13.3** One Server Connected to Several Clients



In this configuration, the data connection is initially between the server and the first client. When this connection is terminated, the server connects to the second client, and so on. The connection can return to the first client after the last client's connection is broken if your program is set up to do so.

You must use the EOFCONNECT option to cause the connection to move properly from one client to the next. Here is an example of using the EOFCONNECT option with one server SAS session and two clients. The clients can be on the same PC or on a PC connected across a network.

In the first SAS session, submit the following statements:

```
   /* Creates a pipe called SALES, acting  */
   /* as a server. The server waits 30     */
   /* seconds for a client to connect.     */
   /* After the client has disconnected,   */
   /* this server SAS session tries to     */
   /* connect to the next available client */
filename daily namepipe '\\.\pipe\sales'
         server eofconnect retry=30;
   /* This program reads in the daily     */
   /* sales figures sent from each client.*/
data totsales;
   infile daily;
   input dept $ item $ total;
run;
```

In the second SAS session, submit the following statements:

```
  /* Creates a pipe called SALES, acting   */
  /* as a client. The client waits forever */
  /* for a server to connect. After the    */
  /* first client has disconnected, the    */
  /* second client connects with the server.*/
  /* The first client is the TOYS dept.    */
filename dept1 namepipe '\\.\pipe\sales'
         client retry=-1;
data toys;
   input item $ total;
   dept='TOYS';
   file dept1;
   put dept item total;
   datalines;
DOLLS 100
MARBLES 10
BLOCKS 50
GAMES 60
CARS 40
;
  /* The second client is the SPORTS dept.*/
  /* These data could come from a separate */
  /* SAS session.                          */
filename dept2 namepipe '\\.\pipe\sales'
         client retry=-1;
data sports;
   input item $ total;
   dept='SPORTS';
   file dept2;
   put dept item total;
   datalines;
BALLS 30
BATS 65
GLOVES 15
RACKETS 75
FISHING 20
TENTS 115
HELMETS 45
;
```

## The NOBLOCK Option

In the following example, the NOBLOCK option is used to specify that if no data are available when the pipe is read, then the program should continue performing. If the default value of BLOCK had been used, then the pipe would wait indefinitely until data were found in the pipe. The EOFCONNECT option is used to tell the server that when a client sends an end-of-file, the server can connect with a new client. The RETRY= option tells the server to look for any new clients for 20 seconds while the client waits indefinitely on a server. The clients can be on the same PC or on a PC connected across a network. A server connects to one client at a time, and the clients queue in a serial order waiting to connect to the server.

In the first SAS session, submit the following statements:

```
   /* Defines a named pipe called LINE.   */
   /* Use the NOBLOCK option to specify   */
```

```
      /* that if no data are available when  */
      /* the read is performed, then continue.*/
      /* Use the EOFCONNECT option to tell   */
      /* the server to try to connect with a */
      /* new client if an end-of-file is     */
      /* encountered. Use the RETRY= option  */
      /* to tell the server to look for any  */
      /* new clients for 20 seconds.         */
   filename data namepipe '\\.\pipe\line' server
            noblock eofconnect retry=20;
      /* This DATA step reads in all data  */
      /* from any clients connected to the */
      /* named pipe called LINE.           */
   data all;
      infile data length=len;
      input @;
         /* If the length of the incoming  */
         /* record is 0, then no data were */
         /* found in the pipe; otherwise,  */
         /* read the incoming data.        */
      if len ne 0 then
         do;
            input machine $ width weight;
            output;
         end;
   run;
   proc print;
   run;
```

Each of the following DATA steps below can be carried out on several different PCs connected across a network:

```
      /* Defines a named pipe called LINE.  */
      /* The RETRY= option is set such that */
      /* the clients wait forever until a   */
      /* server is available                */
      /* (that is, RETRY=-1).               */
      filename data namepipe '\\.\pipe\line'
               client retry=-1;
      /* This is information from the */
      /* first machine/client.        */
   data machine1;
      file data;
      input width weight;
      machine='LINE_1';
      put machine width weight;
      datalines;
   5.3 18.2
   3.2 14.3
   4.8 16.9
   6.4 20.8
   4.3 15.4
   6.1 19.5
   5.6 18.9
   ;
      /* This is information from the */
      /* second machine/client.       */
   filename data namepipe '\\.\pipe\line'
            client retry=-1;
   data machine2;
      file data;
      input width weight;
      machine='LINE_2';
      put machine width weight;
      datalines;
   4.3 17.2
   5.2 18.4
   6.8 19.9
   3.4 14.5
   5.3 18.6
   4.1 17.1
   6.6 19.5
   ;
```

## The CALL RECONNECT Routine

The following example demonstrates how to set up a named pipe server to establish a connection with two clients. (For this example, you need three active SAS sessions.) In this example, the CALL RECONNECT routine is used to reconnect to the next client on the named pipe if it has been at least 30 seconds since the previous client has sent any data. Each client is a data entry operator, sending data to the server SAS session.

In the server SAS session, submit the following statements:

```
filename data namepipe '\\.\pipe\orders'
         server noblock eofconnect retry=30;
data all;
   infile data length=len missover;
   input @;
      /* If the length of the incoming  */
      /* record is 0, then no data were */
      /* found in the pipe; otherwise,  */
      /* read the incoming data         */
   if len ne 0 then
      do;
         input operator $ item $ quantity $;
         if item='' or quantity='' then
            delete;
         else
            output;
         put operator= item= quantity=;
      end;
      /* If no data are being transmitted,*/
      /* try reconnecting to the next     */
      /* available client.                */
   else
      do;
            /* Use the named pipe fileref */
            /* as the argument of */
            /* CALL RECONNECT. */
         call reconnect('data');
      end;
   run;
```

In the second SAS session, which is the first data entry operator, submit the following statements:

```
filename data namepipe '\\.\pipe\orders'
         client retry=-1;
data entry1;
   if _n_=1 then
     do;
        window entry_1
         #1 @2 'ENTER STOP WHEN YOU ARE FINISHED'
         #3 @5 'ITEM NUMBER - ' item $3.
         #5 @5 'QUANTITY - ' quantity $3.;
     end;
     do while (upcase(_cmd_) ne 'STOP');
        display entry_1;
        file data;
        put 'ENTRY_1' +1 item quantity;
        item='';
        quantity='';
     end;
   stop;
run;
```

In the third SAS session, which is the second data entry operator, submit the following statements:

```
filename data namepipe '\\.\pipe\orders'
         client retry=-1;
data entry2;
   if _n_=1 then
     do;
       window entry_2
        #1 @2 'ENTER STOP WHEN YOU ARE FINISHED'
        #3 @5 'ITEM NUMBER - ' item $3.
        #5 @5 'QUANTITY - ' quantity $3.;
     end;
     do while (upcase(_cmd_) ne 'STOP');
        display entry_2;
        file data;
        put 'ENTRY_2' +1 item quantity;
        item='';
        quantity='';
     end;
   stop;
run;
```

# CHAPTER 14 Accessing External DLLs from SAS

*Overview of Dynamic Link Libraries in SAS* **295**
*The SASCBTBL Attribute Table* **296**
  *Syntax of the Attribute Table* **296**
    *ROUTINE Statement* **296**
    *ARG Statement* **299**
  *The Importance of the Attribute Table* **300**
*Special Considerations When Using External DLLs* **301**
  *Using PEEKLONG Functions to Access Character String Arguments* **301**
  *Accessing External DLLs Efficiently* **302**
  *Grouping SAS Variables as Structure Arguments* **303**
  *Using Constants and Expressions as Arguments to MODULE* **304**
  *Specifying Formats and Informats to Use with MODULE Arguments* **305**
    *C Language Formats* **305**
    *FORTRAN Language Formats* **305**
    *PL/I Language Formats* **306**
    *COBOL Language Formats* **306**
    *$CSTRw. Format* **307**
    *$BYVALw. Format* **307**
  *Understanding MODULE Log Messages* **308**
*Examples* **310**
  *Updating a Character String Argument* **310**
  *Passing Arguments by Value* **311**
  *Using PEEKCLONG to Access a Returned Pointer* **311**
  *Using Structures* **312**
  *Invoking a DLL Routine from PROC IML* **313**

## Overview of Dynamic Link Libraries in SAS

Dynamic link libraries (DLLs) are executable files that contain one or more routines written in any of several programming languages. DLLs are a mechanism for storing useful routines that might be needed by many applications. When an application needs a routine that resides in a DLL, it loads the DLL, invokes the routine, and unloads the DLL upon completion. SAS provides routines and functions that let you invoke these external routines from within SAS. You can access the DLL routines from the DATA step, the IML procedure, and SCL code. You use the MODULE family of SAS call routines and functions (including MODULE, MODULEN, MODULEC, MODULEI, MODULEIN, and MODULEIC) to invoke a routine that resides in an external DLL. This documentation refers to the MODULE family of call routines and functions generically as the MODULE functions.

These are general steps for accessing an external DLL routine:

1 Create a text file that describes the DLL routine you want to access, including the arguments it expects and the values it returns (if any). This attribute file must be in a special format, as described in “The SASCBTBL Attribute Table” on page 296.
2 Use the FILENAME statement to assign the SASCBTBL fileref to the attribute file you created.
3 In a DATA step or SCL code, use a call routine or function (MODULE, MODULEN, or MODULEC) to invoke the DLL routine. The specific function you use depends on the type of expected return value (none, numeric, or character). (You can also use MODULEI, MODULEIN, or MODULEIC within a PROC IML step.) The MODULE functions are described in “MODULE Function” on page 406.

***CAUTION:***
**Only experienced programmers should access external DLLs.** By accessing a function in an external DLL, you transfer processing control to the external function. If done improperly, or if the external function is not reliable, you might lose data or have to reset your computer (or both). △

## The SASCBTBL Attribute Table

Because the MODULE routine invokes an external function that SAS knows nothing about, you must supply information about the function’s arguments so that the MODULE routine can validate them and convert them, if necessary. For example, suppose you want to invoke a routine that requires an integer as an argument. Because SAS uses floating point values for all of its numeric arguments, the floating point value must be converted to an integer before you invoke the external routine. The MODULE routine looks for this attribute information in an attribute table referred to by the SASCBTBL fileref.

The attribute table is a sequential text file that contains descriptions of the routines you can invoke with the MODULE function. The function of the table is to define how the MODULE function should interpret its supplied arguments when building a parameter list to pass to the called DLL routine.

The MODULE routines locate the table by opening the file referred to by the SASCBTBL fileref. If you do not define this fileref, the MODULE routines simply call the requested DLL routine without altering the arguments.

***CAUTION:***
**Using the MODULE functions without defining an attribute table can cause SAS to crash or force you to reset your computer.** You need to use an attribute table for all external functions that you want to invoke. △

The attribute table should contain a description for each DLL routine you intend to call (using a ROUTINE statement) plus descriptions of each argument associated with that routine (using ARG statements).

### Syntax of the Attribute Table

At any point in the attribute table file, you can create a comment using an asterisk (*) as the first nonblank character of a line or after the end of a statement (following the semicolon). You must end the comment with a semicolon.

#### ROUTINE Statement

The following is the syntax of the ROUTINE statement:

**ROUTINE** *name* MINARG=*minarg* MAXARG=*maxarg*
<CALLSEQ=BYVALUE | BYADDR>
<STACKORDER=R2L | L2R>
<STACKPOP=CALLER | CALLED>
<TRANSPOSE=YES | NO> <MODULE=*DLL-name*>
<RETURNS=SHORT | USHORT | LONG | ULONG
| DOUBLE | DBLPTR | CHAR<*n*>>
<RETURNREGS=DXAX>;

The following are descriptions of the ROUTINE statement attributes:

ROUTINE *name*

starts the ROUTINE statement. You need a ROUTINE statement for every DLL function you intend to call using the MODULE function. The value for *name* must match the routine name or ordinal you specified as part of the '*module*' argument in the MODULE function, where *module* is the name of the DLL (if not specified by the MODULE attribute, described later) and the routine name or ordinal. For example, to be able to specify **`KERNEL32,GetPath`** in the MODULE function call, the ROUTINE *name* should be **`GetPath`**.

The *name* argument is case sensitive, and is required for the ROUTINE statement.

MINARG=*minarg*

specifies the minimum number of arguments to expect for the DLL routine. In most cases, this value will be the same as MAXARG; but some routines do allow a varying number of arguments. This is a required attribute.

MAXARG=*maxarg*

specifies the maximum number of arguments to expect for the DLL routine. This is a required attribute.

CALLSEQ=BYVALUE | BYADDR

indicates the calling sequence method used by the DLL routine. Specify BYVALUE for call-by-value and BYADDR for call-by-address. The default value is BYADDR.

FORTRAN and COBOL are call-by-address languages; C is usually call-by-value, although a specific routine might be implemented as call-by-address.

The MODULE routine does not require that all arguments use the same calling method; you can identify any exceptions by using the BYVALUE and BYADDR options in the ARG statement, described later in this section.

STACKORDER=R2L | L2R

indicates the order of arguments on the stack as expected by the DLL routine. R2L places the arguments on the stack according to C language conventions. The last argument (right-most in the call syntax) is pushed first, the next-to-last argument is pushed next, and so on, so that the first argument is the first item on the stack when the external routine takes over. R2L is the default value.

L2R places the arguments on the stack in reverse order, pushing the first argument first, the second argument next, and so on, so that the last argument is the first item on the stack when the external routine takes over. Pascal uses this calling convention, as do some C routines.

STACKPOP=CALLER | CALLED

specifies which routine, the caller routine or the called routine, is responsible for popping the stack (updating the stack pointer) upon return from the routine. The default value is CALLER (that is, the code that calls the routine). Some routines, such as those that use Microsoft's __stdcall attribute with 32-bit compilers, require the called routine to pop the stack.

TRANSPOSE=YES | NO

specifies whether to transpose matrices with both more than one row and more than one column before calling the DLL routine. This attribute applies only to routines called from within PROC IML with MODULEI, MODULEIC, and MODULEIN.

TRANSPOSE=YES is necessary when calling a routine written in a language that does not use row-major order to store matrices. (For example, FORTRAN uses column-major order.)

For example, consider this matrix with three columns and two rows:

```
           columns
           1   2   3
       -------------
rows   1 | 10  11  12
       2 | 13  14  15
```

PROC IML stores this matrix in memory sequentially as 10, 11, 12, 13, 14, 15. However, FORTRAN routines will expect this matrix as 10, 13, 11, 14, 12, 15.

The default value is NO.

MODULE=*DLL-name*

names the executable module (the DLL) in which the routine resides. The MODULE function searches the directories named by the PATH environment variable. If you specify the MODULE attribute here in the ROUTINE statement, then you do not need to include the module name in the *module* argument to the MODULE function (unless the DLL routine name you are calling is not unique in the attribute table). The MODULE function is described in "MODULE Function" on page 406.

You can have multiple ROUTINE statements that use the same MODULE name. You can also have duplicate ROUTINE names that reside in different DLLs.

RETURNS=SHORT | USHORT | LONG | ULONG | DOUBLE | DBLPTR | CHAR<*n*>

specifies the type of value that the DLL routine returns. This value will be converted as appropriate, depending on whether you use MODULEC (which returns a character) or MODULEN (which returns a number). The possible return value types are

SHORT
: short integer

USHORT
: unsigned short integer

LONG
: long integer

ULONG
: unsigned long integer

DOUBLE
: double-precision floating point number

DBLPTR
: a pointer to a double-precision floating point number (instead of using a floating point register). Consult the documentation for your DLL routine to determine how it handles double-precision floating point values.

CHAR*n*
: pointer to a character string up to *n* bytes long. The string is expected to be null-terminated and will be blank-padded or truncated as appropriate. If you

do not specify *n*, the MODULE function uses the maximum length of the receiving SAS character variable.

If you do not specify the RETURNS attribute, you should invoke the routine with only the MODULE and MODULEI call routines. You will get unpredictable values if you omit the RETURNS attribute and invoke the routine using the MODULEN/MODULEIN or MODULEC/MODULEIC functions.

The ROUTINE statement must be followed by as many ARG statements as you specified in the MAXARG= option. The ARG statements must appear in the order that the arguments will be specified within the MODULE routines.

## ARG Statement

The syntax for each ARG statement is

**ARG** *argnum* NUM|CHAR <INPUT|OUTPUT|UPDATE> <NOTREQD|REQUIRED> <BYADDR|BYVALUE> <FDSTART> <FORMAT=*format*>;

Here are the descriptions of the ARG statement attributes:

ARG *argnum*
: defines the argument number. This a required attribute. Define the arguments in ascending order, starting with the first routine argument (ARG 1).

NUM | CHAR
: defines the argument as numeric or character. This is a required attribute.

If you specify NUM here but pass the routine a character argument, the argument is converted using the standard numeric informat. If you specify CHAR here but pass the routine a numeric argument, the argument is converted using the BEST12 informat.

INPUT | OUTPUT | UPDATE
: indicates the argument is either input to the routine, an output argument, or both. If you specify INPUT, the argument is converted and passed to the DLL routine. If you specify OUTPUT, the argument is *not* converted, but is updated with an outgoing value from the DLL routine. If you specify UPDATE, the argument is converted and passed to the DLL routine *and* updated with an outgoing value from the routine.

You can specify OUTPUT and UPDATE only with variable arguments (that is, no constants or expressions).

NOTREQD | REQUIRED
: indicates if the argument is required. If you specify NOTREQD, then MODULE can omit the argument. If other arguments follow the omitted argument, indicate the omitted argument by including an extra comma as a placeholder. For example, to omit the second argument to routine XYZ, you would specify:

```
call module('XYZ',1,,3);
```

***CAUTION:***

**Be careful when using NOTREQD; the DLL routine must not attempt to access the argument if it is not supplied in the call to MODULE. If the routine does attempt to access it, your system is likely to crash.** △

The REQUIRED attribute indicates that the argument is required and cannot be omitted. REQUIRED is the default value.

BYADDR | BYVALUE
: indicates the argument is passed by reference or by value.

BYADDR is the default value unless CALLSEQ=BYVALUE was specified in the ROUTINE statement, in which case BYVALUE is the default. Specify BYADDR when using a call-by-value routine that also has arguments to be passed by address.

FDSTART
: indicates that the argument begins a block of values that is grouped into a structure whose pointer is passed as a single argument. Note that all subsequent arguments are treated as part of that structure until the MODULE function encounters another FDSTART argument.

FORMAT=*format*
: names the format that presents the argument to the DLL routine. Any SAS Institute-supplied formats, PROC FORMAT-style formats, or SAS/TOOLKIT formats are valid. Note that this format must have a corresponding valid informat if you specified the UPDATE or OUTPUT attribute for the argument.

The FORMAT= attribute is not required, but is recommended, since format specification is the primary purpose of the ARG statements in the attribute table.

***CAUTION:***
**Using an incorrect format can produce invalid results or cause a system crash.** △

## The Importance of the Attribute Table

MODULE routines rely heavily on the accuracy of the information in the attribute table. If this information is incorrect, unpredictable results can occur (including a system crash).

Consider an example routine **xyz** that expects two arguments: an integer and a pointer. The integer is a code indicating what action takes place. For example, action 1 means that a 20-byte character string is written into the area pointed to by the second argument, the pointer.

Now suppose you call **xyz** using the MODULE routine but indicating in the attribute table that the receiving character argument is only 10 characters long:

```
routine xyz minarg=2 maxarg=2;
arg 1 input num byvalue format=ib4.;
arg 2 output char format=$char10.;
```

Regardless of the value given by the LENGTH statement for the second argument to MODULE, MODULE passes a pointer to a 10-byte area to the **xyz** routine. If **xyz** writes 20 bytes at that location, the 10 bytes of memory following the string provided by MODULE are overwritten, causing unpredictable results:

```
data _null_;
    length x $20;
    call module('xyz',1,x);
    run;
```

The call might work fine, depending on which 10 bytes were overwritten. However, this might also cause you to lose data or cause your system to crash.

Also, note that the PEEKLONG and PEEKCLONG functions rely on the validity of the pointers you supply. If the pointers are invalid, it is possible that SAS could crash. For example, this code would cause a crash:

```
data _null_;
   length c $10;
     /* trying to copy from address 0!!!*/
   c = peekclong(0,10);
```

```
run;
```

Be sure that your pointers are valid when using PEEKLONG and PEEKCLONG.

## Special Considerations When Using External DLLs

### Using PEEKLONG Functions to Access Character String Arguments

Because the SAS language does not provide pointers as data types, you must use the PIB4. format/informat to represent pointers. You can then use the SAS PEEKLONG functions to access the data stored at these address values.

For example, suppose you have a routine named GetPath in a library named SERVICES.DLL. It has two arguments, an integer function code and a pointer to a pointer. The function code determines what action GetPath will take, and the second argument points to a pointer that will be updated by GetPath to refer to a system character string. The calling code in C might be

```
GetPath(1,&stgptr);
printf("GetPath indicates string is
   '%s'.\n",stgptr);
```

Using MODULE, the corresponding attribute table entry would be

```
ROUTINE GetPath MINARG=2 MAXARG=2
  MODULE=SERVICES;
ARG 1 NUM INPUT BYVALUE FORMAT=PIB4.;
ARG 2 NUM OUTPUT BYADDR FORMAT=PIB4.;
```

and could be invoked as follows:

```
call module('GetPath',1,stgptr);
put stgptr= stgptr=hex8.;
```

If the pointer value in STGPTR is 0035F780, STGPTR would actually be set to the decimal value 3536768, which is the decimal equivalent of 0035F780. So the PUT statement above would produce:

```
STGPTR=3536768 STGPTR=0035F780
```

However, you want the data at address 0035F780, not the value of the pointer itself. To access that data, you need to use the PEEKCLONG function.

The PEEKCLONG function is given two arguments, a pointer via a numeric variable (such as STGPTR above) and a length in bytes (characters). PEEKCLONG returns a character string of the specified length containing the characters at the pointer location.

In the example, suppose that GetPath sets the second argument's pointer value to the address of the null-terminated character string C:\XYZ. You can access the character data with:

```
call module('SERVICES,GetPath',1,stgptr);
length path $64;
path = peekclong(stgptr,64);
i = index(path,'00'x);
if i then substr(path,i)=' ';
/* path now contains the string */
```

The PEEKCLONG function copies 64 bytes starting at the location referred to by the pointer in STGPTR. Because you need only the data up to the null terminator (but not including it), you search for the null terminator with the INDEX function, then blank out all characters including and after that point.

You can also use the $CSTR format in this scenario to simplify your code slightly:

```
call module('SERVICES,GetPath',1,stgptr);
length path $64;
path = put(peekclong(stgptr,64),$cstr64.);
```

The $CSTR format accepts as input a character string of a specified width. It looks for a null terminator and pads the output string with blanks from that point.

For more information, see the PEEKLONG function in *SAS Language Reference: Dictionary* and "$CSTR*w*. Format" on page 307.

## Accessing External DLLs Efficiently

The MODULE routine reads the attribute table referenced by the SASCBTBL fileref once per step (DATA step, PROC IML step, or SCL step). MODULE parses the table and stores the attribute information for future use during the step. When you use a MODULE function, SAS searches the stored attribute information for the matching routine and module names. The first time you access a DLL during a step, SAS loads the DLL and determines the address of the requested routine. Each DLL you invoke stays loaded for the duration of the step, and is not reloaded in subsequent calls. All modules and routines are unloaded at the end of the step. For example, suppose the attribute table had the basic form:

```
* routines XYZ and BBB in FIRST.DLL;
ROUTINE XYZ MINARG=1 MAXARG=1 MODULE=FIRST;
ARG 1 NUM INPUT;
ROUTINE BBB MINARG=1 MAXARG=1 MODULE=FIRST;
ARG 1 NUM INPUT;
* routines ABC and DDD in SECOND.DLL;
ROUTINE ABC MINARG=1 MAXARG=1 MODULE=SECOND;
ARG 1 NUM INPUT;
ROUTINE DDD MINARG=1 MAXARG=1 MODULE=SECOND;
ARG 1 NUM INPUT;
```

and the DATA step looked like:

```
filename sascbtbl 'myattr.tbl';
data _null_;
   do i=1 to 50;
      /* FIRST.DLL is loaded only once */
      value = modulen('XYZ',i);
      /* SECOND.DLL is loaded only once */
      value2 = modulen('ABC',value);
      put i= value= value2=;
   end;
run;
```

In this example, MODULEN parses the attribute table during DATA step compilation. In the first loop iteration (i=1), FIRST.DLL is loaded and the XYZ routine is accessed when MODULEN calls for it. Next, SECOND.DLL is loaded and the ABC routine is accessed. For subsequent loop iterations (starting when i=2), FIRST.DLL and SECOND.DLL remain loaded, so the MODULEN function simply accesses the XYZ and ABC routines. SAS unloads both DLLs at the end of the DATA step.

Note that the attribute table can contain any number of descriptions for routines that are not accessed for a given step. This does not cause any additional overhead (apart from a few bytes of internal memory to hold the attribute descriptions). In the above example, BBB and DDD are in the attribute table but are not accessed by the DATA step.

## Grouping SAS Variables as Structure Arguments

A common need when calling external routines is to pass a pointer to a structure. Some parts of the structure might be used as input to the routine, while other parts might be replaced or filled in by the routine. Even though SAS does not have structures in its language, you can indicate to MODULE that you want a particular set of arguments grouped into a single structure. You indicate this by using the FDSTART option of the ARG statement to flag the argument that begins the structure in the attribute table. SAS gathers that argument and all that follow (until encountering another FDSTART option) into a single contiguous block, and passes a pointer to the block as an argument to the DLL routine.

For example, consider the GetClientRect routine, which is part of the Win32 API in USER32.DLL. This routine retrieves the coordinates of a window's client area. This also requires the use of another routine, GetForegroundWindow, to get the window handle for the window you want the coordinates from.

The C prototypes for these routines are

```
HWND GetForegroundWindow(VOID);
BOOL GetClientRect(HWND hWnd, LPRECT lprc);
```

In C, the code to invoke them is:

```
typedef struct tagRECT {
    int left;
    int top;
    int right;
    int bottom;
    } RECT;
  /* RECT is a structure variable */
....                  /* other code */
/* Need the window handle first */
hWnd=GetForegroundWindow();
/* Function call, passing the address */
/* of RECT                            */
GetClientRect(hWnd, &RECT);
```

To call these routines using MODULE, you would use the following attribute table entries:

```
routine GetForegroundWindow
  minarg=0
  maxarg=0
  stackpop=called
  module=USER32
  returns=long;
routine GetClientRect
  minarg=5
  maxarg=5
  stackpop=called
  module=USER32;
arg 1 num input byvalue format=pib4.;
arg 2 num update fdstart format=ib4.;
arg 3 num update         format=ib4.;
arg 4 num update         format=ib4.;
arg 5 num update         format=ib4.;
```

with the following DATA step:

```
filename sascbtbl 'sascbtbl.dat';
data _null_;
  hwnd=modulen('GetForegroundWindow');
  call module('GetClientRect',hwnd,left,
       top,right,bottom);
  put left= top= right= bottom=;
run;
```

The use of the FDSTART option in the ARG statement for argument 2 indicates that argument 2 and all subsequent arguments are to be gathered together into a single parameter block.

The output in the log from the PUT statement would look like:

```
LEFT=2 TOP=2 RIGHT=400 BOTTOM=587
```

## Using Constants and Expressions as Arguments to MODULE

You can pass any kind of expression as an argument to the MODULE functions. The attribute table indicates whether the argument is for input, output, or update.

You can specify input arguments as constants and arithmetic expressions. However, because output and update arguments must be able to be modified and returned, you can pass only a variable for these parameters. If you specify a constant or expression where a value that can be updated is expected, SAS issues a warning message pointing out the error. Processing continues, but the MODULE routine cannot update a constant or expression argument (meaning that the value of the argument you wanted to update will be lost).

Consider these examples. Here is the attribute table:

```
* attribute table entry for ABC;
routine abc minarg=2 maxarg=2;
arg 1 input format=ib4.;
arg 2 output format=ib4.;
```

Here is the DATA step with the MODULE calls:

```
data _null_;
  x=5;
  /* passing a variable as the    */
  /*   second argument - OK       */
  call module('abc',1,x);
  /* passing a constant as the    */
  /*   second argument - INVALID  */
  call module('abc',1,2);
  /* passing an expression as the */
  /*   second argument - INVALID  */
  call module('abc',1,x+1);
run;
```

In the above example, the first call to MODULE is correct because the variable **x** is updated with what the **abc** routine returns for the second argument. The second call to MODULE is not correct because a constant is passed. MODULE issues a warning indicating you have passed a constant, and MODULE passes a temporary area instead.

The third call to MODULE is not correct as an arithmetic expression is passed, causing a temporary location from the DATA step to be used, and the returned value is lost.

## Specifying Formats and Informats to Use with MODULE Arguments

You specify the SAS format and informat for each DLL routine argument by specifying in the attribute table the FORMAT attribute in the ARG statement. The format indicates how numeric and character values should be passed to the DLL routine and how they should be read back upon completion of the routine.

Usually, the format you use corresponds to a variable type for a given programming language. The following sections describe the proper formats that correspond to different variable types in various programming languages.

### C Language Formats

| C Type | SAS Format/Informat |
|---|---|
| double | RB8. |
| float | FLOAT4. |
| signed int | IB4. |
| signed short | IB2. |
| signed long | IB4. |
| char * | IB4. |
| unsigned int | PIB4. |
| unsigned short | PIB2. |
| unsigned long | PIB4. |
| char[*w*] | $CHAR*w*. or $CSTR*w*. (see “$CSTR*w*. Format” on page 307) |

*Note:* For information about passing character data other than as pointers to character strings, see “$BYVAL*w*. Format” on page 307. △

### FORTRAN Language Formats

| FORTRAN Type | SAS Format/Informat |
|---|---|
| integer*2 | IB2. |
| integer*4 | IB4. |
| real*4 | FLOAT4. |
| real*8 | RB8. |
| character**w* | $CHAR*w*. |

The MODULE routines can support FORTRAN character arguments only if they are not expected to be passed by descriptor.

## PL/I Language Formats

| **PL/I Type** | **SAS Format/Informat** |
|---|---|
| FIXED BIN(15) | IB2. |
| FIXED BIN(31) | IB4. |
| FLOAT BIN(21) | RB4. |
| FLOAT BIN(31) | RB8. |
| CHARACTER(*w*) | $CHAR*w*. |

The PL/I descriptions are added here for completeness; this does not guarantee that you will be able to invoke PL/I routines.

## COBOL Language Formats

| **COBOL Format** | **SAS Format/Informat** | **Description** |
|---|---|---|
| PIC S*xxxx* BINARY | IB*w*. | integer binary |
| COMP-2 | RB8. | double-precision floating point |
| COMP-1 | RB4. | single-precision floating point |
| PIC *xxxx* or S*xxxx* | F*w*. | printable numeric |
| PIC *yyyy* | $CHAR*w*. | character |

The following COBOL specifications might not properly match with the Institute-supplied formats because zoned and packed decimal are not truly defined for systems based on Intel architecture.

| **COBOL Format** | **SAS Format/Informat** | **Description** |
|---|---|---|
| PIC S*xxxx* DISPLAY | ZD*w*. | zoned decimal |
| PIC S*xxxx* PACKED-DECIMAL | PD*w*. | packed decimal |

The following COBOL specifications do not have true native equivalents and are only usable in conjunction with the corresponding S370F*xxx* informat and format, which allows for IBM mainframe-style representations to be read and written in the PC environment.

| COBOL Format | SAS Format/Informat | Description |
|---|---|---|
| PIC *xxxx* DISPLAY | S370FZDU*w*. | zoned decimal unsigned |
| PIC S*xxxx* DISPLAY SIGN LEADING | S370FZDL*w*. | zoned decimal leading sign |
| PIC S*xxxx* DISPLAY SIGN LEADING SEPARATE | S370FZDS*w*. | zoned decimal leading sign separate |
| PIC S*xxxx* DISPLAY SIGN TRAILING SEPARATE | S370FZDT*w*. | zoned decimal trailing sign separate |
| PIC *xxxx* BINARY | S370FIBU*w*. | integer binary unsigned |
| PIC *xxxx* PACKED-DECIMAL | S370FPDU*w*. | packed decimal unsigned |

## $CSTR*w*. Format

If you pass a character argument as a null-terminated string, use the $CSTR*w*. format. This format looks for the last nonblank character of your character argument and passes a copy of the string with a null terminator after the last nonblank character. For example, given the attribute table entry:

```
* attribute table entry;
routine abc minarg=1 maxarg=1;
arg 1 input char format=$cstr10.;
```

you can use the following DATA step:

```
data _null_;
     rc = module('abc','my string');
     run;
```

The $CSTR format adds a null terminator to the character string **my string** before passing it to the **abc** routine. This is equivalent to the following attribute entry:

```
* attribute table entry;
routine abc minarg=1 maxarg=1;
arg 1 input char format=$char10.;
```

with the following DATA step:

```
data _null_;
     rc = module('abc','my string'||'00'x);
     run;
```

The first example is easier to understand and easier to use when using variable or expression arguments.

The $CSTR informat converts a null-terminated string into a blank-padded string of the specified length. If the DLL routine is supposed to update a character argument, use the $CSTR informat in the argument attribute.

## $BYVAL*w*. Format

When you use a MODULE function to pass a single character by value, the argument is automatically promoted to an integer. If you want to use a character expression in the MODULE call, you must use the special format/informat called $BYVAL*w*. The $BYVAL*w*. format/informat expects a single character and will produce a numeric value, the size of which depends on *w*. $BYVAL2. produces a short, $BYVAL4. produces a long, and $BYVAL8. produces a double. Consider this example using the C language:

```
long xyz(a,b)
  long a; double b;
  {
  static char c = 'Y';
  if (a == 'X')
     return(1);
  else if (b == c)
     return(2);
  else return(3);
  }
```

In this example, the **xyz** routine expects two arguments, a long and a double. If the long is an **X**, the actual value of the long is 88 in decimal. This is because an ASCII **X** is stored as hex 58, and this is promoted to a long, represented as 0x00000058 (or 88 decimal). If the value of **a** is **X**, or 88, a 1 is returned. If the second argument, a double, is **Y** (which is interpreted as 89), then 2 is returned.

Now suppose that you want to pass characters as the arguments to **xyz**. In C, you would invoke them as follows:

```
x = xyz('X',(double)'Z');
y = xyz('Q',(double)'Y');
```

This is because the **X** and **Q** values are automatically promoted to ints (which are the same as longs for the sake of this example), and the integer values corresponding to **Z** and **Y** are cast to doubles.

To call **xyz** using the MODULEN function, your attribute table must reflect the fact that you want to pass characters:

```
routine xyz minarg=2 maxarg=2 returns=long;
arg 1 input char byvalue format=$byval4.;
arg 2 input char byvalue format=$byval8.;
```

Note that it is important that the BYVALUE option appear in the ARG statement as well. Otherwise, MODULEN assumes that you want to pass a pointer to the routine, instead of a value.

Here is the DATA step that invokes MODULEN and passes it characters:

```
data _null_;
     x = modulen('xyz','X','Z');
     put x= ' (should be 1)';
     y = modulen('xyz','Q','Y');
     put y= ' (should be 2)';
     run;
```

## Understanding MODULE Log Messages

If you specify **i** in the control string parameter to MODULE, SAS prints several informational messages to the log. You can use these messages to determine whether you have passed incorrect arguments or coded the attribute table incorrectly.

Consider this example that uses MODULEIN from within the IML procedure. It uses the MODULEIN function to invoke the **changi** routine (stored in theoretical TRYMOD.DLL). In the example, MODULEIN passes the constant 6 and the matrix x2, which is a 4x5 matrix to be converted to an integer matrix. The attribute table for **`changi`** is as follows:

```
routine changi module=trymod returns=long;
arg 1 input num format=ib4. byvalue;
arg 2 update num format=ib4.;
```

The following IML step invokes MODULEIN:

```
proc iml;
     x1 = J(4,5,0);
     do i=1 to 4;
        do j=1 to 5;
           x1[i,j] = i*10+j+3;
           end;
        end;
     y1= x1;
             x2 = x1;
                      y2 = y1;
     rc = modulein('*i','changi',6,x2);
     ....
```

The `'*i'` control string causes the lines shown in Output 14.1 to be printed in the log.

**Output 14.1** MODULEIN Output

```
---PARM LIST FOR MODULEIN ROUTINE---  CHR PARM 1 885E0AA8 2A69 (*i)
CHR PARM 2 885E0AD0 6368616E6769 (changi)
NUM PARM 3 885E0AE0 0000000000001840
NUM PARM 4 885E07F0
0000000000002C40000000000000002E40000000000000304000000000000031400000000000003240
0000000000003840000000000000003940000000000000003A400000000000003B400000000000003C40
00000000000041400000000000080414000000000
---ROUTINE changi LOADED AT ADDRESS 886119B8 (PARMLIST AT 886033A0)--- PARM 1 06000000     <CALL-BY-VALUE>
PARM 2 88604720
0E0000000F000000100000001100000012000000180000001900000001A0000001B0000001C000000
22000000230000002400000025000000260000002C0000002D0000002E0000002F00000030000000
---VALUES UPON RETURN FROM changi ROUTINE---   PARM 1 06000000     <CALL-BY-VALUE>
PARM 2 88604720
140000001F0000002A00000035000000400000008200000008D00000098000000A3000000AE000000
F0000000FB00000006010000110100001C0100005E01000069010000740100007F0100008A010000
---VALUES UPON RETURN FROM MODULEIN ROUTINE---  NUM PARM 3 885E0AE0 0000000000001840
NUM PARM 4 885E07F0
00000000000034400000000000003F400000000000004540000000000000804A400000000000005040
000000000040604000000000000A06140000000000000063400000000000606440000000000000C06540
0000000000006E400000000000606F4000000000
```

The output is divided into four sections.

1 The first section describes the arguments passed to MODULEIN.

The 'CHR PARM *n*' portion indicates that character parameter *n* was passed. In the example, 885E0AA8 is the actual address of the first character parameter to MODULEIN. The value at the address is hex 2A69, and the ASCII representation of that value ('*i') is in parentheses after the hex value. The second parameter is likewise printed. Only these first two arguments have their ASCII equivalents printed; this is because other arguments might contain unreadable binary data.

The remaining parameters appear with only hex representations of their values (NUM PARM 3 and NUM PARM 4 in the example).

The third parameter to MODULEIN is numeric, and it is at address 885E0AE0. The hex representation of the floating point number 6 is shown. The fourth parameter is at address 885E07F0, which points to an area containing all the values for the 4x5 matrix. The **`*i`** option prints the entire argument; be careful if you use this option with large matrices because the log might become quite large.

2 The second section of the log lists the arguments to be passed to the requested routine and, in this case, changed. This section is important for determining if the arguments are being passed to the routine correctly. The first line of this section contains the name of the routine and its address in memory. It also contains the address of the location of the parameter block that MODULEIN created.

The log contains the status of each argument as it is passed. For example, the first parameter in the example is call-by-value (as indicated in the log). The second parameter is the address of the matrix. The log shows the address, along with the data to which it points.

Note that all the values in the first parameter and in the matrix are long integers because the attribute table states that the format is IB4.

3 In the third section, the log contains the argument values upon return from **`changi`**. The call-by-value argument is unchanged, but the other argument (the matrix) contains different values.

4 The last section of the log output contains the values of the arguments as they are returned to the MODULEIN calling routine.

## Examples

### Updating a Character String Argument

This example uses the Win32 routine GetTempPathA. This routine expects as an argument a pointer to a buffer, along with the length of the buffer. GetTempPathA fills the buffer with a null-terminated string representing the temporary path. Here is the C prototype for the GetTempPathA routine:

```
DWORD WINAPI GetTempPathA
    (DWORD nBufferLength, LPSTR lpBuffer);
```

Here is the attribute table:

```
routine GetTempPathA
  minarg=2
  maxarg=2
  stackpop=called
  returns=long;
arg 1 input  byvalue format=pib4.;
arg 2 update         format=$cstr200.;
```

Note that the STACKPOP=CALLED option is used; all Win32 service routines require this attribute. The first argument is passed by value because it is an input argument only. The second argument is an update argument because the contents of the buffer are to be updated. The $CSTR200. format allows for a 200-byte character string that is null-terminated.

Here is the SAS code to invoke the function. In this example, the DLL name (KERNEL32) is explicitly given in the call (because the MODULE attribute was not used in the attribute file):

```
filename sascbtbl "sascbtbl.dat";
data _null_;
```

```
  length path $200;
  n = modulen( '*i',
      "KERNEL32,GetTempPathA", 199, path );
  put n= path=;
run;
```

*Note:* KERNEL32.DLL is an internal DLL provided by Windows. Its routines are described in the Microsoft Win32 SDK. △

The code produces these log messages:

```
NOTE: Variable PATH is uninitialized.
N=7 PATH=C:\TEMP
```

The example uses 199 as the buffer length because PATH can hold up to 200 characters with one character reserved for the null terminator. The $CSTR200. informat ensures that the null-terminator and all subsequent characters are replaced by trailing blanks when control returns to the DATA step.

## Passing Arguments by Value

This example calls the Beep routine, part of the Win32 API in the KERNEL32 DLL. Here is the C prototype for Beep:

```
BOOL Beep(DWORD dwFreq, DWORD dwDuration)
```

Here is the attribute table to use:

```
routine Beep
  minarg=2
  maxarg=2
  stackpop=called
  callseq=byvalue
  module=kernel32;
arg 1 num format=pib4.;
arg 2 num format=pib4.;
```

Because both arguments are passed by value, the example includes the CALLSEQ=BYVALUE attribute in the ROUTINE statement, so it is not necessary to specify the BYVALUE option in each ARG statement.
Here is the sample SAS code used to call the Beep function:

```
filename sascbtbl 'sascbtbl.dat';
data _null_;
  rc = modulen("*e","Beep",1380,1000);
run;
```

The computer speaker beeps.

## Using PEEKCLONG to Access a Returned Pointer

The following example uses the **lstrcat** routine, part of the Win32 API in KERNEL32.DLL. **lstrcat** accepts two strings as arguments, concatenates them, and returns a pointer to the concatenated string. The C prototype is

```
LPTSTR lstrcat (LPTSTR lpszString1,
                LPCTSTR lpszString2);
```

The following is the proper attribute table:

```
routine lstrcat
  minarg=2
  maxarg=2
  stackpop=called
  module=KERNEL32
  returns=ulong;
  arg 1 char format=$cstr200.;
  arg 2 char format=$cstr200.;
```

To use **lstrcat**, you need to use the SAS PEEKCLONG function to access the data referenced by the returned pointer. Here is the sample SAS program that accesses **lstrcat**:

```
filename sascbtbl 'sascbtbl.dat';
data _null_;
  length string1 string2 conctstr $200;
  string1 = 'This is';
  string2 = ' a test!';
  rc = modulen('lstrcat',string1,string2);
  conctstr = peekclong(rc,200);
  put conctstr=;
run;
```

The following output appears in the log:

```
conctstr=This is a test!
```

Upon return from MODULEN, the pointer value is stored in RC. The example uses the PEEKCLONG function to return the 200 bytes at that location, using the $CSTR200. format to produce a blank-padded string that replaces the null termination.

For more information about the PEEKLONG functions, see the PEEKCLONG function and the PEEKLONG function in *SAS Language Reference: Dictionary*.

## Using Structures

"Grouping SAS Variables as Structure Arguments" on page 303 describes how to use the FDSTART attribute to pass several arguments as one structure argument to a DLL routine. Refer to that section for an example of the GetClientRect attribute table and C language equivalent. This example shows how to invoke the GetClientRect function after defining the attribute table.

The most straightforward method works, but generates a warning message about the variables not being initialized:

```
filename sascbtbl 'sascbtbl.dat';
data _null_;
     hwnd=modulen('GetForegroundWindow');
     call module('GetClientRect',hwnd,
       left,top,right,bottom);
     put _all_;
     run;
```

To remove the warning, you can use the RETAIN statement to initialize the variables to 0. Also, you can use shorthand to specify the variable list in the MODULEN statement:

```
data _null_;
     retain left top right bottom 0;
     hwnd=modulen('GetForegroundWindow');
```

```
      call module('GetClientRect',hwnd,
            of left--bottom);
      put _all_;
      run;
```

Note that the OF keyword indicates that what follows is a list of variables, in this case delimited by the double-dash. The output in the log varies depending on the active window and looks something like the following:

```
HWND=3536768 LEFT=2 TOP=2 RIGHT=400
BOTTOM=587
```

## Invoking a DLL Routine from PROC IML

This example shows how to pass a matrix as an argument within PROC IML. The example creates a 4x5 matrix. Each cell is set to 10*x*+*y*+3, where *x* is the row number and *y* is the column number. For example, the cell at row 1 column 2 is set to (10*1)+2+3, or 15.

The example invokes several routines from the theoretical TRYMOD DLL. It uses the **changd** routine to add 100*x*+10*y* to each element, where *x* is the C row number (0 through 3) and *y* is the C column number (0 through 4). The first argument to **changd** indicates what extra amount to sum. The **changdx** routine works just like **changd**, except that it expects a transposed matrix. The **changi** routine works like **changd** except that it expects a matrix of integers. The **changix** routine works like **changdx** except that integers are expected.

*Note:* A maximum of three arguments can be sent when invoking a DLL routine from PROC IML. △

In this example, all four matrices x1, x2, y1, and y2 should become set to the same values after their respective MODULEIN calls. Here are the attribute table entries:

```
routine changd module=trymod returns=long;
arg 1 input num format=rb8. byvalue;
arg 2 update num format=rb8.;
routine changdx module=trymod returns=long
  transpose=yes;
arg 1 input num format=rb8. byvalue;
arg 2 update num format=rb8.;
routine changi module=trymod returns=long;
arg 1 input num format=ib4. byvalue;
arg 2 update num format=ib4.;
routine changix module=trymod returns=long
  transpose=yes;
arg 1 input num format=ib4. byvalue;
arg 2 update num format=ib4.;
```

Here is the PROC IML step:

```
proc iml;
     x1 = J(4,5,0);
     do i=1 to 4;
        do j=1 to 5;
           x1[i,j] = i*10+j+3;
           end;
        end;
     y1= x1; x2 = x1; y2 = y1;
     rc = modulein('changd',6,x1);
     rc = modulein('changdx',6,x2);
     rc = modulein('changi',6,y1);
     rc = modulein('changix',6,y2);
     print x1 x2 y1 y2;
     run;
```

The following are the results of the PRINT statement:

```
 X1
 20        31        42        53        64
130       141       152       163       174
240       251       262       273       284
350       361       372       383       394
 X2
 20        31        42        53        64
130       141       152       163       174
240       251       262       273       284
350       361       372       383       394
 Y1
 20        31        42        53        64
130       141       152       163       174
240       251       262       273       284
350       361       372       383       394
 Y2
 20        31        42        53        64
130       141       152       163       174
240       251       262       273       284
350       361       372       383       394
```

# CHAPTER 15 Considerations for SAS/AF Programmers

*Controlling the Appearance and Behavior of SAS* **315**
*Controlling the Main SAS Window* **315**
    *SAS System Options That Control the Main SAS Window* **315**
    *SAS Commands That Control the Main SAS Window* **316**
*Accessing External DLLs from SAS* **318**
*Designing, Saving, and Loading Custom Toolbar Controls* **319**
*Invoking SAS/AF Applications Automatically* **319**
*Associating Your Own Logo and Icons with Your SAS/AF Application* **319**
*Incorporating Electronic Mail into Your SAS/AF Application* **320**

## Controlling the Appearance and Behavior of SAS

SAS under Windows provides SAS/AF programmers with extensive control over the appearance and behavior of the main SAS window. You can:

- use SAS system options and windowing environment commands to control the appearance of the main SAS window
- call external dynamic link libraries (DLLs) using the DATA step or SAS Component Language (SCL) commands
- design, save, and load custom toolbar controls
- immediately invoke SAS/AF programs when you start SAS
- distribute the minimum subset of SAS files needed to run a particular SAS/AF application
- associate your own logo and icons with your SAS/AF applications
- use SCL code to send electronic mail to other users
- invoke a web browser to view documents online.

## Controlling the Main SAS Window

SAS system options and windowing environment commands make it possible to change the appearance and behavior of the main SAS window so much that end users might not recognize it as SAS.

### SAS System Options That Control the Main SAS Window

The following table lists the system options that provide control over the main SAS window.

**Table 15.1** SAS System Options for the Main SAS Window

| Option | Description |
|---|---|
| AWSCONTROL | Remove system controls such as the title bar, system menu, and minimize and maximize buttons from the main SAS window. |
| AWSDEF | Specify the location and dimensions of the main SAS window when SAS initializes. |
| AWSMENU | Specify whether to display the main SAS window menu bar. |
| AWSMENUMERGE | Specify whether to embed menu items that are specific to Windows in the main menus. |
| AWSTITLE | Specify the text that appears in the title bar of the main SAS window. |
| ICON | Iconize (minimize) the main SAS window at SAS initialization. |
| REGISTER | Specify other Windows programs to be included as options on the AWS **`File`** menu. |
| SASCONTROL | Remove system controls and the minimize and maximize buttons from SAS application windows. |
| SOLUTIONS | Include or suppress the Solutions menu in the main menu. |
| SPLASH | Specify whether the logo (splash screen) is displayed when SAS starts. |
| SPLASHLOC | Specify the location of the bitmap that contains the splash screen you want to display when SAS starts. |
| TOOLSMENU | Include or suppress the Tools menu in the main menu. |
| VIEWMENU | Include or suppress the View menu in the main menu. |
| WEBUI | Specifies whether the web enhancements that select objects by pointing the mouse pointer and using a single mouse-click to invoke the default action are enabled. |
| WINDOWSMENU | Include or suppress the Window menu in the main menu. WINDOWSMENU is valid only if NOAWSMENUMERGE is specified. |

## SAS Commands That Control the Main SAS Window

Table 15.2 on page 316 lists the SAS commands that you can use to control the appearance and behavior of the main SAS window.

**Table 15.2** SAS Windowing Environment Commands for the Main SAS Window

| Option | Description |
|---|---|
| AWSMAXIMIZE | Maximizes the main SAS window. |
| AWSMINIMIZE | Minimizes the main SAS window. |
| AWSRESTORE | Restores the main SAS window to its previous state. |
| COLOR | Sets the color for various components of the application windows. |

| Option | Description |
|---|---|
| COMMAND | Controls the appearance of the command bar or dialog box. |
| DLGABOUT | Invokes the About dialog box. |
| DLGCONVERT | Invokes the Convert dialog box for a selected OLE object. |
| DLGCDIR | Invokes the Change Folder dialog box. |
| DLGENDR | Invokes the Exit SAS confirmation dialog box. |
| DLGFIND | Invokes the Find dialog box. |
| DLGFONT | Invokes the Fonts Selection dialog box. |
| DLGLIB | Invokes the Libraries dialog box. |
| DLGLINKS | Invokes the OLE Links dialog box. |
| DLGOPEN | Invokes the Open dialog box for the Program Editor. |
| DLGPAGESETUP | Invokes the Page Setup dialog box. |
| DLGPREF | Invokes the Preferences dialog box. |
| DLGPRT | Invokes the Print dialog box. |
| DLGPRTPREVIEW | Invokes the Print Preview window. |
| DLGPRTSETUP | Invokes the Print Setup dialog box. |
| DLGREPLACE | Invokes the Replace dialog box. |
| DLGRUN | Invokes the Run dialog box. |
| DLGSAVE | Invokes the Save As dialog box. |
| DLGSMAIL | Invokes the E-mail dialog box. |
| FILEOPEN | Invokes the Open dialog box for the Enhanced Editor. |
| NEXTWIND | Displays the next open SAS window. |
| PMENU | Toggles the command lines on or off in the windowing environment. |
| PREVWIND | Display the previous open SAS window. |
| RESHOW | Redisplays the SAS windows that are currently open. |
| TOOLCLOSE | Closes the toolbox or toolbar. |
| TOOLEDIT | Invokes the Customize dialog box for toolbars or tool boxes. |
| TOOLLARGE | Toggles the size of the toolbar or toolbox buttons. |
| TOOLLOAD | Opens the toolbox or toolbar with the specified configuration. |
| TOOLSWITCH | Toggles whether the toolbar or toolbox associated with the active window is automatically loaded. |
| TOOLTIPS | Toggles the ToolTips on or off. |
| WATTACH | Toggles whether the contents of the active window are attached to e-mail that you send through SAS. |
| WDOCKVIEW | Enables dockable windows. |
| WEDIT | Invokes a new Enhanced Editor window. |
| WEMAILFMT | Specifies the format (.RTF or .TEXT) of any text window you attach to an e-mail message. |

| Option | Description |
|---|---|
| WHIDECURSOR | Suppress display of the cursor. |
| WHSBAR | Toggles the horizontal scroll bars on or off. |
| WINSERT | Toggles the insert mode on or off. |
| WMENUPOP | Enables or disables the pop-up menus in the SAS application windows. |
| WMRU | Specifies how many filenames to retain in the list under the File menu. |
| WNEWTITLE | Clears the contents of the active window and removes its title. |
| WNEXTEDIT | Switches the active Enhanced Editor window to another Enhanced Editor window. |
| WPGM | |
| WPOPUP | Causes the pop-up menu for the active window to appear. |
| WSCREENTIPS | Toggles the ScreenTips on or off. |
| WSTATUSLN | Toggles the status line on or off, and controls the area proportions. |
| WUNDO | Undoes the previous editing action. |
| WVSBAR | Toggles the vertical scroll bars on or off. |
| WWINDOWBAR | Displays the window bar at the bottom of the main SAS window. |
| ZOOM | Maximizes the active SAS application window. |

## Accessing External DLLs from SAS

You can access routines that reside in external dynamic link libraries (DLLs) by using the SAS MODULE family of functions within a DATA step or SCL. This lets you access DLLs that you create or purchase; you can even access operating system DLLs.

To access an external DLL, you must know:

- □ the name of the DLL
- □ the function name or ordinal
- □ a description of the function's arguments
- □ a description of the return code.

***CAUTION:***

**Only experienced programmers should access external DLLs.** When you access an external DLL, you are passing control of your computer from SAS to the DLL function. If done improperly, or if the DLL function is unreliable, you might lose data or have to reset your computer (or both). △

The general steps for accessing an external DLL routine are:

1 Create a text file that describes the DLL routine you want to access, including the arguments it expects and the values it returns (if any). This attribute file must be in a special format.

2 Use the FILENAME statement to assign the SASCBTBL fileref to the attribute file you created.

3 In a DATA step or SCL code, use MODULE, MODULEN, or MODULEC to invoke the DLL routine. The specific function you use depends on the type of expected return value (none, numeric, or character). (You can also use MODULEI, MODULEIN, or MODULEIC within a PROC IML step.)

*Note:* The MODULE routines can be a flexible and powerful tool, especially when used with the SASCBTBL file, SAS formats and informats, and other SAS routines. As such, you should be extremely careful when invoking external routines; if done improperly, you might lose data or have to reset your computer. △

For complete information about accessing DLLs from within SAS, see Chapter 14, "Accessing External DLLs from SAS," on page 295.

## Designing, Saving, and Loading Custom Toolbar Controls

You can provide the users of your SAS/AF application with easy-to-use tools by creating a custom toolbar configuration. You can assign these tools to represent any windowing environment command. For complete information about creating and saving custom toolbars, see "Customizing the Toolbar" on page 66.

If you distribute your SAS/AF application to other machines, be sure to include the catalog entry that contains your custom tool configuration.

By default, tool switching is enabled, which allows the use of a custom toolbar in your SAS/AF application. Tool switching can be disabled by issuing the TOOLSWITCH OFF command.

## Invoking SAS/AF Applications Automatically

SAS provides a system option, INITCMD, that lets you invoke SAS/AF programs automatically. When you use this option, SAS does not create the PROGRAM EDITOR, LOG, or OUTPUT windows but instead runs the SAS/AF applications and windowing environment commands that you specify.

The general syntax of the INITCMD option is:

-INITCMD "*af-command*" <*DM-command-1*...*DM-command-n*>

where *af-command* is a command to start an AF application, and *DM-command-1* through *DM-command-n* are any windowing environment commands.

For example, the following option specification starts a SAS/AF application and loads a custom tool bar:

```
-initcmd "AF c=mylib.myapp.myfirst.frame;
  toolload bar mylib.myapp.profile.toolbox"
```

For more information about the INITCMD system option, see *SAS Language Reference: Dictionary*.

## Associating Your Own Logo and Icons with Your SAS/AF Application

You can substitute your own logo screen and icons in place of those provided by SAS.

*Note:* These procedures involve creating resources for and building your own dynamic link libraries (DLLs). For more information on creating DLLs, see the Microsoft Win32 Software Development Kit. △

To display your own logo when SAS starts:

1 Create the logo you want to display and save it either as a Windows bitmap (which has a BMP file extension), or compile it as resource and build it into a DLL.

2 When you invoke SAS, specify the SPLASHLOC option with the full pathname of the file that contains your bitmap. If the bitmap is in a DLL, be sure to specify the resource number as well. The default resource number is 1. For more information, see “SPLASHLOC System Option” on page 558.

Your logo will display when you start SAS.

To use your own icons with your SAS/AF application:

1 Use the USERICON system option when you start SAS to specify the resource file that contains the icons you want to include. You must use the Windows software development tool to compile the resource file. For more information about the USERICON option, see “USERICON System Option” on page 571.

2 Use SAS/AF software to create a FRAME entry.

3 On the FRAME entry, select **`Make`** from the popup menu. (Alternatively, you can create a region and then select **`Fill`** from the popup menu.)

4 Select **`Icon`** from the selection list.

5 From the Icon Attributes display, select the **`Current Icon`** button.

6 Select the **`User-defined`** radio button to display the icons in your resource file.

7 From the Select an Icon display, choose the icon you want to use. If there are more icons than will fit in the window, use the scrollbars to see the rest of the icons.

## Incorporating Electronic Mail into Your SAS/AF Application

You can associate SCL code with buttons and fields in a FRAME entry to create your own interface to electronic mail. SAS provides methods to interface with VIM-, MAPI-, and SMTP-compatible electronic mail programs.

Using SCL code, you can specify who should receive mail (TO, CC, and BCC), the subject of the mail, the body of the message, and any files you want to attach to the message. “Sending E-Mail Using SAS” on page 40 describes the e-mail functions that SAS facilitates and contains examples of DATA step and SCL code.

PART 3

# Features of the SAS Language for Windows

*Chapter* ***16***. . . . . . . . . **Commands under Windows** *323*

*Chapter* ***17***. . . . . . . . . **Formats under Windows** *375*

*Chapter* ***18***. . . . . . . . . **Functions and CALL Routines under Windows** *385*

*Chapter* ***19***. . . . . . . . . **Informats under Windows** *413*

*Chapter* ***20***. . . . . . . . . **Procedures under Windows** *423*

*Chapter* ***21***. . . . . . . . . **Statements under Windows** *441*

*Chapter* ***22***. . . . . . . . . **System Options under Windows** *465*

*Chapter* ***23***. . . . . . . . . **Length and Precision of Variables** *579*

*Chapter* ***24***. . . . . . . . . **Macro Facility under Windows** *581*

CHAPTER
16

# Commands under Windows

SAS Commands under Windows 324
Commands Not Supported in the Windows Operating Environment 325
AUTOSCROLL Command 325
AWSMAXIMIZE Command 326
AWSMINIMIZE Command 326
AWSRESTORE Command 327
CAPS Command 327
COLOR Command 328
COMMAND Command 328
CUT Command 330
DLGABOUT Command 330
DLGCDIR Command 331
DLGCOLUMNSIZE Command 331
DLGCOLUMNSORT Command 331
DLGCONVERT Command 332
DLGENDR Command 332
DLGFIND Command 333
DLGFONT Command 333
DLGLIB Command 334
DLGLINKS Command 334
DLGOPEN Command 334
DLGPAGESETUP Command 336
DLGPREF Command 337
DLGPRT Command 337
DLGPRTPREVIEW Command 338
DLGPRTSETUP Command 339
DLGREPLACE Command 339
DLGRUN Command 340
DLGSAVE Command 340
DLGSMAIL Command 342
FILE Command 342
FILEOPEN Command 344
FILL Command 344
GSUBMIT Command 345
HOME Command 345
ICON Command 346
INCLUDE Command 347
PMENU Command 348
SAVE Command 349
STORE Command 349
SUBTOP Command 350

*TOOLCLOSE Command* **351**
*TOOLEDIT Command* **351**
*TOOLLARGE Command* **352**
*TOOLLOAD Command* **352**
*TOOLSWITCH Command* **353**
*TOOLTIPS Command* **354**
*WATTACH Command* **354**
*WATTENTION Command* **355**
*WAUTOSAVE Command* **355**
*WBROWSE Command* **356**
*WCOPY Command* **357**
*WCUT Command* **357**
*WDOCKVIEW Command* **358**
*WDOCKVIEWMINIMIZE Command* **358**
*WDOCKVIEWRESIZE Command* **359**
*WDOCKVIEWRESTORE Command* **359**
*WEDIT Command* **360**
*WEMAILFMT Command* **360**
*WEXITSAVE Command* **361**
*WFILE Command* **361**
*WHIDECURSOR Command* **362**
*WHSBAR Command* **363**
*WINSERT Command* **363**
*WMENUPOP Command* **364**
*WMRU Command* **364**
*WNAVKEYUNMARK Command* **365**
*WNEWTITLE Command* **366**
*WNEXTEDIT Command* **366**
*WPASTE Command* **367**
*WPGM Command* **367**
*WPOPUP Command* **368**
*WRTFSAVE Command* **368**
*WSCREENTIPS Command* **369**
*WSTATUSLN Command* **369**
*WUNDO Command* **371**
*WVSBAR Command* **371**
*WWINDOWBAR Command* **372**
*X Command* **372**
*ZOOM Command* **373**

## SAS Commands under Windows

During an interactive SAS session, you can issue commands from the command bar, from the command line within a SAS window, from the keyboard, or from the toolbar. SAS supports many commands that help you to navigate your session and accomplish certain tasks. In many cases, the command is simply another way to invoke an action that you can also accomplish by using the SAS menus and windows. However, advanced users might find the supported commands to be a more efficient way to work. Commands provide a more flexible way to accomplish a task if the parameters of your task are different from what the SAS interface supports.

Most SAS windowing environment commands are described in the SAS Help and Documentation. The commands that are described here have syntax or behavior that is specific to the Windows operating environment.

For more information about issuing commands, see "Issuing SAS Commands" on page 38.

## Commands Not Supported in the Windows Operating Environment

The following SAS commands are not supported under Windows:

PCLEAR

PLIST

SCROLLBAR

SMARK

WDRAG

WGROW

WMOVE

WSHRINK

These commands are not supported under Windows because it is more efficient to use Windows features. For example, the SCROLLBAR command and window sizing commands are not needed in the Windows operating environment as scrollbars and window sizing bars are an integral part of the graphical user interface.

## AUTOSCROLL Command

**Controls how often the Log and Output windows scroll to display output**

**Windows specifics:** default values

### Syntax

AUTOSCROLL <*number-of-lines* | PAGE | MAX>

### Details

Under Windows, the default value for the AUTOSCROLL command in the OUTPUT window is 0 (meaning that no output is written to that window while statements are executing, which provides the best performance). The default value for the LOG window is half the number of lines of the LOG window when SAS is started.

Scrolling can increase the length of time that SAS takes to run your program. The less scrolling that the LOG and OUTPUT windows have to do, the faster that your program will run.

You can also set scrolling options in the Preferences dialog box **`Advanced`** page.

### See Also

- "Setting Session Preferences" on page 57

- “AUTOSCROLL Command” in the SAS Help and Documentation

## AWSMAXIMIZE Command

**Maximizes the main SAS window**

**Windows specifics:** all

### Syntax

AWSMAXIMIZE <ON | OFF>

**no argument**
toggles the main SAS window between the maximized and the restored state.

**ON**
maximizes the main SAS window. This option has the same effect as clicking on the maximize button.

**OFF**
restores the main SAS window to its previous state.

### Details

The AWSMAXIMIZE command allows you to enlarge the main SAS window to use the complete Windows desktop.

## AWSMINIMIZE Command

**Minimizes the main SAS window**

**Windows specifics:** all

### Syntax

AWSMINIMIZE <ON | OFF>

**no argument**
toggles the main SAS window between the minimized and the restored state.

**ON**
minimizes the main SAS window. This option has the same effect as clicking on the minimize button.

**OFF**
restores the main SAS window to its previous state.

## AWSRESTORE Command

**Restores the main SAS window to its previous state**

**Windows specifics:** all

### Syntax

AWSRESTORE <ON | OFF>

**no argument**
toggles the main SAS window between the maximized and the restored state.

**ON**
restores the main SAS window to its previous state. This option has the same effect as selecting `Restore` from the main SAS window's title bar menu.

**OFF**
restores the main SAS window to its default state.

### Details

You can use either the AWSRESTORE command or the AWSMAXIMIZE command to toggle the main SAS window between maximized and its previous state.

## CAPS Command

**Causes characters to be translated to uppercase**

**Windows specifics:** all

### Syntax

CAPS

### Details

The CAPS command changes the case for text not yet entered or for text modified in a window.

Under Windows, characters are translated to uppercase when you move the cursor off the line *or* when you press ENTER.

### See Also

□ "CAPS Command" in the SAS Help and Documentation

## COLOR Command

**Controls the color of window components**

**Windows specifics:** affected window components

### Syntax

COLOR *field-type* <*color* | NEXT <*highlight*>>

***field-type***
specifies the area of the window or the type of text whose color is to be changed.

***color***
specifies a color for the window or for selected portions of the window.

**NEXT**
changes the color to the next available color. The value of NEXT is based on the most recent color entered. The order of the colors depends on your monitor.

***highlight***
specifies the highlighting attribute.

### Details

Under Windows, you cannot use the COLOR command to change the colors of the following display components: border, menu bar, pop-up menu background, and title bar. Use the Windows Control Panel to change the colors of these display components.

In addition, the HIGHLIGHT and BLINK highlight attributes are not supported for any Windows window component.

### See Also

- Other COLOR commands in the SAS Help and Documentation

## COMMAND Command

**Sets the options for the command bar**

**Windows specifics:** valid options

### Syntax

COMMAND <<WINDOW <"*title*"> | BAR
<SORT=MCU|MRU><FOCUS><MAX=*max-commands*><AUTOCOMPLETE | NOAUTOCOMPLETE>> | CLOSE>

**no arguments**
toggles the command line on and off for the active window.

**WINDOW <"*title*">**
specifies to display the command bar as a separate window that can be moved anywhere on the desktop. The "*title*" argument is optional and must be enclosed in double quotes. When you specify *title*, the command window appears with *title* as the title.

**BAR**
specifies to display the command bar in a stationary location, underneath the menu bar.

**SORT=MCU|MRU**
specifies how you want SAS to sort the commands in the command bar drop-down list. You can sort commands in the order that you most commonly use them (MCU) or that you most recently used them (MRU).

You must specify the WINDOW or BAR argument in the command before specifying the SORT argument.

**FOCUS**
specifies to place the window focus in the command bar.

**MAX=*max-commands***
specifies the maximum number of commands to “remember” in the command bar drop- down list. Valid values are 0 through 50.

You must specify the WINDOW or BAR argument in the command before specifying the MAX argument.

**AUTOCOMPLETE | NOAUTOCOMPLETE**
specifies whether the command bar attempts to match the command that is being typed with commands that were previously typed.

You must specify the WINDOW or BAR argument in the command before specifying the SORT argument.

**CLOSE**
specifies to close the command bar.

## Details

You can set some of these options by using the Customize Tools dialog box. However, you can specify a title for the Command window only by using this command.

If you issue COMMAND FOCUS when the command bar is closed

- The command bar is opened in the state that it was in before it was closed, either docked to the main SAS window or undocked as a separate window.
- The window focus is placed in the command bar.

## See Also

- “COMMAND Command” in the SAS Help and Documentation
- “Using the Command Bar to Issue Commands” on page 39
- “View Preferences” on page 59
- “DLGPREF Command” on page 337

## CUT Command

**Cuts selected text from a window**

**Windows specifics:** supported options

### Syntax

CUT <LAST | ALL>

**LAST**
cuts the most recently marked text and unmarks all other marks when more than one area of text is marked. To cut one area of text when more than one mark exists, you must use either the LAST or the ALL argument.

**ALL**
cuts all current marks when more than one area of text is marked.

### Details

The CUT command removes marked text from the current window and stores it in the Windows clipboard.

Under Windows, the APPEND and BUFFER= options are not supported for the CUT command.

### See Also

- "CUT Command" in the SAS Help and Documentation
- "Using the Clipboard" on page 53
- "WCUT Command" on page 357

## DLGABOUT Command

**Opens the About SAS System dialog box**

**Windows specifics:** all

### Syntax

DLGABOUT

### Details

To access the About SAS System dialog box from the pull-down menus, select the **`Help`** menu and then select **`About SAS System`**.

## DLGCDIR Command

**Opens the Change Folder dialog**

**Windows specifics:** all

### Syntax

DLGCDIR

### Details

From the Change Folder dialog, you can select a new working folder.

### See Also

- “Changing the SAS Current Folder” on page 37
- “SASINITIALFOLDER System Option” on page 546

## DLGCOLUMNSIZE Command

**Opens the Columns Settings dialog box**

**Windows specifics:** all

### Syntax

DLGCOLUMNSIZE

### Details

When a SAS window contains a List view with details, you can specify the size of a column in pixels using the Columns Settings dialog box. An example of a window that can be a List view is the SAS Explorer window.

### See Also

- “Resizing the Detail Columns of a List View” on page 79

## DLGCOLUMNSORT Command

**Opens the Sort Columns dialog box**

**Windows specifics:** all

### Syntax

DLGCOLUMNSORT

### Details

When a SAS window contains a List view, you can sort the columns using the Sort Columns dialog box. An example of a window that can be a List view is the SAS Explorer window.

### See Also

- □ "Sorting Window List Views by a Specific Column" on page 78

## DLGCONVERT Command

**Opens the Convert dialog box**

**Windows specifics:** all

### Syntax

DLGCONVERT

### Details

You can use this command from the SAS/AF BUILD window with an OLE object selected. The Convert dialog box lets you convert the selected OLE object from one type to another, with the available types depending on what the OLE server application supports for that object.

### See Also

- □ "Converting OLE Objects" on page 249

## DLGENDR Command

**Opens the Exit dialog box**

**Windows specifics:** all

### Syntax

DLGENDR

### Details

The Exit dialog box prompts you to confirm that you want to exit SAS. If you select `OK` in the dialog box, the SAS session ends. If `Confirm exit` is not selected in the Preferences dialog box `General` tabbed page, SAS closes when you enter the DLGENDR command.

### See Also

- "Setting Session Preferences" on page 57

## DLGFIND Command

**Opens the Find dialog box**

**Windows specifics:** all

### Syntax

DLGFIND

### Details

The Find dialog box allows you to search for text strings.

### See Also

- "DLGREPLACE Command" on page 339

## DLGFONT Command

**Opens the Fonts dialog box**

**Windows specifics:** all

### Syntax

DLGFONT

### Details

The Fonts Selection dialog box allows you to dynamically change the SAS windowing environment font.

## DLGLIB Command

**Opens the Libraries dialog box**

**Windows specifics:** all

### Syntax

DLGLIB

### Details

The Libraries dialog box lets you define or modify SAS libraries. The DLGLIB command is supported for compatibility with previous releases.

You can use the SAS Explorer window to browse or assign SAS libraries.

### See Also

- □ SAS Help and Documentation for more information about using the SAS Explorer window to manage SAS libraries

## DLGLINKS Command

**Opens the Links dialog box**

**Windows specifics:** all

### Syntax

DLGLINKS

### Details

The DLGLINKS command opens the Links dialog box, allowing you to update a linked object.

### See Also

- □ “Using Linked OLE Objects” on page 247

## DLGOPEN Command

**Opens the Open dialog box for the default editor**

**Windows specifics:** all

## Syntax

DLGOPEN <LONGFILTER=“*filters*” | FILTER=’*filters*’ <REPLACE> ><SUBMIT | NOSUBMIT> <IMPORT> <VERIFY> <ALTCMD=’*command*’>

**no arguments**
opens the Open dialog box with the default settings

**LONGFILTER=“*filters*” | FILTER=’*filters*’**
LONGFILTER=“*filters*” specifies one or more file filters to use as search criteria for displaying files in the Open dialog box. The first filter in the argument list is the default filter and is used as the search criteria. All of the filters in the argument list are added to the list of filters in the **`Files of type:`** combo box. To search for additional file types, you would select another filter from the **`Files of type:`** combo box.

You must enclose the filter list in double quotation marks. Note that you can specify long file names that include spaces and single quotes. Separate each filter that you specify with a vertical bar (|). For example, if you specify

```
dlgopen longfilter="*.text|*.Bob’s work|*.*XX"
```

the dialog box displays all files in the current folder that have .text as their file extension, and the dialog box adds *.text, *.Bob’s work and *.XX to the **`Files of type:`** combo box.

*Note:* When you are using the DLGOPEN command in the DM statement, do not use single quotation marks as part of a longfilter. The DM statement requires single quotation marks around the command it submits. A single quotation mark in the longfilter indicates to the DM statement the end of the command. △

FILTER=’*filters*’ specifies one or more file filters to use as search criteria for displaying files in the Open dialog box. The first filter in the argument list is the default filter and is used as the search criteria. All of the filters in the argument list are added to the list of filters in the **`Files of type:`** combo box. To search for additional file types, you would select another filter from the **`Files of type:`** combo box. You must enclose the filter list quotation marks. Separate multiple lists with a space. For example, if you specify

```
dlgopen filter=’*.bak *.txt’
```

the dialog box displays all files in the current folder that have a .BAK file extension, and adds both *.BAK and *.TXT to the **`Files of Type:`** combo box.

*Note:* The difference between LONGFILTER=“*filters*” and FILTER=’*filters*’ is the use of spaces and quotation marks. Use LONGFILTER=“*filters*” if *filters* contain spaces and single quotation marks. If you use FILTER=’*filters*’, *filters* cannot contain spaces and single quotation marks. △

**REPLACE**
replaces the filter list with the specified filters instead of concatenating the list with the default filters. This option is valid only when you specify the LONGFILTER= or FILTER= argument as well. For example, the command

```
dlgopen longfilter="*.txt" replace
```

will load the **`Files of type:`** box with the *.TXT specification (instead of the default file types).

**SUBMIT | NOSUBMIT**
specifies whether the `Submit` check box is checked when the dialog box opens. By default, the `Submit` check box (which indicates that the contents of the opened file should be immediately submitted as a SAS program) is not checked. To automatically submit a file when it is opened, select `Submit contents of file opened` from the Preferences dialog box `General` page.

**IMPORT**
invokes the Import dialog box, allowing you to import graphics files into your SAS session. For more information on importing graphics, see "Importing a Graphics File from within a SAS/GRAPH Window" on page 188.

**VERIFY**
verifies whether the active window contains a `File` pull-down menu with an `Open` item. If it does, the Open dialog box invokes the Open item command instead of invoking the default INCLUDE command.

The VERIFY argument is not valid when specified with ALTCMD or IMPORT.

**ALTCMD='*command*'**
specifies a command to be applied to the file that is selected from the Open dialog box. For example, the command

```
dlgopen altcmd='x' longfilter="*.bat"
```

allows you to select a DOS batch file, which is then run in a DOS shell. The INCLUDE command is the default command.

## Details

The Open dialog box enables you to open files in the default editor. The default editor is determined from the `Use Enhanced Editor` option on the Preferences dialog box `Edit` tabbed page. If this option is selected, the Enhanced Editor is the default editor. Otherwise, the Program Editor is the default editor.

To access the Open dialog box from the pull-down menus, select the `File` menu and then select `Open`.

## See Also

- "Opening Files" on page 108
- "Setting Session Preferences" on page 57
- "Importing Graphics from Other Applications" on page 188
- "FILEOPEN Command" on page 344

## DLGPAGESETUP Command

**Opens the Page Setup dialog box**

**Windows specifics:** all

### Syntax

DLGPAGESETUP

### Details

The Page Setup dialog box allows you to define page attributes such as paper size, source, orientation, and margins.

### See Also

- □ "Setting Up the Printed Page" on page 170

## DLGPREF Command

**Opens the Preferences dialog box**

**Windows specifics:** all

### Syntax

DLGPREF

### Details

The Preferences dialog box allows you to configure your SAS session to accommodate the way that you like to work.

### See Also

- □ "Setting Session Preferences" on page 57

## DLGPRT Command

**Opens the Print dialog**

**Windows specifics:** all

### Syntax

DLGPRT <NOSOURCE | ACTIVEBITMAP | SCREENBITMAP | AWSBITMAP | CLIPBITMAP | CLIPTEXT | ALTCMD='*command*' | BITMAPONLY | NODISPLAY | VERIFY>

**no argument**
prints the active window with the default print settings.

**ACTIVEBITMAP**
suppresses the Print dialog box and prints the active window as a bitmap.

**ALTCMD='*command*'**

uses the Print dialog box to issue a command other than PRINT.

**AWSBITMAP**
suppresses the Print dialog box and prints the main SAS window as a bitmap.

**BITMAPONLY**
allows only bitmap printing from the Print dialog box.

**CLIPBITMAP**
suppresses the Print dialog box and prints the contents of the Windows clipboard as a bitmap.

**CLIPTEXT**
suppresses the Print dialog box and prints the contents of the Windows clipboard as text.

**NODISPLAY**
suppresses the Print dialog box and prints using the default settings.

**NOSOURCE**
prevents the user from specifying a source (application window) from which to print.

**SCREENBITMAP**
suppresses the Print dialog box and prints the entire screen as a bitmap.

**VERIFY**
checks to see if the active application window supports text printing (that is, whether the **`File`** pull-down menu contains a **`Print`** item). If it does not, the Print dialog box allows only bitmap printing.

### Details

The Print dialog box allows you to print the contents of the active window.

### See Also

- "Printing from within a SAS Window" on page 166

## DLGPRTPREVIEW Command

**Invokes the Print Preview window**

**Windows specifics:** all

### Syntax

DLGPRTPREVIEW <VERIFY>

**VERIFY**
checks to see if the active application window supports printing (that is, whether the **`File`** pull-down menu contains a **`Print`** item). If it does not, the Print Preview window will not be displayed. You can still print these windows as bitmaps. Preview the output by issuing the DLGPRT VERIFY command and then clicking **`Preview`**.

### Details

Not all SAS application windows support the Print Preview feature.

### See Also

- □ "Previewing Your Output Before You Print" on page 174
- □ "Printing" on page 166

---

## DLGPRTSETUP Command

**Opens the Print Setup dialog box or programmatically sets printer settings**

**Windows specifics:** all

### Syntax

DLGPRTSETUP <ORIENT=PORTRAIT | LANDSCAPE > <NODISPLAY>

**ORIENT=PORTRAIT | LANDSCAPE**
sets the default page orientation for the current printer. The orient parameter is to support backward compatibility of SAS. The preferred method to specify the orientation is with the ORIENTATION system option.

**NODISPLAY**
suppresses the display of the Print Setup dialog box. This option is intended to be used only when you use other options to explicitly set printer settings.

### Details

The Print Setup dialog box allows you to name the printer to which you want to print, specify that you want to use SAS forms to print, and to access dialog boxes that control how SAS prints information, such as paper orientation, margins, and fonts.

### See Also

- □ "Changing the Printer" on page 169
- □ "Changing the Print Font" on page 170
- □ "Setting Up the Printed Page" on page 170

---

## DLGREPLACE Command

**Opens the Replace dialog box**

**Windows specifics:** all

### Syntax

DLGREPLACE

### Details

The Replace dialog box allows you to find a text string and replace it with another text string.

### See Also

- □ "DLGFIND Command" on page 333

## DLGRUN Command

**Opens the Run dialog**

**Windows specifics:** all

### Syntax

DLGRUN

### Details

The Run dialog box allows you to start another application from within SAS. For example, if you typed excel.exe in the **`Command line:`** field of the Run dialog box, Microsoft Excel would open.

## DLGSAVE Command

**Opens the Save As dialog box**

**Windows specifics:** all

### Syntax

DLGSAVE <LONGFILTER="*filters*" | FILTER='*filters*' <REPLACE>> <EXPORT>
<NOPROMPT> <VERIFY> <ALTCMD='*command*'>

***no arguments***
opens the Save As dialog box with the default settings.

**LONGFILTER="*filters*" | FILTER='*filters*'**
LONGFILTER="*filters*" specifies one or more file filters to use as search criteria for displaying files in the Save as dialog box. The first filter in the argument list is the default filter and is used as the search criteria. All of the filters in the argument list are added to the list of filters in the **`Files of type:`** combo box. To search for

additional file types, you would select another filter from the **`Files of type:`** combo box.

You must enclose the filter in double quotation marks. Note that you can specify long filename extensions that include spaces and single quotes, and each filter that you specify must be separated by a vertical bar (|). For example, if you specify

```
dlgsave longfilter="*.text|*.Bob's work|*.**XX"
```

the dialog displays all files in the current folder that have .TEXT as their file extension, and the dialog box adds *.text, *.Bob's work, and *.**XX to the **`Files of type:`** combo box.

*Note:* When you are using the DLGSAVE command in the DM statement, do not use single quotation marks as part of a longfilter. The DM statement requires single quotation marks around the command it submits. A single quotation mark in the longfilter indicates to the DM statement the end of the command. △

FILTER='*filters*' specifies one or more file filters to use as search criteria for displaying files in the Open dialog box. The first filter in the argument list is the default filter and is used as the search criteria. All of the filters in the argument list are added to the list of filters in the **`Files of type:`** combo box. To search for additional file types, you would select another filter from the **`Files of type:`** combo box. You must enclose the filter list in quotation marks. Separate multiple lists with a space. For example, if you specify

```
dlgsave filter='*.bak *.txt'
```

the dialog box displays all files in the current folder that have a .BAK file extension, and the dialog adds both *.BAK and *.TXT to the **`Files of type:`** combo box.

*Note:* The difference between LONGFILTER="*filters*" and FILTER='*filters*' is that with LONGFILTER="*filters*" you can use spaces and quotes in the filters, where in FILTER='*filters*' you cannot use spaces and quotes. △

**REPLACE**

replaces the filter list with the specified filters instead of concatenating the list with the default filters. This option is valid only when you specify the LONGFILTER= or FILTER= arguments as well. For example, the command

```
dlgsave longfilter="*.txt" replace
```

will load the **`Files of type:`** combo box with only the *.TXT specification (instead of the default file types).

**EXPORT**

invokes the Export dialog box, allowing you to export graphics files from your SAS session. For more information about the Export dialog box, see "Exporting Graphics for Use with Other Applications" on page 189.

**NOPROMPT**

does not prompt the user to replace or append an existing file.

**VERIFY**

verifies whether the active window contains a **`File`** pull-down menu with a **`Save`** item. If it does, the Save As dialog box invokes the **`Save`** item instead of the default FILE command.

The VERIFY argument is not valid when specified with ALTCMD or EXPORT.

**ALTCMD='*command*'**

specifies the command to be applied to the file that is selected from the Save As dialog box. For example, the command

```
dlgsave altcmd='prtfile'
```

sets the file selected from the Save As dialog box as the current print file. The FILE command is the default command.

### Details

The Save As dialog box lets you save the contents of the active window to a file. To access the Save As dialog box from the pull-down menus, select the **`File`** menu and then select **`Save As`**.

### See Also

- □ The Enhanced Editor section, “Saving Files” on page 85
- □ The Program Editor section, “Saving Files” on page 114

## DLGSMAIL Command

**Opens the Send Mail dialog box**

**Windows specifics:** all

### Syntax

DLGSMAIL

### Details

The DLGSMAIL command opens the e-mail dialog box based on the value of the EMAILDLG system option. If the value of the EMAILDLG option is **`sas`**, the DLGSMAIL opens the Send Mail dialog box. If the value of the EMAILDLG option is **`native`**, the DLGSMAIL opens the MAPI-compliant e-mail dialog box.

### See Also

- □ “Sending E-Mail Using SAS” on page 40
- □ “WEMAILFMT Command” on page 360
- □ “EMAILDLG System Option” on page 498

## FILE Command

**Saves the contents of a window to an external file**

**Windows specifics:** valid options

### Syntax

FILE *file-specification* <ENCODING=’*encoding-value*’><*portable-options*>
 <*host-options*>

***file-specification***
specifies a valid Windows external file specification, such as a fileref, a file shortcut, a Windows filename that is enclosed in quotation marks, an environment variable, or an unquoted filename that resides in the current directory.

**ENCODING='*encoding-value*'**
specifies the encoding to use when writing to the output file. The value for ENCODING= indicates that the output file has a different encoding from the current session encoding.

When you write data to the output file, SAS transcodes the data from the session encoding to the specified encoding.

For valid encoding values, see "Encoding Values in SAS Language Elements" in *SAS National Language Support (NLS): User's Guide*.

***portable-options***
specifies one or more portable options, which are documented under the FILE command in SAS Help and Documentation.

***host-options***

BLKSIZE=*block-size*
BLK=*block-size*
specifies the number of bytes that are physically read or written in an I/O operation. The default is 8192. The maximum is 1megabytes.

IGNOREDOSEOF
is used in the context of I/O operations on variable record format files. When this option is specified, any occurrence of ^Z is interpreted as character data and not as an end-of-file marker.

LRECL=*record-length*
specifies the record length (in bytes). Under Windows, the default is 256. The value of *record-length* can range from 1 to 1,048,576 (1 megabyte).

RECFM=*record-format*
controls the record format. Under Windows, the following values are valid:

| | |
|---|---|
| F | indicates fixed format. |
| N | indicates binary format and causes the file to be treated as a byte stream. |
| P | indicates print format. |
| S370V | indicates the variable S370 record format (V). |
| S370VB | indicates the variable block S370 record format (VB). |
| S370VBS | indicates the variable block with spanned records S370 record format (VBS). |
| V\|D | indicates variable format. This is the default. |

## Details

The FILE command writes the entire contents of the active window to an external file without removing text from the window.

If you do not specify a *file-specification*, then SAS uses the filename from the previous FILE or INCLUDE command. In this case, SAS first asks you if you want to overwrite the file. If you have not issued any FILE or INCLUDE commands, you receive an error message indicating no default file exists.

In the Enhanced Editor, if the filename is eight characters or less, the file extension of .SAS is appended to *file-specification*. No extension is appended for a *file-specification* longer than eight characters.

### See Also

- "FILE Command" in the SAS Help and Documentation
- ENCODING System Option in *SAS National Language Support (NLS): User's Guide*
- "Referencing External Files" on page 146
- "Using the FILE Command" on page 159
- For an example of using some of these options, see "Advanced External I/O Techniques" on page 160.

## FILEOPEN Command

**Opens the Open dialog box for the Enhanced Editor or opens a file in the Enhanced Editor**

**Windows specifics** all

### Syntax

FILEOPEN <*"file specification"*>

***"file specification"***
specifies a valid Windows path, filename, and file extension. If the file resides in the current working folder, the path is not required.

### Details

The Open dialog box opens if you do not include a file-specification on the FILEOPEN command. If the FILEOPEN command does include a file-specification, the Open dialog box is bypassed and the file opens in the Enhanced Editor. You must include single or double quotation marks around the specified file.

*Note:* To open a file in the Program Editor, use the DLGOPEN command. △

### See Also

- "Opening Files" on page 83
- "DLGOPEN Command" on page 334

## FILL Command

**Specifies the fill character**

**Windows specifics:** default character

### Syntax

FILL *fill-character*

***fill-character***
specifies the character to be used to fill out a line.

### Details

The fill characters are placed beginning at the current cursor position. Under Windows, the default fill character is an underscore (_).

### See Also

- □ "FILL Command-line Command" in the SAS Help and Documentation

## GSUBMIT Command

**Submits SAS code stored in the Windows clipboard**

**Windows specifics:** valid value for *paste-buffer-name*

### Syntax

GSUBMIT BUF=*paste-buffer-name* | "*SAS-statement-1;...SAS-statement-n;*"

### Details

Under Windows, if the *paste-buffer-name* argument is specified, it must be DEFAULT. The Windows clipboard is the default paste buffer.

SAS statements in the Windows clipboard will not be submitted using the GSUBMIT command if a procedure that you submitted using the Enhanced Editor is still running. You can copy the SAS statements to a new Enhanced Editor window and then submit them.

### See Also

- □ "Using the GSUBMIT Command" on page 160

## HOME Command

**Moves the cursor position from the current position to the home position**

**Windows specifics:** keyboard equivalent

### Syntax

HOME

### Details

Under Windows, the HOME command is equivalent to the HOME key on your keyboard, which moves your cursor between the last cursor position and the home position in the window. If the Command line displays in the window, the home position is the Command line.

You can also define a function key to execute the CURSOR command, which positions the cursor at the home position in the window but has no toggle effect.

### See Also

- □ “HOME Command” in the SAS Help and Documentation

## ICON Command

**Minimizes the active window**

**Windows specifics:** all

### Syntax

ICON <ALL>

**no argument**
specifies that the active window be minimized.

**ALL**
specifies that all windows except the main SAS window be minimized.

### Details

If the window bar is active, the ICON command minimizes windows to the window bar. Otherwise, windows are minimized to the application workspace.

*Note:* Do not confuse this command with the ICON system option, which minimizes the main SAS window. △

### See Also

- □ “ICON Command” in the SAS Help and Documentation

## INCLUDE Command

**Copies lines from an external file into a SAS window**

**Windows specifics:** valid options

### Syntax

INCLUDE *file-specification* <ENCODING='*encoding-value*'><*portable-options*>
<*host-options*>

***file-specification***
specifies a valid Windows external file specification, such as a fileref, a file shortcut, a Windows filename that is enclosed in quotation marks, an environment variable, or an unquoted filename that resides in the current directory.

**ENCODING='*encoding-value*'**
specifies the encoding to use when reading from the external file. The value for ENCODING= indicates that the external file has a different encoding from the current session encoding.

When you read data from an external file, SAS transcodes the data from the specified encoding to the session encoding.

For valid encoding values, see "Encoding Values in SAS Language Elements" in *SAS National Language Support (NLS): User's Guide*.

***portable-options***
specifies one or more portable options, which are documented under the INCLUDE command in SAS Help and Documentation.

***host-options***

BLKSIZE=*block-size*
BLK=*block-size*
specifies the number of bytes that are physically read or written in an I/O operation. The default is 8K. The maximum is 1M.

IGNOREDOSEOF
is used in the context of I/O operations on variable record format files. When this option is specified, any occurrence of ^Z is interpreted as character data and not as an end-of-file marker.

LRECL=*record-length*
specifies the record length (in bytes). Under Windows, the default is 256. The value of *record-length* can range from 1 to 1,048,576 (1 megabyte).

NOTABS
is used only in the context of Dynamic Data Exchange. This option enables you to use nontab character delimiters between variables. For more information on this option, see "Using the NOTAB Option with DDE" on page 279.

RECFM=*record-format*
controls the record format. Under Windows, the following values are valid:

| | |
|---|---|
| F | indicates fixed format. |
| N | indicates binary format and causes the file to be treated as a byte stream. |

| | |
|---|---|
| P | indicates print format. |
| S370V | indicates the variable S370 record format (V). |
| S370VB | indicates the variable block S370 record format (VB). |
| S370VBS | indicates the variable block with spanned records S370 record format (VBS). |
| V\|D | indicates variable format. This is the default. |

### Details

The INCLUDE command copies the entire contents of an external file into the active window.

If you do not specify a *file-specification*, then SAS uses the filename from the previous FILE or INCLUDE command. If you have not issued any FILE or INCLUDE commands, you receive an error message indicating no default file exists.

In the Enhanced Editor, if the filename is eight characters or less, the file extension of .SAS is appended to *file-specification*. No extension is appended for a *file-specification* longer than eight characters.

### See Also

- "INCLUDE Command" in the SAS Help and Documentation
- "ENCODING System Option" in *SAS National Language Support (NLS): User's Guide*
- "Referencing External Files" on page 146
- "Using the INCLUDE Command" on page 159
- For an example of using some of these options, see "Advanced External I/O Techniques" on page 160.

## PMENU Command

**Toggles the command line in the SAS application windows on and off**

**Windows specifics:** command behavior

### Syntax

PMENU <ON | OFF>

**no argument**
toggles the command lines on and off.

**ON**
turns the command lines off.

**OFF**
turns the command lines on.

### Details

Under the Windows operating environment, the pull-down menus and pop-up menus are always enabled. Use can use either the PMENU command or the COMMAND command to specify whether you want the command line to display in SAS windows.

### See Also

- □ "PMENU Command" in the SAS Help and Documentation
- □ "COMMAND Command" on page 328
- □ "WMENUPOP Command" on page 364

## SAVE Command

**Writes the entire contents of the Enhanced Editor, Program Editor, Log, Output, Notepad, and Keys windows to a catalog entry.**

### Syntax

**SAVE** <*catalog-entry*><ATTR> <TABS> <APPEND | REPLACE>

**(no argument)**
writes the contents of the window to the catalog entry that was most recently specified in a COPY or SAVE command during the current SAS session.

***catalog-entry***
specifies the four-level name.

**ATTR**
stores attributes with the entry.

**TABS**
compresses spaces as tabs during storage instead of storing the file with the default spacing.

**APPEND**
appends the contents of the window to the contents of the catalog entry. When it is specified, this catalog entry becomes the default until another catalog entry is specified.

**REPLACE**
replaces the contents of the catalog entry with the contents of the window. Once specified, this replacement becomes the default until another catalog entry is specified.

## STORE Command

**Copies selected text or graphics to the Windows clipboard**

**Windows specifics:** valid options; not supported by the Enhanced Editor

### Syntax

STORE <LAST | ALL>

**LAST**
copies only the most recently marked text and unmarks all other marks when more than one area of text is marked. To store one area of text when more than one mark exists, you must use either the LAST or ALL argument.

**ALL**
stores all current marks when more than one area of text has been marked.

### Details

The STORE command copies marked text or graphics in the active window and stores the copy in the Windows clipboard.

The APPEND and BUFFER= options are not supported under Windows for the STORE command.

### See Also

- "STORE Command" in the SAS Help and Documentation
- "Using the Clipboard" on page 53

## SUBTOP Command

**Submits the first *n* lines of a SAS program for processing**

**Windows specifics:** valid in the Enhanced Editor and the Program Editor

### Syntax

SUBTOP <*n*>

**no argument**
specifies to submit only the top line of the program for processing.

***n***
specifies to submit the first *n* lines of the program for processing.

### Details

When the `Clear text on submit` check box is selected in the Enhanced Editor Options dialog box, all of the submitted lines are deleted from the window when you issue the SUBTOP command.

### See Also

- □ "Submitting Your Program" on page 93

## TOOLCLOSE Command

**Closes the application toolbar or toolbox**

**Windows specifics:** all

### Syntax

TOOLCLOSE

### Details

Use the TOOLCLOSE command to close the toolbar or toolbox.

## TOOLEDIT Command

**Opens the Customize Tools dialog box**

**Windows specifics:** all

### Syntax

TOOLEDIT <*library.catalog.entry*>

**no argument**
edits the currently loaded set of tools.

***library.catalog.entry***
specifies the TOOLBOX entry you want to edit.

### Details

The TOOLEDIT command invokes the Customize Tools dialog box with the TOOLBOX entry specified by *library.catalog.entry*. If a TOOLBOX entry is not specified, the currently loaded set of tools is used.

### See Also

- □ "Customizing the Toolbar" on page 66

## TOOLLARGE Command

**Sets the size of the toolbar or toolbox buttons**

**Windows specifics:** all

### Syntax

TOOLLARGE <ON | OFF>

**no argument**
toggles the size of the toolbar or toolbox buttons between large and normal.

**ON**
sets the size of the toolbar or toolbox buttons to large.

**OFF**
sets the size of the toolbar or toolbox buttons to normal.

### Details

The TOOLLARGE command toggles the size of the toolbar buttons between normal and large. You might find the large buttons easier to use with high-resolution displays.

### See Also

- “Resetting the Tools to the Default Settings” on page 71

## TOOLLOAD Command

**Loads a specific toolbox**

**Windows specifics:** all

### Syntax

TOOLLOAD <WINDOW> <BOX | BAR> <*libref.catalog.member*>

**no arguments**
loads the toolbar for the active window. The tools are displayed as a toolbar or toolbox, depending on the setting in the Customize tools dialog box.

**WINDOW**
associates the toolbox entry you specify with the active window, so that the particular set of tools that you load apply only to that window. This association lasts until you close the window. If you reopen the window later, the window will revert to its default toolbar.

If the WINDOW option is not specified on the TOOLLOAD command, the toolbar or toolbox that is loaded applies to all windows that do not have a specific toolset definition stored for them in the Sasuser.Profile catalog. Such specific toolsets must be named to match the window. For instance, the Explorer window toolset is named Sasuser.Profile.Explorer. If the WINDOW option is not specified, the toolset definition will persist throughout the current SAS session regardless of how many times a particular window is closed and reopened.

**BOX | BAR**
controls whether the icons are displayed as a toolbox in a separate window or as a toolbar integrated with the main SAS window.

***libref.catalog.member***
specifies the catalog entry to load. TOOLBOX is the default catalog entry type.

### Details

After the TOOLLOAD command is processed, the specified toolbox is the active toolbox.

### See Also

- “Customizing and Saving a Toolbar for Use with a Particular Application or Window” on page 70

## TOOLSWITCH Command

**Toggles the tool switching feature on and off**

**Windows specifics:** all

### Syntax

TOOLSWITCH ON | OFF

**ON**
automatically loads the toolbar (if one is defined) for the active window.

**OFF**
uses the default toolbar (Sasuser.Profile.Toolbox) for all windows unless you explicitly load another one.

### Details

The TOOLSWITCH command allows you to switch between a toolbar defined for the active window and the SAS default toolbar.

### See Also

- “Setting Session Preferences” on page 57
- “TOOLEDIT Command” on page 351

## TOOLTIPS Command

**Toggles the ToolTips feature on and off**

**Windows specifics:** all

### Syntax

TOOLTIPS <ON | OFF>

**no argument**
toggles the ToolTips feature on and off.

**ON**
turns the ToolTips feature on.

**OFF**
turns the ToolTips feature off.

### Details

ToolTips are the helpful cues that appear over toolbar or toolbox buttons, (and over some other controls in the main SAS window) as you position the mouse pointer over them.

The TOOLTIPS command specifies whether the ToolTips text is displayed when you move the cursor over an icon in the toolbox or some other control. If you do not specify ON or OFF, the TOOLTIPS command toggles the text on and off, depending on the current setting.

*Note:* Do not confuse ToolTips with ScreenTips. ScreenTips display helpful cues for the status line, the window bar, and tabs in the main SAS window. △

### See Also

- “WSCREENTIPS Command” on page 369

## WATTACH Command

**Toggles whether the contents of the active window are attached to an electronic mail message that you initiate using SAS**

**Windows specifics:** all

### Syntax

WATTACH <ON | OFF>

**no argument**
toggles the attach mode on and off

**ON**
specifies to attach the active window

**OFF**
specifies to not attach the active window

### Details

If you specify ON, the contents of the active window are sent as an attached file. For text windows, the format is either text or RTF (as determined by the WEMAILFMT command or the Preferences dialog box settings). Graphic windows are sent as Windows bitmap (BMP) files.

You can also toggle this setting in the Preferences dialog box `General` page.

### See Also

- "Sending E-Mail Using SAS" on page 40
- "WEMAILFMT Command" on page 360
- "Setting Session Preferences" on page 57

## WATTENTION Command

**Displays the Tasking Manager window, which allows you to select which SAS process to terminate**

**Windows specifics:** all

### Syntax

WATTENTION

### Details

The WATTENTION command allows you to select a SAS process to terminate. This is the equivalent of pressing CTRL + Break.

## WAUTOSAVE Command

**Controls how often SAS automatically saves work in the SAS editor windows**

**Windows specifics:** all

### Syntax

WAUTOSAVE <<ON | OFF> INTERVAL=*minutes*>

**no arguments**
turns the autosave feature on and resets the autosave timer (so that work will automatically be saved after the defined time interval).

**ON | OFF**
specifies to turn the autosave feature on or off.

**INTERVAL=*minutes***
specifies to automatically save work every certain number of minutes. The default interval is 10 minutes. Specify the interval as an integer.

### Details

Use the WAUTOSAVE command if you want SAS to automatically save your work more often or less often than the default interval of every 10 minutes. SAS saves the Program Editor contents to 'pgm.asv' in the current working folder or in the folder specified by the AUTOSAVELOC system option. Contents of the Enhanced Editor windows are saved to the operating environment temporary folder with the filename of 'Autosave of *filename*.$AS'. You can also set the autosave feature in the Preferences dialog box **`Edit`** page.

### See Also

- □ "Setting Session Preferences" on page 57
- □ "AUTOSAVELOC System Option" in *SAS Language Reference: Dictionary*

## WBROWSE Command

**Invokes your preferred web browser**

**Windows specifics:** all

### Syntax

WBROWSE <"*URL*">

**no argument**
invokes the preferred web browser as defined in the Preferences dialog box **`Web`** page.

***URL***
specifies a URL (Uniform Resource Locator) which contains the server and path information needed to find a document on the Internet or on a local intranet.

### Details

By default, the WBROWSE command invokes the default web browser, which displays SAS Institute's home page (www.sas.com). If you specify a URL then that location is displayed instead. Note that you must enclose the URL in double quotations. The default page the web browser opens can be changed in the Preferences dialog box **`Web`** page.

### See Also

□ “Setting Session Preferences” on page 57

## WCOPY Command

**Copies the marked contents of the active window to the Windows clipboard**

**Windows specifics:** all

### Syntax

WCOPY

### Details

WCOPY is intended to be used with the toolbar commands. When you enter the WCOPY command and the active window is a text window, the active window’s menu is searched for a COPY item. If there is a COPY item, the marked contents is copied to the Windows clipboard. If there is no COPY item, WCOPY will execute the STORE command.

### See Also

□ “STORE Command” on page 349

## WCUT Command

**Moves the marked contents of the active window to the Windows clipboard**

**Windows specifics:** all

### Syntax

WCUT

### Details

WCUT is intended to be used with the toolbar commands and is valid only when the active window is an editor window, such as the PROGRAM EDITOR window. When you enter the WCUT command, the active window’s menu is searched for a CUT item. If there is a CUT item, the marked contents of the active window are moved to the Windows clipboard. If there is no CUT item, WCUT will execute the CUT command.

### See Also

- □ “CUT Command” on page 330

## WDOCKVIEW Command

**Toggles the Docking View on and off**

**Windows specifics:** all

### Syntax

WDOCKVIEW <ON | OFF>

**no argument**
toggles the Docking View on and off.

**ON**
turns the Docking View on.

**OFF**
turns the Docking View off

### Details

The Docking View allows for easy navigation within the main SAS window. When the Docking View is enabled, windows that can be docked (integrated with the main SAS window) such as the SAS Explorer and Results windows, display on the left side of the main SAS window. When you click on an item in a docked window that opens another window, such as the output from a procedure listed in the Results window, the window opens on the right side of the main SAS window. You navigate between docked windows using tabs.

### See Also

- □ “Using the Docking View” on page 35
- □ “Setting Session Preferences” on page 57

## WDOCKVIEWMINIMIZE Command

**Minimizes the Docking View window**

**Windows specifics:** all

### Syntax

WDOCKVIEWMINIMIZE

### Details

WDOCKVIEWMINIMIZE minimizes the Docking View window.

### See Also

- □ "WDOCKVIEWRESTORE Command" on page 359
- □ "Using the Docking View" on page 35

## WDOCKVIEWRESIZE Command

**Start Resize mode for moving the Docking View split bar**

**Windows specifics:** all

### Syntax

WDOCKVIEWRESIZE

### Details

When you type WDOCKVIEWRESIZE in the command bar, SAS starts a resize mode. In Resize mode, you can move the Docking View split bar either by using the mouse or by using the left and right arrow keys on the keyboard. When you press the Ctrl key followed by either the left or right arrow keys, the amount of space that the split bar moves is increased. To end Resize mode, press Enter.

You can also start the Docking View Resize mode by typing Alt + W + S or by selecting

Window ► Size Docking View

### See Also

- □ "Resizing the Docking View in the Main SAS Window" on page 78

## WDOCKVIEWRESTORE Command

**Restores the Docking View window from the task bar**

**Windows specifics:** all

### Syntax

WDOCKVIEWRESTORE

### Details

WDOCVIEWRESTORE restores the Docking View window to the left side of the main SAS window.

### See Also

- "WDOCKVIEWMINIMIZE Command" on page 358
- "Using the Docking View" on page 35

## WEDIT Command

**Opens an Enhanced Editor window and optionally enables or disables the Enhanced Editor**

**Windows specifics:** all

### Syntax

WEDIT <"*filename*"> <USE|NOUSE>

**no argument**
opens an Enhanced Editor window.

***"filename"***
specifies the name of a file to open in the Enhanced Editor. The filename should be in double quotation marks. If specified, *filename* must be the first argument.

**USE**
specifies to enable the Enhanced Editor and to open an Enhanced Editor window.

**NOUSE**
specifies to disable the Enhanced Editor. An Enhanced Editor window is not opened.

### Details

When you use the WEDIT command to enable the Enhanced Editor, the **`Use Enhanced Editor`** check box is selected in the **`Edit`** page of the Preferences dialog box. Similarly, when you use the WEDIT command to disable the Enhanced Editor, the **`Use Enhanced Editor`** check box is deselected.

### See Also

- "Setting Session Preferences" on page 57
- "Using the Enhanced Editor" on page 82

## WEMAILFMT Command

**Specifies the format to use when attaching the contents of a text window to an electronic mail message**

**Windows specifics:** all

### Syntax

WEMAILFMT TEXT|RTF

**TEXT**
attaches the contents of the current SAS text window as a plain text file.

**RTF**
attaches the contents of the current SAS text window as a rich text format (RTF) file.

### Details

If the current SAS window contains graphics, the contents of the windows are automatically attached as a Windows bitmap file.

When you use the WEMAILFMT command, the **`Mail current window as attachment`** check box is updated in the General tabbed page of the Preferences dialog box.

### See Also

- □ "Sending E-Mail Using SAS" on page 40
- □ "WATTACH Command" on page 354
- □ "DLGSMAIL Command" on page 342

## WEXITSAVE Command

**Toggles saving your settings when you exit SAS**

**Window specifics:** all

### Syntax

WEXITSAVE <ON | OFF>

**no argument**
toggles the saving of your settings when you exit SAS.

**ON**
saves your settings when you exit SAS.

**OFF**
does not save your settings when you exit SAS.

### Details

You can also toggle this setting in the Preferences dialog box **`General`** page.

### See Also

- □ "Setting Session Preferences" on page 57

## WFILE Command

**Saves the contents of the active window**

**Windows specifics:** all

### Syntax

WFILE

### Details

The WFILE command saves the contents of the active window to a file.

### See Also

- □ Enhanced Editor section "Saving Files" on page 85
- □ Program Editor section "Saving Files" on page 114

## WHIDECURSOR Command

**Suppresses the display of the cursor in SAS windows that do not allow text input**

**Windows specifics:** all

### Syntax

WHIDECURSOR <ON | OFF>

**no argument**
toggles between hiding and displaying the cursor.

**ON**
hides the cursor.

**OFF**
displays the cursor.

### Details

The WHIDECURSOR command inhibits the display of the default text cursor in windows that do not allow text input, such as those in SAS/EIS and SAS/AF software. You can also toggle the WHIDECURSOR setting in the Preferences dialog box **`Advanced`** page.

### See Also

- □ "Setting Session Preferences" on page 57

## WHSBAR Command

**Toggles the horizontal scroll bars on and off**

**Windows specifics:** all

### Syntax

WHSBAR <ON | OFF>

**no argument**
toggles the horizontal scroll bars on and off.

**ON**
displays the horizontal scroll bars.

**OFF**
hides the horizontal scroll bars.

### Details

You can also toggle this setting in the Preferences dialog box **`View`** page.

### See Also

- "Setting Session Preferences" on page 57

## WINSERT Command

**Toggles insert mode on and off**

**Windows specifics:** all

### Syntax

WINSERT <ON | OFF>

**no argument**
toggles the insert mode on and off.

**ON**
enables the insert mode.

**OFF**
enables the overstrike mode.

### Details

You can also toggle this setting by pressing the INSERT key on your keyboard or by modifying the **`Overtype mode`** option in the Preferences dialog box **`Edit`** tabbed page.

### See Also

- □ "Setting Session Preferences" on page 57

## WMENUPOP Command

**Toggles the pop-up menus in the SAS application windows on and off**

**Windows specifics:** all

### Syntax

WMENUPOP <ON | OFF>

**no argument**
toggles the pop-up menus on and off.

**ON**
turns the pop-up menus on.

**OFF**
turns the pop-up menus off.

### Details

By default, the pop-up menus are on. You can access the pop-up menu for a window by clicking the right mouse button inside the window client area.

When used with the -NOAWSMENU system option, this command makes all menu selections unavailable to the user. This can be a useful technique when developing SAS/AF applications in which you want to restrict the actions of the end user.

### See Also

- □ "AWSMENU System Option" on page 488
- □ "PMENU Command" on page 348
- □ "WPOPUP Command" on page 368

## WMRU Command

**Retains the names of the most recently used files in the File menu**

**Windows specifics:** all

### Syntax

WMRU <<ON> <NUM=*number-of-filenames*><CASCADE>>| <OFF>

**no arguments**
toggles the file list on and off.

***ON NUM=number-of-filenames***
turns the file list on and maintains *number-of-filenames* filenames in the list. The *number-of-filenames* argument can be an integer from 1 to 30. If you omit *number-of-filenames*, the last number specified for the most recently used files is used.

**CASCADE**
specifies that the most recently used files list can be accessed from the **`File`** menu **`Recent Files`** submenu.

**OFF**
turns the file list off.

### Details

When you open or save a file using the Open or Save As dialog boxes, SAS adds the filename to the recently used file list in the **`File`** pull-down menu or the **`Recent Files`** submenu. You can open a recently used file in a SAS editor window by making the editor the active window and selecting its name from the **`File`** menu or the **`Recent Files`** submenu. By default, SAS retains four filenames in the list.

You can also configure these settings in the Preferences dialog box **`General`** page.

### See Also

- "Setting Session Preferences" on page 57

## WNAVKEYUNMARK Command

**Toggles the setting for enabling unmarking of text using navigational keys**

**Windows specifics:** all

### Syntax

WNAVKEYUNMARK <ON | OFF>

**no argument**
toggles the **`Enable unmarking with navigation keys`** setting on and off.

**ON**
turn the **`Enable unmarking with navigation keys`** setting on.

**OFF**
turns the **`Enable unmarking with navigation keys`** setting off.

### Details

You can access the **`Enable unmarking with navigation keys`** setting by selecting

Tools ► Options ► Preferences ► Edit

When the **`Enable unmarking with navigation keys`** setting is selected, you can unmark text by using the up, down, left, and right navigation keys.

### See Also

- □ "Setting Session Preferences" on page 57

## WNEWTITLE Command

**Clears the contents of the active window and removes its title**

**Windows specifics:** all

### Syntax

WNEWTITLE

### Details

When you save the contents of a SAS window to a file, SAS assigns the filename as the title of the window. You can use the WNEWTITLE command to clear the active window and remove that title (reverting to Untitled).

If used in the LOG or OUTPUT window, this command clears the contents of the window and changes the name to Untitled. If this command is used in the Program Editor window, SAS prompts you to save the contents of the window before clearing it and removing the title. If this command is used in the Enhanced Editor window, SAS opens a new, untitled, Enhanced Editor window.

## WNEXTEDIT Command

**Toggles between all Enhanced Editor windows that are currently open**

**Windows specifics:** all

### Syntax

WNEXTEDIT

### Details

You can use the WNEXTEDIT command to move between Enhanced Editor windows.

### See Also

- “WPGM Command” on page 367

## WPASTE Command

**Pastes the contents of the Windows clipboard into the active window**

**Windows specifics:** all

### Syntax

WPASTE

### Details

WPASTE is intended to be used with the toolbar commands. When you enter the WPASTE command, the active window’s menu is searched for a PASTE item. If there is a PASTE item and the clipboard contains text, WPASTE will execute as if you selected PASTE from the pull-down menu. If there is no PASTE item, WPASTE will execute the PASTE command.

### See Also

- “PASTE Command” in the SAS Help and Documentation

## WPGM Command

**Changes the active window to the editor window that was most recently edited**

**Windows specifics:** all

### Syntax

WPGM

### Details

The behavior of the WPGM command depends on the setting of the **`Use Enhanced Editor`** check box. The check box is available from the **`Edit`** tab in the Preferences dialog box. If the **`Use Enhanced Editor`** check box is selected and you issue the WPGM command, the active window becomes the Enhanced Editor window that was most recently edited. If the **`Use Enhanced Editor`** check box is not selected, the active window becomes the Program Editor.

Issuing the WPGM command repeatedly displays the open Enhanced Editor windows in the order of the most recently edited window to the least recently edited.

### See Also

- "WNEXTEDIT Command" on page 366

## WPOPUP Command

**Causes the pop-up menus for a window to appear**

**Windows specifics:** all

### Syntax

WPOPUP

### Details

You can access the pop-up menu for a window by clicking the right mouse button inside the window client area. By default under Windows, this command is associated with the right mouse button.

### See Also

- "WMENUPOP Command" on page 364

## WRTFSAVE Command

**Saves the contents of the current text window to an RTF file**

**Windows specifics:** all

### Syntax

WRTFSAVE "*filename*" <NOPROMPT>

***filename***
is a required argument and can include a file path. If you specify a filename without a path, the file is saved in the current SAS working folder. The WRTFSAVE command does not automatically append the .RTF file extension. If you want the resulting filename to end in .RTF, be sure to include it as part of the filename that you specify.

**NOPROMPT**
specifies that if a file with the same filename already exists, that file will be overwritten without prompting you with a confirmation dialog box.

### Details

The WRTFSAVE command saves the contents of the active window in .RTF format. The contents of the active window must be text. This command performs the same action as the Save As dialog box when you select **`.rtf`** from the **`Save file as type`** list. However, WRTFSAVE saves the file without displaying an intermediate dialog box.

## WSCREENTIPS Command

**Toggles the ScreenTips on and off**

**Windows specifics:** all

### Syntax

WSCREENTIPS <ON | OFF>

**no argument**
toggles the ScreenTips on and off.

**ON**
displays helpful cues for the status line, window bar and tabs within the main SAS window.

**OFF**
turns off the ScreenTips.

### Details

ScreenTips are the helpful cues that appear over the status line, window bar and tabs in the main SAS window as you position the mouse pointer over them.

The WSCREENTIPS command specifies whether the ScreenTips text is displayed when you move the cursor over the status line, window bar, or tabs in the main SAS window.

*Note:* Do not confuse ScreenTips with ToolTips. ToolTips display helpful cues for tools. You can also toggle the ScreenTips setting in the Preferences dialog box **`View`** page. △

### See Also

- "TOOLTIPS Command" on page 354
- "Setting Session Preferences" on page 57

## WSTATUSLN Command

**Toggles the status line on and off, and specifies the area proportions**

**Windows specifics:** all

### Syntax

WSTATUSLN <ON | OFF> <ALL | MSG<=*percent-msg*> |
CDIR<=*percent-cdir*><CURPOS>>

**no arguments**
toggles the status line on and off.

The first argument is optional, but if you specify it, you must include it before the second group of options:

**ON**
displays the status line in its most recent active state. If the status line includes a message area, the message lines in the SAS application windows are disabled. ON is the default setting.

**OFF**
turns off the status line. This enables the message lines in the SAS application windows.

The second group of arguments is also optional. Note that specifying these options without specifying the ON option first does not automatically turn the status line on if it is currently off.

**ALL**
includes both the message area and the current folder areas on the status line. If you do not specify the MSG and CDIR options with percentage values, the status line proportions revert to the most recent settings. ALL is the default setting.

**MSG<=*percent-msg*>**
includes the message area as part of the status line. If you specify this option without the CDIR option, the message area occupies the entire status line. If you specify a percentage with this option and with the CDIR option, the message area occupies the proportion of the line that you specify.

**CDIR<=*percent-cdir*>**
includes the current folder as part of the status line. If you specify this option without the MSG option, the current folder area occupies the entire status line. If you specify a percentage with this option and with the MSG option, the current folder area occupies the proportion of the line that you specify.

**CURPOS**
includes the Enhanced Editor cursor position (line and column) in the status line when the Enhanced Editor is the active window.

## Details

The WSTATUSLN command specifies whether the status line of the active window is on or off and specifies the proportions of the status line that the message area and the current folder area occupy. You can also toggle the status line in the Preferences dialog box `View` page.

## Example

To display a status line that is evenly divided between the message display and the current folder display, issue the following command:

```
wstatusln on msg=50 cdir=50
```

## See Also

- "Setting Session Preferences" on page 57

## WUNDO Command

**Undoes the last CUT, COPY, or PASTE toolbar action**

**Windows specifics:** all

### Syntax

WUNDO

### Details

When you enter the WUNDO command, the active window's menu is searched for an undo item. If there is an undo item, WUNDO will execute as if you selected **UNDO** from the pull-down menu. If there is no undo item, WUNDO will execute the UNDO command. Some windows may not have an undo command.

## WVSBAR Command

**Toggles the vertical scroll bars on and off**

**Windows specifics:** all

### Syntax

WVSBAR <ON | OFF>

**no argument**
toggles the vertical scroll bars on and off.

**ON**
displays the vertical scroll bars.

**OFF**
hides the vertical scroll bars.

### Details

You can also toggle this setting in the Preferences dialog box **View** page.

### See Also

- "Setting Session Preferences" on page 57

## WWINDOWBAR Command

**Toggles the window bar on and off**

**Windows specifics:** all

### Syntax

WWINDOWBAR <ON | OFF>

**no argument**
toggles the window bar on or off.

**ON**
displays the window bar in the main SAS window.

**OFF**
does not display the window bar in the main SAS window.

### Details

If the window bar is on, it displays at the bottom of the main SAS window just above the status bar. SAS windows minimize to the window bar. You can bring a window to the front by clicking on the window's button in the window bar. To open a file in an open application, such as one of the editors, you drag the file to the application's button in the window bar (which brings the application to the front) and then drag the file to the application's window. When the window bar is off, SAS windows minimize to small title bars.

You can also turn the window bar on and off using the Preferences dialog box **`View`** page.

### See Also

- "Customizing Your Windowing Environment with Commands" on page 62
- "Setting Session Preferences" on page 57

## X Command

**Enters host-system mode or allows you to submit a Windows command without ending your SAS session**

**Windows specifics:** valid values for *command* syntax

### Syntax

X <'*command* '>;

**no argument**

open a DOS command window.

***command***
specifies the command that you want to execute.

### Details

This form of the X command issues one command. The command is passed to the operating environment and executed. If errors occur, the appropriate error messages are displayed.

### See Also

- "X Command" in the SAS Help and Documentation
- "Running Windows or MS-DOS Commands from within SAS" on page 25
- "XCMD System Option" on page 575
- "XSYNC System Option" on page 576
- "XWAIT System Option" on page 577

## ZOOM Command

**Maximizes the active window**

**Windows specifics:** all

### Syntax

ZOOM <ON | OFF>

**ON**
maximizes the active window.

**OFF**
returns the active window to the default size.

### Details

When you maximize one application window, the SAS windowing environment enters a maximized mode. As you switch between active windows, each window you select will be maximized. When you restore one of the application windows to its original size, all windows are restored.

### See Also

- "ZOOM Command" in the SAS Help and Documentation

CHAPTER

# 17

# Formats under Windows

*SAS Formats under Windows* **375**
*Writing Binary Data* **375**
*Converting User-Written Formats from Earlier Releases to SAS 9.1* **376**
*HEXw. Format* **376**
*$HEXw. Format* **377**
*IBw.d Format* **377**
*PDw.d Format* **379**
*PIBw.d Format* **380**
*RBw.d Format* **381**
*ZDw.d Format* **382**

## SAS Formats under Windows

A SAS format is an instruction or template that SAS uses to write data values. Most SAS formats are described completely in *SAS Language Reference: Dictionary*. The formats that are described here have behavior that is specific to Windows.

Many of the SAS formats that have details specific to the Windows operating environment are used to write binary data. In using these formats, it is important that you understand the concepts that are presented in "Writing Binary Data" on page 375.

If you have formats that you created for use in earlier releases of SAS, see "Converting User-Written Formats from Earlier Releases to SAS 9.1" on page 376 for information about how to convert those formats for use with SAS 9.1.

## Writing Binary Data

IBM mainframes, Hewlett Packard 9000, and most other UNIX systems store bytes in one order, called big-endian. Intel-based, or IBM compatible microcomputers and the VAX and Alpha computers manufactured by Compaq store bytes in a different order called byte-reversed, or little-endian.

Binary data stored in one order cannot be read by a computer that stores binary data in the other order without additional processing taking place. When you are designing SAS applications, try to anticipate how your data will be read and choose your formats and informats accordingly.

SAS provides two sets of informats for reading binary data and corresponding formats for writing binary data.

- The IB*w.d*, PD*w.d*, PIB*w.d*, and RB*w.d* informats and formats read and write in native mode, that is, using the byte-ordering system that is standard for the machine.

- □ The S370FIB*w.d*, S370FPD*w.d*, S370FRB*w.d*, and S370FPIB*w.d* informats and formats read and write according to the IBM 370 standard, regardless of the native mode of the machine. These informats and formats allow you to write SAS programs that can be run in any SAS environment, regardless of how numeric data are stored.

If a SAS program that reads and writes binary data runs on only one type of machine, you can use the native mode informats and formats. However, if you want to write SAS programs that can be run on multiple machines using different byte-storage systems, use the IBM 370 formats and informats. The purpose of the IBM 370 informats and formats is to enable you to write SAS programs that can be run in any SAS environment, no matter what standard you use for storing numeric data.

For example, suppose you have a program that writes data with the PIB*w.d* format. You execute the program on a microcomputer so that the data are stored in byte-reversed mode. Then on the microcomputer you run another SAS program that uses the PIB*w.d* informat to read the data. The data are read correctly because both of the programs are run on the microcomputer using byte-reversed mode. However, you cannot upload the data to a Hewlett Packard 9000-series machine and read the data correctly because they are stored in a form native to the microcomputer but foreign to the Hewlett Packard 9000. To avoid this problem, use the S370FPIB*w.d* format to write the data; even on the microcomputer, this causes the data to be stored in IBM 370 mode. Then read the data using the S370FPIB*w.d* informat. Regardless of what type of machine you use when reading the data, they are read correctly.

## Converting User-Written Formats from Earlier Releases to SAS 9.1

You must convert Release 6.09E and Release 6.12 user-written formats to their SAS 9.1 counterparts before you can use them in a SAS 9.1 program. You can convert these catalogs by using the CPORT and CIMPORT procedures. For more information on the CPORT and CIMPORT procedures, see *Base SAS Procedures Guide*.

## HEX*w*. Format

**Converts real binary (floating-point) values to hexadecimal values**

**Category** numeric

**Width range:** 1–16

**Default width:** 8

**Alignment:** left

**Windows specifics:** native floating–point representation

**See:** HEXw. in *SAS Language Reference: Dictionary*

### Syntax

HEX*w*.

***w***

specifies the width of the output field. When you specify a *w* value of 1 through 15, the real binary number is truncated to a fixed-point integer before being converted to hexadecimal notation. When you specify 16 for the *w* value, the floating-point value of the number is used; in other words, the number is not truncated.

### See Also

- "$HEX*w*. Format" on page 377
- "HEX*w*. Informat" on page 415

## $HEX*w*. Format

**Converts character values to hexadecimal values**

**Category** character

**Width range:** 1–32767

**Default width:** 4

**Alignment:** left

**Windows specifics:** ASCII character–encoding system

**See:** $HEXw. in *SAS Language Reference: Dictionary*

### Syntax

$HEX*w*.

***w***
specifies the width of the output field.

### Details

The $HEX*w*. format is like the HEX*w*. format in that it converts a character value to hexadecimal notation, with each byte requiring two columns. Under Windows, the $HEX*w*. format produces hexadecimal representations of ASCII codes for characters.

### See Also

- "HEX*w*. Format" on page 376
- "$HEX*w*. Informat" on page 416

## IB*w.d* Format

**Writes integer binary (fixed-point) numbers**

**Category** numeric

**Width range:** 1–8

**Default width:** 4

**Decimal range:** 0–10

**Alignment:** left

**Windows specifics:** native floating-point representation

**See:** IBw.d in *SAS Language Reference: Dictionary*

## Syntax

IB*w*.*d*

***w***

specifies the width of the output field in bytes (not digits).

***d***

optionally specifies a scaling factor. When you specify a *d* value, the IB*w*.*d* format multiplies the number by $10^d$, and then applies the integer binary format to that value.

## Details

The IB*w*.*d* format converts a double-precision number and writes it as an integer binary (fixed-point) value. Integers are stored in integer-binary (fixed-point) form.

For more information about microcomputer fixed-point values, see Intel's developer Web site.

## Examples

**Example 1: Processing a Positive Number** If you format 1.0 as the double-precision number, it is stored as an integer:

```
01 00 00 00 00 00 00 00
```

(Remember, Windows stores binary data in byte-reversed order.) The value written depends on the *w* value you specify.

If you specify the IB4. format, you receive the following value:

```
01 00 00 00
```

If you specify the IB2. format, you receive the following value:

```
01 00
```

**Example 2: Processing a Negative Number** If you try to format –1 with the IB4. format, you receive the following value:

```
FF FF FF FF
```

If you specify the IB2. format, you receive the following value:

```
FF FF
```

**Example 3: Processing a Number That Is Too Large to Format** When a numeric value is too large to format, the result is largest interger value that can be stored in four bytes, which is 2,147,483,647.

In the following code

```
data a;
   x = 9999999999999999999;
```

```
      y = put(x, IB8.);
      put y = hex16.;
   run;
```

SAS returns the hexadecimal representation of 2147483647

```
 y=FFFFFFFFFFFFFF7F
```

## See Also

- □ "IB*w.d* Informat" on page 416
- □ "Writing Binary Data" on page 375

## PD*w.d* Format

**Writes packed decimal data**

**Category** numeric

**Width range:** 1–16

**Default width:** 1

**Decimal range:** 1–31

**Alignment:** left

**Windows specifics:** How the values are interpreted as negative or positive

**See:** PDw.d in *SAS Language Reference: Dictionary*

### Syntax

PD*w.d*

***w***
specifies the width of the output field in bytes (not digits).

***d***
optionally specifies a scaling factor. When you specify a *d* value, the PD*w.d* format multiplies the number by $10^d$, and then applies the packed decimal format to that value.

### Details

The PD*w.d* format writes double-precision numbers in packed decimal format. In packed decimal data, each byte contains two digits. The *w* value represents the number of bytes, not the number of digits. The value's sign is in the uppermost bit of the first byte (although the entire first byte is used for the sign).

### Examples

**Example 1: Processing a Positive Number** If you format 1143.0 using the PD2. format, you receive the following value:

```
00 43
```

If you specify PD4., you receive the following value:

```
00 00 11 43
```

**Example 2: Processing a Negative Number** If you format –1143.0 using the PD2. format, you receive the following value:

```
80 43
```

If you specify the PD4. format, you receive the following value:

```
80 00 11 43
```

**Example 3: Processing a Number That Is Too Large To Format** When a numeric value is too large to format, as in this example

```
data a;
   x = 1i308;
   y = put(x, PD16.2);
   put y = hex16.;
run;
```

the result is

```
y=0099999999999999
```

## See Also

- □ "PD*w.d* Informat" on page 418
- □ "Writing Binary Data" on page 375

## PIB*w.d* Format

**Writes positive integer binary data**

**Category** numeric

**Width range:** 1–8

**Default width:** 1

**Decimal range:** 0–10

**Alignment:** left

**Windows specifics:** native byte-swapped integers

**See:** PIBw.d in *SAS Language Reference: Dictionary*

### Syntax

PIB*w.d*

***w***
specifies the width of the output field in bytes (not digits).

***d***
optionally specifies a scaling factor. When you specify a *d* value, the PIB*w.d* format multiplies the number by $10^d$, and then applies the positive integer binary format to that value.

## Details

The PIB*w.d* format converts a fixed-point value to an integer binary value. If the fixed-point value is negative, the PIB*w.d* format writes the integer representation for –1.

For more information about microcomputer fixed-point values, see Intel's developer Web site.

## Examples

**Example 1: Processing a Number That Is Too Large To Format** When a numeric value is too large to format, the result is the largest integer value that can be stored in four bytes, which is 2,147,483,647.

In the following code

```
data a;
   x = 9999999999999999999;
   y = put(x, PIB8.);
   put y = hex16.;
run;
```

SAS returns the hexadecimal representation of 2147483647

```
y=FFFFFFFFFFFFFF7F
```

## See Also

- □ "PIB*w.d* Informat" on page 418
- □ "Writing Binary Data" on page 375

# RB*w.d* Format

**Writes real binary (floating-point) data**

**Category** numeric

**Width range:** 2–8

**Default width:** 4

**Decimal range:** 0–10

**Alignment:** left

**Windows specifics:** native floating–point representation

**See:** RBw.d in *SAS Language Reference: Dictionary*

## Syntax

RB*w.d*

***w***

specifies the width of the output field.

***d***

optionally specifies a scaling factor. When you specify a *d* value, the RB*w.d* format multiplies the number by $10^d$, and then applies the real binary format to that value.

## Details

The RB*w.d* format writes numeric data in real binary (floating-point) notation. Numeric data for scientific calculations are commonly represented in floating-point notation. (SAS stores all numeric values in floating-point notation.) A floating-point value consists of two parts: a mantissa that gives the value and an exponent that gives the value's magnitude.

Real binary is the most efficient format for representing numeric values because SAS already represents numbers this way and no conversion is needed.

For more information about Windows floating-point notation, see Intel's developer Web site.

## Examples

**Example 1: Processing a Number That Is Too Large To Format** When a numeric value is too large to format, as in this example

```
data a;
   x = 1i308;
   y = put(x, RB8.2);
   put y = hex16.;
run;
```

the result is

```
y=0000000000D1FFFF
```

## See Also

- "RB*w.d* Informat" on page 420
- "Writing Binary Data" on page 375

## ZD*w.d* Format

**Writes zoned decimal data**

**Category** numeric

**Width range:** 1–32

**Default width:** 1

**Decimal range:** 1–10

**Alignment:** left

**Windows specifics:** Last byte includes the sign.

**See:** ZDw.d in *SAS Language Reference: Dictionary*

## Syntax

ZD*w.d*

***w***
: specifies the number of bytes (not the number of digits).

***d***
: optionally specifies the number of digits to the right of the decimal point in the numeric value.

## Details

The ZD*w.d* format writes zoned decimal data. This is also known as an overprint trailing numeric format. In the Windows operating environment, the last byte of the field contains the sign information of the number. The following table gives the conversion for the last byte.

| Digit | ASCII Character | Digit | ASCII Character |
|---|---|---|---|
| 0 | { | –0 | } |
| 1 | A | –1 | J |
| 2 | B | –2 | K |
| 3 | C | –3 | L |
| 4 | D | –4 | M |
| 5 | E | –5 | N |
| 6 | F | –6 | O |
| 7 | G | –7 | P |
| 8 | H | –8 | Q |
| 9 | I | –9 | R |

## Examples

**Example 1: Processing a Number That Is Too Large To Format** When a numeric value is too large to format, as in this example

```
data a;
   x = 1e308;
   y = put(x, ZD32.2);
   put y = hex16.;
run;
```

the result is a sequence of 39s:

```
y=3939393939393939
```

## See Also

□ "ZD*w.d* Informat" on page 420

# CHAPTER 18 Functions and CALL Routines under Windows

*SAS Functions and Call Routines under Windows* **385**
*BYTE Function* **386**
*CALL SYSTEM Routine* **387**
*COLLATE Function* **388**
*DINFO Function* **390**
*DOPEN Function* **391**
*DOPTNAME Function* **392**
*DOPTNUM Function* **392**
*FDELETE Function* **393**
*FEXIST Function* **394**
*FILEEXIST Function* **394**
*FILENAME Function* **395**
*FILEREF Function* **396**
*FINFO Function* **396**
*FOPTNAME Function* **400**
*FOPTNUM Function* **402**
*LIBNAME Function* **403**
*MCIPISLP Function* **404**
*MCIPISTR Function* **405**
*MODULE Function* **406**
*PEEKLONG Function* **408**
*RANK Function* **408**
*SLEEP Function* **409**
*TRANSLATE Function* **410**
*WAKEUP Function* **411**

## SAS Functions and Call Routines under Windows

A SAS function returns a value from a computation or system operation. SAS CALL routines are used to alter variable values or perform other system functions. Most SAS functions and CALL routines are completely described in the SAS functions and CALL routines portion of *SAS Language Reference: Dictionary*. The functions and CALL routines that are described here have syntax or behavior specific to the Windows operating environment.

## BYTE Function

**Returns one character in the ASCII collating sequence**

**Category:** Character

**Windows specifics:** Uses the ASCII code sequence

**See:** BYTE Function in *SAS Language Reference: Dictionary*

### Syntax

**BYTE**(*n*)

***n***
specifies an integer that represents a specific ASCII character. The value of *n* can range from 0 to 255.

### Details

If the BYTE function returns a value to a variable that has not yet been assigned a length, by default the variable is assigned a length of 1.

Because Windows is an ASCII system, the BYTE function returns the *n*th character in the ASCII collating sequence. The value of *n* can range from 0 to 255.

Any programs using the BYTE function with characters above ASCII 127 (the hexadecimal notation is `'7F'x`) may return a different value when used on a PC from another country as characters above ASCII 127 are national characters and they vary from country to country.

## CALL SOUND Routine

**Generates a sound with a specific frequency and duration**

**Category:** Special

**Windows specifics:** all

### Syntax

**CALL SOUND**(*frequency*,*duration*);

***frequency***
specifies the sound frequency in terms of cycles per second. The frequency must be at least 20 and no greater than 20,000.

***duration***
specifies the sound duration in milliseconds. The default is -1.

### Example

**Example 1: Producing a Tone** The following statement produces a tone of frequency 523 cycles per second (middle C) lasting 2 seconds:

```
data _null_;
   call sound(523,2000);
run;
```

## CALL SYSTEM Routine

**Submits an operating system command or a Windows application for execution**

**Category:** Special

**Windows specifics:** *command* must be a valid Windows command

**See:** CALL SYSTEM Routine in *SAS Language Reference: Dictionary*

### Syntax

**CALL SYSTEM**(*command*);

***command***

can be any of the following:

- an operating system command enclosed in quotes or the name of a Windows application that is enclosed in quotes.
- an expression whose value is an operating system command or the name of a Windows application.
- the name of a character variable whose value is an operating system command or the name of a Windows application.

### Details

If you are running SAS interactively, the command executes in a command prompt window. By default, you must type **`exit`** to return to your SAS session.

### Comparison

The CALL SYSTEM routine is similar to the X command. However, the CALL SYSTEM routine is callable and can therefore be executed conditionally.

The values of the XSYNC and XWAIT system options affect how the CALL SYSTEM routine works.

### Examples

**Example 1: Executing Operating System Commands Conditionally** If you want to execute operating system commands conditionally, use the CALL SYSTEM routine:

```
options noxwait;
data _null_;
   input flag $ name $8.;
```

```
      if upcase(flag)='Y' then
         do;
            command='md c:\'||name;
            call system(command);
         end;
      cards;
   Y mydir
   Y junk2
   N mydir2
   Y xyz
   ;
```

This example uses the value of the variable FLAG to conditionally create directories. After the DATA step executes, three directories have been created: C:\MYDIR, C:\JUNK2, and C:\XYZ. The directory C:\MYDIR2 is not created because the value of FLAG for that observation is not **Y**.

The X command is a global SAS statement. Therefore, it is important to realize that you cannot conditionally execute the X command. For example, if you submit the following code, the X statement is executed:

```
data _null_;
   answer='n';
   if upcase(answer)='y' then
      do;
         x 'md c:\extra';
      end;
run;
```

In this case, the directory C:\EXTRA is created regardless of whether the value of ANSWER is equal to **'n'** or **'y'**.

**Example 2: Obtaining a Directory Listing** You can use the CALL SYSTEM routine to obtain a directory listing:

```
data _null_;
   call system('dir /w');
run;
```

In this example, the /W option for the DIR command instructs Windows to print the directory in the wide format instead of a vertical list format.

## See Also

- "X Command" on page 372
- "XSYNC System Option" on page 576
- "XWAIT System Option" on page 577

## COLLATE Function

**Returns an ASCII collating sequence character string**

**Category:** Character

**Windows specifics:** Uses the ASCII code sequence

**See:** COLLATE Function in *SAS Language Reference: Dictionary*

## Syntax

**COLLATE** (*start-position*<,*end-position*>) |
(*start-position*<,,*length*>)

***start-position***
specifies the numeric position in the collating sequence of the first character to be returned.

***end-position***
specifies the numeric position in the collating sequence of the last character to be returned.

***length***
specifies the number of characters in the returned collating sequence.

## Details

The COLLATE function returns a string of ASCII characters that range in value from 0 to 255. The string that is returned by the COLLATE function begins with the ASCII character that is specified by the *start-position* argument. If the *end-position* argument is specified, the string returned by the COLLATE function contains all the ASCII characters between the *start-position* and *end-position* arguments. If the *length* argument is specified instead of the *end-position* argument, then the COLLATE function returns a string that contains a value for *length*. The returned string ends, or truncates, with the character having the value 255 if you request a string length that contains characters exceeding this value.

The default length of the return string value is 200 characters. To return a length of 201 to 256 ASCII characters, use a format such as $256 for the return string variable or explicitly define the variable's length, such as **length y $260**.

Any programs using the COLLATE function with characters above ASCII 127 (the hexadecimal notation is **'7F'x**) may return a different value when used on a PC from another country. Characters above ASCII 127 are national characters and they vary from country to country.

## Examples

**Example 1: Returning an ASCII String Using the Return Variable Default String Length** In this example, the return code variable **y** uses the default return string length of 200. Therefore, the COLLATE function returns 200 characters of the collating sequence.

```
data _null_;
   y = collate(1,256);
   put y;
run;
```

**Example 2: Returning an ASCII String Larger Than the Default Return Variable String Length** By formatting the return code variable to a length greater than 256, the COLLATE function returns 256 characters of the collating sequence.

```
data _null_;
   format y $260.;
```

```
      y = collate(1,256);
      put y;
   run;
```

**Example 3: Returning an ASCII String of a Specific Length** In this example, the return code variable **y** uses a return string length of 56, and the COLLATE function returns the first 56 characters of the collating sequence.

```
data _null_;
   y = collate(,,56);
   put y;
run;
```

## DINFO Function

**Returns information about a directory**

**Category:** External Files

**Windows specifics:** directory pathname is the only information available

**See:** DINFO Function in *SAS Language Reference: Dictionary*

### Syntax

**DINFO**(*directory-id*, *info-item*)

***directory-id***
specifies the identifier that was assigned when the directory was opened, generally by the DOPEN function.

***info-item***
specifies the information item to be retrieved. DINFO returns a blank if the value of *info-item* is invalid.

### Details

Directories that are opened with the DOPEN function are identified by a directory–id. Use DOPTNAME to determine the names of the available system–dependent directory information items. Use DOPTNUM to determine the number of directory information items available.

Under Windows, the only *info-item* that is available is Directory, which is the pathname of *directory-id*. If *directory-id* points to a list of concatenated directories, then Directory is the list of concatenated directory names.

### Example of Obtaining Directory Information

```
data a;
  rc=filename("tmpdir", "c:");
  put "rc = 0 if the directory exists: " rc=;
  did=dopen("tmpdir'');
  put did=;
```

```
    numopts=doptnum(did);
    put numopts=;
    do i = 1 to numopts;
      optname = doptname(did,i);
      put i= optname=;
      optval=dinfo(did,optname);
      put optval=;
    end;
  run;
```

**Output 18.1** The SAS Log Displays the Directory Information

```
NOTE: PROCEDURE PRINTTO used (Total process time):
      real time           0.03 seconds
      cpu time            0.00 seconds

446  data a;
447    rc=filename("tmpdir", "c:");
448    put "rc = 0 if the directory exists: " rc=;
449    did=dopen("tmpdir");
450    put did=;
451    numopts=doptnum(did);
452    put numopts=;
453    do i = 1 to numopts;
454      optname = doptname(did,i);
455      put i= optname=;
456      optval=dinfo(did,optname);
457      put optval=;
458    end;
459  run;
rc = 0 if the directory exists: rc=0
did=1
numopts=1
i=1 optname=Directory
optval=C:\TEMP\elimal
NOTE: The data set WORK.A has 1 observations and 6 variables.
NOTE: DATA statement used (Total process time):
      real time           0.08 seconds
      cpu time            0.04 seconds

460  proc printto; run;
```

## See Also

- "DOPEN Function" on page 391

## DOPEN Function

**Opens a directory and returns a directory identifier value**

**Category:** External Files

**Windows specifics:** *fileref* can be assigned by using an environment variable

**See:** DOPEN Function in *SAS Language Reference: Dictionary*

### Syntax

**DOPEN**(*fileref*)

***fileref***
specifies the fileref that is assigned to the directory.

### Details

DOPEN opens a directory and returns a directory identifier value (a number greater than 0) that is used to identify the open directory in other SAS external file access functions. If the directory could not be opened, DOPEN returns 0. The directory to be opened must be identified by a fileref.

## DOPTNAME Function

**Returns the name of a directory information item**

**Category:** External Files

**Windows specifics:** directory is the only item available

**See:** DOPTNAME Function in *SAS Language Reference: Dictionary*

### Syntax

**DOPTNAME**(*directory-id*, *nval* )

***directory-id***
specifies the identifier that was assigned when the directory was opened, generally by the DOPEN function.

***nval***
specifies the sequence number of the option.

### Details

Under Windows, the only directory information item that is available is Directory, which is the pathname of *directory-id*. The *nval*, or sequence number, of Directory is 1. If *directory-id* points to a list of concatenated directories, then Directory is the list of concatenated directory names.

### Example

For an example of using DOPTNAME, see "Example of Obtaining Directory Information" on page 390.

## DOPTNUM Function

**Returns the number of information items that are available for a directory**

**Category:** External Files

**Windows specifics:** directory is the only item available

**See:** DOPTNUM Function in *SAS Language Reference: Dictionary*

## Syntax

**DOPTNUM**(*directory-id*)

***directory-id***
specifies the identifier that was assigned when the directory was opened, generally by the DOPEN function.

## Details

Under Windows, only one information item is available for a directory. The name of the item is Directory; its value is the pathname or list of pathnames for *directory-id*, and its sequence number is 1. Since only one information item is available for a directory, this function will return a value of `1`.

## Example

For an example of the DOPTNUM function, see "Example of Obtaining Directory Information" on page 390.

# FDELETE Function

**Deletes an external file or an empty directory**

**Category:** External Files

**Windows specifics:** *fileref* can be assigned by using an environment variable

**See:** FDELETE Function in *SAS Language Reference: Dictionary*

## Syntax

**FDELETE**("*fileref*")

***fileref***
specifies the fileref that is assigned to the external file or directory. The fileref cannot be associated with a list of concatenated filenames or directories. If the fileref is associated with a directory, the directory must be empty. You must have permission to delete the file. Under Windows, *fileref* can be an environment variable. The fileref or the environment variable that you specify must be enclosed in quotation marks.

## Details

FDELETE returns 0 if the operation was successful, and a non-zero number if it was not successful.

## FEXIST Function

**Verifies the existence of an external file by its fileref**

**Category:** External Files

**Windows specifics:** *fileref* can be assigned with an environment variable

**See:** FEXIST Function in *SAS Language Reference: Dictionary*

### Syntax

**FEXIST**("*fileref*")

***fileref***
specifies the fileref that is assigned to an external file. Under Windows, *fileref* can also be an environment variable. The fileref or the environment variable that you specify must be enclosed in quotation marks.

### Details

FEXIST returns 1 if the external file that is associated with fileref exists, and 0 if the file does not exist.

### Example

For an example of using the FINFO function, see "Example of Obtaining File Information" on page 397.

## FILEEXIST Function

**Verifies the existence of an external file by its physical name**

**Category:** External Files

**Windows specifics:** *filename* can be assigned with an environment variable

**See:** FILEEXIST Function in *SAS Language Reference: Dictionary*

### Syntax

**FILEEXIST**("*filename*")

***filename***
specifies a fully qualified physical filename of the external file. In a DATA step, *filename* can be a character expression, a string in quotation marks, or a DATA step variable. In a macro, *filename* can be any expression.

Under Windows, *filename* can also be an environment variable. The filename or environment variable that you specify must be enclosed in quotation marks.

### Details

FILEEXIST returns 1 if the external file exists and 0 if the external file does not exist.

## FILENAME Function

**Assigns or deassigns a fileref for an external file, directory, or output device**

**Category:** External Files

**Windows specifics:** device types and host options

**See:** FILENAME Function in *SAS Language Reference: Dictionary*

### Syntax

**FILENAME** (*"fileref"*, *"filename"* <,*device-type*<,*host-options*<,*dir-ref*>>>)

***fileref***

in a DATA step, specifies the fileref to assign to the external file. In a macro (for example, in the %SYSFUNC function), fileref is the name of a macro variable (without an ampersand) whose value contains the fileref to assign to the external file. (See *SAS Language Reference: Dictionary* for details.)

Under Windows, *fileref* can also be a Windows environment variable. The fileref or the environment variable that you specify must be enclosed in quotation marks.

***filename***

specifies the external file. Specifying a blank filename clears the fileref that was previously assigned.

Under Windows, the *filename* differs according to the device type. Table 5.2 on page 155 shows the information that is appropriate to each device. The filename that you specify must be enclosed in quotation marks.

***device-type***

specifies type of device or the access method that is used if the fileref points to an input or output device or location that is not a physical file. It can be any one of the devices that are listed in FILENAME statement device-type argument on page 446. DISK is the default device type.

***host-options***

are options that are specific to Windows. You can use any of the options that are available in the FILENAME statement. See the FILENAME statement host-option-list on page 447.

***dir-ref***

specifies the fileref that is assigned to the directory in which the external file resides.

### Details

FILENAME returns a value of 0 if the operation was successful, and a non-zero number if the operation was not successful.

### Example

For an example of using the FILENAME function, see "Example of Obtaining File Information" on page 397.

## FILEREF Function

**Verifies that a fileref has been assigned for the current SAS session**

**Category:** External Files

**Windows specifics:** the fileref argument can specify a Windows environment variable

**See:** FILEREF Function in *SAS Language Reference: Dictionary*

### Syntax

**FILEREF**("*fileref*")

***fileref***
specifies the fileref to be validated. Under Windows, *fileref* can also be a Windows environment variable. The fileref or the environment variable that you specify must be enclosed in quotation marks.

### Details

A negative return code indicates that the fileref exists but the physical file associated with the fileref does not exist. A positive value indicates that the fileref is not assigned. A value of zero indicates that the fileref and external file both exist.

### Example

For an example of using the FILEREF function, see "Example of Obtaining File Information" on page 397.

## FINFO Function

**Returns the value of an information item for an external file**

**Category:** External Files

**Windows specifics:** available *info-items*

**See:** FINFO Function in *SAS Language Reference: Dictionary*

### Syntax

**FINFO**(*file-id*, *info-item*)

***file-id***
specifies the identifier that was assigned when the file was opened, generally by the FOPEN function.

***info-item***

specifies the name of the file information item to be retrieved. This is a character value. *Info-item* is either a variable that contains a file information name or the file information name that has been enclosed in quotation marks.

*info-item* for disk files can be one of these file information items:

- File Name
- RECFM
- LRECL

*info-item* for pipe files can be one of these file information items:

- Unnamed pipe access device
- PROCESS
- RECFM
- LRECL

## Details

The FINFO function returns the value of a system-dependent information item for an external file that was previously opened and assigned a file-id by the FOPEN function. FINFO returns a blank if the value given for *info-item* is invalid.

## Example of Obtaining File Information

```
data a;

/* Does fileref "curdirfl" exist?  No = 0 */

rc=fexist ("curdirfl");
put;
put "Fileref curdirfl exist? rc should be 0 (no); " rc=;

/* assign fileref */

rc=filename("curdirfl", "c:\tmp333");

/* RC=0 indicates success in assigning fileref */

put "Fileref assigned - rc should be 0; " rc=;
rc=fexist ("curdirfl");

/* Does file which "curdirfl" points to exist?  No = 0 */
/* Assigning a fileref doesn't create the file. */

put "File still doesn't exist - rc should be 0; " rc=;
rc=fileref ("curdirfl");

/* Does fileref "curdirfl" exist?  */
/* Negative means fileref exists, but file does not */
/* Positive means fileref does not exist           */
/* Zero means both fileref and file exist          */

put "Fileref now exists - rc should be negative; " rc=;
put;

/* Does the file that the fileref points to exist?  Should be no. */
```

```
if ( fileexist ("./tmp333") ) then
    /* if it does, open it for input */
      do;
        put "Open file for input";
        fid=fopen ("curdirfl", "i") ;
      end;
   else       /* most likely scenario */
      do;
        put "Open file for output";
        fid=fopen ("curdirfl", "o");
      end;

/* fid should be non-zero.  0 indicates failure. */
put "File id is: " fid=;
numopts = foptnum(fid);
put "Number of information items should be 3; " numopts=;
do i = 1 to numopts;
optname = foptname (fid,i);
put i= optname=;
optval  = finfo (fid, optname);
put optval= ;
end;
rc=fclose (fid);
rc=fdelete ("curdirfl");
put "Closing the file, rc should be 0; "
rc=; run;
```

**Output 18.2** The Resulting SAS Log

```
NOTE: PROCEDURE PRINTTO used (Total process time):
      real time           0.36 seconds
      cpu time            0.00 seconds

291  data a;
292
293  /* Does fileref "curdirfl" exist?  No = 0 */
294
295  rc=fexist ("curdirfl");
296  put;
t297  put "Fileref curdirfl exist? rc should be 0 (no); " rc=;
298
299  /* assign fileref */
300
301  rc=filename("curdirfl", "c:\tmp333");
302
303  /* RC=0 indicates success in assigning fileref */
304
305  put "Fileref assigned - rc should be 0; " rc=;
306  rc=fexist ("curdirfl");
307
308  /* Does file which "curdirfl" points to exist?  No = 0 */
309  /* Assigning a fileref doesn't create the file. */
310
311  put "File still doesn't exist - rc should be 0; " rc=;
312  rc=fileref ("curdirfl");
313
314  /* Does fileref "curdirfl" exist?  */
315  /* Negative means fileref exists, but file does not */
316  /* Positive means fileref does not exist            */
317  /* Zero means both fileref and file exist           */
318
319  put "Fileref now exists - rc should be negative; " rc=;
320  put;
321
322  /* Does the file that the fileref points to exist?  Should be no. */
323
324  if ( fileexist ("./tmp333") ) then
325      /* if it does, open it for input */
326        do;
327          put "Open file for input";
328          fid=fopen ("curdirfl", "i") ;
329        end;
330     else       /* most likely scenario */
331        do;
332          put "Open file for output";
333          fid=fopen ("curdirfl", "o");
334        end;
```

```
335
336  /* fid should be non-zero.  0 indicates failure. */
337  put "File id is: " fid=;
338  numopts = foptnum(fid);
339  put "Number of information items should be 3; " numopts=;
340  do i = 1 to numopts;
341  optname = foptname (fid,i);
342  put i= optname=;
343  optval  = finfo (fid, optname);
344  put optval= ;
345  end;
346  rc=fclose (fid);
347  rc=fdelete ("curdirfl");
348  put "Closing the file, rc should be 0; "
349  rc=; run;

Fileref curdirfl exist? rc should be 0 (no); rc=0
Fileref assigned - rc should be 0; rc=0
File still doesn't exist - rc should be 0; rc=0
Fileref now exists - rc should be negative; rc=-20006

Open file for output
File id is: fid=1
Number of information items should be 3; numopts=3
i=1 optname=File Name
optval=c:\tmp333
i=2 optname=RECFM
optval=V
i=3 optname=LRECL
optval=256
Closing the file, rc should be 0; rc=0
NOTE: The data set WORK.A has 1 observations and 6 variables.
NOTE: DATA statement used (Total process time):
      real time           0.12 seconds
      cpu time            0.09 seconds

350  proc printto; run;
```

## See Also

- "FOPEN Function" in *SAS Language Reference: Dictionary*

## FOPTNAME Function

**Returns the name of an information item for an external file**

**Category:** External Files

**Windows specifics:** available information items

**See:** FOPTNAME Function in *SAS Language Reference: Dictionary*

### Syntax

**FOPTNAME**(*file-id*, *nval*)

***file-id***
specifies the identifier that was assigned when the file was opened, generally by the FOPEN function.

***nval***

specifies the number of the file information item to be retrieved. The following table shows the values that *nval* can have for single and concatenated files under Windows operating environments.

| *nval* | Single File | Pipe Files | Concatenated Files |
|---|---|---|---|
| 1 | File Name | Unnamed pipe access device | File Name |
| 2 | RECFM | PROCESS | RECFM |
| 3 | LRECL | RECFM | LRECL |
| 4 | | LRECL | |

## Details

FOPTNAME returns a blank if an error occurred.

## Example: File Attributes When Using the Pipe Device Type

The following example creates a data set that contains the name and value attributes that are returned by the FOPTNAME function when you are using pipes:

```
data fileatt;
   filename mypipe pipe 'dir';
   fid=fopen("mypipe","s");
   /* fid should be non---zero. 0 indicates failure */
   put "File id is: " fid=;
   numopts=foptnum(fid);
   put "Number of information items should be 4; " numopts=;
   do i=1 to numopts;
      optname=foptname(fid,i);
      put i= optname=;
      optval=finfo(fid,optname);
      put optval=;
   end;

rc=fclose(fid);
run;
```

**Output 18.3** The SAS LOG Displays Pipe File Information

```
NOTE: PROCEDURE PRINTTO used (Total process time):
      real time           0.03 seconds
      cpu time            0.01 seconds

6    data fileatt;
7       filename mypipe pipe 'dir';
8       fid=fopen("mypipe","s");
9       /* fid should be non-zero. 0 indicates failure */
10      put "File id is: " fid=;
11      numopts=foptnum(fid);
12      put "Number of information items should be 4; " numopts=;
13      do i=1 to numopts;
14         optname=foptname(fid,i);
15         put i= optname=;
16         optval=finfo(fid,optname);
17         put optval=;
18      end;
19
20   rc=fclose(fid);
21   run;
File id is: fid=1
Number of information items should be 4; numopts=4
i=1 optname=Unnamed Pipe Access Device
optval=
i=2 optname=PROCESS
optval=dir
i=3 optname=RECFM
optval=V
i=4 optname=LRECL
optval=256
NOTE: The data set WORK.FILEATT has 1 observations and 6 variables.
NOTE: DATA statement used (Total process time):
      real time           9.64 seconds
      cpu time            1.16 seconds

22   proc printto; run;
```

## FOPTNUM Function

**Returns the number of information items that are available for a file**

**Category:** External Files

**Windows specifics:** information items available

**See:** FOPTNUM Function in *SAS Language Reference: Dictionary*

### Syntax

**FOPTNUM**(*file-id*)

***file-id***
specifies the identifier that was assigned when the file was opened, generally, by the FOPEN function.

### Details

Three information items are available for files:

- File Name
- RECFM
- LRECL

These information items are available for pipes:

- Unnamed pipe access device
- PROCESS
- RECFM
- LRECL

FOPTNUM returns the following values:

| | |
|---|---|
| For files: | 3 |
| For pipes: | 4 |

### Example

For an example of the FOPTNUM functions, see "Example of Obtaining File Information" on page 397.

## LIBNAME Function

**Assigns or clears a libref for a SAS data library**

**Category:** SAS File I/O

**Windows specifics:** behavior of the ' 'libref (a space between single quotation marks)

**See:** LIBNAME Function in *SAS Language Reference: Dictionary*

### Syntax

**LIBNAME**(*libref*<,*SAS-data-library*<,*engine*<,*options*>>>)

***libref***
specifies the libref that is assigned to a SAS data library. Under Windows, the value of *libref* can be an environment variable.

***SAS-data-library***
specifies the physical name of the SAS data library that is associated with the libref.

***engine***
specifies the engine that is used to access SAS files opened in the data library.

***options***
names one or more options honored by the specified engine, delimited with blanks.

### Details

If the LIBNAME function returns a 0, then the function was successful. However, you could receive a non-zero value, even if the function was successful. A non-zero value is returned if an error, warning, or note is produced. To determine if the function was successful, look through the SAS Log and use the following guidelines:

- If a warning or note was generated, then the function was successful.
- If an error was generated, then the function was not successful.

Under Windows, if you do not specify a *SAS-data-library* or if you specify a *SAS-data-library* as ' '(a space between single quotation marks) or ''(no space between single quotation marks), SAS deassigns the libref.

## MCIPISLP Function

**Causes SAS to wait for a piece of multimedia equipment to become active**

**Category:** Special

**Windows specifics:** all

### Syntax

**MCIPISLP**(*number-of-seconds*)

***number-of-seconds***
specifies the number of seconds you want SAS to wait. This number must be an integer.

### Details

The MCIPISLP function is especially useful when you have used the MCIPISTR function to open a piece of equipment, but you know it is going to take a few seconds for the equipment to be ready.

The *number-of-seconds* argument must be an integer and represents how many seconds you want to wait. The return value is the number of seconds slept.

The MCIPISLP function can be used in the DATA step and in SCL code.

### Example

This example uses both the MCIPISTR and MCIPISLP functions to play a CD and a video. The PUT statements display the return values of these functions. This allows you to see in the SAS log whether there was a problem with any of your equipment.

```
data _null_;
  /* Open a CD player. */
  msg=mcipistr("open cdaudio alias mytunes");
  put msg=;
  /* Wait one second for the CD player     */
  /* to become active.                     */
  slept=mcipislp(1);
  /* Begin playing your favorite tunes     */
  /* from the beginning of the CD.         */
  msg=mcipistr("play mytunes");
```

```
    put msg=;
    /* Now open a video file. */
    msg=mcipistr("open c:\movies\amovie.avs
                  alias myshow");
    put msg=;
    /* Begin the show and wait for it to     */
    /* complete.                             */
    msg=mcipistr("play myshow wait");
    put msg=;
    /* When the show is complete,            */
    /* close the instance.                   */
    msg=mcipistr("close myshow");
    put msg=;
    /* Stop and close the instance of the CD */
    /* player.                               */
    msg=mcipistr("stop mytunes");
    put msg=;
    msg=mcipistr("close mytunes");
    put msg=;
  run;
```

## See Also

- "MCIPISTR Function" on page 405

## MCIPISTR Function

**Submits an MCI string command to a piece of multimedia equipment**

**Category:** Special

**Windows specifics:** all

### Syntax

**MCIPISTR**(*MCI-string-command*)

***MCI-string-command***
is any valid SAS string; that is, a character variable, a character literal enclosed in quotes, or other character expression.

### Details

The MCIPISTR function submits an MCI (Media Control Interface) string command.

You can use MCI to control many types of multimedia equipment, such as CD players, mixers, videodisc players, and so on. Windows provides MCI support. For more information about valid MCI string commands, refer to the Windows multimedia SDK documentation in the MSDN Library and your MCI-compliant device documentation.

The return value is a string that contains return information from the MCI string command. Examples of return information include "invalid instance" and "1".

*Note:* Not all MCI commands supply return codes that are usable from SAS. △

The MCIPISTR function can be used in the DATA step and in SCL code.

## Example

To use a CD player, you could submit the following statements in your DATA step:

```
msg=mcipistr("open cdaudio alias cd");
msg=mcipistr("play cd");
msg=mcipistr("stop cd");
msg=mcipistr("close cd");
```

## See Also

□ "MCIPISLP Function" on page 404

## MODULE Function

**Calls a specific routine or module that resides in an external dynamic link library (DLL)**

**Category:** External Routines

**Windows specifics:** all

### Syntax

**CALL MODULE**(<*cntl*>,*module*,*arg-1*,*arg-2*. . . ,*arg-n*);

*num*=**MODULEN**(<*cntl*>,*module*,*arg-1*,*arg-2*…,*arg-n*);

*char*=**MODULEC**(<*cntl*>,*module*,*arg-1*…,*arg-2*,*arg-n*);

*Note:* The following functions permit vector and matrix arguments; you can use them within the IML procedure. △

**CALL MODULEI** <*cntl*>,*modulearg-1*,*arg-2*. . . ,*arg-n*);

*num*=**MODULEIN**(<*cntl*>,*module*,*arg-1*,*arg-2*. . .,*arg-n*)

*char*=**MODULEIC**(<*cntl*>,*module*,*arg-1*,*arg-2*. . .,*arg-n*);

***cntl***

is an optional control string whose first character must be an asterisk (*), followed by any combination of the following characters:

| | |
|---|---|
| I | prints the hexadecimal representations of all arguments to the MODULE function and to the requested DLL routine before and after the DLL routine is called. You can use this option to help diagnose problems that are caused by incorrect arguments or attribute tables. If you specify the **I** option, the **E** option is implied. |
| E | prints detailed error messages. Without the **E** option (or the **I** option, which supersedes it), the only error message that the MODULE function generates is "Invalid argument to function," |

|  |  |
|---|---|
|  | which is usually not enough information to determine the cause of the error. |
| S*x* | uses *x* as a separator character to separate field definitions. You can then specify *x* in the argument list as its own character argument to serve as a delimiter for a list of arguments that you want to group together as a single structure. Use this option only if you do not supply an entry in the SASCBTBL attribute table. If you do supply an entry for this module in the SASCBTBL attribute table, you should use the FDSTART option in the ARG statement in the table to separate structures. |
| H | provides brief help information about the syntax of the MODULE routines, the attribute file format, and the suggested SAS formats and informats. |

For example, the control string **`'*IS/'`** specifies that parameter lists be printed and that the string **`'/'`** is to be treated as a separator character in the argument list.

***module***
is the name of the external module to use, specified as a DLL name and the routine name or ordinal value, separated by a comma. The module must reside in a dynamic link library (DLL) and it must be externally callable. For example, the value **`'KERNEL32,GetProfileString'`** specifies to load KERNEL32.DLL and to invoke the GetProfileString routine. Note that while the DLL name is not case sensitive, the routine name is based on the restraints of the routine's implementation language, so the routine name is case sensitive.

If the DLL supports ordinal-value naming, you can provide the DLL name followed by a decimal number, such as **`'XYZ,30'`**.

You do not need to specify the DLL name if you specified the MODULE attribute for the routine in the SASCBTBL attribute table, as long as the routine name is unique (that is, no other routines have the same name in the attribute file).

You can specify *module* as a SAS character expression instead of as a constant; most often, though, you will pass it as a constant.

***arg-1, arg-2, ...arg-n***
are the arguments to pass to the requested routine. Use the proper attributes for the arguments (that is, numeric arguments for numeric attributes and character arguments for character attributes).

***CAUTION:***
**Be sure to use the correct arguments and attributes.** Using incorrect arguments or attributes for a DLL function can cause SAS, and possibly your operating system, to fail. △

## Details

The MODULE functions execute a routine *module* that resides in an external (outside SAS) dynamic link library with the specified arguments *arg-1* through *arg-n*.

The MODULE call routine does not return a value, while the MODULEN and MODULEC functions return a number *num* or a character *char*, respectively. Which routine you use depends on the expected return value of the DLL function that you want to execute.

MODULEI, MODULEIC, and MODULEIN are special versions of the MODULE functions that permit vector and matrix arguments. Their return values are still scalar. You can invoke these functions only from PROC IML.

Other than this name difference, the syntax for all six routines is the same.

The MODULE function builds a parameter list by using the information in *arg-1* to *arg-n* and by using a routine description and argument attribute table that you define in a separate file. Before you invoke the MODULE routine, you must define the fileref of SASCBTBL to point to this external file. You can name the file whatever you want when you create it.

This way, you can use SAS variables and formats as arguments to the MODULE function and ensure that these arguments are properly converted before being passed to the DLL routine.

### See Also

- "The SASCBTBL Attribute Table" on page 296

## PEEKLONG Function

**Stores the contents of a memory address in a numeric variable on 32-bit and 64-bit platforms**

**Category** Special

**Windows specifics:** all

**See:** PEEKLONG Function and PEEKCLONG Function in *SAS Language Reference: Dictionary*

### Syntax

PEEKLONG(*address*<,*length*)>

*address*
specifies a character string that is the memory address.

*length*
specifies the length of the character data.

### Details

***CAUTION:***
**The PEEKLONG functions can directly access memory addresses. Improper use of the PEEKLONG functions can cause SAS, and your operating system, to fail.** Use the PEEKLONG functions only to access information that is returned by one of the MODULE functions. △

The PEEKLONG function returns a value of length *length* that contains the data that start at memory address *address*.

The variations of the PEEKLONG functions are

| | |
|---|---|
| PEEKCLONG | accesses character strings. |
| PEEKLONG | accesses numeric values. |

Usually, when you need to use one of the PEEKLONG functions, you will use PEEKCLONG to access a character string.

## RANK Function

**Returns the position of a character in the ASCII collating sequence**

**Category:** Character
**Windows specifics:** Uses the ASCII sequence
**See:** RANK Function in *SAS Language Reference: Dictionary*

### Syntax

**RANK**(*x*)

***x***
is a character expression (or character string) that contains a character in the ASCII collating sequence. If the length of *x* is greater than 1, you receive the rank of the first character in the string.

### Details

Because Windows uses the ASCII character set, the RANK function returns an integer that represents the position of a character in the ASCII collating sequence.

*Note:* Any program that uses the RANK function with characters above ASCII 127 (the hexadecimal notation is `'7F'x`) is not portable because these are national characters and they vary from country to country. △

## SLEEP Function

**Suspends execution of a SAS DATA step for a specified period of time**

**Category:** Special
**Windows specifics:** all
**See:** SLEEP Function in *SAS Language Reference: Dictionary*

### Syntax

**SLEEP**(*n*<,*unit*>)

***n***
specifies the number of seconds that you want to suspend execution of a DATA step. The *n* argument is a numeric constant that must be greater than or equal to 0. Negative or missing values for *n* are invalid.

***unit***
specifies the unit of seconds, as a power of 10, which is applied to *n*. For example, 1 corresponds to a second, and .001 to a millisecond. The default is 1.

### Details

The SLEEP function suspends execution of a DATA step for a specified number of seconds. When the SLEEP function uses the default *unit* value, a pop-up window appears that indicates how long SAS is going to sleep.

The return value of the *n* argument is the number of seconds slept. The maximum sleep period for the SLEEP function is 46 days.

When you submit a program that calls the SLEEP function, the SLEEP window appears telling you when SAS is going to wake up. You can inhibit the SLEEP window by starting SAS with the NOSLEEPWINDOW system option. Your SAS session remains inactive until the sleep period is over. To cancel the call to the SLEEP function, use the CTRL+BREAK attention sequence.

You should use a null DATA step to call the SLEEP function; follow this DATA step with the rest of the SAS program. Using the SLEEP function in this manner enables you to use the CTRL+BREAK attention sequence to interrupt the SLEEP function and to continue with the execution of the rest of your SAS program.

### Example

The following example tells SAS to delay the execution of the program for 12 hours and 15 minutes:

```
data _null_;
   /* argument to sleep must be expressed in seconds */
   slept= sleep((60*60*12)+(60*15));
run;
data monthly;
   /*... more data lines */
run;
```

### See Also

- "SLEEPWINDOW System Option" on page 551

## TRANSLATE Function

**Replaces specific characters in a character expression**

**Category:** Character

**Windows specifics:** Required syntax; pairs of *to* and *from* arguments are optional

**See:** TRANSLATE Function in *SAS Language Reference: Dictionary*

### Syntax

**TRANSLATE**(*source,to-1,from-1* <,...*to-n,from-n*>)

***source***
specifies the SAS expression that contains the original character value.

***to***
specifies the characters that you want TRANSLATE to use as substitutes.

***from***
specifies the characters that you want TRANSLATE to replace.

### Details

Under Windows, you do not have to provide pairs of *to* and *from* arguments. However, if you do not use pairs, you must supply a comma as a place holder.

## WAKEUP Function

**Specifies the time a SAS DATA step begins execution**

**Category:** Special

**Windows specifics:** all

### Syntax

**WAKEUP**(*until-when*)

***until-when***
specifies the time when the WAKEUP function will be executed.

### Details

Use the WAKEUP function to specify the time a DATA step begins to execute. The return value is the number of seconds slept.

The *until-when* argument can be a SAS datetime value, a SAS time value, or a numeric constant, as explained in the following list:

- □ If *until-when* is a datetime value, the WAKEUP function sleeps until the specified date and time. If the specified date and time have already passed, the WAKEUP function does not sleep, and the return value is 0.
- □ If *until-when* is a time value, the WAKEUP function sleeps until the specified time. If the specified time has already passed in that 24-hour period, the WAKEUP function sleeps until the specified time occurs again.
- □ If the value of *until-when* is a numeric constant, the WAKEUP function sleeps for that many seconds before or after the next occurring midnight. If the value of *until-when* is a positive numeric constant, the WAKEUP function sleeps for *until-when* seconds past midnight. If the value of *until-when* is a negative numeric constant, the WAKEUP function sleeps until *until-when* seconds before midnight.

Negative values for the *until-when* argument are allowed, but missing values are not. The maximum sleep period for the WAKEUP function is approximately 46 days.

When you submit a program that calls the WAKEUP function, a the SLEEP window appears telling you when SAS is going to wake up. You can inhibit the SLEEP window by starting SAS with the NOSLEEPWINDOW system option. Your SAS session remains inactive until the waiting period is over. If you want to cancel the call to the WAKEUP function, use the CTRL + BREAK attention sequence.

You should use a null DATA step to call the WAKEUP function; follow this DATA step with the rest of the SAS program. Using the WAKEUP function in this manner enables you to use the CTRL+BREAK attention sequence to interrupt the waiting period and continue with the execution of the rest of your SAS program.

## Examples

**Example 1: Delaying Program Execution until a Specified Date or Time** The code in this example tells SAS to delay execution of the program until 1:00 p.m. on January 1, 2004:

```
data _null_;
   slept=wakeup('01JAN2004:13:00:00'dt);
run;
data compare;
   /* ...more data lines */
run;
```

The following example tells SAS to delay execution of the program until 10:00 p.m.:

```
data _null_;
   slept=wakeup("22:00:00"t);
run;
data compare;
   /* ...more data lines */
run;
```

**Example 2: Delaying Program Execution until a Specified Time Period after Midnight** The following example tells SAS to delay execution of the program until 35 seconds after the next occurring midnight:

```
data _null_;
   slept=wakeup(35);
run;
data compare;
   /* ...more data lines */
run;
```

**Example 3: Using a Variable as an Argument to the WAKEUP Function** This example illustrates using a variable as the argument of the WAKEUP function:

```
data _null_;
   input x;
   slept=wakeup(x);
   datalines;
1000
;
data compare;
   input article1 $ article2 $ rating;
   /* ...more data lines */
run;
```

Because the instream data indicate that the value of X is 1000, the WAKEUP function sleeps for 1,000 seconds past midnight.

## See Also

□ "SLEEPWINDOW System Option" on page 551

# CHAPTER 19 Informats under Windows

*SAS Informats under Windows* **413**
*Reading Binary Data* **413**
*Converting User-Written Informats from Earlier Releases to SAS 9.1* **414**
*Converting Version 6 User-Written Informats* **414**
*Converting Version 5 User-Written Informats* **415**
*HEXw. Informat* **415**
*$HEXw. Informat* **416**
*IBw.d Informat* **416**
*PDw.d Informat* **418**
*PIBw.d Informat* **418**
*RBw.d Informat* **420**
*ZDw.d Informat* **420**

## SAS Informats under Windows

A SAS informat is an instruction or template that SAS uses to read data values into a variable. Most SAS informats are described completely in *SAS Language Reference: Dictionary*. The informats that are described here have behavior that is specific to SAS under Windows.

Many of the SAS informats that have details specific to the Windows operating environment are used to read binary data. In using these informats, it is important that you understand the concepts that are presented in "Reading Binary Data" on page 413.

If you have informats that you created for use in earlier releases of SAS, see "Converting User-Written Informats from Earlier Releases to SAS 9.1" on page 414 for information about how to convert those informats for use with SAS 9.1.

## Reading Binary Data

IBM mainframes, Hewlett Packard 9000, and most other UNIX systems store bytes in one order, called big-endian. Intel-based, or IBM compatible microcomputers and the VAX and Alpha computers manufactured by Compaq store bytes in a different order called byte-reversed, or little-endian.

Binary data stored in one order cannot be read by a computer that stores binary data in the other order without additional processing taking place. When you are designing SAS applications, try to anticipate how your data will be read and choose your formats and informats accordingly.

SAS provides two sets of informats for reading binary data and corresponding formats for writing binary data.

- □ The IB*w.d*, PD*w.d*, PIB*w.d*, and RB*w.d* informats and formats read and write in native mode, that is, using the byte-ordering system that is standard for the machine.
- □ The S370FIB*w.d*, S370FPD*w.d*, S370FRB*w.d*, and S370FPIB*w.d* informats and formats read and write according to the IBM 370 standard, regardless of the native mode of the machine. These informats and formats allow you to write SAS programs that can be run in any SAS environment, regardless of how numeric data are stored.

If a SAS program that reads and writes binary data runs on only one type of machine, you can use the native mode informats and formats. However, if you want to write SAS programs that can be run on multiple machines using different byte-storage systems, use the IBM 370 formats and informats. The purpose of the IBM 370 informats and formats is to enable you to write SAS programs that can be run in any SAS environment, no matter what standard you use for storing numeric data.

For example, suppose you have a program that writes data with the PIB*w.d* format. You execute the program on a microcomputer so that the data are stored in byte-reversed mode. Then on the microcomputer you run another SAS program that uses the PIB*w.d* informat to read the data. The data are read correctly because both the programs are run on the microcomputer using byte-reversed mode. However, you cannot upload the data to a Hewlett Packard 9000-series machine and read the data correctly because they are stored in a form native to the microcomputer but foreign to the Hewlett Packard 9000. To avoid this problem, use the S370FPIB*w.d* format to write the data; even on the microcomputer, this causes the data to be stored in IBM 370 mode. Then read the data using the S370FPIB*w.d* informat. Regardless of what type of machine you use when reading the data, they are read correctly.

## Converting User-Written Informats from Earlier Releases to SAS 9.1

You must convert Release 6.04, Release 6.06, and Release 6.08 user-written informats and formats to their SAS 9.1 counterparts before you can use them in a SAS 9.1 program. The only exception to this rule is user-written informats and formats created by Release 6.08 or later under Windows; these informats and formats can be read directly from your Windows SAS session. *

### Converting Version 6 User-Written Informats

You can convert Release 6.04, 6.06, and 6.08 SAS catalogs that contain user-written informats and formats by using one of the following methods:

Converting Release 6.04 catalogs
: use the CNTLOUT= option in the PROC FORMAT statement in Release 6.04 to create an output data set, and then use the CNTLIN= option in the PROC FORMAT statement in SAS 9.1 to create the SAS 9.1 informats or formats. You must use the V604 engine in your SAS 9.1 session to read the data set. This method also works for converting from Release 6.06 or Release 6.08.

Converting Release 6.06 or Release 6.08 catalogs
: use the CPORT and CIMPORT procedures to convert the informats and formats. For more information about the CPORT and CIMPORT procedures, see *Base SAS*

* However, it is recommended that you use PROC CPORT and PROC CIMPORT to convert older Windows catalogs that contain user-written informats and formats to SAS 9.1 if you no longer need to use them in previous releases.

*Procedures Guide*. This method works for converting from Release 6.06 or Release 6.08 only; it does not work for converting from Release 6.04.

## Converting Version 5 User-Written Informats

You must also convert Version 5 user-written informats and formats to their SAS 9.1 counterparts before you can use them in a SAS 9.1 program. (This implies that you are not only converting these files, but are also transferring them from a remote operating system to your PC). You can convert them using one of the following methods:

- □ Use the V5TOV6 procedure on the remote operating environment to convert the informats and formats to Version 6 format. This implies that the remote operating environment has access to Version 6 SAS software. Then, transport the converted informats and formats (as binary files) to your Windows operating environment and use the CIMPORT procedure to complete the conversion.

  *Note:* The V5TOV6 procedure is not available in SAS 9.1. You must use this procedure in Release 6 of SAS. △
- □ Use the SUGI supplemental procedure FMTLIB under Version 5 on the remote operating environment to create an output data set, transport that data set to your PC, and then use the CNTLIN= option in the PROC FORMAT statement in SAS 9.1 to create the SAS 9.1 informats or formats.

## HEX*w*. Informat

**Converts hexadecimal positive binary values to fixed-point or floating-point binary values**

**Category** numeric

**Width range:** 1–16

**Default width:** 8

**Alignment:** left

**Windows specifics:** native floating-point representation

**See:** HEXw. Informat in *SAS Language Reference: Dictionary*

### Syntax

HEX*w*.

***w***

specifies whether the input represents an integer (fixed-point) or a real (floating-point) binary number. When you specify a *w* value of 1 through 15, the input hexadecimal value represents an integer binary number. When you specify 16 for the *w* value, the input hexadecimal value represents a floating-point value.

### Details

The HEX*w*. informat expects input that is not byte-reversed, that is, not in Windows form. (The IB, PIB, and RB informats for binary numbers expect the bytes to be reversed.) This means that you can use the HEX*w*. informat to read hexadecimal literals from SAS programs that were created in another environment.

### See Also

- □ "$HEX*w*. Informat" on page 416
- □ "HEX*w*. Format" on page 376

## $HEX*w*. Informat

**Converts hexadecimal data to character data**

**Category** character

**Width range:** 1–32767

**Default width:** 2

**Alignment:** left

**Windows specifics:** ASCII character-encoding system

**See:** $HEXw. Informat in *SAS Language Reference: Dictionary*

### Syntax

$HEX*w*.

***w***
specifies width of the input value.

### Details

The $HEX*w*. informat is like the HEX*w*. informat in that it reads values in which each hexadecimal digit occupies 1 byte. Use the $HEX*w*. informat to encode hexadecimal information into a character variable when your input data are limited to printable characters. The conversion is based on the ASCII character set.

### See Also

- □ "HEX*w*. Informat" on page 415
- □ "$HEX*w*. Format" on page 377

## IB*w.d* Informat

**Reads integer binary (fixed-point) data**

**Category** numeric

**Width range:** 1–8

**Default width:** 4

**Decimal range:** 0–10

**Windows specifics:** native floating-point representation

**See:** IBw.d Informat in *SAS Language Reference: Dictionary*

## Syntax

IB*w.d*

***w***

specifies the width of the input field.

***d***

optionally specifies the power of 10 by which to divide the input value. SAS uses the *d* value even if the input data contain decimal points.

## Details

For integer binary data, the high-order bit is the value's sign: 0 for positive values, 1 for negative. Negative values are represented in twos-complement notation. If the informat includes a *d* value, the data value is divided by $10^d$.

Using the IB*w.d* informat requires you to understand twos complements and byte-swapped data format.

For more information about microcomputer fixed-point values, see Intel's developer Web site.

## Comparison of IB and PIB

The IB*w.d* informat and the PIB*w.d* informat give you different results. The IB*w.d* informat processes both positive and negative numbers and it uses the high-order bit as the sign bit. In contrast, the PIB*w.d* informat is used only for positive numbers and it does not look for a sign bit. As an example, suppose your data contain the following two-byte (byte-swapped) value:

```
01 80
```

When you read this value using the IB2. informat, the informat looks for the sign bit, sees that it is on, and reads the value as –32,767. However, if you read this value with the PIB2. informat, no sign bit is used, and the result is 32,769.

## Example

Suppose that your data contain the following 6-byte (byte-swapped) value:

```
64 00 00 00 00 00
```

If you read this value using the IB6. informat, it is read as the fixed-point value 100.0. Now suppose that your data contain the following (byte-swapped) value:

```
01 80
```

Because the sign bit is set, the value is read as –32,767.

## See Also

- "IB*w.d* Format" on page 377
- "Reading Binary Data" on page 413

## PD*w.d* Informat

**Reads packed decimal data**

**Category** numeric

**Width range:** 1–16

**Default width:** 1

**Decimal range:** 0–31

**Windows specifics:** How values are interpreted as negative or positive

**See:** PDw.d Informat in *SAS Language Reference: Dictionary*

### Syntax

PD*w.d*

***w***
specifies the width of the input field.

***d***
optionally specifies the power of 10 by which to divide the input value. If the data contain decimal points, then SAS ignores the *d* value.

### Details

In packed decimal data, each byte contains two digits. The value's sign is in the first bit of the first byte (although the entire first byte is used for the sign). Although it is usually impossible to key in packed decimal data directly from a terminal, many programs write packed decimal data. The decimal range is 1 through 31.

### Example

Suppose your data contain the following packed decimal number:

```
80 00 11 43
```

If you use the PD4. informat, this value is read as the double-precision value –1143.0. Similarly, the following value is read as 1500.0:

```
00 00 15 00
```

### See Also

- □ "PD*w.d* Format" on page 379
- □ "Reading Binary Data" on page 413

## PIB*w.d* Informat

**Reads positive integer binary (fixed-point) data**

**Category** numeric

**Width range:** 1–8

**Default width:** 1

**Decimal range:** 0–10

**Windows specifics:** native byte-swapped integers

**See:** PIBw.d Informat in *SAS Language Reference: Dictionary*

## Syntax

PIB*w.d*

***w***

specifies the width of the input field.

***d***

optionally specifies the power of 10 by which to divide the input value. SAS uses the *d* value even if the input data contain decimal points.

## Details

Positive integer binary values are the same as integer binary (see the informat "IB*w.d* Informat" on page 416), except that all values are treated as positive. Thus, the high-order bit is part of the value rather than the value's sign.

## Comparison of PIB and IB

The PIB*w.d* informat and the IB*w.d* informat give you different results, and you should differentiate carefully between these two informats. The IB*w.d* informat processes both positive and negative numbers and uses the high-order bit as the sign bit. In contrast, the PIB*w.d* informat is used only for positive numbers and it does not look for a sign bit. As an example, suppose your data contain the following two-byte (byte-swapped) value:

```
01 80
```

When you read this value using the IB2. informat, the informat looks for the sign bit, sees that it is on, and reads the value as –32,767. However, if you read this value with the PIB2. informat, no sign bit is used, and the result is 32,769.

## Example

Suppose your data contain the following one-byte value:

```
FF
```

If you read this value using the PIB1. informat, it is read as the double-precision value 255.0. Using this informat requires you to understand twos complements and byte-swapped data format.

## See Also

- "PIB*w.d* Format" on page 380

- □ “Reading Binary Data” on page 413

## RB*w.d* Informat

**Reads real binary (floating-point) data**

**Category** numeric

**Width range:** 2–8

**Default width:** 4

**Decimal range:** 0–10

**Windows specifics:** native floating-point representation

**See:** RBw.d Informat in *SAS Language Reference: Dictionary*

### Syntax

RB*w.d*

***w***
specifies the width of the input field.

***d***
optionally specifies the power of 10 by which to divide the input value. SAS uses the *d* value even if the input data contain decimal points.

### Details

The RB*w.d* informat reads numeric data that are stored in microcomputer real binary (floating-point) notation. Numeric data for scientific calculations are often stored in floating-point notation. (SAS stores all numeric values in floating-point notation.) A floating-point value consists of two parts: a mantissa that gives the value and an exponent that gives the value’s magnitude. It is usually impossible to key in floating-point binary data directly from a terminal, but many programs write floating-point binary data.

### See Also

- □ “RB*w.d* Format” on page 381
- □ “Reading Binary Data” on page 413

## ZD*w.d* Informat

**Reads zoned decimal data**

**Category** numeric

**Width range:** 1–32

**Default width:** 1

**Decimal range:** 1–10

**Windows specifics:** Last byte includes the sign

**See:** ZDw.d Informat in *SAS Language Reference: Dictionary*

## Syntax

ZD*w.d*

***w***
specifies the width of the input field.

***d***
optionally specifies the power of 10 by which to divide the input value. If the data contain decimal points, then SAS ignores the *d* value.

## Details

This is also known as an overprint trailing numeric format. Under Windows, the last byte of the field contains the sign information of the number. The following table gives the conversion for the last byte:

| Digit | ASCII Character | Digit | ASCII Character |
|---|---|---|---|
| 0 | { | –0 | } |
| 1 | A | –1 | J |
| 2 | B | –2 | K |
| 3 | C | –3 | L |
| 4 | D | –4 | M |
| 5 | E | –5 | N |
| 6 | F | –6 | O |
| 7 | G | –7 | P |
| 8 | H | –8 | Q |
| 9 | I | –9 | R |

## See Also

- "ZD*w.d* Format" on page 382

# CHAPTER 20
# Procedures under Windows

*SAS Procedures under Windows* **423**
*CATALOG Procedure* **423**
*CIMPORT Procedure* **424**
*CONTENTS Procedure* **425**
*CONVERT Procedure* **426**
*CPORT Procedure* **429**
*DATASETS Procedure* **430**
*OPTIONS Procedure* **431**
*PMENU Procedure* **432**
*PRINTTO Procedure* **433**
*SORT Procedure* **435**

## SAS Procedures under Windows

Base SAS procedures enable you to perform statistical computations, create reports, and manage your data. Most of the Base SAS procedures are described in *Base SAS Procedures Guide*. The procedures described here have syntax or behavior that is specific to Windows.

## CATALOG Procedure

**Manages entries in SAS catalogs**

**Windows specifics:** FILE= option in the CONTENTS statement

**See:** CATALOG Procedure in *Base SAS Procedures Guide*

### Syntax

**PROC CATALOG** CATALOG=<*libref.*>*catalog* <ENTRYTYPE=*etype*> <KILL>;
  **CONTENTS** <OUT=*SAS-data-set*> <FILE=*fileref*;>

*Note:* This is a simplified version of the CATALOG procedure syntax. For the complete syntax and its explanation, see the CATALOG procedure in *Base SAS Procedures Guide*. △

***fileref***
names a file specification that is specific to the Windows operating environment.

### Details

The CATALOG procedure manages entries in SAS catalogs.

The FILE= option in the CONTENTS statement of the CATALOG procedure accepts a file specification that is specific to the Windows operating environment. If an unquoted file specification is given in the FILE= option, but no FILENAME statement, SET system option, or Windows environment variable is used to define the file specification, the file is named *file-specification*.LST and is stored in the working directory. For example, if MYFILE is not a fileref defined by the FILENAME statement, the SET system option, or a Windows environment variable, and you submit the following statements, the file MYFILE.LST, containing the list of contents for Sasuser.Profile, is created in your working directory:

```
proc catalog catalog=sasuser.profile;
   contents file=myfile;
run;
```

## CIMPORT Procedure

**Restores a transport file created by the CPORT procedure**

**Windows specifics:** Name and location of transport file

**See:** CIMPORT Procedure in *Base SAS Procedures Guide*

### Syntax

**PROC CIMPORT** *destination=libref* | *<libref.>member-name <option(s)>*;

*Note:* This is a simplified version of the CIMPORT procedure syntax. For the complete syntax and its explanation, see the CIMPORT procedure in *Base SAS Procedures Guide*. △

***destination***
identifies the file(s) in the transport file as a single SAS data set, single SAS catalog, or multiple members of a SAS data library.

***libref | <libref.>member-name***
specifies the name of the SAS data set, catalog, or library to be created from the transport file.

### Details

The CIMPORT procedure *imports* a transport file that was created (*exported*) by the CPORT procedure.

Coupled with the CPORT procedure, the CIMPORT procedure enables you to move catalogs and data sets from one operating environment to another.

*Note:* PROC CIMPORT processes a file generated by PROC CPORT, not a transport file generated by the XPORT engine. △

When you use the CIMPORT procedure under Windows, remember the following:

- The value of the INFILE= option can be a fileref defined in a FILENAME statement, a quoted Windows pathname, or an environment variable.
- If you omit the INFILE= option and have not defined the reserved fileref SASCAT, SAS tries to read from a file named SASCAT.DAT in your working directory. If no file by that name exists, the following error message is issued and the procedure terminates, assuming that C:\SAS has been defined as the working directory:

  ```
  ERROR: Physical file does not exist, C:\SAS\SASCAT.DAT
  ```
- If you have not transferred the file created by PROC CPORT in binary format, PROC CIMPORT cannot read the file, and you receive the following message:

  ```
  ERROR: Given transport file is bad.
  ```

## CONTENTS Procedure

**Prints descriptions of the contents of one or more SAS data library files**

**Windows specifics:** Engine/Host Dependent Information output

**See:** CONTENTS Procedure in *Base SAS Procedures Guide*

### Syntax

**PROC CONTENTS** <*option(s)*>;

***option(s)***
For an explanation of the available options, see the CONTENTS procedure in *Base SAS Procedures Guide*.

### Details

The CONTENTS procedure shows the contents of a SAS data set and prints the directory of the SAS data library.

While most of the printed output generated by the CONTENTS procedure is the same across all operating environments, the Engine/Host Dependent Information output depends on both the operating environment and the engine. The following example output shows the Engine/Host Dependent Information that is generated for the V9 engine from these statements:

```
DATA SCHOOL;
   INPUT NAME $ Y GRADE CLASS $ ID;
   DATALINES;
 PHIL 1 85 MATH 234107589
 ROBERTO 1 90 ENGLISH 190873452
 CAROL 2 70 MATH 257902348
 THOMAS 2 71 ENGLISH 234567823
 JUANITA 3 98 FRENCH 876345290
 CEDRIC 3 75 HISTORY 231987222
 MARIA 4 89 PE 87654321
 ;
```

```
PROC CONTENTS DATA=SCHOOL OUT=SCHOUT(DROP=CRDATE MODATE);
   TITLE 'SCHOOL DATASET';
   RUN;
```

**Output 20.1** Engine/Host Dependent Information from PROC CONTENTS Using the V9 Engine

```
                                        SCHOOL DATASET                07:52 Monday, march 17, 2003   1

                                    The CONTENTS Procedure

              Data Set Name        WORK.SCHOOL                     Observations          7
              Member Type          DATA                            Variables             5
              Engine               V9                              Indexes               0
              Created              Monday, March 17, 2003 00:39:17 Observation Length    40
              Last Modified        Monday, March 17, 2003 00:39:17 Deleted Observations  0
              Protection                                           Compressed            NO
              Data Set Type                                        Sorted                NO
              Label
              Data Representation  WINDOWS_32
              Encoding             wlatin1 Western (Windows)

                                   Engine/Host Dependent Information

Data Set Page Size          4096
Number of Data Set Pages    1
First Data Page             1
Max Obs per Page            101
Obs in First Data Page      7
Number of Data Set Repairs  0
File Name                   C:\DOCUME~1\sasusr1\LOCALS~1\Temp\SAS Temporary Files\_TD1904\school.sas7bdat
Release Created             9.0101B0
Host Created                XP_PRO


                               Alphabetic List of Variables and Attributes

                                      #    Variable    Type    Len

                                      4    CLASS       Char      8
                                      3    GRADE       Num       8
                                      5    ID          Num       8
                                      1    NAME        Char      8
                                      2    Y           Num       8
```

The engine name (V9) is listed in the header information. The Engine/Host Dependent Information describes attributes of the data set, such as the data set page size and the maximum number of observations per page. For more information about how to interpret the data set size information, see “Calculating Data Set Size” on page 208.

## See Also

- the section on starting with SAS data sets in *Step-by-Step Programming with Base SAS Software*

## CONVERT Procedure

**Converts BMDP, OSIRIS system files and SPSS export files to SAS data sets**

**Windows specifics:** All

## Syntax

**PROC CONVERT** *product-specification <option(s)>*

***product-specification***
is required and can be one of the following:

BMDP=*fileref* <(CODE=*code* CONTENT=*content-type*)>
converts into a SAS data set the first member a BMDP save file created under DOS. Here is an example:

```
filename save 'c:\myidr\bmdp.dat';
proc convert bmdp=save;
run;
```

If you have more than one save file in the BMDP file referenced by the *fileref* argument, you can use two options in parentheses after *fileref*. The CODE= option lets you specify the code of the save file you want, and the CONTENT= option lets you give the content of the save file. For example, if a file with CODE=JUDGES has a content of DATA, you can use the following statement:

```
filename save 'c:\mydir\bmdpl.dat';
proc convert bmdp=save(code=judges
                       content=data);
run;
```

OSIRIS=*fileref*
specifies a fileref for the OSIRIS file to be converted into a SAS data set. If you use this product specification, you must also use the DICT= option, which specifies the OSIRIS dictionary to use.

SPSS=*fileref*
specifies a fileref for the SPSS export file to be converted into a SAS data set. The SPSS export file must be created by using the SPSS EXPORT command from any operating environment.

***option-list***

DICT=*fileref*
specifies a fileref of the dictionary file for the OSIRIS file. The DICT= option is valid only when used with the OSIRIS product specification.

FIRSTOBS=*n*
gives the number of the observation where the conversion is to begin. This enables you to skip over observations at the beginning of the OSIRIS or SPSS/PC system file.

OBS=*n*
specifies the number of the last observation to convert. This option enables you to exclude observations at the end of the file.

OUT= *SAS-data-set*
names the SAS data set created to hold the converted data. If the OUT= option is omitted, SAS still creates a Work data set and automatically names it DATA*n*, just as if you omitted a data set name in a DATA statement. If it is the first such data set in a job or session, SAS names it DATA1, the second is DATA2, and so on. If

the OUT= option is omitted or if you do not specify a two-level name (including a libref) in the OUT= option, the converted data set is stored in your Work data library and by default it is not permanent.

## Details

The CONVERT procedure converts a BMDP or OSIRIS system file or an SPSS export file to a SAS data set. It produces one output data set, but no printed output. The new data set contains the same information as the input system file; exceptions are noted in "Output Data Sets" on page 428. The BMDP, OSIRIS and SPSS engines provide more extensive capabilities.

Because the BMDP, OSIRIS and SPSS products are maintained by other companies or organizations, changes may be made that make the system files incompatible with the current version of PROC CONVERT. SAS upgrades PROC CONVERT only to support changes that are made to these products when a new version of SAS is available.

**Missing Values** If a numeric variable in the input data set has either no value or a system missing value, PROC CONVERT assigns it a missing value.

**Output Data Sets** This section describes the attributes of the output SAS data set for each *product-specification* value.

***CAUTION:***

**Be sure that the translated names are unique.** Variable names can sometimes be translated by SAS. To ensure the procedure works correctly, be sure your variables are named in such a way that translation results in unique names. △

**BMDP output** Variable names from the BMDP save file are used in the SAS data set, but nontrailing blanks and all special characters are converted to underscores in the SAS variable names. The subscript in BMDP variable names, such as x(1), becomes part of the SAS variable name, with the parentheses omitted: X1. Alphabetic BMDP variables become SAS character variables of corresponding length. Category records from BMDP are not accepted.

**OSIRIS Output** For single-response variables, the V1-V9999 name becomes the SAS variable name. For multiple-response variables, the suffix R*n* is added to the variable name, where *n* is the response. For example, V25R1 is the first response of the multiple-response variable V25. If the variable after V1000 has 100 or more responses, responses above 99 are eliminated. Numeric variables that OSIRIS stores in character, fixed-point binary, or floating-point binary mode become SAS numeric variables. Alphabetic variables become SAS character variables; any alphabetic variable of length greater than 200 is truncated to 200. The OSIRIS variable description becomes a SAS variable label, and OSIRIS print formats become SAS formats.

**SPSS Output** SPSS variable names and variable labels become variable names and labels without change. SPSS alphabetic variables become SAS character variables. SPSS blank values are converted to SAS missing values. SPSS print formats become SAS formats, and the SPSS default precision of no decimal places becomes part of the variables' formats. SPSS value labels are not copied. DOCUMENT data are copied so that PROC CONTENTS can display them.

## Comparison

The CONVERT procedure is closely related to the BMDP, OSIRIS and SPSS interface library engines. (In fact, the CONVERT procedure uses these engines.) For example, the following two sections of code provide identical results:

-
  ```
  filename myfile 'myspss.por';
  proc convert spss=myfile out=temp;
   run;
  ```

-
  ```
  libname myfile spss 'myspss.por';
  data temp;
      set myfile._first_;
   run;
  ```

However, the BMDP, OSIRIS and SPSS engines have more extensive capabilities than PROC CONVERT.

### See Also

- "Reading BMDP, OSIRIS and SPSS Files" on page 140

## CPORT Procedure

**Writes SAS data sets and catalogs into a special format in a transport file**

**Windows specifics:** Name and location of transport file

**See:** CPORT Procedure in *Base SAS Procedures Guide*

### Syntax

**PROC CPORT** *source-type=libref* | *<libref.>member-name<option(s)>*;

*Note:* This is a simplified version of the CPORT procedure syntax. For the complete syntax and its explanation, see the CPORT procedure in *Base SAS Procedures Guide*. △

***libref***
specifies the name and location of the file to be transported.

### Details

The CPORT procedure writes SAS data sets, SAS catalogs, or SAS data libraries to sequential file formats (transport files). Use PROC CPORT with the CIMPORT procedure to move files from one environment to another.

The value of the FILE= option can be a fileref defined in a FILENAME statement, a quoted Windows pathname, or an environment variable.

If you do not use the FILE= option and have not defined the reserved fileref SASCAT, a file named SASCAT.DAT is created in your working directory.

### See Also

- "CIMPORT Procedure" on page 424
- "Transferring SAS Files between Operating Environments" on page 143

# DATASETS Procedure

**Lists, copies, renames, and deletes SAS files and also manages indexes for and appends SAS data sets in a SAS data library**

**Windows specifics:** Directory information; CONTENTS statement output

**See:** DATASETS Procedure in *Base SAS Procedures Guide*

## Syntax

**PROC DATASETS** <*options(s)*>;
  **CONTENTS**<*options(s)*>;

***option(s)***
  This is a simplified version of the DATASETS procedure syntax. For the complete syntax, see the DATASETS procedure in *Base SAS Procedures Guide*.

## Details

The DATASETS procedure is a utility procedure that manages your SAS files.

The SAS data library information that is displayed in the SAS log by the DATASETS procedure depends on the operating environment and the engine. The following example SAS log shows the information (for the V9 engine) that the DATASETS procedure generates under Windows.

**Output 20.2** SAS Data Library Information from PROC DATASETS

```
PROC DATASETS library=work;
                           Directory

       Libref         WORK
       Engine         V9
       Physical Name  C:\DOCUME~1\sasusr\LOCALS~1\Temp\SAS Temporary Files\_TD2663
       File Name      C:\DOCUME~1\sasusr\LOCALS~1\Temp\SAS Temporary Files\_TD2663


                   Member     File
       #  Name     Type       Size  Last Modified

       1  GSEG     CATALOG   54272   17MAR2003:13:20:23
       2  HAT      DATA      46080   17MAR2003:13:20:20
       3  SASGOPT  CATALOG    5120   17MAR2003:13:20:21
```

The output shows you the libref, engine, and physical name that are associated with the library, as well as the names and other properties of the SAS files that are contained in the library.

The CONTENTS statement in the DATASETS procedure generates the same Engine/Host Dependent Information as the CONTENTS procedure.

## See Also

□ "CONTENTS Procedure" on page 425

- □ the section about modifying data set names and variable attributes in *Step-by-Step Programming with Base SAS Software*

## OPTIONS Procedure

**Lists the current values of all SAS system options**

**Windows specifics:** Host options

**See:** OPTIONS Procedure in *Base SAS Procedures Guide*

### Syntax

**PROC OPTIONS** <*options(s)*>;

***option(s)***
This is a simplified version of the OPTIONS procedure syntax. For the complete syntax and its explanation, see the OPTIONS procedure in *Base SAS Procedures Guide*.

### Details

The OPTIONS procedure lists the current settings of the SAS system options.

The options displayed by the OPTIONS procedure that are not operating environment specific (session and configuration) are the same for every operating environment, although the default values may differ slightly. However, the environment-specific options displayed by this procedure are different for each operating environment. The following display shows some sample operating environment options for the Windows environment, as generated by this code:

```
proc options host;
run;
```

**Output 20.3** Windows Operating Environment Options Displayed by PROC OPTIONS

```
Host Options:

 ACCESSIBILITY=STANDARD
                  Enable Extended Accessibility
 ALTLOG=          Specifies the destination for a copy of the SAS log
 ALTPRINT=        Specifies the destination for a copy of the SAS procedure output file
 AUTHPROVICERDOMAIN=
                  Authentication providers associated with domain suffixes
 AUTHSERVER=      Specify the authentication server or domain.
 AUTOEXEC=        Specifies the autoexec file to be used
 AWSCONTROL=(SYSTEMMENU MINMAX TITLE)
                  Used to customize the appearance for the SAS AWS.  Valid parameters are:
                  TITLE/NOTITLE SYSTEMMENU/NOSYSTEMMENU MINMAX/NOMINMAX
 AWSDEF=(  0   0  79 79)
                  Specify the initial size and position of the SAS AWS. This should be
                  specified as follows: 0 0 100 100
 AWSMENU          Show the main window's (AWS) menu.
 AWSMENUMERGE     Add host specific menu items to the main window's (AWS) menu.

...
 NOTE:  PROCEDURE OPTIONS used:( Total process time)
        real time           0.01 seconds
        cpu time            0.01 seconds
```

The option values listed in is display are examples only. The output of PROC OPTIONS depends on many things. Some option values depend on what method you use to run SAS. For example, the default line size under the SAS windowing environment is 75 lines on a VGA display, while it is 132 lines in batch mode. Also, the way you have set up your process affects the default values of system options. For example, the default value of the SASAUTOS= option depends on where you store your autocall macros.

Using PROC OPTIONS, you can check the values of all system options. If you want more information about a particular operating environment option, refer to "SAS System Options under Windows" on page 467 or Using SAS Software in Your Operating Environment in the SAS Help and Documentation.

## See Also

- □ "SAS System Options under Windows" on page 467

## PMENU Procedure

**Defines pull-down menu facilities for windows created with SAS software**

**Windows specifics:** ACCELERATE= option accepted for several key combinations

**See:** PMENU Procedure in *Base SAS Procedures Guide*

### Syntax

**PROC PMENU** <CATALOG=<*libref.*>*catalog*> <DESC '*entry-description*'>;

**ITEM** *command* <*option(s)*>;

**ITEM** '*menu-item*' <*option(s)*>;
ACCELERATE=*name-of-key*;

This is a simplified version of the PMENU procedure syntax. For the complete syntax, see the PMENU procedure in *Base SAS Procedures Guide*.

**ACCELERATE=*name-of-key***

defines a key sequence that can be used instead of selecting an item. When you press the key sequence, it has the same effect as selecting the item from the menu bar or pull-down menu.

Under Windows, the ACCELERATE= option in the ITEM statement is accepted only for the following key combinations:

- Ctrl + A (Select All)
- Ctrl + C (Copy)
- Ctrl + F (Find)
- Ctrl + N (New)
- Ctrl + O (Include)
- Ctrl + P (Print)
- Ctrl + S (File)
- Ctrl +V (Paste)
- Ctrl +X (Cut).
- Ctrl + Z (Undo)
- Del (Clear)

## Details

The PMENU procedure defines pull-down menus that can be used in DATA step windows, macro windows, SAS/AF and SAS/FSP windows, or in any SAS application that enables you to specify customized menus.

If you want your program to be portable between Windows and OS/2, you can use the following alternate key combinations:

- Ctrl+Insert
- Shift+Insert
- Alt+Backspace
- Shift+Delete.

If you use these alternate key combinations in your SAS program, the **`Edit`** pull-down menu shows the standard key combination; however, you can use either the standard or alternate key combination to activate the menu item.

## PRINTTO Procedure

### Defines destinations for SAS procedure output and the SAS log

**Windows specifics:** Valid values for *file-specification*; UNIT= option

**See:** PRINTTO Procedure in *Base SAS Procedures Guide*

### Syntax

**PROC PRINTTO** <*option(s)*>;

*Note:* This is a simplified version of the PRINTTO procedure syntax. For the complete syntax and its explanation, see the PRINTTO procedure in *Base SAS Procedures Guide*. △

***option(s)***

LOG=*file-specification*
PRINT=*file-specification*
can be

- □ a fileref defined in a FILENAME statement or function. To send SAS output or log directly to the printer, use a FILENAME statement or function with the PRINTER device-type keyword.
- □ a quoted Windows pathname
- □ an alphanumeric text string. The destination filename is *file-specification*.LOG or *file-specification*.LST and it is stored in the current directory.
- □ a SAS or Windows environment variable

UNIT=*nn*
sends your SAS procedure output to the file FT*nn*F001.LST, where *nn* represents the UNIT= value, which can range from 1 to 99. The file is located in the SAS working directory.

## Details

The PRINTTO procedure defines destinations for SAS procedure output and for the SAS log.

## Examples

**Example 1: Redirecting SAS Log Output** The following statements redirect any SAS log entries that are generated after the RUN statement to an output file with a fileref of TEST, which is associated with the LPT1: device:

```
filename test printer 'lpt1:';
proc printto log=test;
run;
```

When these statements are issued, a dialog box is opened that informs you PROC PRINTTO is running. All SAS log entries are redirected to the TEST output file as specified; however, they are not printed on the LPT1: device until the output file is closed, either by redirecting the SAS log entries back to the default destination or to another file.

The following statements send any SAS log entries that are generated after the RUN statement to the external file associated with the fileref MYFILE:

```
filename myfile 'c:\mydir\mylog.log';
proc printto log=myfile;
run;
```

**Example 2: Redirecting SAS Procedure Output** The following statements send any SAS procedure output to a file named MYPRINT.LST in your working directory (assuming MYPRINT is not a previously defined fileref or environment variable):

```
proc printto print=myprint;
run;
```

The following statements send any SAS procedure output to the printer port, which is usually defined by the system as LPT1:

```
proc printto print='lpt1:';
run;
```

**Example 3: Restoring the Output Destinations to the Default** The following statements (including a PROC PRINTTO statement with no options) redirect the SAS log and procedure output to the original default destinations:

```
proc printto;
run;
```

## SORT Procedure

**Sorts observations in a SAS data set by one or more variables, then stores the resulting sorted observations in a new SAS data set or replaces the original data set**

**Windows specifics:** Sort utilities available; SORTSIZE= and TAGSORT statement options

**See:** SORT Procedure in *Base SAS Procedures Guide*

### Syntax

**PROC SORT** <*option(s)*> <*collating-sequence-option*>;

*Note:* This is a simplified version of the SORT procedure syntax. For the complete syntax and its explanation, see the SORT procedure in *Base SAS Procedures Guide* △

**SORTSIZE=*memory-specification***
specifies the maximum amount of memory available to the SORT procedure. For further explanation of the SORTSIZE= option, see the following Details section.

**TAGSORT**
stores only the BY variables and the observation number in temporary files. When you specify TAGSORT, the sort is a single-threaded sort. Do not specify TAGSORT if you want SAS to use multiple threads to sort. For details about TAGSORT option, see the following Details section.

### Details

The SORT procedure sorts observations in a SAS data set by one or more character or numeric variables, either replacing the original data set or creating a new, sorted data set. By default under Windows, the SORT procedure uses the ASCII collating sequence.

The SORT procedure uses the sort utility specified by the SORTPGM system option. Sorting can be done by SAS, your database, or the Windows SyncSort utility. You can use all the options available to the SAS sort utility, such as the SORTSEQ and NODUPKEY options. For a complete list of all options available, see the list of sort options in the See Also section.

## SORTSIZE= Option

Under Windows, you can use the SORTSIZE= option in the PROC SORT statement to limit the amount of memory that is available to the SORT procedure. This option might reduce the amount of swapping SAS must do to sort the data set. If PROC SORT needs more memory than you specify, it creates a temporary utility file to store the data in. The SORT procedure's algorithm can swap data more efficiently than Windows can.

The syntax of the SORTSIZE= option is as follows:

SORTSIZE=*memory-specification*

where *memory-specification* can be one of the following:

| | |
|---|---|
| *n* | specifies the amount of memory in bytes. |
| *n*K | specifies the amount of memory in 1-kilobyte multiples. |
| *n*M | specifies the amount of memory in 1-megabyte multiples. |

The default SAS configuration file sets this option to 64MB using the SORTSIZE= system option.

You can override the default value of the SORTSIZE= system option by specifying a different SORTSIZE= value in the PROC SORT statement, or by submitting an OPTIONS statement that sets the SORTSIZE= system option to a new value.

The SORTSIZE= option is also discussed in "Improving Performance of the SORT Procedure" on page 206.

## TAGSORT Option

The TAGSORT option in the PROC SORT statement is useful in sorts when there may not be enough disk space to sort a large SAS data set. When you specify TAGSORT, the sort is a single-threaded sort. Do not specify TAGSORT if you want the SAS to use multiple threads to sort.

When you specify the TAGSORT option, only sort keys (that is, the variables specified in the BY statement) and the observation number for each observation are stored in the temporary files. The sort keys, together with the observation number, are referred to as *tags*. At the completion of the sorting process, the tags are used to retrieve the records from the input data set in sorted order. Thus, in cases where the total number of bytes of the sort keys is small compared with the length of the record, temporary disk use is reduced considerably. You should have enough disk space to hold another copy of the data (the output data set) or two copies of the tags, whichever is greater. Note that while using the TAGSORT option may reduce temporary disk use, the processing time may be much higher. However, on PCs with limited available disk space, the TAGSORT option may allow sorts to be performed in situations where they would otherwise not be possible.

## Creating Your Own Collating Sequences

If you want to provide your own collating sequences or change a collating sequence that has been provided for you, use the TRANTAB procedure to create or modify translate tables. For more information about the TRANTAB procedure, see *SAS National Language Support (NLS): User's Guide*. When you create your own translate tables, they are stored in your Sasuser.Profile catalog and they override any translate tables by the same name that are stored in the HOST catalog.

*Note:* System managers can modify the HOST catalog by copying newly created tables from the Sasuser.Profile catalog to the HOST catalog. Then all users can access the new or modified translate table. △

If you want to see the names of the collating sequences stored in the HOST catalog (using the SAS Explorer), submit the following statement:

```
dm 'catalog sashelp.host' catalog;
```

Alternatively, you can select the **`View`** pull-down menu, then select the **`Libraries`** item, then double-click on the Sashelp library, and then double-click on the HOST catalog. In batch mode, you can use the following statements to generate a list of the contents of the HOST catalog:

```
proc catalog catalog=sashelp.host;
   contents;
run;
```

Entries of type TRANTAB are the collating sequences.

If you want to see the contents of a particular translate table, use the following statements:

```
proc trantab table=table-name;
   list;
run;
```

The contents of the collating sequence are displayed in the SAS log.

## Using Syncsort for Windows

**Introduction to Using SyncSort with SAS** If you have SyncSort installed at your site, you can use Syncsort as an alternative sorting algorithm to the database sort or the SAS sort. SAS determines which sort to use by the values that are set for the SORTPGM, SORTCUT, and SORTCUTP system options.

The SyncSort installation process adds the SyncSort directory to the Windows PATH statement. As long as the SyncSort directory is included in the Windows PATH statement, SAS is able to launch SyncSort. SyncSort is developed by Syncsort, Inc.

**Setting SyncSort as the Sort Algorithm** To always sort using the SyncSort sort routine, the value of the SORTPGM system option must be HOST. To set this option, submit the following OPTIONS statement:

```
options sortpgm=host;
```

*Note:* The SORTPGM option can also be set from the SAS System Options window, in the SAS configuration file, or during SAS invocation. This example shows how to specify the SORTPGM system option at invocation or in the SAS configuration file:

```
-sortpgm host
```

△

**Sorting Based on Size or Observations** The sort routine that SAS uses can be based on either the number of observations in a data set or on the size of the data set. When the SORTPGM option is set to BEST, SAS uses the first available and pertinent sorting algorithm based on this order of precedence:

- database sort utility
- host sort utility
- SAS sort utility

If sorting is not to be done by the database, SAS looks at the values for the SORTCUT and SORTCUTP options to determine which sort to use.

The SORTCUT option specifies the number of observations above which SyncSort is used instead of the SAS sort. The SORTCUTP option specifies the number of bytes in the data set above which SyncSort is used.

If SORTCUT and SORTCUTP are set to zero, SAS uses the SAS sort routine. If you specify both options and either condition is met, SAS uses SyncSort.

When the following OPTIONS statement is in effect, the SyncSort routine is used when the number of observations is 501 or greater:

```
options sortpgm=best sortcut=500;
```

Here, the SyncSort routine is used when the size of the data set is greater than 40M:

```
options sortpgm=best sortcutp=40M;
```

For more information about these sort options, see "SORTPGM System Option" on page 555, "SORTCUT System Option" on page 552 and "SORTCUTP System Option" on page 553

**Changing the Location of SyncSort Temporary Files** By default, SyncSort uses the location that is specified in the -WORK option for temporary files. To change the location of SyncSort temporary files, specify a new location by using the SORTDEV option. Here is an example:

```
options sortdev="c:\temp\sortsync";
```

For more information about the SORTDEV options, see "SORTDEV System Option" on page 554.

**Passing Options to SyncSort** Use the SORTANOM option to specify the options that you want to use for SyncSort:

**Table 20.1** SORTANOM Options for SyncSort

| Task | SORTANOM Option |
|---|---|
| Run in multi-call mode instead of single-call mode | SORTANOM=b |
| Print statistics in the SAS log about the sorting process | SORTANOM=t |
| Print in the SAS log the commands that have been passed to Syncsort | SORTANOM=v |

Multiple options can be specified by concatenating the options:

```
options sortdev=btv;
```

For more information about the SORTANOM option, see "SORTANOM System Option" on page 551.

**Passing Parameters to SyncSort** Use the SORTPARM option to pass Syncsort options to SyncSort. Enclose the options in quotations marks as in this OPTIONS statement:

```
options sortparm="SyncSort-options";
```

For information about the SORTPARM option, see "SORTPARM System Option" on page 555. See the SyncSort documentation for a description of the SyncSort options.

## Specifying the SORTSEQ= Option with a Host Sort Utility

The SORTSEQ= option enables you to specify the collating sequence for your sort. For a list of valid values, see the SORT procedure in *Base SAS Procedures Guide*.

***CAUTION:***

**If you are using a host sort utility to sort your data, then specifying the SORTSEQ= option might corrupt the character BY variables if the sort sequence translation table and its inverse are not one-to-one mappings.** The translation table must map each character to a unique weight, and the inverse table must map each weight to a unique character variable. △

If your translation tables are not one-to-one mappings, then you can use one of the following methods to perform your sort:

- □ create a translation table that maps one-to-one. When you create a translation table that maps one-to-one, you can easily create a corresponding inverse table by using the TRANTAB procedure. If your table is not mapped one-to-one, then you will receive the following note in the SAS log when you try to create an inverse table:

  ```
  NOTE:  This table cannot be mapped one to one.
  ```

  For more information, see the TRANTAB procedure in *SAS National Language Support (NLS): User's Guide*.
- □ use the SAS sort. You can specify the SAS sort by using the SORTPGM system option. For more information, see “SORTPGM System Option” on page 555.
- □ specify the collation order options of your host sort utility. See the documentation for your host sort utility for more information.
- □ create a view with a dummy BY variable. For an example, see “Example: Creating a View with a Dummy BY Variable” on page 439.

*Note:* After using one of these methods, you might need to perform subsequent BY processing using either the NOTSORTED option or the NOBYSORTED system option. For more information about the NOTSORTED option, see the BY statement in *SAS Language Reference: Dictionary*. For more information about the NOBYSORTED system option, see the BYSORTED system option in *SAS Language Reference: Dictionary*. △

**Example: Creating a View with a Dummy BY Variable** The following code is an example of creating a view using a dummy BY variable:

```
options no date nostimer ls-78 ps-60;
options sortpgm=host msglevel=i;

data one;
   input name $ age;
   datalines;
   anne 35
   ALBERT 10
   JUAN 90
   janet 5
   bridget 23
   BRIAN 45
   ;
run;
```

```
data oneview / view=oneview;
   set one;
   name1=upcase(name);
run;

proc sort data=oneview out=final(drop=name1);
   by name1;
run;

proc print data=final;
run;
```

The output is the following:

**Output 20.4** Creating a View with a Dummy BY Variable

```
  The SAS System
Obs        name       age
 1         ALBERT      10
 2         anne        35
 3         BRIAN       45
 4         bridget     23
 5         janet        5
 6         JUAN        90
```

## See Also

- □ TRANTAB Procedure" in *SAS National Language Support (NLS): User's Guide*
- □ "SORTANOM System Option" on page 551
- □ "SORTCUT System Option" on page 552
- □ "SORTCUTP System Option" on page 553
- □ "SORTDEV System Option" on page 554
- □ "SORTPARM System Option" on page 555
- □ "SORTPGM System Option" on page 555
- □ "SORTSIZE System Option" on page 556
- □ "Improving Performance of the SORT Procedure" on page 206

# CHAPTER 21

# Statements under Windows

*SAS Statements under Windows* **441**
*ABORT Statement* **441**
*ATTRIB Statement* **442**
*FILE Statement* **443**
*FILENAME Statement* **445**
*FOOTNOTE Statement* **450**
*%INCLUDE Statement* **451**
*INFILE Statement* **453**
*LENGTH Statement* **455**
*LIBNAME Statement* **456**
*SYSTASK Statement* **458**
*TITLE Statement* **461**
*WAITFOR Statement* **461**
*X Statement* **462**

## SAS Statements under Windows

A SAS statement is a directive to SAS that either requests that SAS perform a certain operation or provides information to the system that might be necessary for later operations.

All SAS statements are described in *SAS Language Reference: Dictionary*. Those that are described here have syntax and usage that are specific to Windows.

## ABORT Statement

**Stops executing the current DATA step, SAS job, or SAS session**

**Valid in:** a DATA step

**Windows specifics:** Action of the ABEND and RETURN options; maximum value of *condition-code*

**See:** ABORT Statement in *SAS Language Reference: Dictionary*

### Syntax

**ABORT** <ABEND | RETURN> <*n*>;

**ABEND**
causes abnormal termination of the current SAS job or session for the current process. Further action is based on how your operating environment and site treat jobs that end abnormally.

**RETURN**
causes the immediate normal termination of the current SAS job or session. A condition code is returned indicating an error if a job ends abnormally.

***n***
allows you to specify a condition code that SAS returns to its calling program. The value of *n* must be an integer. Return codes 0 - 6 and those greater than 997 are used by SAS.

### Details

The ABORT statement causes SAS to stop processing the current DATA step.
The ABEND and RETURN options both terminate the SAS process, job, or session.

### See Also

- "Return Codes and Completion Status" on page 597

## ATTRIB Statement

**Associates a format, informat, label, and length with one or more variables**

**Valid in:** a DATA step

**Windows specifics:** length specification

**See:** ATTRIB Statement in *SAS Language Reference: Dictionary*

### Syntax

**ATTRIB** *variable-list-1 attribute-list-1*…<*variable-list-n attribute-list-n*>;

*Note:* Following is a simplified explanation of the ATTRIB statement syntax. For the complete syntax and its explanation, see the ATTRIB statement in *SAS Language Reference: Dictionary*. △

***attribute-list***

LENGTH=<$>*length*
specifies the length of the variables in *variable-list*. Under Windows, the length you can specify for a numeric variable ranges from 3 to 8 bytes.

### Details

Using the ATTRIB statement in the DATA step permanently associates attributes with variables by changing the descriptor information of the SAS data set that contains the variables.

## FILE Statement

**Specifies the current output file for PUT statements**

**Valid in:** a DATA step

**Windows specifics:** Valid values for *file specification*; valid values for *encoding-value*; valid options for *host-option-list*

**See:** FILE Statement in *SAS Language Reference: Dictionary*

### Syntax

**FILE** *file-specification*<ENCODING=*'encoding-value'*><*option-list*> <*host-option-list*>;

***file-specification***

can be any of the file specification forms discussed in "Referencing External Files" on page 146.

*Note:* The words AUX, CON, NUL, PRN, LPT1 - LPT9, and COM1 - COM9 are reserved words under Windows. Do not use them as filenames. △

**ENCODING=*'encoding-value'***

specifies the encoding to use when writing to the output file. The value for ENCODING= indicates that the output file has a different encoding from the current session encoding.

When you write data to the output file, SAS transcodes the data from the session encoding to the specified encoding.

For valid encoding values, see "Encoding Values in SAS Language Elements" in *SAS National Language Support (NLS): User's Guide*.

***host-option-list***

names external I/O statement options that are specific to the Windows operating environment. They can be any of the following:

BLKSIZE=*block-size*
BLK=*block-size*

specifies the number of bytes that are physically read or written in an I/O operation. The default is 8K. The maximum is 1M.

BLOCK | NOBLOCK

is used only in the context of named pipes. This option indicates whether the client is to wait if no data is currently available. BLOCK is the default value.

BYTE | MESSAGE

is used only in the context of named pipes. This option indicates the type of pipe. BYTE is the default value.

COMMAND

is used only in the context of Dynamic Data Exchange (DDE). This option enables you to issue a remote command for applications that do not use the SYSTEM topic name. For more information, see "Referencing the DDE External File" on page 274 and "Controlling Another Application Using DDE" on page 276.

EOFCONNECT

is used only in the context of named pipes and is valid only when defining the server. This option indicates that if an end-of-file (EOF) character is received from a client, the server should try to connect to the next client.

HOTLINK
: is used only in the context of Dynamic Data Exchange (DDE). For a complete description and an example of using this option, see “Using the DDE HOTLINK” on page 280.

IGNOREDOSEOF
: is used in the context of I/O operations on variable record format files. When this option is specified, any occurrence of ^Z is interpreted as character data and not as an end-of-file marker.

LRECL=*record-length*
: specifies the record length (in bytes). Under Windows, the default is 256. The value of *record-length* can range from 1 to 1,048,576 (1 megabyte).

MOD
: specifies that output should be appended to an existing file.

NOTAB
: is used only in the context of Dynamic Data Exchange (DDE). This option enables you to use non-tab character delimiters between variables. For more information about this option, see “Using the NOTAB Option with DDE” on page 279.

RECFM=*record-format*
: controls the record format. The following values are valid under Windows:

| | |
|---|---|
| F | indicates fixed format. |
| N | indicates binary format and causes the file to be treated as a byte stream. |
| P | indicates print format. |
| S370V | indicates the variable S370 record format (V). |
| S370VB | indicates the variable block S370 record format (VB). |
| S370VBS | indicates the variable block with spanned records S370 record format (VBS). |
| V \| D | indicates variable format. This is the default. |

The S370 values are valid with z/OS types of files only. That is, files that are binary, have variable-length records, and are in EBCDIC format. If you want to use a fixed-format z/OS, first copy it to a variable-length, binary z/OS file.

RETRY=*seconds*
: is used only in the context of named pipes. This option specifies how long a named pipe client should wait for a busy pipe. The minimum (and default) value for *seconds* is 10.

SERVER | CLIENT
: is used only in the context of named pipes. This option specifies the mode of a named pipe. The default value is SERVER.

TERMSTR=
: specifies the end-of-line character for the file. Use this option to share files between the UNIX and Windows operating environments. Valid values are:

CRLF
: Carriage return line feed. Use TERMSTR=CRLF to write Windows formatted files. CRLF is the default.

LF
: Line feed. Use TERMSTR=LF to write UNIX formatted files.

NL
New line. Use TERMSTR=NL to write UNIX formatted files.

### Details

The FILE statement routes the output from the PUT statement to either the same external file to which procedure output is written or to a different external file.

If the FILE statement includes the ENCODING argument and the reserved filerefs LOG or PRINT as the file-specification, SAS issues an error message. The ENCODING value in the FILE statement overrides the value of the ENCODING system option.

### See Also

- "ENCODING System Option" on page 499
- "Named Pipe Examples" on page 288 for examples of using some of these options
- "DDE Examples" on page 276 for examples of using some of these options

## FILENAME Statement

**Associates a SAS fileref with an external file or a logical file device**

**Valid in:** anywhere in a SAS program

**Windows specifics:** Valid values for *access-method*; valid values for *device-type*; valid filenames for *external-file*; valid values for *encoding*; valid options in *host-option-list*

**See:** FILENAME Statement in *SAS Language Reference: Dictionary*

### Syntax

**FILENAME** *fileref* <*device-type*> '*external-file*'
<ENCODING='*encoding-value*'><*host-option-list*>;

**FILENAME** *fileref device-type* <'*external-file*'>
<ENCODING=*encoding-value*><*host-option-list*>;

**FILENAME** *fileref* <*device-type*> ('*directory-1*'<,...*directory-n*'>)
<ENCODING=*encoding-value*><*host-option-list*>;

*Note:* This is a simplified version of the FILENAME statement syntax. For the complete syntax and its explanation, see the FILENAME statement in *SAS Language Reference: Dictionary*. △

***fileref***
is any valid fileref, as discussed in "Using a Fileref" on page 147.

For examples of using filerefs in member-name syntax (also called aggregate syntax), see "Assigning a Fileref to a Directory" on page 149. For a discussion of the rules SAS uses when accessing files through filerefs, see "Understanding How Concatenated Directories Are Accessed" on page 153.

*Note:* The words AUX, CON, NUL, PRN, LPT1 - LPT9, and COM1 - COM9 are reserved words under Windows. Do not use them as filerefs. △

***device-type***

enables you to read and write data from devices rather than files. The following values are valid:

CATALOG
: reads a SAS catalog as an external flat file.

COMMPORT
: reads data from and writes data to a communications port.

DDE
: reads data from and writes data to another application using Dynamic Data Exchange. For more information, see "DDE Syntax within SAS" on page 273.

DISK
: reads data from and writes data to a disk file. Under Windows, DISK is the default for *device-type*.

DRIVEMAP
: displays information about the available hard drives (local and networked).

DUMMY
: specifies a null output device. This value is especially useful in testing situations.

EMAIL
: lets you send electronic mail programmatically from SAS. For more information, see "Sending E-Mail Using SAS" on page 40.

FTP
: lets you access information on other machines using TCP/IP. You must have TCP/IP software and a WINSOCK.DLL installed on your local machine. You must also be able to connect to a machine that can function as an FTP server. For more information about using the FTP access method, see the FILENAME statement in *SAS Language Reference: Dictionary*.

NAMEPIPE
: writes data to a named pipe. For more information, see "Using Named Pipes" on page 286.

NOTESDB
: writes data to a Lotus Notes database. For more information, see "Populating a Lotus Notes Database Using the DATA Step and SCL Code" on page 214.

PIPE
: writes data to an unnamed pipe. For more information, see "Using Unnamed Pipes" on page 284.

PLOTTER
: indicates that you are accessing a plotter. Windows printing is not used. This device-type keyword is used solely in conjunction with SAS/GRAPH software.

PRINTER
: indicates that you are accessing a printer file or device. By default, output is routed through Windows printing when you use this device-type keyword. For more information about altering your default printer, see the system option "SYSPRINT System Option" on page 566.

SOCKET
: lets you read and write information over a TCP/IP socket. You must have TCP/IP software and a WINSOCK.DLL installed on your local machine. The SOCKET access method uses the nonblocking method of issuing socket requests. For more

information about using the SOCKET access method, see the FILENAME statement and FILENAME function in *SAS Language Reference: Dictionary*.

TEMP

creates a temporary file that exists only as long as the filename is assigned. The temporary file can be accessed only through the logical name and is available only while the logical name exists. A physical path name is never shown to the user. If a physical path name is specified, an error is returned. Files that are manipulated by the TEMP device can have the same attributes and behave identically to DISK files.

For an example of specifying a device type in the FILENAME statement, see "Advanced External I/O Techniques" on page 160.

*Note:* The TAPE and TERMINAL device-type keywords (documented in *SAS Language Reference: Dictionary*) are not applicable to the Windows operating environment. If you use one of these device-type keywords in your SAS program under Windows, you receive an error message. Also, while the DISK device-type keyword is accepted under Windows, it is ignored because disk files are the default under Windows. △

***external-file***

can be any valid Windows file specification that is enclosed in quotes. (for more information, see "Referencing External Files" on page 146).

**ENCODING=*'encoding-value'***

specifies the encoding to use when reading from or writing to the external file. The value for ENCODING= indicates that the external file has a different encoding from the current session encoding.

When you read data from an external file, SAS transcodes the data from the specified encoding to the session encoding. When you write data to an external file, SAS transcodes the data from the session encoding to the specified encoding.

For valid encoding values, see "Encoding Values in SAS Language Elements" in *SAS National Language Support (NLS): User's Guide*.

***host-option-list***

names external I/O statement options that are specific to Windows. They can be any of the following:

ALTDEST=*filename*

is for use only with the PRINTER device type. *Filename* specifies a file destination to write to when you direct output to the fileref. Although the output is written to disk and not to the printer, the output is still formatted by using the printer driver that is associated with the printer that you specified with the *external-file* argument. For example,

```
filename groupHP printer
   "HP LaserJet 4si, 1st floor"
   altdest=
      "C:\My SAS Files\Printer output\out.prn";
```

uses the printer driver that is associated with the named printer (an HP LaserJet 4si) to create the output in **`out.prn`**. No output is actually sent to the printer when you use this fileref.

BAUD=

sets the baud rate. The value for baud-rate depends on your communications hardware. It must be an integer. This host option is valid only if you specify the COMMPORT device-type keyword.

BITS=

sets the transmission bits. Values are 5 through 8. This host option is valid only when you specify the COMMPORT device-type keyword.

*Note:* For the 8250 serial port, invalid combinations are 5 data bits with 2 stop bits and 6, 7, or 8 data bits with 1.5 stop bits. △

BLKSIZE=*block-size*
BLK=*block-size*
specifies the number of bytes that are physically read or written in an I/O operation. The default is 8K. The maximum is 1M.

BLOCK | NOBLOCK
is used only in the context of named pipes. This option indicates whether the client is to wait if no data is currently available. BLOCK is the default value.

BYTE | MESSAGE
is used only in the context of named pipes. This option indicates the type of pipe. BYTE is the default value.

COMMAND
is used only in the context of Dynamic Data Exchange (DDE). This option enables you to issue a remote command for applications that do not use the SYSTEM topic name. For more information, see "Referencing the DDE External File" on page 274 and "Controlling Another Application Using DDE" on page 276.

COMTIMEOUT=*value*
controls how a communications port timeout is handled. A timeout occurs when no data is available at the communications port for a period of time, usually 60 seconds. The COMTIMEOUT= option can have the following values:

| | |
|---|---|
| EOF | returns an end-of-file (EOF) character when a timeout occurs. This is the default behavior. The EOF character causes the current DATA step to terminate. |
| WAIT | instructs the communications port to wait forever for data. In other words, this value overrides the timeout. In this case, no record is returned to the DATA step until data are available. This can cause your program to go into an infinite loop, so use this value with caution. |
| ZERO | returns a record length of 0 bytes when a timeout occurs. However, the DATA step does not terminate; it simply tries to read data again. |

This host-option is valid only if you specify the COMMPORT device-type keyword.

CONSOLE=*state*
specifies the state of the DOS window when an application is opened using pipes. Valid states are:

| | |
|---|---|
| MAX | opens the DOS window maximized |
| MIN | opens the DOS window minimized |
| NORMAL | opens the DOS window using the default for the machine. |

This host-option is valid only if you specify the PIPE keyword.

DROPNULL=
is used to discard null bytes when they are received. The valid values are

| | |
|---|---|
| ON | specifies to discard null bytes when they are received. |

| | |
|---|---|
| OFF | specifies not to discard null bytes when they are received. OFF is the default value. |

This host option is valid only if you specify the COMMPORT device-type keyword. For example:

```
filename portin commport 'com1:' dropnull=off;
```

EOFCONNECT
is used only in the context of named pipes and is valid only when you are defining the server. This option indicates that if an end-of-file (EOF) character is received from a client, the server should try to connect to the next client.

FLOW=
controls the transmission control flow. Values are: XONXOFF, DTRDSR, or RTSCTS. This host option is valid only if you specify the COMMPORT device-type keyword.

HOTLINK
is used only in the context of Dynamic Data Exchange (DDE). For a complete description and an example of how to use this option, see “Using the DDE HOTLINK” on page 280.

IGNOREDOSEOF
is used in the context of I/O operations on variable record format files. When this option is specified, any occurrence of ^Z is interpreted as character data and not as an end-of-file marker.

LRECL=*record-length*
specifies the record length (in bytes). Under Windows, the default is 256. The value of *record-length* can range from 1 to 1,048,576 (1 megabyte).

MOD
specifies that output should be appended to an existing file.

NOTAB
is used only in the context of Dynamic Data Exchange (DDE). This option enables you to use nontab character delimiters between variables. For more information about this option, see “Using the NOTAB Option with DDE” on page 279.

PARITY=
sets the parity check bit. Values are NONE, ODD, EVEN, MARK, or SPACE. This host option is valid only if you specify the COMMPORT device-type keyword.

RECFM=*record-format*
controls the record format. The following values are valid under Windows:

| | |
|---|---|
| F | indicates fixed format. |
| N | indicates binary format and causes the file to be treated as a byte stream. N is not valid for the PIPE or the NAMEPIPE device types. |
| P | indicates print format. |
| S370V | indicates the variable S370 record format (V). |
| S370VB | indicates the variable block S370 record format (VB). |
| S370VBS | indicates the variable block with spanned records S370 record format (VBS). |
| V \| D | indicates variable format. This is the default. |

The S370 values are valid with z/OS types of files only. That is, files that are binary, have variable-length records, and are in EBCDIC format. If you want to use a fixed-format z/OS file, first copy it to a variable-length, binary z/OS file.

RETRY=*seconds*
is used only in the context of named pipes. This option specifies how long a named pipe client should wait for a busy pipe. The minimum (and default) value for *seconds* is 10.

RCONST=*seconds*
specifies the initial read time-out value in 0.001 of a second (1000 = 1 second). The default is 8 seconds. This host-option is valid only if you specify the COMMPORT device-type keyword.

RMULTI= *seconds*
specifies the subsequent read time-out value in 0.001 of a second (1000 = 1 second). This host-option is valid only if you specify the COMMPORT device-type keyword.

SERVER | CLIENT
is used only in the context of named pipes. This option specifies the mode of a named pipe. The default value is SERVER.

STOP=
sets the stop bit. Values are ONE, TWO, ONEHALF. This host option is valid only if you specify the COMMPORT device-type keyword.

*Note:* For the 8250 serial port, invalid combinations are 5 data bits with 2 stop bits and 6, 7, or 8 data bits with 1.5 stop bits. △

WCONST=*seconds*
specifies the initial time-out value in 0.001 of a second (1000 = 1 second). This host option is valid only if you specify the COMMPORT device-type keyword.

WMULTI=*seconds*
specifies the subsequent time-out value in 0.001 of a second (1000 = 1 second). This host option is valid only if you specify the COMMPORT device-type keyword.

## Details

The FILENAME statement temporarily associates a valid SAS name with an external file or an output device. An external file is a file created and maintained in the Windows operating environment from which you need to read data.

*Operating Environment Information:* Under Windows NT, when the FILENAME statement is issued and the **`Start in`** field of the SAS properties **`Shortcut`** page is blank, Windows adds a shortcut for the FILEREF to the desktop. To avoid this, be sure that the **`Start in`** field is not blank. △

## See Also

□ "Advanced External I/O Techniques" on page 160

## FOOTNOTE Statement

**Prints up to ten lines of text at the bottom of the procedure output**

**Valid in:** anywhere in a SAS program

**Windows specifics:** Maximum length of footnote

**See:** FOOTNOTE Statement in *SAS Language Reference: Dictionary*

### Syntax

**FOOTNOTE** <*n*> <'*text*' | "*text*">;

***n***
specifies the relative line to be occupied by the footnote.

***text***
specifies the text of the footnote in single or double quotation marks.

### Details

The FOOTNOTE statement takes effect when the step or RUN group with which it is associated executes. Once you specify a footnote for a line, SAS repeats the same footnote on all pages until you cancel or redefine the footnote for that line.

The maximum footnote length under Windows is 256 characters. If the specified footnote is greater than the LINESIZE system option, the footnote is truncated to the line size.

## %INCLUDE Statement

**Includes and executes SAS statements and data lines**

**Valid in:** anywhere in a SAS program

**Windows specifics:** *source*, if a file specification is used; valid options for *encoding-value* and *host-options*

**See:** %INCLUDE Statement in *SAS Language Reference: Dictionary*

### Syntax

**%INCLUDE** *source* </<ENCODING='*encoding-value*'><*host-options*>>;

*Note:* This is a simplified version of the %INCLUDE statement syntax. For the complete syntax and its explanation, see the %INCLUDE statement in *SAS Language Reference: Dictionary*. △

***source***
describes the location of the information you want to access. The two possible sources are a file specification or internal lines. Under Windows, an asterisk (*) cannot be used to specify keyboard entry. The file specification can be any of the file specification forms discussed in "Referencing External Files" on page 146.

*Note:* When using member-name syntax and the member name contains a leading digit, enclose the member name in quotation marks. If the member name contains a macro variable reference, use double quotation marks. △

**ENCODING='*encoding-value*'**

specifies the encoding to use when reading from the specified source. The value for ENCODING= indicates that the specified source has a different encoding from the current session encoding.

When you read data from the specified source, SAS transcodes the data from the specified encoding to the session encoding.

For valid encoding values, see "Encoding Values in SAS Language Elements" in *SAS National Language Support (NLS): User's Guide*.

***host-options***

consists of statement options that are valid under Windows. Remember to precede the options list with a forward slash (/). The following options are available under Windows:

BLKSIZE=*block-size*
BLK=*block-size*

specifies the number of bytes that are physically read or written in an I/O operation. The default is 8K. The maximum is 1M.

BLOCK | NOBLOCK

is used only in the context of named pipes. This option indicates whether the client is to wait if no data is currently available.

BYTE | MESSAGE

is used only in the context of named pipes. This option indicates the type of pipe; BYTE is the default value.

EOFCONNECT

is used only in the context of named pipes and is valid only when defining the server. This option indicates that the server should try to connect to the next client if an end-of-file (EOF) character is received from a client.

IGNOREDOSEOF

is used in the context of I/O operations on variable record format files. When this option is specified, any occurrence of ^Z is interpreted as character data and not as an end-of-file marker.

LRECL=*record-length*

specifies the record length (in bytes). Under Windows, the default is 256. The value of *record-length* can range from 1 to 1,048,576 (1 megabyte).

NOTAB

is used only in the context of Dynamic Data Exchange. This option enables you to use non-tab character delimiters between variables. For more information about this option, see "Using the NOTAB Option with DDE" on page 279

RECFM=*record-format*

controls the record format. The following values are valid under Windows:

| | |
|---|---|
| F | indicates fixed format. |
| N | indicates binary format and causes the file to be treated as a byte stream. |
| P | indicates print format. |
| S370V | indicates the variable S370 record format (V). |
| S370VB | indicates the variable block S370 record format (VB). |
| S370VBS | indicates the variable block with spanned records S370 record format (VBS). |

| | |
|---|---|
| V \| D | indicates variable format. This is the default. |

The S370 values are valid with files laid out as z/OS files only—that is, files that are binary, have variable-length records, and are in EBCDIC format. If you want to use a fixed-format z/OS file, first copy it to a variable-length, binary z/OS file.

### Details

When you execute a program that contains the %INCLUDE statement, SAS executes your code, including any statements or data lines that you bring into the program with %INCLUDE.

## INFILE Statement

**Specifies an external file to read with an INPUT statement**

**Valid in:** a DATA step

**Windows specifics:** Valid values for *encoding-value*, *file-specification*, and *host-options*

**See:** INFILE Statement in *SAS Language Reference: Dictionary*

### Syntax

**INFILE** *file-specification* <ENCODING=*'encoding-value'*><*options*> <*host-options*>;

***file-specification***
identifies the source of input data records, usually an external file. The *file-specification* argument can be any of the file specification forms that are discussed in "Referencing External Files" on page 146. The reserved fileref CARDS enables the INFILE statement to reference instream data.

*Note:* The words AUX, CON, NUL, PRN, LPT1 - LPT9, and COM1 - COM9 are reserved words under Windows. Do not use them as filerefs. △

**ENCODING=*'encoding-value'***
specifies the encoding to use when reading from the external file. The value for ENCODING= indicates that the external file has a different encoding from the current session encoding.

When you read data from an external file, SAS transcodes the data from the specified encoding to the session encoding.

For valid encoding values, see "Encoding Values in SAS Language Elements" in *SAS National Language Support (NLS): User's Guide*.

***host-options***
names external I/O statement options that are specific to the Windows operating environment. They can be any of the following:

BLKSIZE= *block-size*
BLK=*block-size*
specifies the number of bytes that are physically read or written in an I/O operation. The default is 8K. The maximum is 1M.

BLOCK | NOBLOCK

is used only in the context of named pipes. This option indicates whether the client is to wait if no data is currently available. The default value is BLOCK.

BYTE | MESSAGE
: is used only in the context of named pipes. This option indicates the type of pipe. The default value is BYTE.

EOFCONNECT
: is used only in the context of named pipes and is valid only when defining the server. This option indicates that if an end-of-file (EOF) character is received from a client, the server should try to connect to the next client.

HOTLINK
: is used only in the context of Dynamic Data Exchange (DDE). For a complete description and an example of using this option, see "Using the DDE HOTLINK" on page 280.

IGNOREDOSEOF
: is used in the context of I/O operations on variable record format files. When this option is specified, any occurrence of ^Z is interpreted as character data and not as an end-of-file marker.

LRECL=*record-length*
: specifies the record length (in bytes). Under Windows, the default is 256. The value of *record-length* can range from 1 to 1,048,576 (1 megabyte).

NOTAB
: is used only in the context of Dynamic Data Exchange (DDE). This option enables you to use nontab character delimiters between variables. For more information about this option, see "Using the NOTAB Option with DDE" on page 279.

RECFM=*record-format*
: controls the record format. The following are valid values under Windows:

| | |
|---|---|
| F | indicates fixed format. |
| N | indicates binary format and causes the file to be treated as a byte stream. |
| P | indicates print format. |
| S370V | indicates the variable S370 record format (V). |
| S370VB | indicates the variable block S370 record format (VB). |
| S370VBS | indicates the variable block with spanned records S370 record format (VBS). |
| V \| D | indicates variable format. This is the default. |

The S370 values are valid with z/OS types of files only. That is, they are valid in files that are binary, have variable-length records, and are in EBCDIC format. If you want to use a fixed-format z/OS file, first copy it to a variable-length, binary z/OS file.

RETRY=*seconds*
: is used only in the context of named pipes. This option specifies how long a named pipe client should wait for a busy pipe. The minimum (and default) value for *seconds* is 10.

SERVER | CLIENT
: is used only in the context of named pipes. This option specifies the mode of a named pipe. The default value is SERVER.

TERMSTR=

specifies the end-of-line character for the file. Use this option to share files between the UNIX and Windows operating environments. Valid values are:

CRLF
: Carriage return line feed. Use TERMSTR=CRLF to read Windows formatted files. CRLF is the default.

LF
: Line feed. Use TERMSTR=LF to read UNIX formatted files.

NL
: New line. Use TERMSTR=NL to read UNIX formatted files.

## Details

If the INFILE statement includes the ENCODING argument and CARDS, CARDS4, DATALINES, or DATALINES4 as the file-specification, then SAS issues an error message. The ENCODING value in the INFILE statement overrides the value of the ENCODING system option.

## See Also

- □ "ENCODING System Option" on page 499
- □ "Named Pipe Examples" on page 288 for examples of using some of these options
- □ "DDE Examples" on page 276 for examples of using some of these options.

## LENGTH Statement

**Specifies the number of bytes SAS uses to store numeric variables**

**Valid in:** a DATA step

**Windows specifics:** Valid numeric variable lengths; valid values for *length*; valid values for *n*

**See:** LENGTH Statement in *SAS Language Reference: Dictionary*

### Syntax

**LENGTH** <*variable-1*><*...variable-n*> <$> <*length*> <DEFAULT=*n*>;

***length***
: Under Windows, can range from 3 to 8 bytes for numeric variables.

**DEFAULT=*n***
: changes the default number of bytes used for storing the values of newly created numeric variables from 8 to the value of *n*. Under Windows, the value of *n* can range from 3 to 8 bytes.

### Details

The LENGTH statement specifies the number of bytes SAS is to use for storing values of variables in each data set being created.

***CAUTION:***
**Any length less than 8 bytes may result in a loss of precision for the value of the variable.** △

## See Also

□ "Length and Precision of Variables under Windows" on page 579

## LIBNAME Statement

**Associates a libref with a SAS data library and lists file attributes for a SAS data library**

**Valid in:** anywhere in a SAS program

**Windows specifics:** Valid values for *engine*; specifications for *SAS-data-library*

**See:** LIBNAME Statement in *SAS Language Reference: Dictionary*

### Syntax

**LIBNAME** *libref <engine>'('SAS-data-library-1' <,...'SAS-data-library-n'> )'* <MEMLIB>;

**LIBNAME** *libref* _ALL_ LIST;

**LIBNAME** *libref* _ALL_ CLEAR;

*Note:* This is a simplified version of the LIBNAME statement syntax. For the complete syntax and its explanation, see the LIBNAME statement in *SAS Language Reference: Dictionary*. △

***libref***
is any valid libref, as documented in *SAS Language Reference: Dictionary*.

***engine-name***
is one of the following library engines supported under Windows:

| | |
|---|---|
| V9 | accesses SAS System 9 and SAS 9.1 data sets. You can use the nickname BASE for this engine. |
| V8 | accesses Version 8, Release 8.1, and Release 8.2 data sets. |
| V7 | accesses Version 7 data sets. |
| V6 | accesses Release 6.08 through Release 6.12 data sets. |
| V604 | accesses Release 6.03 and Release 6.04 data sets. |
| XML | generates an XML document from a SAS data set. |
| XPORT | accesses transport format files. |
| BMDP | accesses BMDP data files in a 32–bit operating environment. |
| OSIRIS | accesses OSIRIS data files. |
| SPSS | accesses SPSS system files. |

For more information about these engines, see "Multiple Engine Architecture" on page 122 and the discussion of engines in *SAS Language Reference: Dictionary*.

***SAS-data-library***
: is the physical name of a SAS data library under Windows. It must be a valid Windows pathname or an environment variable that is set to a valid Windows pathname. You can concatenate several Windows directories to serve as a single SAS data library. When you specify multiple libraries, use parentheses around the first and last library pathnames. For more information about concatenated SAS data libraries, see "Understanding How Multi-Folder SAS Data Libraries Are Accessed" on page 131.

**MEMLIB**
: specifies to use extended server memory for this library. For more information about using extended memory, see "Memory-Based Libraries" on page 199.

## Details

The LIBNAME statement associates a libref with a permanent SAS data library. It also can be used to list the file attributes of a SAS data library. (The LIBNAME statement is also used to clear a libref. For more information, see "Clearing Librefs" on page 130.)

*Note:* The words AUX, CON, NUL, PRN, LPT1 - LPT9, and COM1 - COM9 are reserved words under Windows. Do not use them as librefs. △

**Associating Librefs** Use one of the following forms of the LIBNAME statement to associate a libref or an engine with a SAS data library:

**LIBNAME** *libref <engine>* '*SAS-data-library*'

**LIBNAME** *libref <engine>* ('*SAS-data-library-1*' <,...'*SAS-data-library-n*')>;

Use quotation marks when *SAS-data-library* is a physical path. Quotation marks are not needed when you concatenate librefs.

You can use the same arguments with these forms of the LIBNAME statement as shown in the LIBNAME statement syntax.

**Listing Data Library Attributes** With the LIST option, you can use the LIBNAME statement to list attributes of SAS data libraries. Output 21.1 shows the results of the following LIBNAME statement:

```
libname sashelp list;
```

**Output 21.1** Data Library Attributes Listed by the LIBNAME Statement

```
5    libname sashelp list;
1    libname sashelp list;
NOTE: Libref=   SASHELP
      Scope=    Kernel
      Levels=   27
        -Level 1-
      Engine=   V9
      Physical Name= C:\Program Files\SAS\SAS 9.1\nls\en\SASCFG
      File Name= C:\Program Files\SAS\SAS 9.1\nls\en\SASCFG
        -Level 2-
      Engine=   V9
      Physical Name= C:\Program Files\SAS\SAS 9.1\core\sashelp
      File Name= C:\Program Files\SAS\SAS 9.1\core\sashelp

       . . .

      -Level 27-
      Engine=   V9
      Physical Name= C:\Program Files\SAS\SAS 9.1\webhound\sashelp
      File Name= C:\Program Files\SAS\SAS 9.1\webhound\sashelp
2    run;
```

## See Also

- "LIBNAME Statement" in *SAS Output Delivery System: User's Guide*
- "LIBNAME Statement" in *SAS Metadata LIBNAME Engine User's Guide*
- "LIBNAME Statement" in *SAS XML LIBNAME Engine User's Guide*
- "LIBNAME Statement" in *SAS/ACCESS for Relational Databases: Reference*
- "Using Data Libraries" on page 125

## SYSTASK Statement

**Executes, lists, or terminates asynchronous tasks**

**Valid in:** anywhere in a SAS program

**Windows specifics:** all

### Syntax

SYSTASK COMMAND "*operating system-command*"
 <WAIT | NOWAIT>
 <TASKNAME=*taskname*>
 <MNAME=*name-var*>
 <STATUS=*stat-var*>
 <SHELL<="*shell-command*">>;

SYSTASK LIST <_ALL_ | *taskname*> <STATE> <STATVAR>;

SYSTASK KILL *taskname* <*taskname...*>;

**COMMAND**
executes the *operating system-command*

**LIST**
lists either a specific active task or all of the active tasks in the system.

**KILL**
forces the termination of the specified task(s).

***operating system command***
specifies the name of a Windows command (including any command-specific options). Enclose the command in either single or double quotation marks. If the command options require quotes, repeat the quotes. For example:

```
systask command "find ""my text"" c:\mydir\myfile.sas"
```

*Note:* The operating system command that you specify cannot require input from the keyboard. △

**WAIT | NOWAIT**
determines whether SYSTASK COMMAND suspends execution of the current SAS session until the task has completed. NOWAIT is the default. For tasks that are started with the NOWAIT argument, you can use the WAITFOR statement when necessary to suspend execution of the SAS session until the task has finished.

**TASKNAME=*taskname***
specifies a name that identifies the task. Task names must be unique among all active tasks. A task is active if it is running, or if it has completed and has not been waited for using the WAITFOR statement. Duplicate task names generate an error in the SAS log. If you do not specify a task name, SYSTASK will automatically generate a name. If the task name contains a blank character, enclose the task name in quotes.

**MNAME=*name-var***
specifies a macro variable in which you want SYSTASK to store the task name that it automatically generated for the task. If you specify both the TASKNAME option and the MNAME option, SYSTASK copies the name you specified with TASKNAME into the variable that you specified with MNAME.

**STATUS=*stat-var***
specifies a macro variable in which you want SYSTASK to store the status of the task. Status variable names must be unique among all active tasks.

**SHELL<="*shell-command*">**
specifies that the *operating system-command* should be executed with the operating system shell command. If you specify a shell-command, SYSTASK uses the shell command that you specify to invoke the shell; otherwise, SYSTASK uses the default shell. Enclose the shell command in quotes.

**_ALL_**
specifies all active tasks in the system.

**STATE**
specifies to display the status of the task, which can be Start Failed, Running, or Complete.

**STATVAR**
specifies to display the status variable associated with the task. The status variable is the variable that you assigned with the STATUS option in the SYSTASK COMMAND statement.

## Details

SYSTASK allows you to execute operating system-specific commands from within your SAS session or application. Unlike the X statement, SYSTASK runs these commands as asynchronous tasks, which means that these tasks execute independently of all other tasks that are currently running. Asynchronous tasks run in the background, so you can perform additional tasks while the asynchronous task is still running.

For example, to copy a SAS program, you might use this statement:

```
systask command "copy myprog.sas myprog1.sas"
        taskname="copyfile" status=copystat;
```

The return code from the **copy** command is saved in the macro variable COPYSTAT.

*Note:* Windows command output is not written to the SAS log. △

Program steps that follow the SYSTASK statements in SAS applications usually depend on the successful execution of the SYSTASK statements. Therefore, syntax errors in some SYSTASK statements will cause your SAS application to abort.

There are two types of tasks that can be run with SYSTASK:

Task
: All tasks started with SYSTASK COMMAND are of type Task. For these tasks, if you do not specify STATVAR or STATE, then SYSTASK LIST displays the task name, type, and state, and the name of the status macro variable. To terminate tasks of type Task, use SYSTASK KILL.

SAS/Connect Process
: Tasks started from SAS/Connect with the SIGNON statement or command, and RSUBMIT statement are of type SAS/Connect Process. To display SAS/Connect processes, use the LISTTASK statement to display the task name, type, and state. To terminate a SAS/Connect process, use the KILLTASK statement. For information on SAS/Connect processes, see *SAS/CONNECT User's Guide*.

*Note:* The preferred method for displaying any task (not just SAS/Connect processes) is to use the LISTTASK statement instead of SYSTASK LIST.

The preferred method for ending a task is using the KILLTASK statement in place of SYSTASK KILL. △

The SYSRC macro variable contains the return code for the SYSTASK statement. The status variable that you specify with the STATUS option contains the return code of the process started with SYSTASK COMMAND. To ensure that a task executes successfully, you should monitor both the status of the SYSTASK statement and the status of the process that is started by the SYSTASK statement.

If a SYSTASK statement cannot execute successfully, the SYSRC macro variable will contain a non-zero value. For example, there may be insufficient resources to complete a task, or the SYSTASK statement may contain syntax errors. With the SYSTASK KILL statement, if one or more of the processes cannot be terminated, SYSRC is set to a non-zero value.

When a task is started, its status variable is set to NULL. You can use the status variables for each task to determine which tasks failed to complete. Any task whose status variable is NULL did not complete execution. See WAITFOR for more information about the status variables.

Unlike the X statement, you cannot use the SYSTASK statement to start a new interactive session.

## See Also

- □ "WAITFOR Statement" on page 461
- □ "X Statement" on page 462

## TITLE Statement

**Specifies title lines for SAS output**

**Valid in:** anywhere in a SAS program

**Windows specifics:** Maximum length of the title

**See:** TITLE Statement in *SAS Language Reference: Dictionary*

### Syntax

**TITLE** <*n*> <'*text*' | "*text*">;

***n***
specifies the relative line that contains the title line.

***'text'* | *"text"***
specifies text that is enclosed in single or double quotation marks.

### Details

The TITLE statement specifies title lines to be printed on SAS print files and other SAS output. A TITLE statement takes effect when the DATA or PROC step or RUN group with which it is associated executes. Once you specify a title for a line, it is used for all subsequent output until you cancel the title or define another title for that line.

Under Windows, the maximum title length is 256 characters. If the specified title is greater than the LINESIZE system option, the title is truncated to the line size.

## WAITFOR Statement

**Suspends execution of the current SAS session until the specified tasks finish executing**

**Valid in:** anywhere in a SAS program

**Windows specifics:** all

### Syntax

WAITFOR<_ANY_ | _ALL_> *taskname* <*taskname*...><TIMEOUT=*seconds*>;

***taskname***
specifies the name of the task(s) that you want to wait for. See "SYSTASK Statement" on page 458 for information about task names. The task name(s) that you specify must match exactly the task names assigned through the SYSTASK COMMAND statement. You cannot use wildcards to specify task names.

**_ANY_ | _ALL_**

suspends execution of the current SAS session until either one or all of the specified tasks finishes executing. The default setting is _ANY_, which means that as soon as one of the specified task(s) completes executing, the WAITFOR statement will finish executing.

**TIMEOUT=*seconds***
specifies the maximum number of seconds that WAITFOR should suspend the current SAS session. If you do not specify the TIMEOUT option, WAITFOR will suspend execution of the SAS session indefinitely.

### Details

The WAITFOR statement suspends execution of the current SAS session until the specified task(s) finish executing or until the TIMEOUT interval (if specified) has elapsed. If the specified task was started with the XWAIT option, then the WAITFOR statement ignores that task.

For example, the following statements start three different SAS jobs and suspend the execution of the current SAS session until those three jobs have finished executing:

```
systask command "sas myprog1.sas" taskname=sas1;
systask command "sas myprog2.sas" taskname=sas2;
systask command "sas myprog3.sas" taskname=sas3;
waitfor _all_ sas1 sas2 sas3;
```

The SYSRC macro variable contains the return code for the WAITFOR statement. If a WAITFOR statement cannot execute successfully, the SYSRC macro variable will contain a non-zero value. For example, the WAITFOR statement may contain syntax errors. If the number of seconds specified with the TIMEOUT option elapses, then the WAITFOR statement finishes executing, and SYSRC is set to a non-zero value if

- you specify a single task that does not finish executing
- you specify more than one task and the _ANY_ option (which is the default setting), but none of the tasks finishes executing
- you specify more than one task and the _ALL_ option, and any one of the tasks does not finish executing.

Any task whose status variable is still NULL after the WAITFOR statement has executed did not complete execution.

### See Also

- "SYSTASK Statement" on page 458
- "X Statement" on page 462
- "XWAIT System Option" on page 577

## X Statement

**Runs an operating system command or a Windows application from within a SAS session**

**Valid in:** anywhere in a SAS program

**Windows specifics:** Valid values for *command*

**See:** X Statement in *SAS Language Reference: Dictionary*

## Syntax

**X** <'*command*'>;

**no argument**
places you in a Command prompt session, with an operating system prompt. Here you can execute Windows commands in the context of SAS working directory. There are some things that you cannot do from the Command prompt in this situation, such as define environment variables for use by your SAS session. (Environment variables must be defined before you invoke SAS). Type EXIT at the Command prompt and press ENTER to return to your SAS session.

***command***
specifies a Windows command or a Windows application. This argument can be anything you can specify at a DOS prompt (including the SAS command). Therefore, you can use the X statement to execute Windows applications. The command can be enclosed in quotes, but this is not required syntax.

The command is passed to Windows and executed in the context of the working directory. If errors occur, the appropriate error messages are displayed.

By default, you must type EXIT to return to your SAS session after the command has completed execution. Also by default, if you execute a Windows application such as Notepad, you must close the application before you can return to your SAS session. Specify NOXWAIT in an OPTIONS statement if you do not want to have to type EXIT. With NOXWAIT in effect, as soon as the command finishes execution, control is returned to your SAS session. Note, however, that if you execute a Windows application with the X statement, specifying NOXWAIT does not let you return to your SAS session until you close the application.

Another system option, XSYNC, controls whether you have to wait for the command to finish executing before you can return to your SAS session. If you specify NOXSYNC, you can start a Windows application with the X statement and return to your SAS session without closing the application. For additional details about these two system options, see "XWAIT System Option" on page 577 and "XSYNC System Option" on page 576.

## Details

The X statement issues a host command from within a SAS session when you run SAS in windowing mode. SAS executes the X statement immediately.

Under Windows, you can issue the X statement without the *command* argument.

There are other ways of running operating environment commands besides the X statement (and the X command) under Windows.

## See Also

- "X Command" on page 372
- "XSYNC System Option" on page 576
- "XWAIT System Option" on page 577
- "CALL SYSTEM Routine" on page 387
- The %SYSEXEC statement in "Macro Statements" on page 583
- "Running Windows or MS-DOS Commands from within SAS" on page 25
- "Adding Applications to the Tools Menu" on page 64

CHAPTER
22

# System Options under Windows

*SAS System Options under Windows* **467**
*Displaying SAS System Option Settings* **468**
*Changing SAS System Option Settings* **468**
  *Syntax for System Options in the SAS Invocation or SAS Configuration File* **468**
  *Syntax for Concatenating Libraries in SAS System Options* **469**
  *Syntax for System Options in the OPTIONS Statement* **470**
*Processing System Options That Are Set in Several Places* **470**
*Summary of System Options for Windows* **470**
  *Options Not Applicable to the Windows Environment* **482**
  *ACCESSIBILITY System Option* **482**
  *ALTLOG System Option* **483**
  *ALTPRINT System Option* **484**
  *AUTHSERVER System Option* **484**
  *AUTOEXEC System Option* **485**
  *AWSCONTROL System Option* **486**
  *AWSDEF System Option* **487**
  *AWSMENU System Option* **488**
  *AWSMENUMERGE System Option* **488**
  *AWSTITLE System Option* **489**
  *BUFNO System Option* **490**
  *BUFSIZE System Option* **491**
  *CATCACHE System Option* **492**
  *CLEANUP System Option* **493**
  *COMDEF System Option* **494**
  *CONFIG System Option* **494**
  *DBCS System Option* **495**
  *DBCSLANG System Option* **495**
  *DBCSTYPE System Option* **496**
  *DEVICE System Option* **496**
  *ECHO System Option* **497**
  *EMAILDLG System Option* **498**
  *EMAILSYS System Option* **498**
  *ENCODING System Option* **499**
  *ENGINE System Option* **499**
  *ENHANCEDEDITOR System Option* **501**
  *FILTERLIST System Option* **501**
  *FONT System Option* **502**
  *FONTALIAS System Option* **503**
  *FONTSLOC System Option* **504**
  *FORMCHAR System Option* **505**
  *FULLSTIMER System Option* **505**

*GISMAPS System Option* **507**
*HELPINDEX System Option* **507**
*HELPLOC System Option* **508**
*HELPREGISTER System Option* **509**
*HELPTOC System Option* **511**
*HOSTPRINT System Option* **512**
*ICON System Option* **513**
*INITSTMT System Option* **513**
*JREOPTIONS System Option* **514**
*LINESIZE System Option* **515**
*LOADMEMSIZE System Option* **516**
*LOCALE System Option* **517**
*LOG System Option* **517**
*MAPS System Option* **518**
*MAXMEMQUERY System Option* **518**
*MEMBLKSZ System Option* **519**
*MEMCACHE System Option* **520**
*MEMLIB System Option* **521**
*MEMMAXSZ System Option* **522**
*MEMSIZE System Option* **523**
*MSG System Option* **523**
*MSGCASE System Option* **524**
*MSYMTABMAX System Option* **525**
*MVARSIZE System Option* **525**
*NEWS System Option* **526**
*NLSCOMPATMODE System Option* **527**
*NUMKEYS System Option* **527**
*NUMMOUSEKEYS System Option* **528**
*OBS System Option* **529**
*PAGENO System Option* **529**
*PAGESIZE System Option* **530**
*PAPERTYPE System Option* **531**
*PATH System Option* **532**
*PFKEY System Option* **532**
*PRINT System Option* **535**
*PRTABORTDLGS System Option* **536**
*PRTPERSISTDEFAULT System Option* **536**
*PRTSETFORMS System Option* **537**
*REALMEMSIZE System Option* **538**
*REGISTER System Option* **539**
*RESOURCESLOC System Option* **539**
*RSASUSER System Option* **540**
*RTRACE System Option* **541**
*RTRACELOC System Option* **541**
*S System Option* **542**
*S2 System Option* **543**
*SASAUTOS System Option* **544**
*SASCONTROL System Option* **545**
*SASHELP System Option* **546**
*SASINITIALFOLDER System Option* **546**
*SASUSER System Option* **547**
*SCROLLBARFLASH System Option* **548**
*SET System Option* **549**
*SGIO System Option* **550**

*SORTANOM System Option* **551**
*SORTCUT System Option* **552**
*SORTCUTP System Option* **553**
*SORTDEV System Option* **554**
*SORTPARM System Option* **555**
*SORTPGM System Option* **555**
*SORTSIZE System Option* **556**
*SPLASH System Option* **557**
*SPLASHLOC System Option* **558**
*SSLCERTISS System Option* **559**
*SSLCERTSERIAL System Option* **559**
*SSLCERTSUBJ System Option* **560**
*SSLCLIENTAUTH System Option* **561**
*SSLCRLCHECK System Option* **562**
*STIMEFMT System Option* **562**
*STIMER System Option* **563**
*SYSGUIFONT System Option* **564**
*SYSIN System Option* **565**
*SYSPARM System Option* **565**
*SYSPRINT System Option* **566**
*SYSPRINTFONT System Option* **567**
*TOOLDEF System Option* **569**
*UPRINTMENUSWITCH System Option* **570**
*USER System Option* **570**
*USERICON System Option* **571**
*VERBOSE System Option* **572**
*WEBUI System Option* **573**
*WINDOWSMENU System Option* **573**
*WORK System Option* **574**
*XCMD System Option* **575**
*XMIN System Option* **575**
*XSYNC System Option* **576**
*XWAIT System Option* **577**

## SAS System Options under Windows

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 data libraries. System options can be specified various ways: in the SAS command, in a SAS configuration file, in an OPTIONS statement (either in a SAS program or in a SAS autoexec file), in the SAS System Options window, or in SCL programs. "Summary of System Options for Windows" on page 470 gives specific information about where each SAS system option can be specified.

Once a system option is set, it affects all subsequent DATA and PROC steps in a program or SAS session until it is respecified. For example, the CENTER system option affects all subsequent output from a program, regardless of the number of steps in the program.

Some SAS system options have the same effect (and usually the same name) as data set or statement options. For example, the BUFSIZE system option is analogous to the BUFSIZE= data set option. In the case of overlapping options, SAS uses the following rules of precedence:

□ data set option values (highest precedence)

- statement option values (precedence over system options)
- system option values (lowest precedence).

## Displaying SAS System Option Settings

SAS system options are set to the default values. To display the settings of the SAS system options in the SAS log, use the OPTIONS procedure. For example, the following statement produces a list of options, one option per line, with a brief explanation of what each option does:

```
proc options; run;
```

You can specify the SHORT option in the PROC OPTIONS statement to produce a list of option settings with no explanation of the options. For more information, see the OPTIONS procedure in *Base SAS Procedures Guide*.

In an interactive SAS session, the SAS System Options window displays the settings of many SAS system options. However, it does not list the system options that are valid only at SAS invocation or the system options that are not available in all operating environments. To open the SAS System Options window, enter

Tools ► Options ► System

## Changing SAS System Option Settings

There are several ways to specify values for SAS system options:

- as part of the command that invokes SAS
- as part of a SAS configuration file that is processed when SAS initializes
- in a Windows environment variable (SAS_OPTIONS) that is processed when SAS initializes
- as part of a custom option set that is processed when you launch a new SAS process
- as part of the OPTIONS statement from within your SAS session
- using the interactive SAS System Options window, which you can access by selecting

  Tools ► Options ► System
- within SCL or SAS/AF programs, using the OPTSETC and OPTSETN SCL functions.

Some system options can be specified only when a SAS session or process is initialized (starts up), while other options can be changed as needed during your SAS session.

It is important to remember the differences in syntax between specifying a system option in the SAS command when you start SAS or in the SAS configuration file, and specifying a system option in the OPTIONS statement. The syntax for these situations is different, and if you use the wrong syntax, SAS generates an error message. For information on the OPTIONS statement, see *SAS Language Reference: Dictionary*.

### Syntax for System Options in the SAS Invocation or SAS Configuration File

When you specify a system option at initialization, it must be preceded by a hyphen (-). For on or off options, just list the keyword corresponding to the appropriate setting.

For example, the following command invokes SAS and indicates that SAS output should not be centered:

```
c:\sas\sas.exe -nocenter
```

For options that take a value, do not use an equal sign; follow the option name with a space and then the value. For example, the following SAS command invokes SAS with a line length of 132:

```
c:\sas\sas.exe -linesize 132
```

Physical names (that is, directory names or filenames) should be enclosed in double quotes when you use them in the SAS command or in the SAS configuration file. The quotes are especially necessary when the file or path name that you are specifying contains a space or single quote character, which are valid characters in Windows filenames. For example, the following SAS command invokes SAS and indicates that autocall macros are stored in the C:\SAS\CORE\SASMACRO directory:

```
c:\sas\sas.exe -sasautos "c:\sas\core\sasmacro"
```

Double quotation marks are also needed when an option value contains '=', as shown in this example:

```
c:\sas\sas.exe -set fruit "navel=orange"
```

To specify more than one option in the SAS command, simply separate each option with a space. For example, the following SAS command combines the three options shown previously in this section:

```
c:\sas\sas.exe -linesize 132 -nocenter
               -sasautos "c:\sas\core\sasmacro"
```

The SAS configuration file must contain only option settings; it cannot contain SAS statements. For example, a configuration file named MySASConfig.CFG may contain these option specifications (among others):

```
-nocenter
-noxwait
-pagesize 60
```

All SAS system options can appear in a SAS configuration file. For more information on SAS configuration files, see "SAS Configuration Files" on page 13.

## Syntax for Concatenating Libraries in SAS System Options

To provide more flexibility for storing SAS files across different drives, such as multiple logical drives on your hard disk or on a network, SAS lets you concatenate SAS libraries. The concept of concatenation within SAS means that you can specify multiple drives or directories when you specify certain system options in the SAS configuration file or in the SAS command. To specify concatenated directories, specify the directory names inside parentheses, enclose each directory name in double quotes, and separate the directory names with spaces.

One practical use of concatenation is the storage of SAS help catalogs. If you want to partition your SAS products among two or more directories, simply specify these multiple directories with the SASHELP option in the SAS configuration file, as in the following example:

```
-sashelp ("c:\sas\core\sashelp"
          "d:\sas\stat\sashelp")
```

## Syntax for System Options in the OPTIONS Statement

You can specify many 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 with another OPTIONS statement. For more information about the OPTIONS statement, see *SAS Language Reference: Dictionary*.

When you specify a system option in the OPTIONS statement, do not precede the option name with a hyphen (-). Also, for system options that take a value, use an equal sign (=), not a space. For example, the following statement specifies that output is not to be labeled with a date and that the line size should be 132:

```
options nodate linesize=132;
```

Physical names (that is, directory names or filenames) must be enclosed in quotes when used in the OPTIONS statement. For example, the following OPTIONS statement indicates that autocall macros are stored in the C:\SAS\CORE\SASMACRO directory:

```
options sasautos="c:\sas\core\sasmacro";
```

Any file specification that is not enclosed in quotes in the OPTIONS statement is assumed to be a logical name, that is, a fileref or an environment variable name. If no logical name is found, SAS issues an error message.

Not all system options can be specified in the OPTIONS statement. To find out whether a system option can be specified in the OPTIONS statement, look up the option name in Table 22.1 on page 471, which summarizes all SAS system option information, including where you can specify the options.

## Processing System Options That Are Set in Several Places

When the same system option is set in more than one place, the most recent specification is used. Therefore, the SAS System Options window or OPTIONS statement takes precedence over the SAS autoexec file; the SAS autoexec file takes precedence over the SAS command; and the SAS command takes precedence over the SAS configuration file and environment variable settings.

## Summary of System Options for Windows

Table 22.1 on page 471 lists all the system options available to SAS users under the Windows operating environment. Many of these options have no system-dependent behavior and are described completely in *SAS Language Reference: Dictionary*. Others are available only under Windows and are completely described here. Some system options are described here and in *SAS Language Reference: Dictionary*.

*Note:* Some system options in *SAS Language Reference: Dictionary* indicate that the system option may have additional operating environment information and to refer to the SAS documentation for your operating environment. If such a system option is not described in *SAS Companion for Windows*, the system option is to be used as described in *SAS Language Reference: Dictionary*. △

Use the following legend to determine where to find more information on a system option:

| | |
|---|---|
| ACC | indicates that the option is described in *SAS/ACCESS for Relational Databases: Reference* |

| | |
|---|---|
| COMP | indicates that the option is completely described in this section. Some options are not applicable to the Windows operating environment; these options are listed in “Options Not Applicable to the Windows Environment” on page 482. |
| CONN | indicates that the option is described in *SAS/CONNECT User's Guide* |
| DQ | indicates that the option is described in *SAS Data Quality Server: Reference* |
| IT | indicates that the option is described in the documentation for Integration Technologies, either with the Integration Technologies software or on the SAS web site. |
| LR | indicates that the option is not described here but is described in the system options portion of *SAS Language Reference: Dictionary*. |
| MACRO | indicates that the option is described in *SAS Macro Language: Reference* |
| METH | indicates that the option is described in *Communications Access Methods for SAS/CONNECT and SAS/SHARE* |
| NLS | indicates that the option is described in *SAS National Language Support (NLS): User's Guide* |
| SHR | indicates that the option is described in *SAS/SHARE User's Guide* |
| SPDE | indicates that the option is described in *SAS Scalable Performance Data Engine: Reference* |

**Table 22.1** Summary of SAS System Options

| Options Specification | Default Value | SAS invocation | Configuration file | SAS System Options window | OPTIONS statement | See |
|---|---|---|---|---|---|---|
| ACCESSIBILITY | STANDARD | X | X | | | COMP |
| ALTLOG arg | none | X | X | | | COMP |
| ALTPRINT arg | NOALTPRINT | X | X | | | COMP |
| APPLETLOC | none | X | X | X | X | LR |
| ARMAGENT | none | X | X | X | X | LR |
| ARMLOC | ARMLOC.LOG | X | X | X | X | LR |
| ARMSUBSYS | ARM_NONE | X | X | X | X | LR |
| ASYNCHIO | See LR | X | X | | | LR |
| AUTHPROVIDER DOMAIN | none | X | X | | | LR |
| AUTHSERVER | local and trusted servers | X | X | X | X | COMP |
| AUTOEXEC arg | AUTOEXEC.SAS if file is available; otherwise none | X | X | | | COMP |
| AUTOSAVELOC | none | X | X | X | X | LR |

| Options Specification | Default Value | SAS invocation | Configuration file | SAS System Options window | OPTIONS statement | See |
|---|---|---|---|---|---|---|
| AUTOSIGNON | NOAUTOSIGNON | X | X | X | X | CONN |
| AWSCONTROL | SYSTEMMENU MINMAX TITLE | X | X | X | X | COMP |
| AWSDEF arg | 80% of display height and width | X | X | X | X | COMP |
| AWSMENU | AWSMENU | X | X | X | X | COMP |
| AWSMENUMERGE | AWSMENUMERGE | X | X | X | X | COMP |
| AWSTITLE arg | none | X | X | | | COMP |
| BATCH | NOBATCH (interactive mode); BATCH (batch mode) | X | X | | | LR |
| BINDING arg | DEFAULT | X | X | X | X | LR |
| BOTTOMMARGIN | 0.000IN | X | X | X | X | LR |
| BUFNO arg | 1 | X | X | X | X | LR, COMP |
| BUFSIZE arg | 0 | X | X | X | X | LR, COMP |
| BYERR | BYERR | X | X | X | X | LR |
| BYLINE | BYLINE | X | X | X | X | LR |
| BYSORTED | BYSORTED | X | X | X | X | LR |
| CAPS | NOCAPS | X | X | X | X | LR |
| CARDIMAGE | NOCARDIMAGE | X | X | X | X | LR |
| CATCACHE arg | 0 | X | X | | | LR, COMP |
| CBUFNO arg | 0 | X | X | X | X | LR |
| CENTER | CENTER | X | X | X | X | LR |
| CHARCODE | NOCHARCODE | X | X | X | X | LR |
| CLEANUP | CLEANUP | X | X | X | X | LR, COMP |
| CMDMAC | NOCMDMAC | X | X | X | X | MACRO |
| CMPLIB arg | none | X | X | X | X | LR |
| CMPOPT arg | (NOEXTRAMATH NOMISSCHECK NOPRECISE NOGUARDCHECK) | X | X | X | X | LR |
| COLLATE | NOCOLLATE | X | X | X | X | LR |
| COLORPRINTING | COLORPRINTING | X | X | X | X | LR |

| **Options Specification** | **Default Value** | **SAS invocation** | **Configuration file** | **SAS System Options window** | **OPTIONS statement** | **See** |
|---|---|---|---|---|---|---|
| COMAMID arg | TCP | X | X | | X | CONN, SHR, METH |
| COMAUX1 arg | none | X | X | | | METH |
| COMAUX2 arg | none | X | X | | | METH |
| COMDEF arg | BOTTOM CENTER | X | X | | | COMP |
| COMPRESS arg | NO | X | X | X | X | LR |
| CONFIG arg | !*sasroot*\SASV9.CFG | X | X | | | COMP |
| CONNECTPERSIST | YES | X | X | | | CONN |
| CONNECTREMOTE arg | none | X | X | X | X | CONN |
| CONNECTSTATUS | CONNECTSTATUS | X | X | X | X | CONN |
| CONNECTWAIT | CONNECTWAIT | X | X | X | X | CONN |
| COPIES arg | 1 | X | X | X | X | LR |
| CPUCOUNT arg | 1 | X | X | X | X | LR |
| CPUID | CPUID | X | X | | | LR |
| DATASTMTCHK arg | COREKEYWORDS | X | X | X | X | LR |
| DATE | DATE | X | X | X | X | LR |
| DATESTYLE | MDY | X | X | X | X | LR |
| DBCS | NODBCS | X | X | | | NLS |
| DBCSLANG arg | NONE | X | X | | | NLS |
| DBCSTYPE arg | WINDOWS | X | X | | | NLS |
| DBSLICEPARM | THREADED_APPS, 2 | X | X | X | X | ACC |
| DBSRVTP | NONE | X | X | | | ACC |
| DETAILS | NODETAILS | X | X | X | X | LR |
| DEVICE arg | none | X | X | X | X | LR, COMP |
| DFLANG arg | ENGLISH | X | X | X | X | NLS |
| DKRICOND arg | ERROR | X | X | X | X | LR |
| DKROCOND arg | WARN | X | X | X | X | LR |
| DLDMGACTION | FAIL for batch mode; REPAIR for interactive mode | X | X | X | X | LR |
| DMR | NODMR | X | X | | | LR, CONN |

| Options Specification | Default Value | SAS invocation | Configuration file | SAS System Options window | OPTIONS statement | See |
|---|---|---|---|---|---|---|
| DMS | DMS | X | X | | | LR |
| DMSLOGSIZE arg | 99999 | X | X | | | LR |
| DMSOUTSIZE arg | 99999 | X | X | | | LR |
| DMSSYNCHK | NODMSSYNCHK | X | X | X | X | LR |
| DQLOCALE arg | none | X | X | X | X | DQ |
| DQSETUPLOC arg | none | X | X | X | X | DQ |
| DSNFERR | DSNFERR | X | X | X | X | LR |
| DTRESET | NODTRESET | X | X | X | X | LR |
| DUPLEX arg | NODUPLEX | X | X | X | X | LR |
| ECHO arg | NOECHO | X | X | | | COMP |
| ECHOAUTO | NOECHOAUTO | X | X | | | LR |
| EMAILAUTH PROTOCOL arg | NONE | X | X | | | LR |
| EMAILDLG arg | NATIVE | X | X | | | COMP |
| EMAILHOST arg | LOCALHOST | X | X | | | LR |
| EMAILID arg | none | X | X | | | LR |
| EMAILPORT arg | 25 | X | X | | | LR |
| EMAILPW arg | none | X | X | | | LR |
| EMAILSYS arg | MAPI | X | X | | | COMP |
| ENCODING arg | wlatin1 | X | X | | | NLS |
| ENGINE arg | V9 | X | X | | | LR, COMP |
| ENHANCEDEDITOR | ENHANCEDEDITOR | X | X | | | COMP |
| ERRORABEND | NOERRORABEND | X | X | X | X | LR |
| ERRORBYABEND | NOERRORBYABEND | X | X | X | X | LR |
| ERRORCHECK arg | NORMAL | X | X | X | X | LR |
| ERRORS arg | 20 | X | X | X | X | LR |
| EXPLORER | NOEXPLORER | X | X | | | LR |
| FILTERLIST arg | none | X | X | | | COMP |
| FIRSTOBS arg | 1 | X | X | X | X | LR |
| FMTERR | FMTERR | X | X | X | X | LR |
| FMTSEARCH arg | Work Library | X | X | X | X | LR |
| FONT arg | 'Sasfont' 8 | X | X | X | X | COMP |
| FONTALIAS arg | varies | X | X | | | COMP |
| FONTSLOC arg | !*sasroot*\core\resource | X | X | | | LR, COMP |

| Options Specification | Default Value | SAS invocation | Configuration file | SAS System Options window | OPTIONS statement | See |
|---|---|---|---|---|---|---|
| FORMCHAR arg | (see SASV9.CFG) | X | X | X | X | LR, COMP |
| FORMDLIM arg | none | X | X | X | X | LR |
| FORMS arg | DEFAULT | X | X | X | X | LR |
| FULLSTIMER | NOFULLSTIMER | X | X |  | X | COMP |
| GISMAPS | none | X | X | X | X | LR, COMP |
| GWINDOW | GWINDOW | X | X | X | X | LR |
| HELPENCMD | HELPENCMD | X | X |  |  | LR |
| HELPINDEX arg | See COMP | X | X |  |  | COMP |
| HELPLOC arg | ("!*sasuser*\classdoc" "!*sasroot*\core\help" "!*sasroot*\nls\en\help") | X | X |  |  | LR, COMP |
| HELPREGISTER arg | none | X | X |  |  | COMP |
| HELPTOC arg | See COMP | X | X |  |  | COMP |
| HOSTPRINT | HOSTPRINT | X | X | X | X | COMP |
| IBUFSIZE | 0 | X | X | X | X | LR |
| ICON | NOICON | X | X | X | X | COMP |
| IMPLMAC | NOIMPLMAC | X | X | X | X | MACRO |
| INITCMD arg | none | X | X |  |  | LR |
| INITSTMT arg | none | X | X |  |  | LR, COMP |
| INVALIDDATA arg | a period (.) | X | X | X | X | LR |
| JREOPTIONS | See SASV9.CFG | X | X |  |  | COMP |
| LABEL | LABEL | X | X | X | X | LR |
| _LAST_ arg | _NULL_ | X | X | X | X | LR |
| LEFTMARGIN arg | 0.000IN | X | X | X | X | LR |
| LINESIZE arg | varies | X | X | X | X | LR, COMP |
| LOADMEMSIZE | 0 | X | X |  |  | COMP |
| LOCALE | EN_US | X | X | X | X | NLS |
| LOG arg | *filename*.LOG in batch mode | X | X |  |  | COMP |
| LOGPARM | none | X | X |  |  | LR |
| MACRO | MACRO | X | X |  |  | MACRO |

| Options Specification | Default Value | SAS invocation | Configuration file | SAS System Options window | OPTIONS statement | See |
|---|---|---|---|---|---|---|
| MAPS arg | !*sasext0*\maps | X | X | X | X | LR COMP |
| MAUTOLOCDISPLAY | NOMAUTOLOC DISPLAY | X | X | X | X | MACRO |
| MAUTOSOURCE | MAUTOSOURCE | X | X | X | X | MACRO |
| MAXMEMQUERY | 0 | X | X | X | X | COMP |
| MAXSEGRATIO | 75 | X | X | X | X | SPDE |
| MCOMPILENOTE | none | X | X | X | X | MACRO |
| MEMBLKSIZE | 16 MB | X | X | | | COMP |
| MEMCACHE | OFF | X | X | X | X | COMP |
| MEMLIB | NOMEMLIB | X | X | | | COMP |
| MEMMAXSZ | 2G | X | X | | | COMP |
| MEMSIZE | 0 | X | X | | | COMP |
| MERGENOBY | NOWARN | X | X | X | X | LR |
| MERROR | MERROR | X | X | X | X | MACRO |
| METAAUTORESOURCES | none | X | X | | | LR |
| METACONNECT | none | X | X | X | X | LR |
| METAENCRYPTALG arg | none | X | X | | | LR |
| METAENCRYPT LEVEL | EVERYTHING | X | X | | | LR |
| METAID | see SASV9.CFG | X | X | | | LR |
| METAPASS | none | X | X | X | X | LR |
| METAPORT | see SASV9.CFG | X | X | X | X | LR |
| METAPROFILE | none | X | X | | | LR |
| METAPROTOCOL | BRIDGE | X | X | X | X | LR |
| METAREPOSITIORY | see SASV9.CFG | X | X | X | X | LR |
| METASERVER | see SASV9.CFG | X | X | X | X | LR |
| METAUSER | none | X | X | X | X | LR |
| MFILE | NOMFILE | X | X | X | X | MACRO |
| MINDELIMITER | none | X | X | X | X | MACRO |
| MINPARTSIZE | 0 | X | X | | | SPDE |
| MISSING arg | a period (.) | X | X | X | X | LR |
| MLOGIC | NOMLOGIC | X | X | X | X | MACRO |
| MLOGICNEST | NOMLOGICNEST | X | X | X | X | MACRO |
| MPRINT | NOMPRINT | X | X | X | X | MACRO |

| Options Specification | Default Value | SAS invocation | Configuration file | SAS System Options window | OPTIONS statement | See |
|---|---|---|---|---|---|---|
| MPRINTNEST | NOMPRINTNEST | X | X | X | X | MACRO |
| MRECALL | NOMRECALL | X | X | X | X | MACRO |
| MSG arg | !*sasroot*\core\sasmsg | X | X | | | COMP |
| MSGCASE | NOMSGCASE | X | X | | | COMP |
| MSGLEVEL arg | N | X | X | X | X | LR |
| MSTORED | NOMSTORED | X | X | X | X | MACRO |
| MSYMTABMAX arg | 4,194,304 bytes | X | X | X | X | COMP, MACRO |
| MULTENVAPPL | NOMULTENVAPPL | X | X | | X | LR |
| MVARSIZE arg | 4,096 bytes | X | X | X | X | COMP, MACRO |
| NETENCRYPT | NONETENCRYPT | X | X | X | X | CONN, SHR |
| NETENCRYPT ALGORITHM arg | none | X | X | X | X | CONN, SHR |
| NETENCRYPT KEYLEN arg | 0 | X | X | X | X | CONN, SHR |
| NETMAC | NETMAC | X | X | X | X | CONN, SHR |
| NEWS arg | none | X | X | | | LR, COMP |
| NLSCOMPATMODE | NONLSCOMPAT-MODE | X | X | | | NLS |
| NOTES | NOTES | X | X | X | X | LR |
| NUMBER | NUMBER | X | X | X | X | LR |
| NUMKEYS arg | varies | X | X | | | COMP |
| NUMMOUSEKEYS arg | 3 | X | X | | | COMP |
| OBJECTSERVER | NOOBJECTSERVER | X | X | | | LR |
| OBS arg | MAX | X | X | X | X | LR, COMP |
| ORIENTATION arg | PORTRAIT | X | X | X | X | LR |
| OVP | NOOVP | X | X | X | X | LR |
| PAGEBREAKINITIAL | NOPAGEBREAK INITIAL | X | X | | | LR |
| PAGENO arg | 1 | X | X | X | X | LR, COMP |
| PAGESIZE arg | varies | X | X | X | X | LR, COMP |

| Options Specification | Default Value | SAS invocation | Configuration file | SAS System Options window | OPTIONS statement | See |
|---|---|---|---|---|---|---|
| PAPERDEST arg | none | X | X | X | X | LR |
| PAPERSIZE arg | LETTER | X | X | X | X | LR |
| PAPERSOURCE arg | none | X | X | X | X | LR |
| PAPERTYPE arg | STANDARD | X | X | X | X | LR, COMP |
| PARM arg | none | X | X | X | X | LR |
| PARMCARDS arg | FT15F001 | X | X | X | X | LR |
| PATH arg | see SASV9.CFG | X | X | | | COMP |
| PFKEY arg | WIN | X | X | | | COMP |
| PRINT arg | *filename*.LST in batch mode | X | X | | | COMP |
| PRINTERPATH arg | none | X | X | X | X | LR |
| PRINTINIT | NOPRINTINIT | X | X | | | LR |
| PRINTMSGLIST | PRINTMSGLIST | X | X | X | X | LR |
| PRTABORTDLGS | BOTH | X | X | X | X | COMP |
| PRTPERSIST DEFAULT | NOPRTPERSIST DEFAULT | X | X | | | COMP |
| PRTSETFORMS | PRTSETFORMS | X | X | X | X | COMP |
| QUOTELENMAX | QUOTELENMAX | X | X | X | X | LR |
| REALMEMSIZE | 0 | X | X | | | COMP |
| REGISTER arg | none | X | X | | | COMP |
| REPLACE | REPLACE | X | X | X | X | LR |
| RESOURCESLOC arg | see SASV9.CFG | X | X | | | COMP |
| REUSE arg | NO | X | X | X | X | LR |
| RIGHTMARGIN arg | 0.000 IN | X | X | X | X | LR |
| RSASUSER | NORSASUSER | X | X | | | LR, COMP |
| RTRACE | NONE | X | X | | | COMP |
| RTRACELOC arg | none | X | X | X | X | COMP |
| S arg | 0 | X | X | X | X | LR, COMP |
| S2 arg | 0 | X | X | X | X | LR, COMP |
| SASAUTOS arg | SASAUTOS | X | X | X | X | COMP, MACRO |
| SASCMD | none | X | X | X | X | CONN |

| Options Specification | Default Value | SAS invocation | Configuration file | SAS System Options window | OPTIONS statement | See |
|---|---|---|---|---|---|---|
| SASCONTROL | SYSTEMMENU MINMAX | X | X | X | X | COMP |
| SASFRSCR | #LN00003 | | | | | CONN |
| SASHELP arg | see SASV9.CFG | X | X | | | LR, COMP |
| SASINITIALFOLDER | none | X | X | | | COMP |
| SASMSTORE arg | none | X | X | X | X | MACRO |
| SASSCRIPT arg | see SASV9.CFG | X | X | X | X | CONN |
| SASUSER arg | see SASV9.CFG | X | X | | | LR, COMP |
| SCROLLBARFLASH | NOSCROLLBAR FLASH | X | X | X | X | COMP |
| SEQ arg | 8 | X | X | X | X | LR |
| SERROR | SERROR | X | X | X | X | MACRO |
| SET arg | none | X | X | X | X | COMP |
| SETINIT | NOSETINIT | X | X | | | LR |
| SGIO | NOSGIO | X | X | | | COMP |
| SKIP arg | 0 | X | X | X | X | LR |
| SLEEPWINDOW | SLEEPWINDOW | X | X | | | COMP |
| SOLUTIONS | SOLUTIONS | X | X | | | LR |
| SORTANOM | none | X | X | X | X | COMP |
| SORTCUT | 0 | X | X | X | X | COMP |
| SORTCUTP | 0 | X | X | X | X | COMP |
| SORTDEV | the same location as -WORK | X | X | X | X | COMP |
| SORTDUP arg | PHYSICAL | X | X | X | X | LR |
| SORTEQUALS | SORTEQUALS | X | X | X | X | LR |
| SORTPARM | none | X | X | X | X | COMP |
| SORTPGM | BEST | X | X | X | X | COMP |
| SORTSEQ arg | none | X | X | X | X | NLS |
| SORTSIZE arg | 2,097,152 bytes | X | X | X | X | LR, COMP |
| SOURCE | SOURCE | X | X | X | X | LR |
| SOURCE2 | NOSOURCE2 | X | X | X | X | LR |
| SPDEINDEXSORTSIZE | 33554432 | X | X | X | X | SPDE |
| SPDEMAXTHREADS | 0 | X | X | | | SPDE |
| SPDESORTSIZE | 33554432 | X | X | X | X | SPDE |

| Options Specification | Default Value | SAS invocation | Configuration file | SAS System Options window | OPTIONS statement | See |
|---|---|---|---|---|---|---|
| SPDEUTILLOC | none | X | X | | | SPDE |
| SPDEWHEVAL | COST | X | X | | | SPDE |
| SPLASH | SPLASH | X | X | | | COMP |
| SPLASHLOC | none | X | X | | | COMP |
| SPOOL | NOSPOOL | X | X | X | X | LR |
| SSLCERTISS | none | X | X | X | X | COMP, CONN |
| SSLCERTSERIAL | none | X | X | X | X | COMP, CONN |
| SSLCERTSUBJ | none | X | X | X | X | COMP, CONN |
| SSLCLIENTAUTH | NOSSLCLIENTAUTH | X | X | X | X | COMP, CONN |
| SSLCRLCHECK | NOSSLCRLCHECK | X | X | X | X | COMP, CONN |
| STARTLIB | STARTLIB | X | X | | | LR |
| STIMEFMT | M | X | X | X | X | COMP |
| STIMER | STIMER | X | X | X | X | COMP |
| SUMSIZE arg | 0 | X | X | X | X | LR |
| SYMBOLGEN | NOSYMBOLGEN | X | X | X | X | MACRO |
| SYNCHIO | SYNCHIO | X | X | | | LR |
| SYNTAXCHECK | SYNTAXCHECK | X | X | X | X | LR |
| SYSGUIFONT arg | display setting | X | X | | | COMP |
| SYSIN arg | none | X | X | | | COMP |
| SYSPARM arg | none | X | X | X | X | COMP, MACRO |
| SYSPRINT arg | default system printer | X | X | X | X | COMP |
| SYSPRINTFONT arg | none | X | X | X | X | LR, COMP |
| SYSRPUTSYNC | NO | X | X | | | CONN |
| TBUFSIZE arg | 0 | X | X | X | X | CONN, METH |
| TCPPORTFIRST arg | 0 | X | X | X | X | CONN, METH |
| TCPPORTLAST arg | 0 | X | X | X | X | CONN, METH |
| TERMINAL | TERMINAL | X | X | | | LR |

| Options Specification | Default Value | SAS invocation | Configuration file | SAS System Options window | OPTIONS statement | See |
|---|---|---|---|---|---|---|
| TERMSTMT | none | X | X | | | LR |
| TEXTURELOC | none | X | X | X | X | LR |
| THREADS | THREADS | X | X | X | X | LR |
| TOOLDEF arg | TOP RIGHT | X | X | | | COMP |
| TOOLSMENU | TOOLSMENU | X | X | | | LR |
| TOPMARGIN arg | 0.000 IN | X | X | X | X | LR |
| TRAINLOC | none | X | X | | | LR |
| TRANTAB arg | none | X | X | X | X | NLS |
| UNIVERSALPRINT | NOUNIVERSAL PRINT | X | X | | | LR |
| UPRINTMENU SWITCH | NOUPRINTMENU SWITCH | X | X | | | COMP |
| USER arg | none | X | X | X | X | LR, COMP |
| USERICON arg | none | X | X | | | COMP |
| UTILLOC arg | none | X | X | | | LR |
| UUIDCOUNT arg | 100 | X | X | X | X | LR |
| UUIDGENDHOST arg | none | X | X | | | LR |
| V6CREATEUPDATE arg | ERROR | X | X | | | LR |
| VALIDFMTNAME arg | LONG | X | X | X | X | LR |
| VALIDVARNAME arg | V7 | X | X | X | X | LR |
| VERBOSE | NOVERBOSE | X | X | | | COMP |
| VIEWMENU | VIEWMENU | X | X | | | LR |
| VNFERR | VNFERR | X | X | X | X | LR |
| WEBUI | NOWEBUI | X | X | | | COMP |
| WINDOWSMENU | NOWINDOWSMENU | X | X | X | X | COMP |
| WORK arg | !TEMP\SAS Temporary Files | X | X | | | LR, COMP |
| WORKINIT | WORKINIT | X | X | | | LR |
| WORKTERM | WORKTERM | X | X | X | X | LR |
| XCMD | XCMD | X | X | | | COMP |
| XMIN | NOXMIN | X | X | X | X | COMP |
| XSYNC | XSYNC | X | X | X | X | COMP |

| Options Specification | Default Value | SAS invocation | Configuration file | SAS System Options window | OPTIONS statement | See |
|---|---|---|---|---|---|---|
| XWAIT | XWAIT | X | X | X | X | COMP |
| YEARCUTOFF arg | 1920 | X | X | X | X | LR |

## Options Not Applicable to the Windows Environment

The following SAS system options, which may described in the system options portion of *SAS Language Reference: Dictionary*, are not applicable to the Windows operating environment:

- DMS
- DMSEXP
- DOCLOC
- FSDEVICE
- OPLIST
- SORTNAME
- TAPECLOSE.

## ACCESSIBILITY System Option

**Enables the accessibility features on the Customize Tools dialog box**

**Default:** STANDARD

**Valid in:** configuration file, SAS invocation

**Category:** Input control: Data processing

**PROC OPTIONS GROUP=** INPUTCONTROL

**Windows specifics:** all

### Syntax

-ACCESSIBILITY STANDARD | EXTENDED

**STANDARD**
specifies that the standard Customize Tools dialog box and Properties dialog boxes are enabled.

**EXTENDED**
specifies that the accessibility features are enabled in the Customize Tools dialog box and for some Properties dialog boxes.

### Details

When the ACCESSIBILITY option is set to EXTENDED, the Customize Tools Custom tabbed page and some SAS Properties dialog boxes are modified for accessibility.

The Customize tab contains two additional buttons, `File Menu` and `Edit Menu`. These menu buttons enable accessibility to the commands that are available using the toolbar buttons.

The tabs in these dialog boxes are buttons in order to enable some of the SAS Properties dialog boxes for accessibility. Using the Ctrl + Page Up and Ctrl + Page Down keys, you can access all parts of these Properties dialog boxes.

When this system option is set to EXTENDED, you can toggle between the overstrike cursor and the insert cursor. The insert cursor is the default since some accessibility utilities expect the insert cursor.

### See Also

- "Accessibility Features in SAS under Windows" on page 74

## ALTLOG System Option

**Specifies an alternate SAS log**

**Default:** NOALTLOG

**Valid in:** configuration file, SAS invocation

**Category:** Environment control: Files

**PROC OPTIONS GROUP=** ENVFILES

**Windows specifics:** *destination* must resolve to a valid Windows path or filename

### Syntax

-ALTLOG *destination*

-NOALTLOG

**ALTLOG *destination***
specifies the destination for a copy of the SAS log. The *destination* argument can be a valid Windows pathname or filename (including device names) or an environment variable associated with a pathname. If you specify only a pathname, the copy is placed in a file in the specified directory, with a name of *filename*.LOG, where *filename* is the name of your SAS job. If you are running SAS interactively and specify only a pathname, the log is written to a file named SAS.LOG within that path.

**NOALTLOG**
suppresses the creation of a copy of the SAS log.

### Details

The ALTLOG system option specifies a destination to which a copy of the SAS log is written. Use the ALTLOG system option to capture log output for printing.

To send the SAS log to a printer other than the default printer, use a valid Windows printer name for the *destination* value.

*Note:* ALTLOG replaces the following system options from earlier versions of SAS: LDISK, LPRINT, and LTYPE. △

### See Also

- □ “Routing Procedure Output and the SAS Log to a File” on page 182

## ALTPRINT System Option

**Specifies an alternate SAS procedure output file**

**Default:** NOALTPRINT

**Valid in:** configuration file, SAS invocation

**Category:** Environment control: Files

**PROC OPTIONS GROUP=** ENVFILES

**Windows specifics:** *destination* must resolve to a valid Windows pathname or filename

### Syntax

-ALTPRINT *destination*

-NOALTPRINT

**ALTPRINT *destination***
specifies the destination for a copy of the SAS procedure output file. The *destination* argument can be a valid Windows pathname or filename (including device names) or an environment variable associated with a pathname. If you specify only a pathname, the copy is placed in a file in the specified directory, with a name of *filename*.LST, where *filename* is the name of your SAS job. If you are running SAS interactively and specify only a pathname, the filename is SAS.

**NOALTPRINT**
does not create a copy of the SAS procedure output file.

### Details

The ALTPRINT system option specifies a destination to which a copy of the SAS procedure output file is written. Use the ALTPRINT system option to capture procedure output for printing.

To send the procedure output to a printer other than the default printer, use a valid Windows printer name for the *destination* value.

*Note:* ALTPRINT replaces the following system options form earlier versions of SAS: PDISK, PPRINT, and PTYPE. △

### See Also

- □ “Routing Procedure Output and the SAS Log to a File” on page 182
- □ “Printing” on page 166

## AUTHSERVER System Option

**Specifies the authentication domain server to search for secure server logins**

**Default:** local and trusted servers

**Valid in:** configuration file, SAS invocation, OPTIONS statement, SAS System Options window

**Category:** Environment control: Initialization and operation

**PROC OPTIONS GROUP=** EXECMODES

**Windows specifics:** all

## Syntax

-AUTHSERVER <" " | ’*domain-name*’ | ’.’>

AUTHSERVER <" " | ’*domain-name*’ | ’.’>

**" "**
specifies to search the local server first, and then search trusted servers for a valid user login.

***’domain-name’***
specifies a specific domain-name to search for a valid user login. Single quotation-marks are required.

**’.’**
specifies to search only the local server for a valid user login. Single quotation-marks are required.

## Details

The AUTHSERVER system option specifies which servers to search to validate user logins.

## Comparisons

You use the AUTHSERVER system option to specify a single authentication domain. You use the AUTHPROVIDERDOMAIN system option to specify multiple authentication providers and the associated domains.

## See Also

- “AUTHPROVIDERDOMAIN System Option” in *SAS Language Reference: Dictionary*

## AUTOEXEC System Option

**Specifies an alternate SAS autoexec file**

**Default:** AUTOEXEC.SAS, if the file is available; otherwise, none

**Valid in:** configuration file, SAS invocation

**Category:** Environment control: Files

**PROC OPTIONS GROUP=** ENVFILES

**Windows specifics:** *file-specification* must be a valid Windows filename

### Syntax

-AUTOEXEC *file-specification*

-NOAUTOEXEC

**AUTOEXEC *file-specification***
specifies the SAS autoexec file to be used instead of the default AUTOEXEC.SAS file. The *file-specification* argument can be a valid Windows filename or an environment variable associated with a pathname. For more information on the SAS autoexec file, see “SAS Autoexec File” on page 18.

**NOAUTOEXEC**
indicates that no SAS autoexec file is processed, even if one exists.

### Details

The AUTOEXEC system option specifies the autoexec file. The autoexec file contains SAS statements that are executed automatically when you invoke SAS or when you start another SAS process. The autoexec file can contain any valid SAS statements. For example, you can include LIBNAME statements for SAS data libraries you access routinely in SAS sessions.

If no AUTOEXEC.SAS file is found, the default value for this option is NOAUTOEXEC.

### See Also

- “SAS Autoexec File” on page 18

## AWSCONTROL System Option

**Specifies whether the main SAS window includes a title bar, a system/control menu, and minimize/maximize buttons**

**Default:** TITLE SYSTEMMENU MINMAX

**Valid in:** configuration file, SAS invocation, OPTIONS statement, SAS System Options window

**Category:** Environment control: Display

**PROC OPTIONS GROUP=** ENVDISPLAY

**Windows specifics:** all

### Syntax

-AWSCONTROL <TITLE | NOTITLE><SYSTEMMENU | NOSYSTEMMENU ><MINMAX | NOMINMAX>

AWSCONTROL= <TITLE | NOTITLE><SYSTEMMENU | NOSYSTEMMENU><MINMAX | NOMINMAX>

**AWSCONTROL**
specifies to display the title bar, the system menu, and the minimize and maximize buttons on the main SAS window.

**TITLE | NOTITLE**
specifies whether or not to display the title bar on the main SAS window. If NOTITLE is specified, the system menu and the minimize and maximize buttons are automatically omitted as well.

**SYSTEMMENU | NOSYSTEMMENU**
specifies whether or not to display the system menu on the title bar of the main SAS window. If NOSYSTEMMENU is specified, the minimize and maximize buttons are also omitted.

**MINMAX | NOMINMAX**
specifies whether or not to display the minimize and maximize buttons on the title bar of the main SAS window.

### Details

The AWSCONTROL system option controls only the main SAS window, not the windows that are contained inside the main SAS window. The SASCONTROL system option controls those SAS process windows.

This system option is intended for use by SAS/AF programmers to customize the interface of their applications.

### See Also

- □ "SASCONTROL System Option" on page 545

## AWSDEF System Option

**Specifies the location and dimensions of the main SAS window when SAS initializes**

**Default:** 80% of the display height and width

**Valid in:** configuration file, SAS invocation, OPTIONS statement, SAS System Options window

**Category:** Environment control: Display

**PROC OPTIONS GROUP=** ENVDISPLAY

**Windows specifics:** all

### Syntax

-AWSDEF *row-percent-position column-percent-position height-percent width-percent*

AWSDEF=*row-percent-position column-percent-position height-percent width-percent*

***row-percent-position*** **and** ***column-percent-position***
specify screen percentages that control the position of the upper-left corner of the main SAS window. For example, if you specify 50 for each of these, the upper-left corner of the SAS window is positioned in the center of your display.

The valid range of values for these parameters is 0 through 95.

***height-percent* and *width-percent***
specify screen percentages that control the size of the main SAS window. For example, if you specify 100 for each of these, the SAS window occupies your entire display. If you specify 50 for each of these, the SAS window occupies half of your display.
The valid range of values for these parameters is 40 through 100.

### Details

The AWSDEF system option specifies the location and dimensions of the main SAS window when SAS initializes. For an example of how to use the AWSDEF system option, see "Changing the Size and Placement of the Main SAS Window" on page 64.

## AWSMENU System Option

**Specifies whether to display the menu bar in the main SAS window**

**Default:** AWSMENU

**Valid in:** configuration file, SAS invocation, OPTIONS statement, SAS System Options window

**Category:** Environment control: Display

**PROC OPTIONS GROUP=** ENVDISPLAY

**Windows specifics:** all

### Syntax

-AWSMENU | -NOAWSMENU

AWSMENU | NOAWSMENU

**AWSMENU**
specifies to display the menu bar in the main SAS window.

**NOAWSMENU**
specifies to omit the menu bar in the main SAS window.

### Details

The AWSMENU system option is intended for use by SAS/AF programmers to customize the interface of their applications.

## AWSMENUMERGE System Option

**Specifies whether to embed menu items that are specific to Windows in the main menus**

**Default:** AWSMENUMERGE

**Valid in:** configuration file, SAS invocation, OPTIONS statement, SAS System Options window

**Category:** Environment control: Display

**PROC OPTIONS GROUP=** ENVDISPLAY

**Windows specifics:** all

### Syntax

-AWSMENUMERGE | -NOAWSMENUMERGE

AWSMENUMERGE | NOAWSMENUMERGE

**AWSMENUMERGE**
specifies to embed the menu items that are specific to Windows.

**NOAWSMENUMERGE**
specifies to not embed the menu items that are specific to Windows.

### Details

The AWSMENUMERGE system option determines whether the menu items that are specific to the Windows operating environment are included in the main SAS window menus.

This system option is used by SAS/AF programmers to customize the interface of their applications. If SAS is started in batch mode, SAS sets this system option to NOAWSMENUMERGE.

### See Also

- "WINDOWSMENU System Option" on page 573

## AWSTITLE System Option

**Replaces the default text in the main SAS title bar**

**Default:** none

**Valid in:** configuration file, SAS invocation

**Category:** Environment control: Display

**PROC OPTIONS GROUP=** ENVDISPLAY

**Windows specifics:** all

### Syntax

-AWSTITLE "*title-text*"

***"title-text"***

specifies the text that appears in the title bar of the main SAS window. The text must be enclosed in either single or double quotation marks.

### Details

The AWSTITLE system option allows you to replace the default text in the title bar of the main SAS window with the title that you specify.

This system option is intended for use by SAS/AF programmers to customize the interface of their applications.

## BUFNO System Option

**Specifies the number of buffers to be allocated for processing SAS data sets**

**Default:** 1

**Valid in:** configuration file, SAS invocation, OPTIONS statement, SAS System Options window

**Category:** Files: SAS Files

**PROC OPTIONS GROUP=** SASFILES, PERFORMANCE

**Windows specifics:** Default value

**See:** BUFNO System Option in *SAS Language Reference: Dictionary*

### Syntax

-BUFNO *n* | *n*K | *n*M | *n*G | *hex*X | MIN | MAX

BUFNO= *n* | *n*K | *n*M | *n*G | *hex*X | MIN | MAX

***n* | *n*K | *n*M | *n*G**
specifies the number of buffers in multiples of 1 (bytes); 1,024 (kilobytes); 1,048,576 (megabytes); or 1,073,741,824 (gigabytes). You can specify decimal values for the number of kilobytes, megabytes, or gigabytes. For example, a value of `8` specifies 8 buffers, a value of `.782k` specifies 801 buffers, and a value of `3m` specifies 3,145,728 buffers.

For values greater than 1G, use the *n*M option or specify MAX.

***hex*X**
specifies the number of buffers as a hexadecimal value. You must specify the value beginning with a number (0–9), followed by an X. For example, the value `2dx` specifies 45 buffers.

**MIN**
sets the number of buffers to 0, and requires SAS to use the default value of 1.

**MAX**
sets the number of buffers to 2,147,483,647.

### Details

The number of buffers is not a permanent attribute of the data set; it is valid only for the current SAS session or job.

BUFNO= applies to SAS data sets that are opened for input, output, or update.

Using BUFNO= can improve execution time by limiting the number of input/output operations that are required for a particular SAS data set. The improvement in execution time, however, comes at the expense of increased memory consumption.

Under Windows, the maximum number of buffers that you can allocate is determined by the amount of memory available. To request that SAS allocate the number of buffers based on the number of pages for the data set, use the SASFILE statement.

### See Also

- “BUFSIZE System Option” on page 491
- “SASFILE Statement” in *SAS Language Reference: Dictionary*.
- The chapter on optimizing system performance in *SAS Language Reference: Concepts*.

## BUFSIZE System Option

**Specifies the permanent buffer page size for output SAS data sets**

**Default:** 0

**Valid in:** configuration file, SAS invocation, OPTIONS statement, SAS System Options window

**Category:** Files: SAS Files

**PROC OPTIONS GROUP=** SASFILES, PERFORMANCE

**Windows specifics:** Valid values for *n*

**See:** BUFSIZE System Option in *SAS Language Reference: Dictionary*

### Syntax

-BUFSIZE *n* | *n*K | *n*M | *n*G | *hex*X | MIN | MAX

BUFSIZE=*n* | *n*K | *n*M | *n*G | *hex*X | MIN | MAX

***n* | *n*K | *n*M | *n*G**

specifies the buffer page size in multiples of 1; 1,024 (kilobytes); 1,048,576 (megabytes), and 1,073,741,824 (gigabytes), respectively. You can specify decimal values for the number of kilobytes, megabytes, or gigabytes. For example, a value of **`8`** specifies 8 bytes, a value of **`.782k`** specifies 801 bytes, and a value of **`3m`** specifies 3,145,728 bytes.

***hex*X**

specifies the buffer page size as a hexadecimal value. You must specify the value beginning with a number (0–9), followed by an X. For example, the value **`2dx`** sets the buffer page size to 45 bytes.

**MIN**

sets the buffer page size to -2,147,483,648 and requires SAS to use a default value. Under Windows, the default value is 0. The minimum number is -2,147,483,648.

**MAX**

sets the buffer page size to 2,147,483,647 bytes.

### Details

The BUFSIZE system option enables you to specify the permanent buffer page size for output SAS data sets. Under Windows, the value can range from 512 bytes to 2,147,483,647 bytes. Using the default value of 0 optimizes the buffer page size by enabling the engine to pick a value depending on the size of the observation.

Experienced users may want to vary the value of the BUFSIZE system option if you are trying to maximize memory usage or the number of observations per page.

### See Also

- “BUFNO System Option” on page 490
- The chapter about optimizing system performance in *SAS Language Reference: Concepts*.

## CATCACHE System Option

**Specifies the number of SAS catalogs to keep open**

**Default:** 0

**Valid in:** configuration file, SAS invocation

**Category:** Files: SAS Files

**PROC OPTIONS GROUP=** SASFILES

**Windows specifics:** Valid values for *n*

**See:** CATCACHE System Option in *SAS Language Reference: Dictionary*

### Syntax

-CATCACHE *n* | *n*K | MIN | MAX

***n* | *n*K**
specifies the number of open-file descriptors to keep in cache memory in multiples of 1 (*n*) or 1,024 (*n*K). You can specify decimal values for the number of kilobytes. For example, a value of `8` specifies 8 open-file descriptors, a value of `.782k` specifies 801 open-file descriptors, and a value of `3k` specifies 3,072 open-file descriptors.

If *n* > 0, SAS places up to that number of open-file descriptors in cache memory instead of closing the catalogs.

**MIN**
sets the number of open-file descriptors that are kept in cache memory to 0.

**MAX**
sets the number of open-file descriptors that are kept in cache memory to 32,767.

### Details

By using the CATCACHE system option to specify the number of SAS catalogs to keep open, you can avoid the repeated opening and closing of the same catalogs.

If SAS is running on a z/OS server and the MINSTG system option is in effect, SAS sets the value of CATCACHE to 0.

### See Also

- ☐ The chapter about optimizing system performance in *SAS Language Reference: Concepts*.

## CLEANUP System Option

**Specifies how to handle an out-of-resource condition**

**Default:** CLEANUP

**Valid in:** configuration fie, SAS invocation, OPTIONS statement, SAS System Options window

**Category:** Environment control: Error handling

**PROC OPTIONS GROUP=** ERRORHANDLING

**Windows specifics:** behavior when running in batch mode

**See:** CLEANUP System Option in *SAS Language Reference: Dictionary*

### Syntax

-CLEANUP | -NOCLEANUP

CLEANUP | NOCLEANUP

**CLEANUP**
specifies that during the entire session, SAS attempts to perform automatic, continuous clean-up of resources that are not essential for execution. Nonessential resources include those that are not visible to the user (for example, cache memory) and those that are visible to the user (for example, the KEYS windows).

CLEANUP does not prompt you for any out-of-resource condition except for out-of-disk-space conditions. If you do not want to be prompted for out-of-disk-space conditions, use the CLEANUP option in conjunction with the NOTERMINAL option.

**NOCLEANUP**
specifies that SAS allow the user to choose how to handle an out-of-resource condition. When NOCLEANUP is in effect and SAS cannot execute because of a lack of resources, SAS automatically attempts to clean up resources that are not visible to the user (for example, cache memory). However, resources that are visible to the user (for example, the KEYS windows) are not automatically cleaned up. Instead, SAS prompts you before attempting to regain resources.

### Details

The CLEANUP system option indicates whether you are prompted with a menu of items to clean up when SAS encounters an out-of-resource condition.

If you specify NOCLEANUP and are prompted for input, you can select **`Continuous`** on every menu except the out-of-disk-space menu. If you choose **`Continuous`**, the CLEANUP option is turned on and you are not prompted again in out-of-resource conditions, unless SAS runs out of disk space.

## COMDEF System Option

**Specifies the location where the SAS Command window is displayed**

**Default:** BOTTOM CENTER

**Valid in:** configuration file, SAS invocation

**Category:** Environment control: Display

**PROC OPTIONS GROUP=** ENVDISPLAY

**Windows specifics:** all

### Syntax

-COMDEF TOP | CENTER | BOTTOM
<LEFT | CENTER | RIGHT>

**TOP | CENTER | BOTTOM**
specifies the vertical position of the SAS Command window. The default value is BOTTOM.

**LEFT | CENTER | RIGHT**
specifies the horizontal position of the SAS Command window. The default value is CENTER.

### Details

You must specify a vertical position first. You do not have to specify a horizontal position, but if you omit it, CENTER is used.

*Note:* The SAS Command window is positioned with respect to your entire display, not to the main SAS window. Also, the COMDEF system option applies only when the command bar is not docked to the main SAS window. △

### See Also

- "Setting Session Preferences" on page 57
- "Using the Command Bar to Issue Commands" on page 39

## CONFIG System Option

**Specifies an alternative SAS configuration file**

**Default:** !*sasroot*\SASV9.CFG

**Valid in:** configuration file, SAS invocation

**Category:** System administration: Installation

**PROC OPTIONS GROUP=** INSTALL

**Windows specifics:** all

## Syntax

-CONFIG *file-specification*

***file-specification***
specifies the filename of the SAS configuration file that you want to use, or a Windows environment variable that resolves to a valid filename. The *file-specification* must be a valid Windows filename. If *file-specification* contains spaces, it must be enclosed in quotation marks.

## Details

The CONFIG system option specifies the complete filename of your configuration file. This file contains SAS options that are executed automatically whenever SAS is invoked. SAS supplies a default configuration file, but you can create your own configuration file and store it in a location you choose.

## See Also

- "SAS Configuration Files" on page 13

## DBCS System Option

**Determines whether to process text as encoded in a double-byte character set or as a single-byte character set encoding method**

**Default:** NODBCS

**Valid in:** configuration file, SAS invocation

**Category:** Environment control: Language control

**PROC OPTIONS GROUP=** LANGUAGECONTROL

**Windows specifics:** all

**See:** DBCS System Option in *SAS National Language Support (NLS): User's Guide*

## DBCSLANG System Option

**Specifies a double-byte character set (DBCS) language**

**Default:** NONE

**Valid in:** configuration file, SAS invocation

**Category:** Environment control: Language control

**PROC OPTIONS GROUP=** LANGUAGECONTROL

**Windows specifics:** valid values

**See:** DBCSLANG System Option in *SAS National Language Support (NLS): User's Guide*

## DBCSTYPE System Option

**Specifies a double-byte character set (DBCS) encoding method**

**Default** PCIBM

**Valid in:** configuration file, SAS invocation

**Category:** Environment control: Language control

**PROC OPTIONS GROUP=** LANGUAGECONTROL

**Windows specifics:** Valid values

**See:** DBCSTYPE System Option in *SAS National Language Support (NLS): User's Guide*

## DEVICE System Option

**Specifies a device driver for graphics output for SAS/GRAPH software**

**Default:** none

**Valid in:** configuration file, SAS invocation, OPTIONS statement, SAS System Options window

**Alias:** -DEV

**Category:** Graphics: Driver settings

**PROC OPTIONS GROUP=** GRAPHICS

**Windows specifics:** Valid values for *device-driver-name*; default value

**See:** DEVICE System Option in *SAS Language Reference: Dictionary*

### Syntax

-DEVICE *device-driver-name*

DEVICE=*device-driver-name*

***device-driver-name***
specifies the name of a device driver for graphics output.

### Details

To see the list of device drivers that are available under Windows, you can use the GDEVICE procedure. If you are using the SAS windowing environment, submit the following statements:

```
proc gdevice catalog=sashelp.devices;
run;
quit;
```

If you want to write the device list to the SAS log, submit the following statements:

```
proc gdevice catalog=sashelp.devices nofs;
    list _all_;
run;
quit;
```

Your site might have defined additional device catalogs referenced by the GDEVICE0 libref. See your SAS Support Consultant for more information.

### See Also

- "GDEVICE Procedure" in *SAS/GRAPH Reference, Volumes 1 and 2*

---

## ECHO System Option

**Specifies a message to be echoed to the SAS log while initializing SAS**

**Default:** NOECHO

**Valid in:** configuration file, SAS invocation

**Category:** Log and procedure output control: SAS log

**PROC OPTIONS GROUP=** LOGCONTROL

**Windows specifics:** all

### Syntax

-ECHO "*message*" | -NOECHO

**ECHO "*message*"**
specifies the text of the message to be echoed to the SAS log. The text must be enclosed in single or double quotation marks if the message is more than one word. Otherwise, quotation marks are not needed.

**NOECHO**
specifies that no messages are to be echoed to the SAS log.

### Details

Messages that result from errors in the autoexec file are printed in the SAS log regardless of how the ECHO system option is set.

### Example

For example, you can specify the following:

```
-echo "SAS System under Windows
       is initializing."
```

The message appears in the LOG window as SAS initializes.

## See Also

- □ "ECHOAUTO System Option" in *SAS Language Reference: Dictionary*

## EMAILDLG System Option

**Specifies whether to use the native e-mail dialog provided by your e-mail application or the e-mail dialog provided by SAS**

**Default:** NATIVE

**Valid in:** configuration file, SAS invocation

**Category:** Environment control: Email

**PROC OPTIONS GROUP=** EMAIL

**Windows specifics:** all

### Syntax

-EMAILDLG NATIVE | SAS

**NATIVE**
specifies to use the e-mail dialog box provided by your e-mail system vendor. You can use the native dialog box with SAS only if the e-mail system supports the MAPI interface.

**SAS**
specifies to use the e-mail dialog box provided by SAS.

### Details

The EMAILDLG system option specifies whether to use the native e-mail interactive dialog box provided by your e-mail application or the e-mail interface provided by SAS. SAS uses the native dialog box by default.

### See Also

- □ "Sending E-Mail Using SAS" on page 40

## EMAILSYS System Option

**Specifies which e-mail interface to use**

**Default:** MAPI

**Valid in:** configuration file, SAS invocation

**Category:** Environment control: Email

**PROC OPTIONS GROUP=** EMAIL

**Windows specifics:** all

## Syntax

-EMAILSYS MAPI | VIM | SMTP

**MAPI**
specifies to use the Messaging Application Program Interface (MAPI) electronic mail interface. This is the default value.

**VIM**
specifies to use the Vendor Independent Mail (VIM) electronic mail interface.

**SMTP**
specifies to use the Simple Mail Transfer Protocol email interface.

## Details

SAS supports three types of electronic mail interfaces: MAPI (such as Microsoft Exchange), Vendor Independent Mail (VIM—such as Lotus, cc:Mail) and SMTP. The default value is MAPI. If you specify SMTP, you must also specify and configure the EMAILHOST and EMAILPORT system options. SMTP is available only when you are sending e-mail programatically. SMTP is not available using either your e-mail program native dialog box or the SAS e-mail dialog box.

## See Also

- □ "Sending E-Mail Using SAS" on page 40
- □ From *SAS Language Reference: Dictionary*
  - □ "EMAILID System Option"
  - □ "EMAILPW System Option"
  - □ "EMAILAUTHPROTOCOL System Option"
- □ "The SMTP E-mail Interface" in *SAS Language Reference: Concepts*

## ENCODING System Option

**Specifies the default character-set encoding for the SAS session**

**Default:** wlatin1 if DBCS is not active; none if DBCS is active
**Valid in:** configuration file, SAS invocation
**Category:** Environment control: Language control
**PROC OPTIONS GROUP=** LANGUAGECONTROL
**Windows specifics:** Valid values
**See:** ENCODING= System Option in *SAS National Language Support (NLS): User's Guide*

## ENGINE System Option

**Specifies the default access method to use for SAS libraries**

**Default** V9

**Valid in:** configuration file, SAS invocation

**Category:** Files: SAS Files

**PROC OPTIONS GROUP=** SASFILES

**Window specifics:** valid values

## Syntax

-ENGINE *engine-name*

***engine-name***
can be one of the following under Windows:

BASE | V9
specifies the default SAS engine for SAS System 9 and SAS 9.1 files.

BMDP
specifies the engine for BMDP data files.

OSIRIS
specifies the engine for OSIRIS data files.

SPSS
specifies the engine for SPSS data files.

V8
specifies the SAS engine all Version 8 files.

V7
specifies the SAS engine for all Version 7 files.

V6
specifies the default engine for Releases 6.08 - 6.12. The V6 engine is supported only in 32–bit operating environments.

V604
specifies the default engine for Release 6.04 and Release 6.03.

XML
specifies the default engine for XML files.

XPORT
specifies the transport engine.

## Details

The default engine is used when a SAS data library points to an empty directory or a new file. For information about SAS/SHARE and SAS/ACCESS engines, see their respective documentation.

## See Also

- "Types of Library Engines" on page 123
- *SAS Language Reference: Concepts*
- *SAS/ACCESS for Relational Databases: Reference*

- *Communications Access Methods for SAS/CONNECT and SAS/SHARE*

## ENHANCEDEDITOR System Option

**Specifies whether to enable the Enhanced Editor during SAS invocation**

**Default:** ENHANCEDEDITOR

**Valid in:** configuration file, SAS invocation

**Category:** Environment control: Display

**PROC OPTIONS GROUP=** ENVDISPLAY

**Windows specifics:** all

### Syntax

-ENHANCEDEDITOR | -NOENHANCEDEDITOR

**ENHANCEDEDITOR**
specifies to enable the Enhanced Editor during SAS invocation.

**NOENHANCEDEDITOR**
specifies not to enable the Enhanced Editor during SAS invocation.

### Details

By default, the Enhanced Editor is enabled when you start SAS. If you do not want the Enhanced Editor enabled when you start SAS, use the NOENHANCEDEDITOR system option.

### See Also

- "WEDIT Command" on page 360

## FILTERLIST System Option

**Specifies an alternative set of file filter specifications to use for the Open and Save As dialog boxes**

**Default:** none

**Valid in:** configuration file, SAS invocation

**Category:** Environment control: Display

**PROC OPTIONS GROUP=** ENVDISPLAY

**Windows specifics:** all

### Syntax

-FILTERLIST "*filter1* | *filter2*|... | *filter-n*"

***filter1...filter n***
specifies one or more strings of text separated by a "|" and enclosed in double quotation marks, such as "*.Bob's work | SAS*.*" Note that you can specify long filename extensions that include spaces and single quotation marks.

### Details

All filters in the FILTERLIST are added to the application specified filter list displayed in the **`Files of type`** box in the Open dialog box and in the **`Save as type`** box in the Save As dialog box. The first filter in the FILTERLIST becomes the default filter. The FILTERLIST must be enclosed in double quotation marks.

### See Also

- "DLGOPEN Command" on page 334
- "DLGSAVE Command" on page 340

## FONT System Option

**Specifies a font to use for SAS windows**

**Default:** Sasfont 8

**Valid in:** configuration file, SAS invocation, OPTIONS statement, SAS System Options window

**Category:** Environment control: Display

**PROC OPTIONS GROUP=** ENVDISPLAY

**Windows specifics:** all

### Syntax

-FONT "*font-name*" <BOLD | NORMAL><REGULAR | ITALIC><*font-size*><*character-set*>

FONT="*font-name*" <BOLD | NORMAL><REGULAR | ITALIC> <*font-size*><*character-set*>

***"font-name"***
specifies the name of the font for text in the SAS windowing environment. This must be a valid font name (for example, "SAS Monospace" or "Courier"). The *font-name* argument must be enclosed in double quotation marks. This is a required argument.

**BOLD | NORMAL**
specifies the weight of the font. The default is NORMAL.

**REGULAR | ITALIC**
specifies the style of the font. The default is REGULAR.

***font-size***
specifies the font size to use for printing. This must be an integer from 1 to 7200, inclusive. If you omit this argument, SAS uses the last selected size unless there is no previous size, in which case 8 is used.

***character-set***
specifies the character set to use. The default is “Western”. Some possible valid values are Western, Central European, Cyrillic, Greek, Turkish, Arabic, Baltic, and Thai. If the font does not support the specified character set, the default character set is used. If the default character set is not supported by the font, the font’s default character set is used.

### Details

Valid font names are shown in the Fonts folder. To open the Font folder, type **`font`** in the Run dialog box. For example, you can use the following option with the SAS command:

```
-font "sas monospace bold" 12
```

SAS displays output best with a monospace (fixed-pitch) font. If you use a proportional (variable pitch) font, text may display incorrectly. If you specify a *point-size* that is not valid for a font, SAS uses the closest point size for the font you specify.

### See Also

- “SYSGUIFONT System Option” on page 564
- “SYSPRINTFONT System Option” on page 567

## FONTALIAS System Option

**Assigns a Windows font to one of the SAS fonts**

**Default:** varies (see table in “Details” on page 503)

**Valid in:** configuration file, SAS invocation

**Category:** Graphics: Driver settings

**PROC OPTIONS GROUP=** GRAPHICS

**Windows specifics:** all

### Syntax

-FONTALIAS “*SAS-font*” “*host-specific-font*”

**“*SAS-font*”**
specifies the SAS font you want to replace. The name of the font must be enclosed in double quotation marks.

**“*host-specific-font*”**
specifies the Windows font that you want to assign. The name of the font must be enclosed in double quotation marks.

### Details

Use the FONTALIAS system option for each font that you want to override.
The default font aliases for Windows are as follows:

| SAS font | Windows font |
|---|---|
| Times | Times New Roman |
| Helvetica | Arial |
| Courier | Courier New |
| Symbol | Symbol |
| Script | Script |
| AvantGarde | Arial |
| Bookman | Times New Roman |
| Schoolbook | Times New Roman |
| Palatino | Times New Roman |
| Dingbats | Symbol |

## Example

The system option `-fontalias "Times" "Courier New"` tells SAS to use Courier New wherever the Times SAS font is requested.

## FONTSLOC System Option

**Specifies the directory location of the files that contain the SAS fonts that are loaded during the SAS session**

**Default:** !*sasroot*\core\resource

**Valid in:** configuration file, SAS invocation

**Category:** Environment control: Display

**PROC OPTIONS GROUP=** ENVDISPLAY

**Windows specifics:** all

**See:** FONTSLOC System Option in *SAS Language Reference: Dictionary*

### Syntax

-FONTSLOC *directory-specification*

***directory-specification***
specifies the directory that contains the SAS fonts that are loaded during the SAS session. If *directory-specification* contains spaces, it must be enclosed in quotation marks.

### Details

The directory must be a valid Windows pathname.

## FORMCHAR System Option

**Specifies the default output formatting characters**

**Default:** (see the SAS configuration file)

**Valid in:** configuration file, SAS invocation, OPTIONS statement, SAS System Options window

**Category:** Log and procedure output control: Procedure output

**PROC OPTIONS GROUP=** LISTCONTROL

**Windows specifics:** Valid values for *formatting-characters*

**See:** FORMCHAR System Option in *SAS Language Reference: Dictionary*

### Syntax

-FORMCHAR "*formatting-characters*"

FORMCHAR="*formatting-characters*"

***formatting-characters***
specifies any string or list of strings of characters up to 64 bytes long. If fewer than 64 bytes are specified, the string is padded with blanks on the right. The character string must be enclosed in double quotation marks.

### Details

Formatting characters are used to construct tabular output outlines and dividers for various procedures, such as the CALENDAR, FREQ, and TABULATE procedures. If you omit formatting characters as an option in the procedure, the default specifications given in the FORMCHAR= system option are used. Note that you can also specify a hexadecimal character constant as a formatting character. When you use a hex constant with this option, SAS interprets the value of the hex constant as appropriate for the Windows environment.

The configuration file shipped with SAS contains two FORMCHAR system option specifications, with one of them commented out. The default FORMCHAR uses the characters in the SAS Monospace and Sasfont fonts. If you use a code page other than the standard code pages, comment out the FORMCHAR system option that shipped with SAS and use the other FORMCHAR system option.

## FULLSTIMER System Option

**Generates memory usage and image usage statistics for each task SAS completes**

**Default:** NOFULLSTIMER

**Valid in:** configuration file, SAS invocation, OPTIONS statement, SAS System Options window

**Category:** Log and procedure output control: SAS log

**PROC OPTIONS GROUP=** LOGCONTROL

**Windows specifics:** all

## Syntax

-FULLSTIMER | -NOFULLSTIMER

FULLSTIMER | NOFULLSTIMER

**FULLSTIMER**
specifies that SAS write to the SAS log a complete list of computer resources that were used for each step and for the entire SAS session.

**NOFULLSTIMER**
specifies that SAS not write a complete list of computer resources to the SAS log. This is the default.

## Details

The FULLSTIMER system option specifies whether all the performance statistics of your computer system that are available to SAS are written to the SAS log. Data about I/O, memory, and CPU time is available.

This system option gives you time-elapsed statistics if you have not turned off the STIMER option. If you turn off the STIMER option, the FULLSTIMER option does not generate time statistics.

If you need statistics on tasks such as the SAS windowing environment (statistics for the windowing environment are available only when SAS terminates), you should use the "ALTLOG System Option" on page 483 to specify the destination for a copy of the SAS log. If you specify the FULLSTIMER system option before you end your SAS session, you can view statistics for the SAS windowing environment at the destination that you specified.

Some statistics will not be calculated accurately unless the FULLSTIMER system option is specified at start-up time.

The following is an example of the statistics that the SAS log displays when the FULLSTIMER option is on:

```
NOTE: There were 5 observations read from the data set MYSAS.DEPART1.
NOTE: PROCEDURE PRINT used (Total process time):
      real time           0.96 seconds
      user cpu time       0.01 seconds
      system cpu time     0.15 seconds
      Memory                            83k
```

## Comparisons

The FULLSTIMER system option specifies whether all of the available performance statistics are written to the SAS log. The STIMER system option specifies whether time-elapsed statistics for DATA steps or PROC steps are written to the SAS log.

## See Also

- "STIMER System Option" on page 563

- The section optimizing system performance in *SAS Language Reference: Concepts*.

## GISMAPS System Option

**Specifies the location of the SAS data library that contains U.S. Census Tract maps supplied by SAS/GIS**

**Default:** none

**Valid in:** configuration file, SAS invocation, OPTIONS statement, SAS System Options window

**Category:** Graphics: Driver settings

**PROC OPTIONS GROUP=** GRAPHICS

**Windows specifics:** Valid values for *library-specification* and *path-to-library*

**See:** GISMAPS System Option in *SAS Language Reference: Dictionary*

### Syntax

-GISMAPS *library-specification* | *path-to-library*

GISMAPS=*library-specification* | *path-to-library*

***library-specification* | *path-to-library***
specifies either a library or a physical path to a library that contains U.S. Census Tract maps supplied by SAS/GIS. *path-to-library* must be a valid Windows pathname. If the pathname contains spaces, enclose the pathname in quotation marks.

## HELPINDEX System Option

**Specifies one or more index files to be used by SAS Help and Documentation**

**Default:** /help/common.hlp/index.txt, /help/common.hlp/keywords.htm, common.hhk

**Valid in:** configuration file, SAS invocation

**Category:** Environment control: Help

**PROC OPTIONS GROUP=** HELP

**Windows specifics:** *HTML-HELP-index-pathname*

### Syntax

-HELPINDEX <(> "*index-pathname-1*" < " *index-pathname-2*" "*index-pathname-n*")>

***index-pathname***
specifies the partial pathname for the index that is to be used by SAS Help and Documentation. *index-pathname* must be a valid Windows pathname. Pathname must be enclosed in quotation marks. When you specify more than one pathname,

separate the pathnames with a space and enclose the list of pathnames in parentheses.

The *index-pathname* can be any or all of the following:

/help/*applet-index-filename*
: specifies the partial pathname of the index file that is to be used by the SAS Help and Documentation Java applet under a UNIX environment. *applet-index-filename* must have a file extension of .txt, and it must reside in a path that is specified by the HELPLOC system option. The default is /help/common.hlp/index.txt.

  See the default index file for the format that is required for an index file.

/help/*accessible-index-filename*
: specifies the partial pathname of an accessible index file that is to be used by SAS Help and Documentation under UNIX, OpenVMS, or z/OS environments. An accessible index file is an HTML file that can be used by Web browsers. *accessible-index-filename* must have a file extension of .htm and it must reside in a path that is specified by the HELPLOC system option. The default pathname is / help/common.hlp/keywords.htm.

  See the default index file for the format that is required for an index file.

*HTML-Help-index-pathname*
: specifies the pathname of the Microsoft HTML Help index that is to be used by SAS Help and Documentation under Windows environments. The default pathname is common.hhk. For information about creating an index for Microsoft HTML Help, see your Microsoft HTML Help documentation.

## Details

Use the HELPINDEX option if you have a customized index that you want to use instead of the index that SAS supplies. If you use one configuration file to start SAS under more than one operating environment, you can specify all of the partial pathnames in the HELPINDEX option.The order of the pathnames is not important, although only one pathname of each type can be specified.

When the HELPINDEX option specifies a pathname for UNIX, OpenVMS, or z/OS operating environments, SAS determines the complete path by replacing **`/help/`** in the partial pathname with the pathname that is specified in the HELPLOC option. If the HELPLOC option contains more than one pathname, each path is searched for the specified index.

For example, when the value of HELPINDEX is **`/help/common.hlp/myindex.htm`** and the value of HELPLOC is **`/u/myhome/myhelp`**, the complete path to the index is **`/ u/myhome/myhelp/common.hlp/myindex.htm`**.

## See Also

- "HELPLOC System Option" on page 508

## HELPLOC System Option

**Specifies the location of Help files that are used to view SAS Help and Documentation using Microsoft HTML Help**

**Default:** ("!MYSASFILES\classdoc" "!*sasroot*\nls\en\help" "!*sasroot*\core\help")

**Valid in:** configuration file, SAS invocation

**Category:** Environment control: Help

**PROC OPTIONS GROUP=** HELP

**Windows specifics:** valid values for *pathname*

### Syntax

-HELPLOC <(> "*pathname-1*" <"*pathname-2*" "*pathname-n*")>

***pathname***

specifies one or more directory pathnames in which SAS Help and Documentation files are located. *Pathname* must be a valid Windows pathname that contains the installed Microsoft HTML Help files. Pathnames must be enclosed in quotation marks. When more than one pathname is specified, use parentheses around the list of pathnames.

### Details

Specifying a value for the HELPLOC system option causes SAS to insert that value at the start of a concatenated list of values. This enables you to access the help for your site without losing access to SAS Help and Documentation.

The default folders !MYSASFILES\classdoc and !*sasroot*\core\help are used for SAS/AF application Help and SAS Help and Documentation, respectively.

**Example** The following command contains two specifications of HELPLOC:

```
sas -helploc "c:\app1\help" -helploc "c:\app2\help"
```

The value of the system option is of the following form:

```
("c:\app2\help" "c:\app1\help" "!sasuser\classdoc" "!sasroot\nls\en\help"
"!sasroot\core\help")
```

## HELPREGISTER System Option

**Registers help files to access from the main SAS window Help menu**

**Default:** none

**Valid in:** configuration file, SAS invocation

**Category:** Environment control: Help

**PROC OPTIONS GROUP=** HELP

**Windows specifics:** all

### Syntax

-HELPREGISTER "*menu string*" *help file location* <"*help string*"> <*topic*><CHM | HLP | HTML>

**"*menu string*"**

is the text string that appears in the Help menu.

***help file location***
specifies the folder and the filename in which the help file is located. The *help file location* can be omitted if the file resides in a folder that is specified by the HELPLOC system option. The *help file location* may be truncated with !*sasroot*. If *help file location* includes blank spaces, it must be enclosed in quotation marks.

***"help string"***
is the text that appears in the status line when a user places the mouse over the *menu string*.

***topic***
is the topic within the help file that displays when you select *menu string* from SAS help menu. For HTML files, the topic is the anchor (preceded with #) within the document. For CHM files, the topic is the page within the CHM file. For HLP files, topic is the keyword in the file for which WinHelp searches. If *topic* includes blank spaces, it must be enclosed in quotation marks.

**CHM**
specifies an HtmlHelp CHM file on the local system or network.

**HLP**
specifies a WinHelp file on the local system or network.

**HTML**
specifies an HTML file on the local file system or network, or a valid URL.

## Details

Use the HELPREGISTER system option to add up to 20 help files that you would like available from the main SAS window Help menu. All strings containing spaces must be enclosed in double quotation marks. Optional arguments may be omitted by replacing them with a single period (.) or empty double quotation marks ("") . If no further argument is necessary, no place-holder is required.

To add multiple Help files to the Help menu, use multiple HELPREGISTER system options either in the configuration file or at the command prompt when you start SAS.

## Examples

### Example 1: HTML Pages and URLs

```
sas -helpregister ''SAS Institute Inc'' http://www.sas.com
    ''SAS's homepage on the web'' . html

sas -helpregister ''Local HTML Doc'' c:\mypage.htm
    ''My own help'' middle
```

### Example 2: HTML Help Files (.CHM)

```
sas -helpregister ''My CHM file'' \\server\share\HelpStuff.chm .
    ''InternalFile.htm''

sas -helpregister ''SAS Windows Companion'' host.chm .
    ''/host.hlp/chostfutil.htm''
```

### Example 3: WinHelp Files (.HLP)

```
sas -helpregister ''A WinHelp File'' c:\somefile.hlp
    ''simply an .hlp file''
```

```
sas -helpregister ''WinHelp with a Topic'' c:\somefile.hlp .
    ''My Topic''
```

### See Also

□ “Adding Help to the Help Menu” on page 64

## HELPTOC System Option

**Specifies the table of contents files to be used by SAS Help and Documentation**

**Default:** /help/helpnav.hlp/config.txt /help/common.hlp/toc.htm common.hhc

**Valid in:** configuration file, SAS invocation

**Category:** Environment control: Help

**PROC OPTIONS GROUP=** HELP

**Windows specifics:** *HTML-Help-TOC-pathname*

### Syntax

-HELPTOC <(> "*TOC-pathname-1*" < "*TOC-pathname-2*" "*TOC-pathname-3*")>

***TOC-pathname***

specifies a partial pathname for the table of contents that is to be used by SAS Help and Documentation. The *TOC-pathname* must be a valid Windows pathname. Pathnames must be enclosed in quotation marks. When more than one pathname is specified, use parentheses around the list of pathnames.

The *TOC-pathname* can be any or all of the following:

/help/*applet-TOC-filename*

specifies the partial pathname of the table of contents file that is to be used by the SAS Help and Documentation Java applet under a UNIX environment. *applet-TOC-filename* must have a file extension of .txt, and it must reside in a path that is specified by the HELPLOC system option. The default is /help/helpnav.hlp/config.txt.

See the default table of contents file for the format that is required for an index file.

/help/*accessible-TOC-filename*

specifies the partial pathname of an accessible table of contents file that is to be used by SAS Help and Documentation under UNIX, OpenVMS, or z/OS environments. An accessible table of contents file is an HTML file that can be used by Web browsers. *accessible-TOC-filename* must have a file extension of .htm and it must reside in a path that is specified by the HELPLOC system option. The default pathname is /help/common.hlp/toc.htm.

See the default table of contents file for the format that is required for a table of contents.

*HTML-Help-TOC-pathname*

specifies the complete pathname to the Microsoft HTML Help table of contents that is to be used by SAS Help and Documentation in Windows environments. The default pathname is common.hhc. For information about creating an index for Microsoft HTML Help, see your Microsoft HTML Help documentation.

### Details

Use the HELPTOC option if you have a customized table of contents that you want to use instead of the table of contents supplied by SAS. If you use one configuration file to start SAS under more than one operating environment, you can specify all of the partial pathnames in the HELPTOC option. The order of the pathnames is not important, although only one pathname of each type can be specified.

When the HELPTOC option specifies the pathname for UNIX, OpenVMS, and z/OS operating environments, SAS determines the complete path by replacing **`/help/`** in the partial pathname with the pathname that is specified in the HELPLOC option. If the HELPLOC option contains more than one pathname, each path is searched for the table of contents.

For example, when HELPTOC is **`/help/common.hlp/mytoc.htm`** and the value of HELPLOC is **`/u/myhome/myhelp`**, the complete path to the table of contents is **`/u/myhome/myhelp/common.hlp/mytoc.htm`**.

### See Also

- "HELPLOC System Option" on page 508

## HOSTPRINT System Option

**Specifies that the Windows Print Manager is to be used for printing**

**Default:** HOSTPRINT

**Valid in:** configuration file, SAS invocation, OPTIONS statement, SAS System Options window

**Category:** Log and procedure output control: Procedure output

**PROC OPTIONS GROUP=** LISTCONTROL

**Windows specifics:** all

### Syntax

-HOSTPRINT | -NOHOSTPRINT

HOSTPRINT | NOHOSTPRINT

**HOSTPRINT**
specifies to use Windows printing. This is the default.

**NOHOSTPRINT**
specifies to use SAS forms for printing.

### Details

Use the NOHOSTPRINT option to use forms for printing in a batch SAS session. When you specify NOHOSTPRINT, the **`Use Forms`** check box is selected in the Print Setup dialog box, and SAS uses the linesize, pagesize, and font values that are specified in your SAS form.

### See Also

- “Setting Print Options to Use Forms” on page 175

## ICON System Option

**Minimizes the SAS window**

**Default:** NOICON

**Valid in:** configuration file, SAS invocation, OPTIONS statement, SAS System Option window

**Category:** Environment control: Display

**PROC OPTIONS GROUP=** ENVDISPLAY

**Windows specifics:** all

### Syntax

-ICON | -NOICON

ICON | NOICON

**ICON**
specifies to minimize the main SAS window immediately.

**NOICON**
restores the main SAS window immediately.

### Details

If you put the ICON system option in the SAS command or the SAS configuration file, SAS is minimized upon initialization. If you submit the ICON system option in an OPTIONS statement, SAS is immediately minimized. This is equivalent to clicking on the minimize button.

This system option is especially useful for obtaining a minimized SAS session as soon as you start Windows. For example, the ICON system option could be specified in the SAS command as follows:

```
c:\sas\sas.exe -icon
```

## INITSTMT System Option

**Specifies a SAS statement to be executed after any statements in the autoexec file and before any statements from the SYSIN= file**

**Valid in:** configuration file, SAS invocation

**Alias:** IS

**Category:** Environment control: Initialization and operation

**PROC OPTIONS GROUP=** EXECMODES

**Windows specifics:** *statement* must end a DATA or PROC step if you use the Enhanced Editor

**See:** INITSTMT= System Option in *SAS Language Reference: Dictionary*

## Syntax

INITSTMT '*statement*'

***'statement'***
specifies any SAS statement or statements. The value of *statement* must end a DATA or PROC step if you use the Enhanced Editor.

## JREOPTIONS System Option

**Identifies Java Runtime Environment (JRE) options for SAS**

**Default:** -Djava.security.policy=<*pathname*\*filename*> -Dsas.jre=(private | public) -Dsas.jre.home=!*sasroot*\*pathname* -Djava.ext.dirs=*pathname*\*filename*

**Valid in:** configuration file, SAS invocation

**Category:** Environment control: Initialization and operation

**PROC OPTIONS GROUP** EXECMODES

**Windows specifics:** all

### Syntax

-JREOPTIONS (*-JRE-option-1* <*-JRE-option-n*>)

JREOPTIONS (*-JRE-option-1* <*-JRE-option-n*>)

***-JRE-option***
specifies one or more Java Runtime Environment options. JRE options must begin with a hyphen ( - ). Use a space to separate multiple JRE options. Valid values for *JRE-option* depend on your installation's Java Runtime Environment. For information about JRE options, see your installation's Java documentation.

### Details

The set of JRE-options must be enclosed in parentheses. If you specify multiple JREOPTIONS system options, SAS appends JRE-options to JRE-options that are currently defined. Incorrect JRE-options are ignored. To define the classpath, use the Djava.class.path option.

## Examples

- **`-jreoptions (-verbose)`**
- **`-jreoptions (-Djava.class.path= "c:\my java\classes\myclasses.jar";c:\java2\classes2\classes2.jar -oss600k)`**

## LINESIZE System Option

**Specifies the line size of SAS Log and Output windows**

**Default:** Varies depending on display settings

**Valid in:** configuration file, SAS invocation, OPTIONS statement, SAS System Options window

**Category:** Log and procedure output control: SAS log and procedure output

**PROC OPTIONS GROUP=** LOG_LISTCONTROL

**Windows specifics:** Default value

**See:** LINESIZE System Optionin *SAS Language Reference: Dictionary*

### Syntax

-LINESIZE *n* | MIN | MAX

LINESIZE=*n* | MIN| MAX

***n***
specifies the line size in characters. Valid values range between 64 and 256.

**MIN**
sets the line size to 64 characters.

**MAX**
sets the line size to 256 characters.

### Details

The default values are based on the printer resolution and printer font so that generated reports print correctly.

***CAUTION:***
**Modifying print options by using the Windows printing dialog boxes can change the values of SAS printing system options, which might cause unpredictable output.** If you set printing options using SAS system options such as LINESIZE and PAGESIZE, and then use the Windows printing dialog boxes to set printing options. The SAS system options are set to the values that are specified in the Windows print dialog boxes. △

### See Also

- "PAGESIZE System Option" on page 530

- In the *SAS Language Reference: Dictionary*:
  - "ORIENTATION System Option"
  - "PAGESIZE System Option"

## LOADMEMSIZE System Option

**Specifies a suggested amount of memory needed for executable programs loaded by SAS**

**Default:** 0

**Valid in:** configuration file, SAS invocation

**Category:** System administration: Memory

**PROC OPTIONS GROUP=** MEMORY

**Windows specifics:** all

### Syntax

-LOADMEMSIZE *n* | *n*K | *n*M | *n*G | *hex*X | MIN | MAX

***n* | *n*K | *n*M | *n*G**

specifies the memory size in multiples of 1; 1,024 (kilobytes); 1,048,576 (megabytes), and 1,073,741,842 (gigabytes), respectively. You can specify decimal values for the number of kilobytes, megabytes, or gigabytes. For example, a value of `8` specifies 8 bytes, a value of `.782k` specifies 801 bytes, and a value of `3m` specifies 3,145,728 bytes.

***hex*X**

specifies the amount of memory as a hexadecimal value. You must specify the value beginning with a number (0–9), followed by an X. For example, the value `2dx` sets the amount of memory to 45 bytes.

**MIN**

specifies 0 bytes, which indicates that there is no limit on the total amount of memory that can be used.

**MAX**

specifies that the maximum amount of memory for executable programs is limited only by the amount of memory available.

### Details

When LOADMEMSIZE is set to 0, the memory that is used for executable programs that are loaded by SAS is limited only by the amount of system memory available. If LOADMEMSIZE is set to 1, executable programs are purged from memory when they are no longer in use.

For values of two or greater, SAS first checks the amount of memory that is available for SAS executable programs. If the total amount of memory that is available is greater than the value of LOADMEMSIZE, SAS purges the SAS loaded executable programs that are not in use until the memory that is used is less than the value of the LOADMEMSIZE option, or until there are no other SAS loaded executable programs that can be purged. If all executable programs have been purged and more memory is needed, additional system memory is used as long as it is available.

## LOCALE System Option

**Specifies the locale of the SAS session**

**Default:** English

**Valid in:** configuration file, SAS invocation, OPTIONS statement, SAS System Options window

**Category:** Environment control: Language control

**PROC OPTIONS GROUP=** LANGUAGECONTROL

**Windows specifics:** all

**See:** LOCALE= System Option in *SAS National Language Support (NLS): User's Guide*

## LOG System Option

**Controls the creation of the SAS log file for batch mode**

**Default:** *filename*.LOG in batch mode, where *filename* is the name of your SAS job

**Valid in:** configuration file, SAS invocation

**Category:** Environment control: Files

**PROC OPTIONS GROUP=** ENVFILES

**Windows specifics:** *destination* must be a valid Windows filename

### Syntax

-LOG "*destination*" | -NOLOG

**LOG "*destination*"**
specifies the destination for the SAS log. The *destination* argument can be a valid Windows pathname or filename (including device names such as LPT1) or an environment variable that is associated with a pathname. If you specify only a pathname, the log file is created in the specified directory with the default name of *filename*.LOG, where *filename* is the name of your SAS job.

**NOLOG**
routes each log message to a message box, where one message is displayed per message box.

### Details

The LOG system option specifies a destination to which the SAS log is written when executing SAS programs in batch mode.

This system option is valid only in batch mode.

When you are running SAS interactively, the log is sent to the LOG window; in batch mode, it is sent to a file named *filename*.LOG that is located in the current SAS directory, where *filename* is the name of your SAS job. You can use the LOG system option to specify an alternate destination.

To disable the display of the SAS log, use the NOTERMINAL system option.

When SAS is started with the OBJECTSERVER and NOTERMINAL system options and no log is specified, SAS discards all log messages.

### See Also

- "TERMINAL System Option" in *SAS Language Reference: Dictionary*

## MAPS System Option

**Specifies the name of the SAS library that holds the SAS/GRAPH map data sets**

**Default:** !*sasroot*\maps

**Valid in:** configuration file, SAS invocation, OPTIONS statement, SAS System Options window

**Category:** Graphics: Driver settings

**PROC OPTIONS GROUP=** GRAPHICS

**Windows specifics:** default value and *location-of-maps* must resolve to a valid Windows pathname

**See:** MAPS System Option in *SAS Language Reference: Dictionary*

### Syntax

-MAPS *location-of-maps*

MAPS=*location-of-maps*

***location-of-maps***
specifies a libref, a valid Windows pathname, or an environment variable associated with a pathname. Remember that a pathname is only to the directory or subdirectory level. If the pathname contains spaces, enclose the pathname in quotation marks.

## MAXMEMQUERY System Option

**Specifies the limit on the maximum amount of memory that is allocated for procedures**

**Default:** 0

**Valid in:** configuration file, SAS invocation, OPTIONS statement, SAS System Options window

**Category:** System administration: Memory

**PROC OPTIONS GROUP=** MEMORY

**Windows specifics:** all

### Syntax

-MAXMEMQUERY *n* | *n*K | *n*M | *n*G | *hex*X | MIN | MAX

MAXMEMQUERY= *n* | *n*K | *n*M | *n*G | *hex*X | MIN | MAX

***n* | *n*K | *n*M | *n*G**
specifies the limit in multiples of 1; 1,024 (kilobytes); 1,048,576 (megabytes), and 1,073,741,842 (gigabytes), respectively. You can specify decimal values for the number of kilobytes, megabytes, or gigabytes. For example, a value of **`8`** specifies 8 bytes, a value of **`.782k`** specifies 801 bytes, and a value of **`3m`** specifies 3,145,728 bytes.

***hex*X**
specifies the amount of memory as a hexadecimal value. You must specify the value beginning with a number (0–9), followed by an X. For example, the value **`2dx`** sets the amount of memory to 45 bytes.

**MIN**
sets the amount of memory to the minimum setting, which is 0 bytes. This indicates that there is no limit on the total amount of memory that can be used by each procedure.

**MAX**
sets the amount of memory to the maximum setting, which is 2,147,483,647 bytes.

### Details

Some SAS procedures attempt to allocate the maximum amount of the memory that is possible, up to the amount specified by the MEMSIZE option. If this amount of memory is not available, SAS attempts to use paging. If the amount of page space is less than the value of MEMSIZE, SAS generates an error message. The MAXMEMQUERY option specifies the maximum amount of memory that SAS can request at one time. If your system has small system paging devices, you may want to lower the value of MAXMEMQUERY.

## MEMBLKSZ System Option

**Specifies the memory block size for memory-based libraries for Windows operating environments, excluding Windows NT**

**Default:** 16 MB

**Valid in:** configuration file, SAS invocation

**Category:** System administration: Memory

**PROC OPTIONS GROUP** MEMORY

**Windows specifics:** all

### Syntax

-MEMBLKSZ *n* | *n*K | *n*M |*n*G | *n*T | *hex*X

***n* | *n*K | *n*M | *n*G | *n*T**
specifies the memory block size in multiples of 1; 1,024 (kilobytes); 1,048,576 (megabytes); 1,073,741,824 (gigabytes); and 1,099,511,627,776 (terabytes),

respectively. You can specify decimal values for the number of kilobytes, megabytes, gigabytes, or terabytes. For example, a value of **`8`** specifies 8 bytes, a value of **`.782k`** specifies 801 bytes, and a value of **`3m`** specifies 3,145,728 bytes.

***hex*X**
specifies the memory block size as a hexadecimal value. You must specify the value beginning with a number (0–9), followed by an X. For example, the value **`2dx`** sets the memory block size to 45 bytes.

## Details

Beginning with Windows 2000, multiple processes can be run simultaneously in memory. The value of the MEMBLKSZ system option is the amount of memory that is initially allocated. Additional memory can be allocated in the same memory allocation size that is specified in the MEMBLKSZ option, up to the amount of memory that is specified in the MEMMAXSZ option. For example, if MEMBLKSZ is 2M, additional memory can be allocated in 2M blocks.

When memory-based libraries are using extended memory, this value is also used to determine the amount of the process address space that is used to access the extended memory.

*Note:* This option is ignored in Windows NT operating environments. △

*Note:* Specifying a value that is too large could adversely affect performance.

- Beginning with Windows 2000, specifying a value that is too large could adversely affect overall system performance. Try different values for the MEMBLKSZ option to determine the value that gives the best system performance.
- If you are using extended memory in 32-bit environments, then specifying a value that is too large could adversely affect SAS performance. A smaller value may be optimal. A good starting point is 64K; however, try different values for the MEMBLKSZ option to determine the value that gives the best SAS performance.

△

## See Also

- "Memory-Based Libraries" on page 199
- "MEMMAXSZ System Option" on page 522

## MEMCACHE System Option

**Specifies to use the memory-based libraries as a SAS file cache**

**Default:** 0

**Valid in:** configuration file, SAS invocation, OPTIONS statement, SAS System Options window

**Category:** Files: SAS Files

**PROC OPTIONS GROUP=** SASFILES

**Windows specifics:** all

## Syntax

-MEMCACHE 0 | 1 | 4

MEMCACHE= 0 | 1 | 4

**0**
specifies memory cache is off.

**1**
specifies not to add any new files to the cache. Reads and writes to files already in the cache continue as if MEMCACHE is on.

**4**
specifies memory cache is on. Memory is used as a SAS file cache.

### Details

When the MEMCACHE system option is 4 or 1, SAS file cache places data in memory as it is processed. This data is then available for future references by SAS. Files in the cache are kept until SAS is shut down, caching is terminated, or more space is required for new files. Memory is reclaimed on a least recently used basis. Cached data is written to permanent storage. You can control which SAS libraries use the cache by using the MEMCACHE system option in the OPTIONS statement. Memory usage can be monitored using the performance tools.

### See Also

- "Memory-Based Libraries" on page 199
- "MEMLIB System Option" on page 521

## MEMLIB System Option

**Specifies to process the Work library as a memory-based library**

**Default:** NOMEMLIB
**Valid in:** configuration file, SAS invocation
**Category:** Files: SAS Files
**PROC OPTIONS GROUP=** SASFILES
**Windows specifics:** all

### Syntax

-MEMLIB | -NOMEMLIB

**MEMLIB**
specifies to use memory for the Work libraries.

**NOMEMLIB**
specifies not to use memory.

### Details

When the MEMLIB system option is specified, the Work library is processed in memory. Files are kept in memory until SAS is terminated or the files are deleted. You can monitor memory usage by using the performance tools.

### See Also

- □ "Memory-Based Libraries" on page 199
- □ "LIBNAME Statement" on page 456
- □ "MEMCACHE System Option" on page 520
- □ "Performance Tools" on page 226

## MEMMAXSZ System Option

**Specifies the maximum amount of memory to allocate for using memory-based libraries in Windows operating environments, excluding Windows NT**

**Default:** 2G

**Valid in:** configuration file, SAS invocation

**Category:** System administration: Memory

**PROC OPTIONS GROUP=** MEMORY

**Windows specifics:** all

### Syntax

-MEMMAXSZ *n* | *n*K | *n*M |*n*G | *n*T | *hex*X

***n* | *n*K | *n*M | *n*G | *n*T**
specifies the amount of memory to allocate in multiples of 1; 1,024 (kilobytes); 1,048,576 (megabytes); 1,073,741,824 (gigabytes); and 1,099,511,627,776 (terabytes), respectively. You can specify decimal values for the number of kilobytes, megabytes, gigabytes, or terabytes. For example, a value of **8** specifies 8 bytes, a value of **.782k** specifies 801 bytes, and a value of **3m** specifies 3,145,728 bytes.

***hex*X**
specifies the amount of memory to allocate as a hexadecimal value. You must specify the value beginning with a number (0–9), followed by an X. For example, the value **2dx** sets the amount of memory to 45 bytes.

### Details

The MEMMAXSZ system option specifies the total amount of memory that SAS can use for memory-based libraries. You can monitor the memory by using the performance tools.

*Note:* This option is ignored in Windows NT operating environments. △

*CAUTION:*
**Specifying a value that is too large may adversely affect overall system performance.** Try different values for the MEMMAXSZ option to determine the value that gives the best system performance. △

### See Also

- □ “Memory-Based Libraries” on page 199
- □ “MEMBLKSZ System Option” on page 519
- □ “MEMLIB System Option” on page 521
- □ “MEMCACHE System Option” on page 520

## MEMSIZE System Option

**Specifies a limit on the total amount of memory SAS uses at any one time**

**Default:** 0

**Valid in:** configuration file, SAS invocation

**Category:** System administration: Memory

**PROC OPTIONS GROUP=** MEMORY

**Windows specifics:** valid values

### Syntax

-MEMSIZE *n* | *n*K | *n*M | *n*G | *hex*X | MIN | MAX

***n* | *n*K | *n*M | *n*G**
specifies the limit in multiples of 1; 1,024 (kilobytes); 1,048,576 (megabytes), and 1,073,741,824 (gigabytes), respectively. The value of *n* can be a decimal value. For example, a value of **`8`** specifies 8 bytes, a value of **`.782k`** specifies 801 bytes, and a value of **`3m`** specifies 3,145,728 bytes. Under 32–bit operating environments, the largest value that you can specify is 4294967295 (4G–1).

***hex*X**
specifies the limit as a hexadecimal value. You must specify the value beginning with a number (0–9), followed by an X. For example, the value **`2dx`** sets the limit to 45 bytes.

**MIN**
specifies to set the limit to 0. A value of zero indicates there is no limit except the operating system limit.

**MAX**
specifies to set the limit to the largest possible setting.

### Details

The operating system may use additional amounts of memory. The memory used by SAS includes virtual memory and is therefore not limited to RAM. If MEMSIZE is set to a value that is too low and could cause performance problems, the value of MEMSIZE is determined by SAS.

For optimal performance, use the default value of 0 for MEMSIZE.

## MSG System Option

**Specifies the library that contains SAS error messages**

**Default:** !*sasroot*\core\sasmsg

**Valid in:** configuration file, SAS invocation

**Category:** Environment control: Files

**PROC OPTIONS GROUP=** ENVFILES

**Windows specifics:** Valid values for *library-specification*

### Syntax

-MSG *library-specification*

***library-specification***
can be a Windows logical name (including search strings) or pathname. Do not include a filename. If the pathname contains spaces, you must enclose the pathname in quotation marks.

### Details

The MSG system option specifies the name of the library for SAS error messages.

## MSGCASE System Option

**Specifies whether notes, warnings, and error messages that are generated by SAS are displayed in uppercase characters**

**Default:** NOMSGCASE

**Valid in:** configuration file, SAS invocation

**Category:** Log and procedure output control: SAS log

**PROC OPTIONS GROUP=** LOGCONTROL

**Windows specifics:** all

### Syntax

-MSGCASE | -NOMSGCASE

**MSGCASE**
specifies that messages are displayed in uppercase characters.

**NOMSGCASE**
specifies that messages can include uppercase and lowercase characters. This is the default.

### Details

The MSGCASE system option specifies whether or not messages from the message file are uppercased before they are written out. The setting of the MSGCASE option does not affect user-generated messages and source lines.

## MSYMTABMAX System Option

**Specifies the maximum amount of memory available to the macro variable symbol table(s)**

**Default:** 4194304 bytes (4 MB)

**Valid in:** configuration file, SAS invocation, OPTIONS statement, SAS System Options window

**Category:** Macro: SAS macro

**PROC OPTIONS GROUP=** MACRO

**Windows specifics:** Default value

**See:** MSYMTABMAX System Option in *SAS Macro Language: Reference*

### Syntax

-MSYMTABMAX *n* | *n*K | *n*M | *n*G |*n*T | *hex*X | MIN | MAX

MSYMTABMAX=*n* | *n*K | *n*M | *n*G | *n*T |*hex*X | MIN | MAX

***n* | *n*K | *n*M | *n*G | *n*T**
: specifies the amount of memory that is available in multiples of 1; 1,024 (kilobytes); 1,048,576 (megabytes); 1,072,741,824 (gigabytes); and 1,099,511,627,776 (terabytes), respectively. You can specify decimal values for the number of kilobytes, megabytes, gigabytes, or terabytes. For example, a value of `8` specifies 8 bytes, a value of `.782k` specifies 801 bytes, and a value of `3m` specifies 3,145,728 bytes.

***hex*X**
: specifies the amount of memory that is available as a hexadecimal value. You must specify the value beginning with a number (0–9), followed by an X. For example, the value `2dx` sets the amount of memory to 45 bytes.

**MIN**
: sets the amount of memory that is available to the minimum setting, which is 0. This causes all macro variables to be written to disk.

**MAX**
: sets the amount of memory that is available to the maximum setting.

### Details

After the MSYMTABMAX value is reached, SAS writes any additional macro variables to disk.

## MVARSIZE System Option

**Specifies the maximum size for in-memory macro variables**

**Default:** 4096 bytes

**Valid in:** configuration file, SAS invocation, OPTIONS statement, SAS System Options window

**Category:** Macro: SAS macro

**PROC OPTIONS GROUP=** MACRO

**Windows specifics:** Default value

**See:** MVARSIZE System Option in *SAS Macro Language: Reference*

## Syntax

-MVARSIZE *n* | *n*K | *hex*X | MIN | MAX

MVARSIZE=*n* | *n*K | *hex*X | MIN | MAX

***n* | *n*K**
specifies the maximum macro variable size in multiples of 1 or 1,024 (kilobytes), respectively. You can specify decimal values for the number of kilobytes. For example, a value of **8** specifies 8 bytes, a value of **.782k** specifies 801 bytes, and a value of **3k** specifies 3,072 bytes.

***hex*X**
specifies the maximum macro variable size as a hexadecimal value. You must specify the value beginning with a number (0–9), followed by an X. For example, the value **2dx** sets the maximum macro variable size to 45 bytes.

**MIN**
sets the macro variable size to the minimum setting, which is 0 bytes. This causes all macro variables to be written to disk.

**MAX**
sets the macro variable size to the maximum setting, which is 65,534 bytes.

## Details

The MVARSIZE system option specifies the maximum size for macro variables that are stored in memory. If the size of the macro variable is larger than the maximum value that is specified, variables are written out to disk.

The value of the MVARSIZE system option can affect system performance. Before you specify the value for production jobs, run tests to determine the optimum value.

## NEWS System Option

**Specifies a file that contains messages to be written to the SAS log**

**Default:** none

**Valid in:** configuration file, SAS invocation

**Category:** Environment control: Files

**PROC OPTIONS GROUP=** ENVFILES

**Windows specifics:** Valid values for *file-specification*

**See:** NEWS System Option in *SAS Language Reference: Dictionary*

## Syntax

-NEWS *file-specification*

***file-specification***
specifies an external file. The value for *file-specification* can be a valid Windows pathname or shortcut name. If the pathname contains spaces, you must enclose the pathname in quotation marks.

## Details

The NEWS file can contain information for users, including news items about SAS. The contents of the NEWS file are displayed in the SAS log immediately after the SAS header.

## NLSCOMPATMODE System Option

**Provides national language compatibility with previous releases of SAS**

**Default:** NONLSCOMPATMODE

**Valid in:** configuration file, SAS invocation

**Category:** Environment control: Language control

**PROC OPTIONS GROUP=** LANGUAGECONTROL

**Windows specifics:** all

**See:** NLSCOMPATMODE System Option in *SAS National Language Support (NLS): User's Guide*

## NUMKEYS System Option

**Controls the number of available function keys**

**Default:** number of function keys on the keyboard

**Valid in:** configuration file, SAS invocation

**Category:** Input control: Data processing

**PROC OPTIONS GROUP=** INPUTCONTROL

**Windows specifics:** all

### Syntax

-NUMKEYS *number-of-keys*

***number-of-keys***
specifies the number of active keyboard function keys.

### Details

When SAS initializes, it queries your machine to determine the number of keyboard function keys. You can override this setting by specifying a different value with the NUMKEYS system option.

### Example

If you specify the following system option, SAS displays 10 function keys in the KEYS window:

```
-numkeys 10
```

## NUMMOUSEKEYS System Option

**Specifies the number of mouse buttons SAS displays in the KEYS window**

**Default:** 3 buttons

**Valid in:** configuration file, SAS invocation

**Category:** Input control: Data processing

**PROC OPTIONS GROUP=** INPUTCONTROL

**Windows specifics:** all

### Syntax

-NUMMOUSEKEYS *number-of-buttons*

***number-of-buttons***
specifies the number of mouse buttons, ranging from 0 to 3. If *number-of-buttons* is 0 or 1, the KEYS windows lists no mouse buttons (because the left, and in this case the only, mouse button is reserved by SAS). If *number-of-buttons* is 2, the KEYS window lists the right mouse button (RMB), as well as Ctrl + right mouse button and Shift + right mouse button. If *number-of-buttons* is 3, the KEYS window lists both the right mouse button and the middle mouse button.

### Details

Unless you specify the NUMMOUSEKEYS system option, SAS assumes that three mouse buttons are available. If you have a one- or two-button mouse and want the KEYS window to reflect this, specify the NUMMOUSEKEYS system option in your SAS configuration file.

## OBS System Option

**Specifies when to stop processing observations or records**

**Default:** MAX

**Valid in:** configuration file, SAS invocation, OPTIONS statement, SAS System Options window

**Category:** Files: SAS Files

**PROC OPTIONS GROUP=** SASFILES

**Windows specifics:** Valid range

**See:** OBS System Option in *SAS Language Reference: Dictionary*

### Syntax

-OBS *n* | *n*K | *n*M | *n*G | *n*T |*hex*X | MIN | MAX

OBS=*n* | *n*K | *n*M | *n*G | *n*T |*hex*X | MIN | MAX

***n* | *n*K | *n*M | *n*G | *n*T**
specifies a number to indicate when to stop processing, with *n* being an integer. Using one of the letter notations results in multiplying the integer by a specific value. That is, specifying K (kilo) multiplies the integer by 1,024, M (mega) multiplies by 1,048,576, G (giga) multiplies by 1,073,741,824, T (tera) multiplies by 1,099,511,627,776. You can specify a decimal value for *n* when it is used to specify a K, M, G, or T value. For example, a value of **`20`** specifies 20 observations or records, a value of **`.782k`** specifies 801 observations or records, and a value of **`3m`** specifies 3,145,728 observations or records.

***hex*X**
specifies a number to indicate when to stop processing as a hexadecimal value. You must specify the value beginning with a number (0–9), followed by an X. For example, the hexadecimal value F8 must be specified as **`0F8X`** in order to specify the decimal equivalent of 248. The value **`2dx`** specifies the decimal equivalent of 45.

**MIN**
sets the number to indicate when to stop processing to 0.

**MAX**
sets the number to indicate when to stop processing to 2,147,483,647. On 64–bit systems, MAX is 9,223,372,036,854,775,807. MAX is the default.

## PAGENO System Option

**Resets the page number**

**Default:** 1

**Valid in:** configuration file, SAS invocation, OPTIONS statement, SAS System Options window

**Category:** Log and procedure output control: Procedure output

**PROC OPTIONS GROUP=** LISTCONTROL

**Windows specifics:** Valid values for *n*; syntax

**See:** PAGENO System Option in *SAS Language Reference: Dictionary*

### Syntax

-PAGENO *n* | *n*K | *n*M | *n*G | *hex*X | MIN | MAX

PAGENO=*n* | *n*K | *n*M | *n*G | *hex*X | MIN | MAX

***n* | *n*K | *n*M | *n*G**
specifies the page number in multiples of 1(*n*); 1,024 (*n*K); 1,048,576 (*n*M); and 1,073,741,824 (*n*G), respectively. You can specify a decimal value for *n* when it is used to specify a K, M, G, or T value. For example, a value of `8` sets the page number to 8, a value of `.782k` sets the page number to 801, and a value of `3k` sets the page number to 3,072.

***hex*X**
specifies the page number as a hexadecimal value. You must specify the value beginning with a number (0–9), followed by an X. For example, the value 2dx sets the page number to 45.

**MIN**
sets the page number to the minimum number, which is 1.

**MAX**
sets the page number to the maximum number, which is 2,147,483,647.

### Details

The PAGENO system option specifies a beginning page number for the next page of output that SAS produces.

## PAGESIZE System Option

**Specifies the number of lines that compose a page of SAS output**

**Default:** Varies depending on your display settings

**Valid in:** configuration file, SAS invocation, OPTIONS statement, SAS System Options window

**Category:** Log and procedure output control: SAS log and procedure output

**PROC OPTIONS GROUP=** LOG_LISTCONTROL

**Windows specifics:** Default value

**See:** PAGESIZE System Option in *SAS Language Reference: Dictionary*

### Syntax

-PAGESIZE *n* | MIN | MAX

PAGESIZE=*n* | MIN | MAX

***n***
specifies the number of lines that compose a page.

**MIN**
sets the number of lines that compose a page to the minimum setting, which is 15.

**MAX**
sets the number of lines that compose a page to the maximum setting, which is 32,767.

### Details

Under Windows, the default values are based on the printer resolution and printer font so that generated reports print correctly.

***CAUTION:***
**Modifying print options by using the Windows printing dialog boxes might change the values of SAS printing system options, which might cause unpredictable output.** If you set printing options using SAS system options such as LINESIZE and PAGESIZE, and then use the Windows printing dialogs to set printing options, the SAS system options are set to the values that are specified in the Windows print dialog boxes. △

### See Also

- □ “LINESIZE System Option” on page 515

## PAPERTYPE System Option

**Specifies to a printer the type of paper to use for printing**

**Default:** PLAIN

**Valid in:** configuration file, SAS invocation, OPTIONS statement, SAS System Options window

**Category:** Log and procedure output control: ODS printing

**PROC OPTIONS GROUP=** ODSPRINT

**Windows specifics:** valid values

**See:** PAPERTYPE= System Option in *SAS Language Reference: Dictionary*

### Syntax

-PAPERTYPE PLAIN | STANDARD | GLOSSY | TRANSPARENCY | *printer-defined-value*

PAPERTYPE= PLAIN | STANDARD | GLOSSY | TRANSPARENCY | *printer-defined-value*

**PLAIN**
specifies to use plain paper.

**STANDARD**
specifies to use the standard paper for the printer.

**GLOSSY**
specifies to use glossy paper.

**TRANSPARENCY**
specifies to use transparent paper.

***printer-definded-value***
specifies a paper type that is defined by the printer.

### Details

See your printer documentation for the paper types that your printer can use.

*Operating Environment Information: for Windows NT Users:* The PAPERTYPE system option is not supported under Windows NT. △

## PATH System Option

**Specifies one or more search paths for SAS executable files**

**Default:** !*sasroot*\core\sasexe

**Valid in:** configuration file, SAS invocation

**Category:** System administration: Installation

**PROC OPTIONS GROUP=** INSTALL

**Windows specifics:** all

### Syntax

-PATH <(>"*directory-specification-1*" <"*directory-specification-n*")>

***directory-specification***
specifies the path to search. The value *directory-specification* must be a valid Windows pathname or an environment variable associated with a pathname. If the pathname contains spaces, it must be enclosed in quotation marks. If you specify more than one *directory-specification*, enclose the list of *directory-specification* in parentheses.

### Details

You can specify multiple PATH system options to define a search order.

## PFKEY System Option

**Enables you to map your function keys to the mainframe primary, alternate, or SAA keys**

**Default:** WIN

**Valid in:** configuration file, SAS invocation

**Category:** Input control: Data processing

**PROC OPTIONS GROUP=** INPUTCONTROL

**Windows specifics:** all

## Syntax

-PFKEY PRIMARY | ALTERNATE | SAA | WIN

**PRIMARY**

maps F1 through F12 to the mainframe primary settings for PF1 through PF12 and Shift + F1 through Shift + F12 to PF13 through PF24. The right mouse button (RMB) is mapped to MB2. If you have only 10 function keys, F11, F12, Shift + F11, and Shift + F12 are not available and are not shown in the KEYS window.

Following are the primary mainframe key definitions:

| | **Mainframe** | | **Mainframe** |
|---|---|---|---|
| **PC Key** | **Definition** | **Key** | **Definition** |
| F1 | mark | Shift + F1 | help |
| F2 | smark | Shift + F2 | zoom |
| F3 | unmark | Shift + F3 | zoom off; submit |
| F4 | cut | Shift + F4 | pgm; recall |
| F5 | paste | Shift + F5 | rfind |
| F6 | store | Shift + F6 | rchange |
| F7 | prevwind | Shift + F7 | backward |
| F8 | next | Shift + FF8 | forward |
| F9 | pmenu | Shift + F9 | output |
| F10 | command | Shift + F10 | left |
| F11 | keys | Shift + F11 | right |
| F12 | undo | Shift + F12 | home |
| RMB | zoom off; submit | | |

**ALTERNATE**

maps F1 through F12 to the alternate mainframe key settings. That is, F1 through F12 maps to PF13 through PF24. The result is that F1 through F12 are equivalent to Shift + F1 through Sift + F F12. The right mouse button (RMB) is mapped to MB2. If you have only 10 function keys, F11 and F12 are unavailable and are not shown in the KEYS window. F13 through F24 are mapped to F1 through F12 if your keyboard has only 12 function keys instead of 24.

Following are the alternate mainframe key definitions:

| PC Key | Mainframe Definition | Key | Mainframe Definition |
|---|---|---|---|
| F1 | help | F7 | backward |
| F2 | zoom | F8 | forward |
| F3 | zoom off; submit | F9 | output |
| F4 | pgm; recall | F10 | left |
| F5 | rfind | F11 | right |
| F6 | rchange | F12 | home |
| | | RMB | zoom off; submit |

**SAA**

maps F1 through F12 to the IBM SAA values for CUAPF1 through CUAPF12 and Shift + F1 through Shift + F12 to CUAPF13 through CUAPF24. The right mouse button (RMB) is mapped to MB2. If you have only 10 function keys, F11, F12, Shift + F11, and Shift + F12 are unavailable and are not shown in the KEYS window.

*Note:* SAA stands for System Application Architecture, which is a framework for application development and is used across IBM systems. CUA (Common User Access) is a part of SAA that defines the user interface and components that should be identical across applications. △

Following are the IBM SAA key definitions:

| PC Key | Mainframe Definition | Key | Mainframe Definition |
|---|---|---|---|
| F1 | help | Shift + F1 | cut |
| F2 | keys | Shift + F2 | paste |
| F3 | zoom off; submit | Shift + F3 | store |
| F4 | home | Shift + F4 | mark |
| F5 | pgm; recall | Shift + F5 | unmark |
| F6 | zoom | Shift + F6 | smark |
| F7 | backward | Shift + F7 | left |
| F8 | forward | Shift + F8 | right |
| F9 | prevcmd | Shift + F9 | rfind |
| F10 | pmenu | Shift + F10 | rchange |
| F11 | command | Shift + F11 | undo |
| F12 | cancel | Shift + F12 | next |
| RMB | zoom off; submit | | |

**WIN**

specifies to use the default key definitions for SAS under Windows. WIN is the default.

### Details

Use the PFKEY system option when you do not want the default key definitions for SAS under Windows but instead want to use other key mappings (for example, the mappings used by SAS under z/OS).

Note that the function key values shown in the previous key map tables are for the Base SAS windows only. Other windowing SAS products, such as SAS/AF software, have other key definitions.

If you do not specify the PFKEY system option, or if you specify an invalid value, SAS loads the default Windows key definitions. For a list of key definitions, open the KEYS window by typing `keys` in the command bar..

## PRINT System Option

**Controls the creation of the SAS procedure output file for batch mode**

**Default:** *filename*.LST in batch mode, where *filename* is the name of your SAS job

**Valid in:** configuration file, SAS invocation

**Category:** Environment control: Files

**PROC OPTIONS GROUP=** ENVFILES

**Windows specifics:** all

### Syntax

-PRINT *destination* | -NOPRINT

**PRINT *destination***
specifies the destination for the SAS procedure output file. The *destination* argument can be a valid Windows pathname or filename (including device names) or an environment variable associated with a pathname. If you specify a pathname and it contains spaces, it must be enclosed in quotation marks. If you specify only a pathname, the procedure output file is created in the specified directory, with the default name of *filename*.LST, where *filename* is the name of your SAS job.

**NOPRINT**
suppresses the creation of the SAS procedure output file.

### Details

The PRINT system option specifies the destination to which SAS output is written when executing SAS programs in modes other than the interactive windowing environment.

The PRINT system option is valid only in batch mode.

When SAS is running interactively, the procedure output file is sent to the OUTPUT window; when SAS is running in batch mode, output is sent to a file named *filename*.LST, where *filename* is the name of your SAS job. You can use the PRINT option to specify an alternate destination.

## PRTABORTDLGS System Option

**Specifies when to display the Print Abort dialog box**

**Default** BOTH

**Valid in:** configuration file, SAS invocation, OPTIONS statement, SAS System Options window

**Category:** Environment control: Display

**PROC OPTIONS GROUP=** ENVDISPLAY

**Windows specifics:** all

### Syntax

-PRTABORTDLGS BOTH | NEITHER | FILE | PRINTER

PRTABORTDLGS = BOTH | NEITHER | FILE | PRINTER

**BOTH**
specifies to display the Print Abort dialog box when you are printing either to a file or to the printer.

**NEITHER**
specifies not to display the Print Abort dialog box when you are printing either to a file or to the printer.

**FILE**
specifies to display the Print Abort dialog box only when you are printing to a file.

**PRINTER**
specifies to display the Print Abort dialog box only when you are printing to the printer.

### Details

The Print Abort dialog box appears only while SAS is spooling a print job to its destination. Use the NEITHER value to suppress the Print Abort dialog box.

### See Also

□ "Canceling a Print Job" on page 179

## PRTPERSISTDEFAULT System Option

**Specifies to use the same destination printer from SAS session to SAS session**

**Default:** NOPRTPERSISTDEFAULT

**Valid in:** configuration file, SAS invocation

**Category:** Log and procedure output control: ODS printing

**PROC OPTIONS GROUP=** ODSPRINT

**Windows specifics:** all

### Syntax

-PRTPERSISTDEFAULT | -NOPRTPERSISTDEFAULT

**PRTPERSISTDEFAULT**
specifies to use the same destination printer from SAS session to SAS session.

**NOPRTPERSISTDEFAULT**
specifies to use the default printer.

### Details

Typically, when you start SAS, SAS sets the value of the SYSPRINT system option (which specifies the destination printer) to be the Windows default printer. When you start SAS by using the PRTPERSISTDEFAULT system option, SAS sets the value of the SYSPRINT system option to be the destination printer of the last SAS session that was started by using PRTPERSISTDEFAULT.

To use the same destination printer from SAS session to SAS session, you must use the PRTPERSISTDEFAULT system option each time that you start SAS. If you start SAS by using both the SYSPRINT system option and PRTPERSISTDEFAULT system option, the destination printer is the value that is specified by the SYSPRINT system option.

### See Also

- "SYSPRINT System Option" on page 566
- "Printing" on page 166

## PRTSETFORMS System Option

**Specifies whether to include the Use Forms check box in the Print Setup dialog box**

**Default:** PRTSETFORMS

**Valid in:** configuration file, SAS invocation, OPTIONS statement, SAS System Options window

**Category:** Environment control: Display

**PROC OPTIONS GROUP=** ENVDISPLAY

**Windows specifics:** all

### Syntax

-PRTSETFORMS | -NOPRTSETFORMS

PRTSETFORMS | NOPRTSETFORMS

**PRTSETFORMS**

specifies to include the Use Forms check box in the Print Setup dialog box.

**NOPRTSETFORMS**
specifies to exclude the Use Forms check box from the Print Setup dialog box.

### Details

Use the NOPRTSETFORMS system option to suppress the Use Forms check box in the Print Setup dialog box.

### See Also

- "Using SAS Print Forms" on page 175

## REALMEMSIZE System Option

**Indicates the amount of virtual memory SAS can expect to allocate**

**Default:** 0

**Valid in:** configuration file, SAS invocation

**Category:** System administration: Memory

**PROC OPTIONS GROUP=** MEMORY

**Windows specifics:** valid values

### Syntax

-REALMEMSIZE *n* | *n*K | *n*M | *n*G | *hex*X | MIN | MAX

***n* | *n*K | *n*M | *n*G**
specifies the amount of memory to reserve in multiples of 1; 1,024 (kilobytes); 1,048,576 (megabytes); and 1,073,741,824 (gigabytes), respectively. The value of *n* can be a decimal value. For example, a value of **`8`** specifies 8 bytes, a value of **`.782k`** specifies 801 bytes, and a value of **`3m`** specifies 3,145,728 bytes. Under 32-bit operating environments, the largest value that you can specify is 4294967295 (4G–1).

***hex*X**
specifies the amount of memory as a hexadecimal value. You must specify the value beginning with a number (0–9), followed by an X. For example, the value **`2dx`** sets the amount of memory to 45 bytes.

**MIN**
specifies a value of 0 which indicates that the memory usage is determined by SAS when SAS starts.

**MAX**
specifies to set the memory size to the largest permissible value.

### Details

Use the REALMEMSIZE system option to optimize the performance of SAS procedures that alter their algorithms and memory usage. Setting the value of REALMEMSIZE too low or too high may result in less than optimal performance.

## REGISTER System Option

**Adds an application to the Tools menu in the main SAS window**

**Default:** none

**Valid in:** configuration file, SAS invocation

**Category:** Environment control: Display

**PROC OPTIONS GROUP=** ENVDISPLAY

**Windows specifics:** all

### Syntax

-REGISTER *'menu-name'* *'command'* <*'working-directory'*>

***'menu-name'***
specifies the name you want to appear in the menu. The *menu-name* must be enclosed in quotation marks.

***'command'***
specifies the command you want to execute. The *command* argument can either be a .EXE, .COM, or .BAT file, or it can be an operating environment command such as the DIR command. The *command* must be enclosed in quotation marks.

***'working-directory'***
specifies the working directory to use for the application. This argument is optional. Read your application's documentation to see if the application requires a working directory specification. The *working-directory* must be enclosed in quotation marks.

### Details

You can add up to eight commands to the `Tools` pull-down menu in the main SAS window. If your menu name or command does not include blanks or special characters, you can omit the quotes. For more information about adding commands to the list, see "Adding Applications to the Tools Menu" on page 64.

## RESOURCESLOC System Option

**Specifies a directory location of the files that contain SAS resources**

**Default:** !*sasroot*\core\resource

**Valid in:** configuration file, SAS invocation

**Category:** Environment control: Display

**PROC OPTIONS GROUP=** ENVDISPLAY

**Windows specifics:** all

### Syntax

-RESOURCESLOC <(>*'directory-specification-1'* <*'directory-specification-n'*>)> | "."

***'directory-specification'***
specifies a directory location of the files that contain SAS resources. If *directory-specification* contains spaces, it must be enclosed in quotation marks. If you specify more than one *directory-specification*, enclose the list in parenthesis.

**"."**
specifies that the current working folder is to be the default directory for the location of the files that contain SAS resources.

### Details

SAS resources are dynamic link libraries that contain icons, strings, and fonts that are used by SAS. The types of files that reside in the RESOURCESLOC directory are font files (.fon, .ttf) and dynamic link libraries (.dll).

You can specify multiple RESOURCESLOC options to define a search order.

## RSASUSER System Option

**Controls whether members of the Sasuser data library can be opened for update or for read-only access**

**Default:** NORSASUSER

**Valid in:** configuration file, SAS invocation

**Category:** Environment control: Files

**PROC OPTIONS GROUP=** ENVFILES

**Windows specifics:** Network considerations

**See:** RSASUSER System Option in *SAS Language Reference: Dictionary*

### Syntax

-RSASUSER | -NORSASUSER

**RSASUSER**
limits access to the Sasuser data library to read-only access in environments where all users share the Sasuser library.

**NORSASUSER**
enables a user to open a file in the Sasuser library for update access, thus preventing users from sharing members of the Sasuser data library. Update access to the Sasuser library requires exclusive rights to the data library member. NORSASUSER is the default value.

### Details

Specifying RSASUSER enables a group of users to share Sasuser data library members by enabling all users to have read-only access to members. For example, if RSASUSER is in effect, each user can open the Sasuser.Profile catalog for read-only access, enabling other users to concurrently read from the Profile catalog. However, no user can write information out to the Profile catalog; you receive an error message if you try to do so.

Specifying RSASUSER in a SAS session affects only that session's access to files. To enable a group of users to share members in the Sasuser data library, the system administrator should set RSASUSER in the network version of the SAS configuration file, which is shared by all users who share the Sasuser data library.

If you specify RSASUSER but no Profile catalog exists in the Sasuser data library, the Profile catalog is created in the Work data library.

Whether the RSASUSER system option is useful depends on how SAS is being used. While the RSASUSER system option is extremely useful when users must share information (such as the Profile catalog) stored in the Sasuser data library, it is not useful if these same users are using SAS/ASSIST software. SAS/ASSIST software requires update access to the Sasuser data library.

## RTRACE System Option

**Generates a list of the file resources used in a given SAS session**

**Default:** NONE

**Valid in:** configuration file, SAS invocation

**Category:** Log and procedure output control: SAS log

**PROC OPTIONS GROUP=** LOGCONTROL

**Windows specifics:** all

### Syntax

-RTRACE ALL | NONE

**ALL**
specifies to list all the file resources used in a given SAS session.

**NONE**
specifies not to list the file resources.

### Details

Use the RTRACE and the RTRACELOC system options to create a file that lists the resources SAS uses.

### See Also

- "RTRACELOC System Option" on page 541

## RTRACELOC System Option

**Specifies the name of the file to which the file resource tracking system writes its output**

**Default:** none

**Valid in:** configuration file, SAS invocation

**Category:** Environment control: Files

**PROC OPTIONS GROUP=** ENVFILES

**Windows specifics:** all

### Syntax

-RTRACELOC *filename* | *pathname\filename*

***filename* | *pathname\filename***
: specifies a valid Windows filename or a pathname and a filename in which to store the file resource information. If the filename or the pathname contains spaces, enclose the name in quotation marks. If *pathname* is not specified, the file resource information is stored in the current directory.

### Details

You can use the RTRACELOC and the RTRACE system options to determine which resources SAS uses.

### See Also

- "RTRACE System Option" on page 541

## S System Option

**Specifies the length of statements on each line of source statements and the length of data on the line following a DATALINES statement**

**Default:** 0 (no length restrictions)

**Valid in:** configuration file, SAS invocation, OPTIONS statement, SAS System Options window

**Category:** Input control: Data processing

**PROC OPTIONS GROUP=** INPUTCONTROL

**Windows specifics:** Maximum line length

**See:** S System Option in *SAS Language Reference: Dictionary*

### Syntax

-S *n* | *n*K | *n*M | *n*G | *n*T |*hex*X | MIN | MAX

S=*n* | *n*K | *n*M | *n*G | *n*T | *hex*X | MIN | MAX

***n* | *n*K | *n*M | *n*G | *n*T**
: specifies the length of statements and data in multiples of 1; 1,024 (kilobytes); 1,048,576 (megabytes); 1,072,741,824 (gigabytes); and 1,099,511,627,776 (terabytes),

respectively. You can specify decimal values for the number of kilobytes, megabytes, gigabytes, or terabytes. For example, a value of `8` specifies 8 bytes, a value of `.782k` specifies 801 bytes, and a value of `3m` specifies 3,145,728 bytes.

***hex*X**
specifies the length of statements and data as a hexadecimal value. You must specify the value beginning with a number (0–9), followed by an X. For example, the value 2dx sets the length of statements and data to 45 bytes.

**MIN**
sets the length of statements and data to be 0, and requires SAS to use a default value.

**MAX**
sets the length of statements and data to the maximum, which under Windows is 2,147,483,647.

## Details

The S system option specifies the length of statements, exclusive of sequence numbers, on each line of SAS source statements and the length of data, exclusive of sequence numbers, on lines following a DATALINES statement.

The default value of 0 enables SAS to read a file with any line length up to MAX.

## See Also

- "S2 System Option" on page 543

## S2 System Option

**Specifies the length of secondary source statements**

**Default:** 0

**Valid in:** configuration file, SAS invocation, OPTIONS statement, SAS System Options window

**Category:** Input control: Data processing

**PROC OPTIONS GROUP=** INPUTCONTROL

**Windows specifics:** Valid values for *n*; syntax

**See:** S2 System Option in *SAS Language Reference: Dictionary*

### Syntax

-S2 S | *n* | *n*K | *n*M | *n*G | *hex*X | MIN | MAX

S2=S | *n* | *n*K | *n*M | *n*G | *hex*X | MIN | MAX

**S**
uses the current value of the S system option to compute the record length of text that comes from an %INCLUDE statement, an autoexec file, or an autocall macro file.

***n* | *n*K | *n*M | *n*G**

specifies the value by which to compute the record length of text that comes from an %INCLUDE statement, an autoexec file, or an autocall macro file. Specifies this value in multiples of 1; 1,024 (kilobytes); 1,048,576 (megabytes); and 1,073,741,824 (gigabytes), respectively. You can specify decimal values for the number of kilobytes, megabytes, or gigabytes. For example, a value of **`8`** specifies 8 bytes, a value of **`.782k`** specifies 801 bytes, and a value of **`3m`** specifies 3,145,728 bytes.

***hex*X**
specifies the value as a hexadecimal value. You must specify the value beginning with a number (0–9), followed by an X. For example, the value **`2dx`** specifies 45 bytes.

**MIN**
uses the value of 0, indicating no length restriction.

**MAX**
uses the value of 2,147,483,647.

## Details

The S2 system option operates exactly like the S system option, except that the S2 system option controls input from only an %INCLUDE statement, an autoexec file, or an autocall macro file.

## See Also

□ "S System Option" on page 542

## SASAUTOS System Option

**Specifies the autocall macro library**

**Default:** SASAUTOS

**Valid in:** configuration file, SAS invocation, OPTIONS statement, SAS System Options window

**Category:** Environment control: Files
Macro: SAS macro

**PROC OPTIONS GROUP=** ENVFILES
MACRO

**Windows specifics:** Valid values for *library-specification*

**See:** SASAUTOS System Option in *SAS Language Reference: Dictionary*

### Syntax

-SASAUTOS <(>"*library-specification-1*"...<"*library-specification-n*")>

SASAUTOS=<(>"*library-specification-1*"...<"*library-specification-n*")>

***"library-specification-1"... "library-specification-n"***
specifies one or more valid Windows pathnames or environment variables that are associated with pathnames. Remember that a pathname is only to the directory or subdirectory level. Windows pathnames must be enclosed in quotation marks if you

are using the OPTIONS statement or if the pathname contains spaces. If you specify only one library specification, the parentheses are optional. The value for *library-specification* must resolve to a valid Windows pathname.

### Details

The SASAUTOS system option specifies the SAS autocall macro library or libraries.

### See Also

- "SASAUTOS System Option" on page 584
- *SAS Macro Language: Reference*

## SASCONTROL System Option

**Specifies whether the SAS application windows include system/control menus and minimize/ maximize buttons**

**Default:** SYSTEMMENU MINMAX

**Valid in:** configuration file, SAS invocation, OPTIONS statement, SAS System Options window

**Category:** Environment control: Display

**PROC OPTIONS GROUP=** ENVDISPLAY

**Windows specifics:** all

### Syntax

-SASCONTROL SYSTEMMENU | NOSYSTEMMENU <MINMAX | NOMINMAX>

-SASCONTROL <SYSTEMMENU | NOSYSTEMMENU> MINMAX | NOMINMAX

SASCONTROL=SYSTEMMENU | NOSYSTEMMENU <MINMAX | NOMINMAX>

SASCONTROL=<SYSTEMMENU | NOSYSTEMMENU> MINMAX | NOMINMAX

**SYSTEMMENU**
specifies to display the system/control menu in the windows that are contained in the main SAS window.

**NOSYSTEMMENU**
specifies to omit the system/control menu and the minimize, maximize, and close buttons from the title bar in the windows that are contained in the main SAS window.

**MINMAX**
specifies to display the minimize and maximize buttons in the windows that are contained in the main SAS window.

**NOMINMAX**
specifies to omit the minimize and maximize buttons from the windows that are contained in the main SAS window.

### Details

The SASCONTROL system option affects the windows contained inside the main SAS window, but not the main SAS window itself (which is controlled by the AWSCONTROL system option).

The SASCONTROL system option is intended for use by SAS/AF programmers to customize the interface of their applications.

### See Also

- "AWSCONTROL System Option" on page 486

## SASHELP System Option

**Specifies the directory or directories to be searched for SAS default forms, device lists, dictionaries, and other entries in the Sashelp catalog**

**Default:** !*sasroot*\*SAS product*\\sashelp, !*sascfg*\\sascfg

**Valid in:** configuration file, SAS invocation

**Category:** Environment control: Files

**PROC OPTIONS GROUP=** ENVFILES

**Windows specifics:** Valid values for *library-specification*

**See:** SASHELP System Option in *SAS Language Reference: Dictionary*

### Syntax

-SASHELP ("*library-specification-1*"...<"*library-specification-n*")>

***"library-specification-1"... "library-specification-n"***
specifies one or more valid Windows pathnames or environment variables that are associated with pathnames. Remember that a pathname applies only to the directory or subdirectory level. The value for *library-specification* must resolve to a valid Windows pathname. If the pathname contains spaces, it must be enclosed in quotation marks.

### Details

The SASHELP system option is set during the installation process and normally is not changed after installation.

Note that products and their corresponding files can be split across multiple drives and directories. The *library-specification* argument can be a Windows pathname or an environment variable associated with a pathname.

## SASINITIALFOLDER System Option

**Changes the working folder and the default folders for the Open and Save As dialog boxes to the specified folder after SAS initialization is complete**

**Default:** none

**Valid in:** configuration file, SAS invocation

**Category:** Environment control: Files

**PROC OPTIONS GROUP=** ENVFILES

**Windows specifics:** all

## Syntax

-SASINITIALFOLDER *newfolder*

***newfolder***
specifies the path to the current working folder and the default folders for the Open and Save As dialog boxes. If *newfolder* contains spaces, it must be enclosed in quotation marks.

## Details

SAS determines the locations for AUTOEXEC or INITSTMT files before the SASINITIALFOLDER system option is processed. To ensure that SAS can determine the location of these files, place them in a folder other than the folder that is specified by the SASINITIALFOLDER system option.

If you do not specify the SASINITIALFOLDER system option, the current working folder and the default folders for the Open and Save As dialog boxes are set to the Sasuser folder.

## See Also

- "Changing the SAS Current Folder" on page 37

## SASUSER System Option

**Specifies the name of the Sasuser library**

**Default:** `c:\WINNT\Profiles\`*`username`*`\Personal\` for Windows NT

`c:\Documents and Settings\`*`username`*`\My Documents\` for Windows 2000, Windows XP, and Windows Server 2003

**Valid in:** configuration file, SAS invocation

**Category:** Environment control: Files

**PROC OPTIONS GROUP=** ENVFILES

**Windows specifics:** Valid values for *library-specification*; syntax

## Syntax

-SASUSER ("*library-specification-1*"...<"*library-specification-n*">)

***"library-specification-1"... "library-specification-n"***

specifies one or more valid Windows pathnames or environment variables that are associated with pathnames for a SAS data library. Remember that a pathname applies only to the directory or subdirectory level. If you list only one library specification, the parentheses are optional. The value for *library-specification* must resolve to a valid Windows pathname.

### Details

The SASUSER system option specifies the SAS data library that contains a user's profile catalog. The default value for SASUSER is defined in the SAS configuration file, which you can change when you install SAS. If you do not use the SASUSER system option when you invoke SAS (either in the configuration file or as part of the SAS command), the Sasuser data library is set to be equal to the Work data library, which is temporary.

### See Also

- "Profile Catalog" on page 19
- "Using the Sasuser Data Library" on page 133

## SCROLLBARFLASH System Option

**Specifies whether to allow the mouse or keyboard to focus on a scroll bar**

**Default:** NOSCROLLBARFLASH

**Valid in:** configuration file, SAS invocation, OPTIONS statement, SAS System Options window

**Category:** Environment control: Display

**PROC OPTIONS GROUP=** ENVDISPLAY

**Windows specifics:** all

### Syntax

-SCROLLBARFLASH | -NOSCROLLBARFLASH

SCROLLBARFLASH | NOSCROLLBARFLASH

**SCROLLBARFLASH**
specifies to enable mouse and keyboard focus on the scroll bars.

**NOSCROLLBARFLASH**
specifies to disable mouse and keyboard focus on the scroll bars.

### Details

Under certain conditions, the cursor may flash if you select a scroll bar using the mouse or the keyboard. You can turn off the flashing cursor using the NOSCROLLBARFLASH system option. You can also use the Preferences dialog box **`Advanced`** page to disable the flashing cursor by selecting **`Disable scroll bar focus`**.

### See Also

□ "Setting Session Preferences" on page 57

## SET System Option

**Defines a SAS environment variable**

**Default:** none

**Valid in:** configuration file, SAS invocation, OPTIONS statement, SAS System Options window

**Category:** Environment control: Files

**PROC OPTIONS GROUP=** ENVFILES

**Windows specifics:** Values intended to represent files or paths must be valid under Windows

### Syntax

-SET *SAS-variable* "*value*" | ("*value-1*"...<"*value-n*">)

SET=*SAS-variable* "*value*" | (" *value-1*"...<"*value-n*">)

***SAS-variable***
specifies the environment variable to define.

***value***
specifies the value or set of values to assign to the environment variable. If *value* is a pathname that contain spaces, enclose *value* in quotation marks.

### Details

This is analogous to defining a Windows environment variable with the Windows SET command. One way to use the SET system option is to set up environment variables that represent commonly used external files. For example, the following code defines an environment variable for the sample source library:

```
-set sampsrc (!sasroot\base\sample
             !sasroot\stat\sample
             !sasroot\graph\sample)
```

When you refer to SAMPSRC as a library name during your SAS session, SAS automatically assigns the library with the directories listed. Note that !*sasroot* is also a SAS environment variable that represents the root directory of your SAS installation, and is typically assigned in the SAS configuration file.

Environment variables can only be used as a libref if you use the SET system option at SAS invocation and not in an OPTIONS statement.

If you specify SET on the command line when you start SAS, the variable will be set only for that SAS session. To set an environment variable for repeated use, either add the SET system option to your configuration file or create a Windows environment variable.

*Note:* The words AUX, CON, NUL, LPT1 - LPT9, COM1 - COM9, and PRN are reserved words under Windows. Do not use CON or NUL as environment variable names. △

### See Also

- □ "Assigning SAS Libraries Using Environment Variables" on page 128
- □ "Using Environment Variables" on page 148

## SGIO System Option

**Activates the Scatter/Gather I/O feature**

**Default:** NOSGIO

**Valid in:** configuration file, SAS invocation

**Category:** Files: SAS Files

**PROC OPTIONS GROUP=** SASFILES

**Windows specifics:** all

### Syntax

-SGIO | -NOSGIO

**SGIO**
specifies to activate the scatter–read / gather–write feature. The scatter–read / gather–write feature remains active until your SAS session ends.

**NOSGIO**
specifies not to activate the scatter-read/gather-write feature.

### Details

The SGIO system option greatly improves I/O performance for SAS I/O files (data sets, catalogs, indexes, utility files, and other I/O files) when the PC has a large amount of RAM. Scatter-read / gather-write bypasses intermediate buffer transfers between memory and disk.

When SGIO is active, SAS uses the number of buffers that are specified by the BUFNO system option to transfer data between disk and RAM. I/O performance usually improves as the value for the BUFNO increases. Try different values of the BUFNO system option to tune each SAS job or DATA step.

The scatter-read / gather-write feature is active only for SAS I/O files that

- □ contain a 4K-multiple pagesize (for example, 4096 or 8192) on 32–bit systems
- □ contain a 8K-multiple pagesize (for example, 8192 or 16384) on 64-bit systems
- □ were not created by using Version 6 of SAS
- □ are accessed sequentially.

If an I/O file does not meet these criteria, SGIO is inactive for that file even though the SGIO option is specified.

To use the SGIO system option on Windows NT, you must install Service Pack 4.

### See Also

- □ "BUFNO System Option" on page 490
- □ "SAS Features That Optimize Performance" on page 205

## SLEEPWINDOW System Option

**Enables or disables the SLEEP window**

**Default:** SLEEPWINDOW

**Valid in:** configuration file, SAS invocation

**Category:** Environment control: Display

**PROC OPTIONS GROUP=** ENVDISPLAY

**Windows specifics:** all

### Syntax

-SLEEPWINDOW | -NOSLEEPWINDOW

**SLEEPWINDOW**
specifies to display the SLEEP window.

**NOSLEEPWINDOW**
specifies not to display the SLEEP window.

### Details

The SLEEP window appears when the SLEEP function or the WAKEUP function suspends the execution of a DATA step. The SLEEP window displays the time that remains before the DATA step begins running.

### See Also

- □ "SLEEP Function" on page 409
- □ "WAKEUP Function" on page 411

## SORTANOM System Option

**Specifies certain options for the SyncSort utility**

**Default:** none

**Valid in:** configuration file, SAS invocation, OPTIONS statement, SAS System Options window

**Category:** Sort: Procedure options

**PROC OPTIONS GROUP=** SORT

**Windows specifics:** all

### Syntax

-SORTANOM *option(s)*

SORTANOM=*option(s)*

***option(s)***
can be one or more of the following:

b
tells SyncSort to run in multi-call mode instead of single-call mode.

t
prints statistics about the sorting process in the SAS log.

v
prints all of the commands that are passed to the SyncSort utility in the SAS log.

### See Also

- "Passing Options to SyncSort" on page 438
- The documentation for SyncSort for Windows

## SORTCUT System Option

**Specifies the number of observations above which SyncSort is used instead of the SAS sort program**

**Default:** 0

**Valid in:** configuration file, SAS invocation, OPTIONS statement, SAS System Options window

**Category:** Sort: Procedure options

**PROC OPTIONS GROUP=** SORT

**Windows specifics:** all

### Syntax

-SORTCUT *n* | *n*K | *n*M | *n*G | *hex*X | MIN | MAX

SORTCUT=*n* | *n*K | *n*M | *n*G | *hex*X | MIN | MAX

***n* | *n*K | *n*M | *n*G**
specifies the number of observations in multiples of 1 (*n*); 1,024 (*n*K); 1,048,576 (*n*M); and 1,073,741,824 (*n*G), respectively. You can specify decimal values for *n* when it is used to specify a K, M, or G value. For example, a value of `8` specifies 8 observations, a value of `.782k` specifies 801 observations, and a value of `3m` specifies 3,145,728 observations.

***hex*X**

specifies the number of observations as a hexadecimal value. You must specify the value beginning with a number (0–9), followed by an X. For example, the value **`2dx`** specifies 45 observations.

**MIN**
specifies 0 observations.

**MAX**
specifies 2,147,483,647 observations.

### Details

When you specify SORTPGM=BEST and SAS determines that the database sort utility is not to be used, SAS uses the value of the SORTCUT and SORTCUTP options to determine whether to use SyncSort or the SAS sort. If the data set to be sorted is larger than the number of bytes (or kilobytes or megabytes) that you specify with SORTCUTP, SyncSort is used instead of the SAS sort program. The value that you specify must be less than or equal to 2,147,483,647 bytes. If both SORTCUT and SORTCUTP are either not defined or are set to 0, the SAS sort is used. If you specify both options and either condition is true, SAS uses SyncSort.

### See Also

- □ "SORTCUTP System Option" on page 553
- □ "SORTPGM System Option" on page 555
- □ "Sorting Based on Size or Observations" on page 437

## SORTCUTP System Option

**Specifies the number of bytes above which SyncSort is used instead of the SAS sort program**

**Default:** 0

**Valid in:** configuration file, SAS invocation, OPTIONS statement, SAS System Options window

**Category:** Sort: Procedure options

**PROC OPTIONS GROUP=** SORT

**Windows specifics:** all

### Syntax

-SORTCUTP *n* | *n*K | *n*M | *n*G | *hex*X | MIN | MAX

SORTCUTP=*n* | *n*K | *n*M | *n*G | *hex*X | MIN | MAX

***n* | *n*K | *n*M | *n*G**
specifies the number of bytes in multiples of 1; 1,024 (kilobytes); 1,048,576 (megabytes); and 1,073,741,824 (gigabytes), respectively. You can specify decimal values for the number of kilobytes, megabytes, or gigabytes. For example, a value of **`8`** specifies 8 bytes, a value of **`.782k`** specifies 801 bytes, and a value of **`3m`** specifies 3,145,728 bytes.

***hex*X**
specifies the number of bytes as a hexadecimal value. You must specify the value beginning with a number (0–9), followed by an X. For example, the value **`2dx`** specifies 45 bytes.

**MIN**
specifies 0 bytes.

**MAX**
specifies 2,147,483,647 bytes.

### Details

When you specify SORTPGM=BEST and SAS determines that the database sort utility is not to be used, SAS uses the value of the SORTCUTP and SORTCUT options to determine whether to use SyncSort or the SAS sort. If the data set to be sorted is larger than the number of bytes (or kilobytes or megabytes) that you specify with SORTCUTP, SyncSort is used instead of the SAS sort program. The value that you specify must be less than or equal to 2,147,483,647 bytes. If both SORTCUTP and SORTCUT are either not defined or are set to 0, the SAS sort is used. If you specify both options and either condition is true, SAS uses SyncSort.

The following equation computes the number of bytes to be sorted:
*number of bytes= ((length-of-obs) + (length-of-all-keys)) * number-of-obs*

### See Also

- “SORTPGM System Option” on page 555
- “SORTCUT System Option” on page 552
- “Sorting Based on Size or Observations” on page 437

## SORTDEV System Option

**Specifies the pathname used for temporary files created by the SyncSort utility**

**Default:** same location as -WORK, which is set in the configuration file

**Valid in:** configuration file, SAS invocation, OPTIONS statement, SAS System Options window

**Category:** Sort: Procedure options

**PROC OPTIONS GROUP=** SORT

**Windows specifics:** all

### Syntax

-SORTDEV "*pathname*"

SORTDEV ="*pathname*"

**"*pathname*"**
specifies a valid Windows pathname.

### Details

The SORTDEV option specifies an alternative pathname for temporary files created by the SyncSort utility. The pathname must be enclosed in quotation marks.

### See Also

- □ "WORK System Option" on page 574
- □ "Passing Parameters to SyncSort" on page 438

## SORTPARM System Option

**Specifies parameters for the SyncSort utility**

**Default:** none

**Valid in:** configuration file, SAS invocation, OPTIONS statement, SAS System Options window

**Category:** Sort: Procedure options

**PROC OPTIONS GROUP=** SORT

**Windows specifics:** all

### Syntax

SORTPARM="*SyncSort-parameters*"

-SORTPARM "*SyncSort-parameters*"

***SyncSort-parameters***
specifies any parameters that you want to pass to the SyncSort utility. Enclose *SyncSort- parameters* in quotation marks.

### Details

See the SyncSort for Windows documentation for a description of *SyncSort- parameters*.

## SORTPGM System Option

**Specifies the sort utility that is used in the SORT procedure**

**Default:** BEST

**Valid in:** configuration file, SAS invocation, OPTIONS statement, SAS System Options window

**Category:** Sort: Procedure options

**PROC OPTIONS GROUP=** SORT

**Windows specifics:** all

## Syntax

-SORTPGM SAS | BEST | HOST

SORTPGM = SAS | BEST | HOST

**SAS**
tells SAS to sort by using the SAS sort routine.

**BEST**
tells SAS to determine the best sort routine to sort the data: a database sort, the SAS sort, or SyncSort. When SAS determines that the sort is not to be done by the database, SAS looks at the values for both SORTCUT and SORTCUTP. If they both are set to zero, the SAS sort is used. If both options are set and either condition is met, SAS uses the SyncSort routine.

**HOST**
tells SAS to sort by using SyncSort for Windows.

## See Also

- "SORT Procedure" on page 435
- "SORTCUT System Option" on page 552
- "SORTCUTP System Option" on page 553

## SORTSIZE System Option

**Limits the amount of memory available to the SORT procedure**

**Default:** MAX

**Valid in:** configuration file, SAS invocation, OPTIONS statement, SAS System Options window

**Category:** Sort: Procedure options
System administration: Memory

**PROC OPTIONS GROUP=** MEMORY
SORT

**Windows specifics:** Default value

**See:** SORTSIZE System Option in *SAS Language Reference: Dictionary*

## Syntax

-SORTSIZE *n* | *n*K | *n*M | *n*G | *hex*X | MIN | MAX

SORTSIZE= *n* | *n*K | *n*M | *n*G | *hex*X | MIN | MAX

***n* | *n*K | *n*M | *n*G**

specifies the amount of memory in multiples of 1; 1,024 (kilobytes); 1,048,576 (megabytes); and 1,073,741,824 (gigabytes) respectively. You can specify decimal values for the number of kilobytes, megabytes, or gigabytes. For example, a value of `8` specifies 8 bytes, a value of `.782k` specifies 801 bytes, and a value of `3m` specifies 3,145,728 bytes.

***hex*X**
specifies the amount of memory as a hexadecimal value. You must specify the value beginning with a number (0–9), followed by an X. For example, the value 2dx sets the amount of memory to 45 bytes.

**MIN**
specifies the minimum amount of memory available.

**MAX**
specifies the maximum amount of memory available.

### Details

By default, this option is set to the maximum amount of memory available. The SORTSIZE system option can reduce the amount of swapping SAS must do to sort the data set. If PROC SORT needs more memory than you specify, it creates a temporary utility file in your Saswork directory in which to store the data. The SORT procedure's algorithm can swap unneeded data more efficiently than Windows can.

### See Also

- "SORT Procedure" on page 435
- "Improving Performance of the SORT Procedure" on page 206

## SPLASH System Option

**Specifies whether to display the splash screen (logo screen) when SAS starts**

**Default:** SPLASH

**Valid in:** configuration file, SAS invocation

**Category:** Environment control: Display

**PROC OPTIONS GROUP=** ENVDISPLAY

**Windows specifics:** all

### Syntax

-SPLASH | -NOSPLASH

-SPLASH ON | -SPLASH OFF

**SPLASH or SPLASH ON**
specifies to display the logo screen when SAS initiates.

**NOSPLASH or SPLASH OFF**
specifies to not display the logo screen when SAS initiates.

### Details

The SPLASH system option displays the SAS logo screen when SAS initiates.
You can specify a custom splash screen to display with the SPLASHLOC system option.

### See Also

- "SPLASHLOC System Option" on page 558

## SPLASHLOC System Option

**Specifies the location of the splash screen bitmap that appears when SAS starts**

**Default:** none

**Valid in:** configuration file, SAS invocation

**Category:** Environment control: Display

**PROC OPTIONS GROUP=** ENVDISPLAY

**Windows specifics:** all

### Syntax

-SPLASHLOC *DLL-name <res-number>* | *BMP-filename*

***DLL-name***
specifies the dynamic link library (DLL) where your customized logo and copyright screen reside.

***res-number***
specifies the resource number connected with the dynamic link library (DLL) name.

***BMP-filename***
specifies the path and name of a stand-alone Windows bitmap (BMP) file to use as a splash screen.

### Details

You can create a bitmap resource (a customized logo and copyright screen) and build it into a dynamic link library (DLL). The DLL that you use must be 32-bit if you are running a 32–bit version of SAS or it must be 64–bit if you are running a 64–bit version of SAS (that is, created using the libraries from the Microsoft Platform SDK). If you specify a *DLL-name* without a resource number (*res-number*), the default resource number is 1.

Alternatively, you can specify the path and name of a stand-alone Windows bitmap (BMP) file to use as a splash screen. The path must be a valid Windows pathname. If the pathname contains spaces, it must be enclosed in quotation marks.

## SSLCERTISS System Option

**Specifies the name of the issuer of the digital certificate that Secure Sockets Layer (SSL) should use**

**Default:** none

**Valid in:** configuration file, SAS invocation, OPTIONS statement, SAS System Options window

**Category:** Communications: Networking and encryption

**PROC OPTIONS Group=** COMMUNICATIONS

**Windows specifics:** all

### Syntax

SSLCERTISS "*issuer-of-digital-certificate*"

***"issuer-of-digital-certificate"***
specifies the name of the issuer of the digital certificate that should be used by SSL.

### Details

The SSLCERTISS option is used with the SSLCERTSERIAL option to uniquely identify a digital certificate from the Microsoft certificate store.

### See Also

- "SSLCERTSERIAL System Option" on page 559
- "SSLCERTSUBJ System Option" on page 560
- "SSLCLIENTAUTH System Option" on page 561
- "SSLCRLCHECK System Option" on page 562
- Appendix 3, "Using SSL under Windows," on page 605
- *SAS/CONNECT User's Guide*

## SSLCERTSERIAL System Option

**Specifies the serial number of the digital certificate that Secure Sockets Layer (SSL) should use**

**Default:** none

**Valid in:** configuration file, SAS invocation, OPTIONS statement, SAS System Options window

**Category:** Communications: Networking and encryption

**PROC OPTIONS Group=** COMMUNICATIONS

**Windows specifics:** all

### Syntax

SSLCERTSERIAL “*serial-number*”

***“serial-number”***
specifies the serial number of the digital certificate that should be used by SSL.

### Details

The SSLCERTSERIAL options is used with SSLCERTISS option to uniquely identify a digital certificate from the Microsoft certificate store.

### See Also

- “SSLCERTISS System Option” on page 559
- “SSLCERTSUBJ System Option” on page 560
- “SSLCLIENTAUTH System Option” on page 561
- “SSLCRLCHECK System Option” on page 562
- Appendix 3, “Using SSL under Windows,” on page 605
- *SAS/CONNECT User’s Guide*

## SSLCERTSUBJ System Option

**Specifies the subject name of the digital certificate that SSL should use**

**Default:** none

**Valid in:** configuration file, SAS invocation, OPTIONS statement, SAS System Options window

**Category:** Communications: Networking and encryption

**PROC OPTIONS Group=** COMMUNICATIONS

**Windows specifics:** all

### Syntax

SSLCERTSUBJ “*subject-name*”

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

### Details

The subject name of the digital certificate is used to search for a digital certificate from the Microsoft certificate store.

### See Also

- □ "SSLCERTISS System Option" on page 559
- □ "SSLCERTSERIAL System Option" on page 559
- □ "SSLCLIENTAUTH System Option" on page 561
- □ "SSLCRLCHECK System Option" on page 562
- □ Appendix 3, "Using SSL under Windows," on page 605
- □ *SAS/CONNECT User's Guide*

## SSLCLIENTAUTH System Option

**Specifies whether a server should perform client authentication**

**Default:** NOSSLCLIENTAUTH

**Valid in:** configuration file, SAS invocation, OPTIONS statement, SAS System Options window

**Category:** Communications: Networking and encryption

**PROC OPTIONS GROUP=** COMMUNICATIONS

**Windows specifics:** all

### Syntax

-SSLCLIENTAUTH | -NOSSLCLIENTAUTH

SSLCLIENTAUTH | NOSSLCLIENTAUTH

**SSLCLIENTAUTH**
specifies that the server should require SSL to provide client authentication.

**NOSSLCLIENTAUTH**
specifies that the server should not require SSL to provide client authentication.

### Details

Server authentication is always performed, but the SSLCLIENTAUTH option enables a user to control client authentication. This option is meaningful only when used on a server.

### See Also

- □ "SSLCERTISS System Option" on page 559
- □ "SSLCERTSERIAL System Option" on page 559
- □ "SSLCERTSUBJ System Option" on page 560
- □ "SSLCRLCHECK System Option" on page 562
- □ Appendix 3, "Using SSL under Windows," on page 605
- □ *SAS/CONNECT User's Guide*

## SSLCRLCHECK System Option

**Specifies whether Certificate Revocation Lists (CRLs) are checked when a digital certificate is validated**

**Default:** NOSSLCRLCHECK

**Valid in:** configuration file, SAS invocation, OPTIONS statement, SAS System Options window

**Category:** Communications: Networking and encryption

**PROC OPTIONS Group=** COMMUNICATIONS

**Windows specifics:** all

### Syntax

-SSLCRLCHECK | -NOSSLCRLCHECK

SSLCRLCHECK | NOSSLCRLCHECK

**SSLCRLCHECK | NOSSLCRLCHECK**
specifies that Certificate Revocation Lists are checked when digital certificates are validated.

**NOSSLCRLCHECK**
specifies that Certificate Revocation Lists are not checked when digital certificates are validated.

### Details

Certificate Revocation Lists (CRLs) are published by Certificate Authorities (CAs) and contain a list of revoked digital certificates. The list contains only the revoked digital certificates that were issued by that particular certificate authority. This option is relevant for servers only if client authentication is used. Because clients always check server digital certificates, this option is always relevant for clients.

### See Also

- “SSLCERTISS System Option” on page 559
- “SSLCERTSERIAL System Option” on page 559
- “SSLCERTSUBJ System Option” on page 560
- “SSLCLIENTAUTH System Option” on page 561
- Appendix 3, “Using SSL under Windows,” on page 605
- *SAS/CONNECT User’s Guide*

## STIMEFMT System Option

**Specifies the format to use for displaying the time on STIMER output**

**Default:** M

**Valid in:** configuration file, SAS invocation, OPTIONS statement, SAS System Options window

**Category:** Log and procedure output control: SAS log

**PROC OPTIONS GROUP=** LOGCONTROL

**Windows specifics:** all

## Syntax

-STIMEFMT S | M | H | SECONDS | MINUTES | HOURS

STIMEFMT=S | M | H | SECONDS | MINUTES | HOURS

**S, SECONDS**
specifies that SAS software display the STIMER output as **`seconds`**.

**M, MINUTES**
specifies that SAS software display the STIMER output as **`minutes:seconds`**

**H, HOURS**
specifies that SAS software display the STIMER output as **`hours:minutes:seconds`**.

## Details

The STIMEFMT system option specifies the format to use to display STIMER output as either **`seconds`**, **`minutes:seconds`**, or **`hours:minutes:seconds`**.

## STIMER System Option

**Specifies whether to display time-elapsed statistics after each DATA step and procedure**

**Default:** STIMER

**Valid in:** configuration file, SAS invocation, OPTIONS statement, SAS System Options window

**Category:** Log and procedure output control: SAS log

**PROC OPTIONS GROUP=** LOGCONTROL

**Windows specifics:** Reported statistics

### Syntax

-STIMER | -NOSTIMER

STIMER | NOSTIMER

**STIMER**
specifies to write the statistics. When STIMER is in effect, SAS writes to the SAS log a list of computer resources used for each step and the entire SAS session.

**NOSTIMER**

specifies not to write performance statistics to the SAS log.

### Details

The STIMER system option prints to the SAS log the amount of time it took for SAS to complete a DATA step or procedure task.

### Comparisons

The STIMER system option specifies whether a subset of all the performance statistics of your operating environment that are available to SAS are written to the SAS log. The FULLSTIMER system option specifies whether all of the available performance statistics are written to the SAS log.

### See Also

- “FULLSTIMER System Option” on page 505
- The chapter on optimizing system performance in *SAS Language Reference: Concepts*.

## SYSGUIFONT System Option

**Specifies a font to use for the button text and the descriptive text**

**Default:** depends upon display settings

**Valid in:** configuration file, SAS invocation

**Category:** Environment control: Display

**PROC OPTIONS GROUP=** ENVDISPLAY

**Windows specifics:** all

### Syntax

-SYSGUIFONT "*font-name*" <*font-size*>

**"*font-name*"**
specifies the name of the font for text in screen and dialog box text elements. This must be a valid font name (for example, “Times New Roman” or “Courier”) that matches the name of the font as it is installed on your system. The *font-name* must be enclosed in double quotation marks. This is a required argument.

***font-size***
specifies the font size to use for the window text. If you omit *font-size*, SAS uses the default.

### Details

The SYSGUIFONT system option controls the font size of the text for screen text and dialog box text elements. Use the FONT system option to change the fonts for the window contents. You might need to maximize the SAS window in order to allow space for large fonts to be readable.

### See Also

- "FONT System Option" on page 502
- "Selecting Fonts" on page 57

## SYSIN System Option

**Specifies a batch mode source file**

**Default:** none

**Valid in:** configuration file, SAS invocation

**Category:** Environment control: Files

**PROC OPTIONS GROUP=** ENVFILES

**Windows specifics:** Valid values for *file-specification*

### Syntax

-SYSIN *file-specification* | -NOSYSIN

**SYSIN *file-specification***
specifies to start SAS and submit the file in batch mode. The value of *file-specification* must be a valid Windows filename.

**NOSYSIN**
specifies to start SAS in batch mode, but do not submit any files. This is useful for testing your SAS autoexec file; after your autoexec file is processed, SAS exits.

### Details

The SYSIN system option specifies a file containing a SAS program. This option indicates to SAS that you are executing in noninteractive mode and can be specified only in the SAS invocation.

## SYSPARM System Option

**Specifies a character string that can be passed to SAS programs**

**Default:** none

**Valid in:** configuration file, SAS invocation, OPTIONS statement, SAS System Options window

**Category:** Environment control: Files

**PROC OPTIONS GROUP=** ENVFILES

**Windows specifics:** Valid values and syntax for *characters*

**See:** SYSPARM System Option in *SAS Language Reference: Dictionary*

### Syntax

-SYSPARM <">*characters*<">

SYSPARM=<">*characters*<">

***characters***
writes the character string in all uppercase.

***"characters"***
preserves the case of the character string.

### Details

The SYSPARM system option specifies a character string that can be passed to SAS programs.

The character string specified can be accessed in a SAS DATA step by the SYSPARM() function or anywhere in a SAS program by using the automatic macro variable referenced by &SYSPARM.

## SYSPRINT System Option

**Specifies a destination printer for printing SAS output**

**Default:** Default system printer

**Valid in:** configuration file, SAS invocation, OPTIONS statement, SAS System Options window

**Category:** Log and procedure output control: Procedure output

**PROC OPTIONS GROUP=** LISTCONTROL

**Windows specifics:** all

### Syntax

-SYSPRINT *"printer-name"*<*"destination"*>

SYSPRINT=*"printer-name"*<*"destination"*>

***"printer-name"***
specifies the name of the printer as it is installed under Windows (for example, "Charlie's HP LaserJet"). You can find the list of installed printers on your system by selecting the Printers item in the Windows Control Panel. The *printer-name* must be enclosed in double quotation marks.

***"destination"***
optionally specifies a file name to write the print file to disk. If specified, then all printer output generated by SAS is routed to this file, overwriting any existing file with the same name. Even though the output is not sent directly to a printer, it is

still formatted using the printer driver associated with *printer-name*. The *destination* must be enclosed in double quotation marks.

### Details

The SYSPRINT system option specifies the destination of a printer where you want to print your SAS output.

If you select a different printer by using the Print Setup dialog box, the value of the SYSPRINT system option (shown by PROC OPTIONS) reflects that selection.

If you do not specify the SYSPRINT system option or the PRTPERSISTDEFAULT system option, the *printer-name* and *destination* arguments use the default system printer value.

If PRTPERSISTDEFAULT is specified when SAS starts, the value of SYSPRINT persists from SAS session to SAS session. If both SYSPRINT and PRTPERSISTDEFAULT are specified when SAS starts, the value of SYSPRINT is the printer specified by SYSPRINT.

***CAUTION:***
**Modifying print options by using the Windows printing dialog boxes may change the values of SAS printing system options** If you set printing options using SAS system options such as SYSPRINT, and then use the Windows printing dialogs to set printing options, the SAS system options are set to the values specified in the Windows print dialog boxes. △

### See Also

- "Printing" on page 166
- "PRTPERSISTDEFAULT System Option" on page 536
- "SYSPRINTFONT System Option" on page 567

## SYSPRINTFONT System Option

**Sets the font to use when SAS is printing to the current default printer**

**Default:** none

**Valid in:** configuration file, SAS invocation, OPTIONS statement, SAS System Options window

**Category:** Log and procedure output control: Procedure output

**PROC OPTIONS GROUP=** LISTCONTROL

**Windows specifics:** all

**See:** SYSPRINTFONT System Option in *SAS Language Reference: Dictionary*

### Syntax

-SYSPRINTFONT ("*font-name*" <BOLD | NORMAL><REGULAR | ITALIC><*character-set*> <*point-size*> <NAMED "*printer-name*" | DEFAULT | ALL>)

SYSPRINTFONT="*font-name*" <BOLD | NORMAL><REGULAR | ITALIC><*character-set*> <*point-size*> <NAMED "*printer-name*" | DEFAULT | ALL>

***"font-name"***
specifies the name of the font to use for printing. This must be a valid font name (for example, "SAS Monospace" or "Courier") that matches the name of the font as it is installed on your system. The *font-name* must be enclosed in double quotation marks. This is a required argument.

**BOLD | NORMAL**
specifies the weight of the font. The default is NORMAL.

**REGULAR | ITALIC**
specifies the style of the font. The default is REGULAR.

***character-set***
specifies the character set to use for printing. The default is "Windows". Valid values are Western, Central European, Cyrillic, Greek, Turkish, Arabic, Baltic, and Thai. If the font does not support the specified character set, the default character set is used. If the default character set is not supported by the font, the font's default character set is used.

***point-size***
specifies the point size to use for printing. This must be an integer from 1 to 7200, inclusive. If you omit this argument, SAS uses 10 points.

**NAMED *"printer-name"***
updates the font information for the named printer in the Sasuser.Profile2 catalog. The printer name must exactly match the name shown in the Print Setup dialog box (except that the printer name is not case sensitive). The *printer-name* must be enclosed in double quotation marks. This keyword is optional.

**DEFAULT**
specifies the default font information for the printer used by the -SYSPRINT system option in the Sasuser.Profile2 catalog.

**ALL**
updates the font information for all installed printers in the Sasuser.Profile2 catalog. This keyword is optional.

## Details

The SYSPRINTFONT system option sets the font to use when SAS is printing to the current default printer (which might be specified in the SYSPRINT system option) or to the printer identified with the optional keywords NAMED or ALL. This information is stored in the Sasuser.Profile2 catalog.

Enclose the SYSPRINTFONT option arguments in parenthesis when you specify the option in a configuration file, on the command line, or in the SAS System Options window. Parenthesis are not required if you specify the SYSPRINTFONT system option in the OPTIONS statement.

If you use SYSPRINTFONT with either the DEFAULT or no keyword and later use the SYSPRINT system option or the Print Setup dialog box to change the current default printer, then the font used with the current default printer will be

1 The font specified in Sasuser.Profile2 for the given printer, if any.
2 The font specified with SYSPRINTFONT, if the specified font exists on the printer.
3 If there is no font defined for the printer in Sasuser.Profile2, and SYSPRINTFONT doesn't specify a valid font for the printer, and the current display font is scalable, then SAS will use the display font to print.
4 If the current display font is not scalable, SAS will use 10-point SAS Monospace.

**5** If the SAS Monospace font is not available, SAS will use the printer's default font to print.

## Examples

**Example 1: Specifying a Font to the Default Printer** This example specifies to use the 12-point SAS Monospace font on the default printer:

```
-sysprintfont ("SAS Monospace" 12)
```

**Example 2: Specifying a Font to a Named Printer** This example specifies to use 10-point Courier New on the printer named HP LaserJet IIIsi Postscript, attached to LPT1:. Note that the name given for the printer is how it appears in the Print Setup dialog box in SAS:

```
-sysprintfont ("Courier New" named
    "HP LaserJet IIIsi Postscript on LPT1:")
```

## See Also

- "SYSPRINT System Option" on page 566

## TOOLDEF System Option

**Specifies the Toolbox display location**

**Default:** TOP RIGHT

**Valid in:** configuration file, SAS invocation

**Category:** Environment control: Display

**PROC OPTIONS GROUP=** ENVDISPLAY

**Windows specifics:** all

### Syntax

-TOOLDEF TOP | CENTER | BOTTOM <LEFT | CENTER | RIGHT>

**TOP | CENTER | BOTTOM**
specifies the vertical position of the Toolbox. The default value is TOP.

**LEFT | CENTER | RIGHT**
specifies the horizontal position of the Toolbox. The default value is RIGHT.

### Details

The TOOLDEF system option specifies where the Toolbox is located within your display when it is viewable.

You must specify a vertical position first. You do not have to specify a horizontal position, but if you omit it, RIGHT is used.

*Note:* The Toolbox is positioned with respect to your entire display, not to the main SAS window. This option has no effect if you are using the tool bar instead of the Toolbox. △

### See Also

- □ "Customizing a Toolbar" on page 67
- □ "Using the Toolbar to Issue Commands" on page 39

## UPRINTMENUSWITCH System Option

**Enables the universal print commands in the File menu**

**Default:** NOUPRINTMENUSWITCH

**Valid in:** configuration file, SAS invocation

**Category:** Log and procedure output control: ODS printing

**PROC OPTIONS GROUP=** ODSPRINT

**Windows specifics:** all

### Syntax

-UPRINTMENUSWITCH | -NOUPRINTMENUSWITCH

**UPRINTMENUSWITCH**
specifies that the print commands in the File menu invoke the Universal Printing dialog boxes.

**NOUPRINTMENUSWITCH**
specifies that the print commands in the File menu will invoke the Windows dialog boxes.

### Details

To enable the Universal Printing menus and dialog boxes, you must specify both the UNIVERSALPRINT system option and the UPRINTMENUSWITCH system option when you start SAS. Specifying the UPRINTMENUSWITCH option without specifying the UNIVERSALPRINT option will not invoke the Universal Printing menus and dialog boxes.

### See Also

- □ "Introduction to Printing in SAS within the Windows Environment" on page 166

## USER System Option

**Specifies the name of the default permanent SAS data library**

**Default:** none

**Valid in:** configuration file, SAS invocation, OPTIONS statement, SAS System Options window

**Category:** Environment control: Files

**PROC OPTIONS GROUP=** ENVFILES

**Windows specifics:** Valid values for *library-specification*

**See:** USER System Option in *SAS Language Reference: Dictionary*

### Syntax

-USER "*library-specification*"

USER="*library-specification*"

***library-specification***
specifies the default libref, an environment variable, or Windows pathname in which to store data sets that are created during a SAS session. Remember that a pathname is only to the directory or subdirectory level. The value of *library-specification* must resolve to a valid Windows pathname.

### Details

When you specify the USER system option, any data set that you create with a one-level name will be permanently stored in the specified library. If you want to create a temporary data set, use a two-level name for the data set, with the first part being Work (for example, **`work.tempdata`**).

## USERICON System Option

**Specifies the pathname of the resource file associated with your user-defined icon**

**Default:** none

**Valid in:** configuration file, SAS invocation

**Category:** Environment control: Display

**PROC OPTIONS GROUP=** ENVDISPLAY

**Windows specifics:** all

### Syntax

-USERICON *icon-resource-filename number-of-icons*

***icon-resource-filename***
specifies the fully qualified Windows pathname of the resource file associated with your user-defined icons. If the pathname contains spaces, it must be enclosed in quotation marks.

***number-of-icons***

specifies the maximum number of icons stored in the resource file that you specified.

### Details

The USERICON system option specifies the fully qualified Windows pathname of the resource file associated with your icons, along with the maximum number of icons stored in the resource file that you specified.

The icon resource file must be compiled using the Win32 Software Development Kit (SDK). For more information, refer to the SDK documentation. User-defined icons can be incorporated into applications developed with SAS/AF or SAS/EIS software.

### Example

The following USERICON system option specifies 10 icons that are stored in C:\MYSTUFF\MYICONS.DLL:

```
-usericon c:\mystuff\myicons.dll 10
```

## VERBOSE System Option

**Controls whether SAS writes the settings of all the system options specified in the configuration file to either the terminal or the batch log**

**Default:** NOVERBOSE

**Valid in:** configuration file, SAS invocation

**Category:** Log and procedure output control: SAS log

**PROC OPTIONS GROUP=** LOGCONTROL

**Windows specifics:** Amount of information reported

### Syntax

-VERBOSE | -NOVERBOSE

**VERBOSE**
specifies to write the settings of the system options to the log.

**NOVERBOSE**
specifies not to write the settings of the system options to the log. This is the default.

### Details

The VERBOSE system option writes the settings of SAS system options that were set at SAS invocation either on the command line or as part of the configuration file. If you invoke SAS at a terminal, the settings are displayed at the terminal. If you invoke SAS as a part of a batch job, the settings are written to the batch log. You cannot change the settings of the SAS system options with the VERBOSE system option.

The VERBOSE system option is a good error diagnostic tool. If you receive an error message when you invoke SAS, you can use this option to see if you have an error in your system option specifications.

## WEBUI System Option

**Specifies to enable web enhancements**

**Default:** NOWEBUI

**Valid in:** configuration file, SAS invocation

**Category:** Input control: Data processing

**PROC OPTIONS GROUP=** INPUTCONTROL

**Windows specifics:** all

### Syntax

-WEBUI | -NOWEBUI

**WEBUI**
specifies to enable web enhancements.

**NOWEBUI**
specifies to disable web enhancements.

### Details

If you have installed Microsoft Internet Explorer 5.0 (IE) or later and specify the WEBUI system option, certain windows, such as the SAS Explorer window, work like an IE Web page where pointing to an object with the mouse selects the object and a single mouse-click invokes the object's default action.

To select a range of objects, press and hold down the SHIFT key, and point to the first and last objects in the group.

To select multiple items, press and hold down the CTRL key, and point to individual items in the group.

## WINDOWSMENU System Option

**Specifies to include or suppress the Window menu in windows that display menus**

**Default:** NOWINDOWSMENU

**Valid in:** configuration file, SAS invocation, OPTIONS statement, SAS System Options window

**Category:** Environment control: Display

**PROC OPTIONS GROUP=** ENVDISPLAY

**Windows specifics:** all

### Syntax

-WINDOWSMENU | -NOWINDOWSMENU

WINDOWSMENU | NOWINDOWSMENU

**WINDOWSMENU**
specifies to include the Window menu in the main menu if the NOAWSMENUMERGE system option is specified.

**NOWINDOWSMENU**
specifies to suppress the Window menu in the main menu if the NOAWSMENUMERGE system option is specified.

### Details

The WINDOWSMENU system option is valid only if the NOAWSMENUMERGE system option is specified.

### See Also

- "AWSMENUMERGE System Option" on page 488

## WORK System Option

**Specifies the pathname for the directory containing the Work data library**

**Default:** !TEMP\SAS Temporary Files

**Valid in:** configuration file, SAS invocation

**Category:** Environment control: Files

**PROC OPTIONS GROUP=** ENVFILES

**Windows specifics:** Valid values for *library-specification*

**See:** WORK System Option in *SAS Language Reference: Dictionary*

### Syntax

-WORK "*library-specification*"

***"library-specification"***
specifies an environment variable or a Windows pathname. Remember that a pathname is only to the directory or subdirectory level. The value of *library-specification* must resolve to a valid Windows pathname. The *library-specification* must be enclosed in double quotation marks.

### Details

The default SAS configuration file creates the Work data library in a folder named "SAS Temporary Files" located in your system's designated temporary area (as specified by the TEMP environment variable).

SAS creates a subdirectory called TD*nnnnn* for each SAS process under the directory you specify in the WORK option, where *nnnnn* is a unique number.

### See Album

- “Work Data Library” on page 21

## XCMD System Option

**Specifies that the X command is valid in the current SAS session**

**Default:** XCMD

**Valid in:** configuration file, SAS invocation

**Category:** Environment control: Display

**PROC OPTIONS GROUP=** ENVDISPLAY

**Windows specifics:** all

### Syntax

-XCMD | -NOXCMD

–XCMD ON | –XCMD OFF

**XCMD or XCMD ON**
specifies to allow the X command to be valid in the current SAS session.

**NOXCMD or XCMD OFF**
specifies not to allow the X command to be valid in the current SAS session.

### Details

The XCMD allows the X command to be active in the current SAS session.

### See Also

- “X Command” on page 372

## XMIN System Option

**Specifies to open the application specified in the X command in a minimized state or in the default active state.**

**Default:** NOXMIN

**Valid in:** configuration file, SAS invocation, OPTIONS statement, SAS System Options window

**Category:** Environment control: Display

**PROC OPTIONS GROUP=** ENVDISPLAY

**Windows specifics:** all

### Syntax

-XMIN | -NOXMIN

XMIN | NOXMIN

**XMIN**
specifies to start the application specified in the X command in a minimized state.

**NOXMIN**
specifies to start the application specified in the X command in the default active state.

### Details

The XMIN system option allows you to open an application specified in the X command in a minimized state or in the default active state.

## XSYNC System Option

**Controls whether an X command or statement executes synchronously or asynchronously**

**Default:** XSYNC

**Valid in:** configuration file, SAS invocation, OPTIONS statement, SAS System Options window

**Category:** Environment control: Display

**PROC OPTIONS GROUP=** ENVDISPLAY

**Windows specifics:** all

### Syntax

-XSYNC | -NOXSYNC

XSYNC | NOXSYNC

**XSYNC**
specifies that the operating system command execute synchronously with your SAS session. That is, control is not returned to SAS until the command has completed. You cannot return to your SAS session until the process spawned by the X command or statement is closed. This is the default.

**NOXSYNC**
specifies that the operating system command execute asynchronously with your SAS session. That is, control is returned immediately to SAS and the command continues executing without interfering with your SAS session. With NOXSYNC in effect, you can execute an X command or X statement and return to your SAS session without closing the process spawned by the X command or X statement.

### Details

The value of the XSYNC system option affects the execution of the following:

- □ X statement
- □ X command
- □ CALL SYSTEM routine
- □ %SYSEXEC statement.

### See Also

- □ "Running Windows or MS-DOS Commands from within SAS" on page 25
- □ "XWAIT System Option" on page 577
- □ "X Statement" on page 462
- □ "X Command" on page 372
- □ "CALL SYSTEM Routine" on page 387
- □ "Macro Statements" on page 583

## XWAIT System Option

**Specifies whether you have to type EXIT at the DOS prompt before the DOS shell closes**

**Default:** XWAIT

**Valid in:** configuration file, SAS invocation, OPTIONS statement

**Category:** Environment control: Display

**PROC OPTIONS GROUP=** ENVDISPLAY

**Windows specifics:** all

### Syntax

-XWAIT | -NOXWAIT

XWAIT | NOXWAIT

**XWAIT**
specifies that you must type EXIT to return to your SAS session. This is the default.

**NOXWAIT**
specifies that the command processor automatically returns to the SAS session after the specified command is executed. You do not have to type EXIT.

### Details

The XWAIT system option does not affect Windows applications, such as Excel. It only applies to applications that execute in a Command Prompt window.

The XWAIT system option affects the Command Prompt window started by any of the following:

- □ X statement
- □ X command
- □ CALL SYSTEM routine

- □ %SYSEXEC statement.

## See Also

- □ “Running Windows or MS-DOS Commands from within SAS” on page 25
- □ “XSYNC System Option” on page 576
- □ “X Statement” on page 462
- □ “X Command” on page 372
- □ “CALL SYSTEM Routine” on page 387
- □ “Macro Statements” on page 583

# CHAPTER 23 Length and Precision of Variables

*Length and Precision of Variables under Windows* **579**
*Numeric Variables* **579**
*Character Variables* **580**

## Length and Precision of Variables under Windows

For detailed information about how SAS stores representations of numeric and character data, see *SAS Language Reference: Concepts*. Data representation issues vary among different operating environments; the topics in this section discuss how data are represented in SAS under Windows.

## Numeric Variables

The default length of numeric variables in SAS data sets is 8 bytes. (You can control the length of SAS numeric variables with the LENGTH statement in the DATA step.) In SAS under Windows, the Windows data type of numeric values that have a length of 8 is LONG REAL. The precision of floating-point values is always accurate to 15 digits. Depending upon the number, the precision may be 16 digits of accuracy. For more information about the representation of the LONG REAL Windows data type, see Intel's developer Web site. Table 23.1 on page 579 specifies the significant digits and largest integer values that can be stored in SAS numeric variables.

**Table 23.1** Significant Digits and Largest Integer by Length for SAS Variables under Windows

| Length in Bytes | Largest Integer Represented Exactly | Exponential Notation | Significant Digits Retained |
|---|---:|---|---|
| 3 | 8,192 | $2^{13}$ | 3 |
| 4 | 2,097,152 | $2^{21}$ | 6 |
| 5 | 536,870,912 | $2^{29}$ | 8 |
| 6 | 137,438,953,472 | $2^{37}$ | 11 |

| Length in Bytes | Largest Integer Represented Exactly | Exponential Notation | Significant Digits Retained |
|---|---|---|---|
| 7 | 35,184,372,088,832 | $2^{45}$ | 13 |
| 8 | 9,007,199,254,740,992 | $2^{53}$ | 15 |

For example, if you know that a numeric variable always has values between 0 and 100, you can use a length of 3 to store the number and thus save space in your data set. Here is an example:

```
data mydata;
   length num 3;
   more data lines
run;
```

*Note:* *Dummy variables* (those whose only purpose is to hold 0 or 1) can be stored in a variable whose length is 3 bytes. △

***CAUTION:***

**Use the 3-byte limit for only those variables whose values are small, preferably integers.** If the value of a variable becomes large or has many significant digits, you may lose precision when saving the results of arithmetic calculations if the length of a variable is less than 8 bytes. △

The maximum number of variables in a single SAS data set under Windows is 32,767. In addition, an observation under Windows cannot be longer than 5M. Therefore, if you want your data set to contain 32,767 character variables, the longest each variable can be is 160 bytes.

However, a DATA step can reference more than 32,767 variables, if you write only 32,767 or fewer variables to the data set. For example, you could drop some variables with a DROP= data set option. The maximum number of variables a DATA step can reference under Windows is 2,147,483,647.

## Character Variables

In SAS under Windows, character values are sorted using the ASCII collating sequence. As an alternative to the numeric dummy variables discussed previously, you can choose a character variable with a length of 1 byte to serve the same purpose.

The maximum number of variables in a single SAS data set under Windows is 32,767. In addition, an observation under Windows cannot be longer than 5M. Therefore, if you want your data set to contain 32,767 character variables, the longest each variable can be is 160 bytes.

However, a DATA step can reference more than 32,767 variables, if you write only 32,767 or fewer variables to the data set. For example, you could drop some variables with a DROP= data set option. The maximum number of variables a DATA step can reference under Windows is 2,147,483,647.

# CHAPTER 24

# Macro Facility under Windows

*SAS Macro Facility under Windows* **581**
*Automatic Macro Variables* **581**
*Macro Statements* **583**
*Macro Functions* **583**
*Autocall Libraries* **584**
*SASAUTOS System Option* **584**

## SAS Macro Facility under Windows

In general, the SAS macro language is portable across operating environments. This section discusses those components of the macro facility that have system dependencies. For more information, see the *SAS Macro Language: Reference* and *SAS Language Reference: Dictionary*.

*Note:* The words CON, NUL, PRN, COM1 through COM9, and LPT1 through LPT9 are reserved words under Windows. Do not use these reserved words as the name of a macro variable. △

## Automatic Macro Variables

The following automatic macro variables have values that are specific to Windows:

SYSCC
: contains the current SAS condition code that SAS returns to Windows when SAS exits. Upon exit, SAS translates this condition code to a return code that has a meaningful value for the operating environment.

SYSDEVIC
: gives the name of the current graphics device. The current graphics device is determined by the DEVICE system option. Contact your SAS Support Consultant to determine which graphics devices are available at your site. For information about the DEVICE system option, see “DEVICE System Option” on page 496 and *SAS Language Reference: Dictionary*.

SYSENV
: always contains the value **FORE** under Windows.

SYSJOBID
: returns a number that uniquely identifies the SAS task under Windows.

SYSMAXLONG

returns the maximum long integer value that is allowed under Windows, which is 2,147,483,647. On 64–bit systems, the maximum is 9,007,199,254,740,992.

SYSRC
holds the Windows status of Windows commands that are issued during your SAS session. The variable holds a character string that is the text form of the decimal value of the Windows command status.

For example, consider the following statements:

```
options noxwait;
x 'dirf'; /* Invalid Windows command */
%put This Windows status is &sysrc;
x 'dir'; /* Valid Windows command  */
%put The corrected Windows status is &sysrc;
```

The following lines are written to the SAS log:

```
This Windows status is 1
The corrected Windows status is 0
```

The OPTIONS statement turns the XWAIT option off so that the Windows command prompt window closes automatically. That is, you don't have to type **exit** to return to your SAS session. If you run this example with the XWAIT option, you would need to type **exit** before SAS would run the code. After you type **exit**, a value of 0 is returned in both cases; 0 is the return code for the EXIT command. If you use the NOXSYNC system option, the value of SYSRC is automatically 0.

SYSSCP
returns the operating environment abbreviation WIN.

SYSSCPL
returns the name of the specific Windows environment that you are using. The possible return values are

NET_SRV
Microsoft Windows Server 2003 Standard Edition

NET_ASRV
Microsoft Windows Server 2003 Enterprise Edition

NET_DSRV
Microsoft Windows Server 2003 Datacenter Edition

WIN_NT
Microsoft Windows NT 4.0 Workstation

WIN_NTSV
Microsoft Windows NT Server 4.0 or Microsoft Windows NT Server 4.0 Terminal Server Edition or Microsoft Windows NT Server 4.0, Enterprise Edition

WIN_ASRV
Microsoft Windows 2000 Advanced Server

WIN_DSRV
Microsoft Windows 2000 Datacenter Server

WIN_PRO
Microsoft Windows 2000 Professional

WIN_SRV
Microsoft Windows 2000 Server

XP_PRO

Microsoft Windows XP Professional

W64_ASRV
Microsoft Windows Server 2003 Enterprise 64-bit Edition

W64_DSRV
Microsoft Windows Server 2003 Datacenter 64-bit Edition

W64_PRO
Microsoft Windows XP 64-bit edition.

## Macro Statements

The following macro statement has behavior specific to Windows:

%SYSEXEC
executes operating environment commands immediately and places the return code in the SYSRC automatic macro variable. The %SYSEXEC statement is similar to the X statement that is described in "Running Windows or MS-DOS Commands from within SAS" on page 25. You can use the %SYSEXEC statement inside a macro or in open code. The %SYSEXEC statement has the following syntax:

%SYSEXEC <*command*>;

The *command* argument can be any operating environment command or any sequence of macro operations that generates an operating environment command. You can also use the *command* argument to invoke a Windows application such as Notepad.

Omitting the *command* argument launches a command prompt subprocess, which is interactive. To return to your SAS session, type EXIT at the command prompt and press ENTER. The SYSRC automatic variable is set to 0 if you omit the *command* argument in the %SYSEXEC statement.

Here is a simple example of %SYSEXEC:

```
%sysexec time;
```

This statement launches a command prompt session that displays the following lines:

```
The current time is: 16:32:45.16
Enter new time:
```

*Note:* The %SYSEXEC statement uses the XSYNC and XWAIT system option values just like the X statement and X command do. For more information about these system options, see "XSYNC System Option" on page 576 and "XWAIT System Option" on page 577. △

## Macro Functions

The behavior of the %SYSGET macro function is specific to Windows:

%SYSGET
returns the character string that is the value of the Windows environment variable that is passed as the argument. Both Windows and SAS environment variables can be translated by using the %SYSGET function. A warning message is printed if the environment variable does not exist. The %SYSGET function has the following syntax:

%SYSGET(*environment-variable-name*);

Here is an example of using the %SYSGET function:

```
%let var1=%sysget(comspec);
%put The COMSPEC environment variable
     is &var1;
```

The following line is written to the SAS log:

```
The COMSPEC environment variable is
C:\winnt\system\command.exe
```

## Autocall Libraries

This section discusses the system dependencies of using autocall libraries. For general information, see *SAS Macro Language: Reference*.

An autocall library contains files that define SAS macros. SAS supplies some autocall macros. To use the autocall facility, you must have the SAS system option MAUTOSOURCE set. When SAS is installed, the SASAUTOS system option is used in the SAS configuration file to tell SAS where to find the default macros that are supplied by SAS Institute. You can also define your own autocall macros and store them in a Windows directory.

If you store autocall macros in a Windows directory, the file extension must be .SAS. Each macro file in the directory must contain a macro definition that has a macro name that is the same as the filename. For example, a file named PRTDATA.SAS that is stored in a directory must define a macro named PRTDATA.

### SASAUTOS System Option

To use your own autocall macros in your SAS programs, you must tell SAS where to find them using the SASAUTOS system option. The syntax of the SASAUTOS option is given in “SASAUTOS System Option” on page 544.

You can set the SASAUTOS system option when you start SAS, or you can use it in an OPTIONS statement during your SAS session. You should edit your SAS configuration file to add your autocall library to the library concatenation that is supplied by SAS Institute, as in the following example:

```
-sasautos (c:\mymacros
          !sasroot\core\sasmacro
          !sasroot\base\sasmacro
          !sasroot\stat\sasmacro
          more library specifications
          )
```

Autocall libraries are searched in the order that you specify them. If you use the preceding SASAUTOS option setting and call a macro named PRTDATA, the directory C:\MYMACROS is searched first for the macro; then each of the !SASROOT libraries is searched.

# PART 4

# Appendices

*Appendix* ***1*** . . . . . . . . . **SCL Methods for Automating OLE Objects** *587*

*Appendix* ***2*** . . . . . . . . . **Error Messages for SAS under Windows** *597*

*Appendix* ***3*** . . . . . . . . . **Using SSL under Windows** *605*

*Appendix* ***4*** . . . . . . . . . **Graphics Considerations** *613*

*Appendix* ***5*** . . . . . . . . . **Default Key Settings for Interactive SAS Sessions** *615*

*Appendix* ***6*** . . . . . . . . . **Recommended Reading** *623*

# APPENDIX 1
# SCL Methods for Automating OLE Objects

*Summary of OLE Class Methods* **587**
*_COMPUTE_* **588**
*_DISABLE_DEFAULT_ACTION_* **589**
*_DO_* **589**
*_ENABLE_DEFAULT_ACTION_* **590**
*_EXECUTE_* **590**
*_GET_EVENT_* **591**
*_GET_PROPERTY_* **591**
*_GET_REFERENCE_ID_* **592**
*_GET_TYPE_* **592**
*_IN_ERROR_* **593**
*_NEW_* **594**
*_SET_PROPERTY_* **594**
*_UPDATE_* **595**

## Summary of OLE Class Methods

Table A1.1 on page 587 contains a list of SCL methods you can use with object linking and embedding (OLE) and indicates which of the OLE classes they apply to.

**Table A1.1** SCL Methods Valid for OLE and OLE Automation

| Method | SAS OLE class | SAS OLE Automation class |
|---|---|---|
| _COMPUTE_ | Yes | Yes |
| _DISABLE_DEFAULT_ACTION_ | Yes | No |
| _DO_ | Yes | Yes |
| _ENABLE_DEFAULT_ACTION_ | Yes | No |
| _EXECUTE_ | Yes | No |
| _GET_EVENT_ | Yes | No |
| _GET_PROPERTY_ | Yes | Yes |
| _GET_REFERENCE_ID_ | Yes | Yes |
| _GET_TYPE_ | Yes | No |
| _IN_ERROR_ | Yes | Yes |
| _NEW_ | No | Yes |

| Method | SAS OLE class | SAS OLE Automation class |
|---|---|---|
| _SET_PROPERTY_ | Yes | Yes |
| _UPDATE_ | Yes | No |

*Note:* The _NEW_ method can be used with any class, but the OLE Automation class overrides this method because of special requirements. △

The remainder of this section contains the reference information for these methods.

## _COMPUTE_

**Invokes a method on an OLE automation object and returns a value**

### Syntax

**CALL NOTIFY**(*OLE-object-name*,'_COMPUTE_',*in-OLE-method*<,*in-parm*…,*in-parm*>,*out-value*);

**CALL SEND**(*OLE-object-id*,'_COMPUTE_',*in-OLE-method*<,*in-parm*…,*in-parm*>,*out-value*);

| Where... | Is type... | And... |
|---|---|---|
| *in-OLE-method* | C | specifies the OLE method name. |
| *in-parm* | C or N | provides a parameter to the OLE method. |
| *out-value* | C or N | contains the value returned by the OLE method. |

### Details

The _COMPUTE_ method invokes a method (with parameters) that is exposed by an OLE automation server. The number of parameters (*in-parm* arguments) needed varies among different objects and methods. Only methods that have a return value should be used with the _COMPUTE_ method. For methods with no return values, use the _DO_ method.

### Examples

The following example stores the contents of the item in position 2 of an OLE control in the variable *item2obj*:

```
length item2obj $ 200;
call notify('oleobj', '_COMPUTE_',
            'GetItem', 2, item2obj);
```

The following example uses the cells method of a spreadsheet object to compute the location of the cell at row 2, column 5, and then sets the value of that cell to 100:

```
call send(oleobj, '_COMPUTE_', 'Cells',
          2, 5, cellobj1);
call send(cellobj1, '_SET_PROPERTY_',
          'Value', 100);
```

## _DISABLE_DEFAULT_ACTION_

**Disables the OLE object's default action**

### Syntax

**CALL NOTIFY**(*OLE-object-name*,'_DISABLE_DEFAULT_ACTION_');

### Details

This method prevents the default verb for an OLE object from executing when the object is double-clicked. By default, the default action is enabled.

## _DO_

**Invokes a method on an OLE automation object with no return value**

### Syntax

**CALL NOTIFY**(*OLE-object-name*,'_DO_',*in-OLE-method*<,*in-parm*…,*in-parm*>);

**CALL SEND**(*OLE-object-id*,'_DO_',*in-OLE-method*<, *in-parm*…,*in-parm*>);

| Where... | Is type... | And... |
|---|---|---|
| *in-OLE-method* | C | specifies the OLE method name. |
| *in-parm* | C or N | provides a parameter to the OLE method. |

### Details

The _DO_ method invokes a method (with parameters) that is exposed by an OLE automation server. The number of parameters (*in-parm* arguments) needed varies among different OLE objects and methods. Only methods that have no return value should be used with the _DO_ method. For methods with return values, use the _COMPUTE_ method.

### Example

The following example sends the AboutBox method to an OLE control, which displays the About Box for the control:

```
call notify('oleobj', '_DO_', 'AboutBox');
```

## _ENABLE_DEFAULT_ACTION_

**Enables the OLE object's default action**

### Syntax

**CALL NOTIFY**(*OLE-object-name*,'_ENABLE_DEFAULT_ACTION_');

### Details

This method enables the default verb for an OLE object to execute when the object is double-clicked. By default, the default action is enabled.

## _EXECUTE_

**Executes an OLE verb for the object**

### Syntax

**CALL NOTIFY**(*OLE-object-name*,'_EXECUTE_',*in-verb*<,*in-verb*…*in-verb*>);

| Where... | Is type... | And... |
|---|---|---|
| *in-verb* | C | specifies the OLE verb to execute. |

### Details

A list of verbs supported by this object are listed in the Associated Verbs window for the OLE object (after the object has been created). You can specify more than one OLE verb at a time.

If you attempt to execute a verb that is not valid for the object, the SCL program halts and returns a message that the verb does not exist.

## _GET_EVENT_

**Returns the name of the last OLE control event received**

### Syntax

**CALL NOTIFY**(*OLE-object-name*,'_GET_EVENT_',*out-event*);

| Where... | Is type... | And... |
|---|---|---|
| *out-event* | C | contains the returned name of the last OLE control event received. |

### Details

This method returns the name of the event that the specified OLE control most recently sent.

## _GET_PROPERTY_

**Returns the value of a property of an automation object**

### Syntax

**CALL NOTIFY**(*OLE-object-name*,'_GET_PROPERTY_',*in-OLE-property*,*out-property-value*);

**CALL SEND**(*OLE-object-id*,'_GET_PROPERTY_',*in-OLE-property*,*out-property-value*);

| Where... | Is type... | And... |
|---|---|---|
| *in-OLE-property* | C | specifies the name of the OLE property. |
| *out-property-value* | C or N | contains the returned value of the property. |

### Details

The _GET_PROPERTY_ method is used to get the value of a property of an automation object.

## _GET_REFERENCE_ID_

**Returns a reference identifier for use with any automation object method that requires an automation object as one of its parameters**

### Syntax

**CALL NOTIFY**(*OLE-object-name*,'_GET_REFERENCE_ID_',*out-refid*);

**CALL SEND**(*OLE-object-id*,'_GET_REFERENCE_ID_',*out-refid*);

| Where... | Is type... | And... |
|---|---|---|
| *out-refid* | C | contains the returned reference identifier. |

### Details

The _GET_REFERENCE_ID_ method is used to get the automation object identifier. The value returned is used in subsequent _DO_ or _COMPUTE_ calls where the object method requires an automation object as one of its parameters. This value should be used for the object parameter.

### Example

The following example returns the reference identifier for the automation object. This identifier is then sent as a parameter value to an automation method requiring an object identifier.

```
call notify('oleobj1', '_GET_REFERENCE_ID_',
            refid);
call notify('oleobj2', '_DO_', 'NewAppl',
            refid, p1, p2);
```

## _GET_TYPE_

**Returns the object's type**

### Syntax

**CALL NOTIFY**(*OLE-object-name*,'_GET_TYPE_',*out-type*);

| Where... | Is type... | And... |
|---|---|---|
| *out-type* | C | contains the returned object type. |

## Details

The _GET_TYPE_ method is used to get the type of the object. Valid types include Embedded, Linked, Bitmap, Device Independent Bitmap, and Picture.

## _IN_ERROR_

**Returns an object's ERROR status**

### Syntax

**CALL NOTIFY**(*OLE-object-name*,'_IN_ERROR_',*error-status*<,*error-msg*>);

**CALL SEND**(*OLE-object-id*,'_IN_ERROR_',*error-status*<,*error-msg*>);

| Where... | Is type... | And... |
|---|---|---|
| *error-status* | N | returns a value indicating whether an automation error has been encountered for the object. |
| *error-msg* | C | returns the automation error message. |

### Details

Errors encountered from automation calls can be detected using _IN_ERROR_. The _IN_ERROR_ method returns the status of the last automation call and should be called prior to any subsequent automation calls.

### Example

The following example detects that an error was encountered during the previous _GET_PROPERTY_ call:

```
length errmsg $ 200;
call send(objid,'_GET_PROPERTY_',
          'ActiveObject', actobj);
call send(objid,'_IN_ERROR_',inerror, errmsg);
if inerror then
   link handle_err;
```

## _NEW_

**Creates a new instance of an OLE automation server**

### Syntax

**CALL SEND**(*OLE-instance*,'_NEW_',*new-OLE-id*,*init-arg*,*OLE-auto-app*);

### Details

Before you can use SCL code to refer to an OLE Automation server, you must first create an instance of the OLE Automation class.

For more information about the _INIT_ method, see the description of the Object class in the online documentation for SAS/AF software.

### Example

The following example creates a new instance of an OLE Automation server and assigns the SCL identifier **`exclauto`** to the new object. Note that in this example, **`Excel.Application.8`** is the identifier for Microsoft Excel in the system registry:

```
hostcl=loadclass('sashelp.fsp.hauto');
call send (hostcl, '_NEW_', exclauto, 0,
           'Excel.Application.8');
```

## _SET_PROPERTY_

**Assigns a value to a property of an automation object**

### Syntax

**CALL NOTIFY**(*OLE-object-name*,'_SET_PROPERTY_',*in-OLE-property*,*in-value*);

**CALL SEND**(*OLE-object-id*,'_SET_PROPERTY_',*in-OLE-property*,*in-value*);

| Where... | Is type... | And... |
|---|---|---|
| *in-OLE-property* | C | specifies the OLE property name. |
| *in-value* | C or N | contains the value to assign to the OLE property. |

## Details

The _SET_PROPERTY_ method assigns a value to a property of an automation object.

## _UPDATE_

**Updates the object based on its current contents or on the contents of a different HSERVICE entry**

### Syntax

**CALL NOTIFY**(*OLE-object-name*,'_UPDATE_'<,*in-hservice*>);

| Where... | Is type... | And... |
|---|---|---|
| *in-hservice* | C | specifies the name of the HSERVICE entry to use to update the object. |

### Details

The _UPDATE_ method recreates an object and updates its contents based on its current attributes. The *in-hservice* parameter is used only with OLE objects and is the name of an HSERVICE catalog entry. When you specify the *in-hservice* parameter, the object specified by *OLE-object* is changed to the object stored in the HSERVICE entry referenced by the *in-hservice* parameter.

If you use the _UPDATE_ method without specifying *in-hservice*, the object's contents are updated with the current OLE object source. This is useful for manually updating a linked object.

### Example

In the following example, the object stored in OBJ1 is replaced by the Sasuser.Examples.Sound1.Hservice object:

```
length refid $ 30;
call notify('obj1','_update_',
            'sasuser.examples.sound1.hservice');
```

# APPENDIX 2 Error Messages for SAS under Windows

*Overview of SAS Error Messages* **597**
*Return Codes and Completion Status* **597**
*Accessing Files* **598**
*Using SAS Features* **599**
*Using OLE* **600**
*Using Networks* **601**
*Resolving Internal Errors* **601**
*Resolving Operating System and Windows Error Messages* **602**
*Initialization and Termination Error Messages* **603**

## Overview of SAS Error Messages

This section contains completion codes and error messages that you may find helpful. In the error message lists, the messages are in monospace. Words in italic in the messages represent items that are variable, such as a filename or number. Each description tells you where the message comes from and explains its meaning and what you can do to correct the possible problem.

## Return Codes and Completion Status

The return code for the completion of a SAS job is returned in the Windows batch variable, ERRORLEVEL. A value of 0 indicates normal termination. You can affect the value of ERRORLEVEL by using the ABORT statement. The ABORT statement takes an option argument, *n*, which is an integer. The ABORT statement also takes the RETURN or ABEND argument. If you issue these statements without specifying *n*, the ERRORLEVEL variable is set to the following values:

| | |
|---|---|
| abort; | sets the ERRORLEVEL variable to 3. |
| abort return; | sets the ERRORLEVEL variable to 4. |
| abort abend; | sets the ERRORLEVEL variable to 5. |

The *n* argument can range from 1 to 65,535. The ERRORLEVEL variable is used as a condition in the IF command in a Windows batch file. Refer to your Window's user's guide for more information on the ERRORLEVEL variable. The following table summarizes the values of the ERRORLEVEL variable.

**Table A2.1** Values for the ERRORLEVEL Variable

| Condition | Severity | Return Code Value |
|---|---|---|
| All steps terminated normally | SUCCESS | 0 |
| SAS issued warning(s) | WARNING | 1 |
| SAS issued error(s) | ERROR | 2 |
| User issued the ABORT statement | INFORMATIONAL | 3 |
| User issued the ABORT RETURN statement | FATAL | 4 |
| User issued the ABORT ABEND statement | FATAL | 5 |
| SAS internal error | INFORMATIONAL | 6 |

## Accessing Files

This section describes errors you might receive while trying to use SAS to access files (either external files or SAS files). Whenever you have trouble accessing files, always check the validity of your FILENAME and LIBNAME statements and functions to make sure they point to the right files. Also, be sure you are using the correct fileref or libref.

**`Core catalog cannot be initialized. Please verify the system's date/time.`**
The date or time stamp of SAS's CORE catalog is in the future. Make sure the date and time on your machine are set correctly. This message is issued in conjunction with internal error 602 (see "Resolving Internal Errors" on page 601).

**`Error: Date/time is in the future.`**
The date or time stamp of the file you are trying to access is in the future. Make sure the date and time on your machine are set correctly.

**`Error: File is in use,`** ***`filename`*****`.`**
The file you are trying to access is in use by another Windows process, such as another Windows application.

**`Error: File not found loading`** ***`filename-1`*****`. File contributing to error:`** ***`filename-2`*****`.`**
A DLL-dependent file cannot be found when the requested file is loaded. For SAS, the !SASROOT\CORE\SASEXE file (usually specified with the PATHDLL system option in the SAS configuration file) may have become unavailable. This may be due to a network error or other drive failure. Ensure that PATHDLL specifies the location of !SASROOT\CORE\SASDLL.

**`ERROR: Member or library`** ***`filename`*** **`unavailable for use.`**
The file *filename* is being used by another Windows application.

**`ERROR: Module`** ***`module-name`*** **`not found in search paths.`**
This error is caused by one of the following:

- □ incorrect PATH system option in the SAS configuration file
- □ the product you called is not installed
- □ a dependent image is not installed.

**`ERROR: Unable to clear or reassign the library`** ***`library-name`*** **`because it is still in use.`**
You are trying to reassign a libref while the library is in use.

**`ERROR: Operating system error number` *`n`* `occurred while accessing` *`filename`*`.`**
An unexpected return code has been received by SAS from the operating system. For more information, see "Resolving Operating System and Windows Error Messages" on page 602.

**`ERROR: Physical file does not exist` *`filename`*`.`**
The file you are trying to access does not exist. Verify that you have specified the correct drive and directory. This error can also occur if you are trying to write to a write-protected diskette.

**`ERROR: Write access to member member-name is denied.`**
You are trying to update a file on a write-protected diskette or you are trying to update a file marked as read-only.

## Using SAS Features

This section describes errors you may receive while using features of the SAS language and procedures under Windows. Always check the syntax of the statement or procedure you are using. Also, if you do not get the results you expect, check the contents of any external files or SAS files to be sure they are correct.

**`An interrupt has occurred.`**
You have canceled a print job by clicking on **`Abort`**.

**`ERROR: Out of disk space for spooling.`**
Not enough disk space is currently available for spooling a print job. This message implies that no more disk space can be made available.

**`ERROR: Not enough memory is available for spooling.`**
Not enough memory is available for spooling. This message implies, however, that more space may become available at some point.

**`WARNING: SAS option` *`option-name`* `is valid only at startup of the SAS System or startup of an environment within SAS. The SAS option is ignored.`**
Several SAS system options can only be set in the SAS configuration file or in the SAS command. For more information, see "SAS System Options under Windows" on page 467.

**`ERROR: Unknown configuration option` *`option-name`*`.`**
Check your SAS command and SAS configuration file for an invalid SAS system option. For more information on system options under Windows, see "SAS System Options under Windows" on page 467.

**`ERROR: User terminated the job through the Print Manager.`**
You terminated (that is, deleted) your print job by using the Print Manager.

**`Invalid toolbox catalog:` *`catalog-name`*`.`**
You have tried to load a nonexistent toolbox from a SAS catalog with the TOOLLOAD command. Check the spelling of the toolbox name and the catalog name in the command. You can use the SAS Explorer window to see a listing of a SAS catalog.

**`No tools defined.`**
The toolbox you have attempted to load has no tools defined. You must have at least one tool defined in a toolbox.

**`Unable to access the specified printer driver.`**

SAS cannot access the printer driver. Make sure the correct printer driver has been specified by either the SYSPRINT system option or with the Printer Setup dialog box.

**`Unable to find the printer name.`**
SAS cannot find the printer specified in the Printer Setup dialog box.

## Using OLE

This section describes errors you may receive while using the Windows object linking and embedding (OLE) capability in your SAS applications.

**`OLE Error:`** ***`nnnnnnnn <error message text>`***
An OLE error not documented in this section has occurred. If you receive this error message and cannot determine its cause from the given error message text, contact your SAS Installation Representative, who can determine the cause of the error. The SAS Installation Representative may have to call the SAS Institute Technical Support Division.

**`OLE: Unable to Paste Special.  The clipboard contains no supported formats.`**
The Windows clipboard does not contain any formats that SAS/AF software can use in the FRAME entry.

**`...  Do you want to invoke the Links dialog?`**
The data source of the link could not be found. You have the option of invoking the Links dialog box to redirect the link to another data source.

**`OLE: Operation not allowed on static object.`**
Static objects do not support this operation. A static object is basically a picture of an object; it does not contain nor is it linked to any data.

**`Static objects can not be converted.`**
You cannot convert static objects to another type.

**`OLE: Verb is invalid for the object.`**
The OLE verb that was passed to the object was invalid or could not be sent. Check your SCL code for a misspelled verb.

**`OLE: Server application could not be launched.`**
The server application for the object could not be invoked. You may be missing some executables, or perhaps your network connection is down.

**`OLE: Member or one of the named parameters is not known.`**
You specified a member (that is, a property or method) that is not valid for the OLE object you are automating. For information about the members that are supported, see the documentation for the OLE server application.

**`OLE: Access to multi-dimensional arrays not supported.`**
SAS does not support multi-dimensional arrays. If you want to access multi-dimensional data from a server application, you must first use the server application to present the data in a one-dimensional format.

**`OLE: Object can not be automated.`**
This object does not support automation.

**`OLE: One of the parameters is of the wrong type.`**
**`OLE: One of the parameters is not a valid type.`**
**`OLE: One of the parameters is out of the present range.`**

A parameter passed to an OLE automation server's method is not the correct type and cannot be interpreted.

**`OLE: Application is busy.`**
The OLE server application is busy with a task and cannot honor the current request.

**`OLE: Control is not licensed for use.`**
You do not have appropriate licensing to use a control of this type. For more information, see the control's documentation.

**`OLE: Unable to connect to network device:`*`UNC-drive-name`***
The linked object exists on a network drive that is currently not connected. The UNC name for the drive follows the message.

**`There is no object selected for conversion.`**
The Convert dialog box was invoked but there are no selected objects to convert.

**`There are no linked objects in this window.`**
The Links dialog box was invoked but the frame does not contain any linked OLE objects.

## Using Networks

This section describes errors you may receive while using SAS on a network.

Any of the following errors can occur on a network if you do not have proper access rights:

- **`ERROR: Deletion of old member`** ***`filename`*** **`failed.`**
- **`ERROR: File deletion failed for`** ***`filename`*****`.`**
- **`ERROR: Rename of temporary member for`** ***`filename`*** **`failed.`**
- **`ERROR: User does not have appropriate authorization level for file`** ***`filename`***

Network software enables the network supervisor to control access to network files. Access rights can be set up so that you may read a file, but are unable to update that file. The last message in this list can also appear if you try to access a directory as if it were a file.

## Resolving Internal Errors

Internal errors are fatal errors that keep SAS from starting. While you should not usually see these error messages, with a bit of investigation you may be able to solve the problem that generated the message. The following list describes the most common of these messages:

**`10 Host internal error:  10`**
CTRL+BREAK was pressed during the initialization of SAS. Therefore, SAS terminated.

**`11 Host internal error:  11`**
SAS needs more memory. To correct the problem, change swapping to a disk with more free space or delete enough files from the present swap disk to free up at least 20 megabytes of memory.

**`12 Host internal error:  12`**

SAS has determined that there is an error in the specified SAS configuration. A descriptive message is displayed that indicates which system option is in error.

**`13 Host internal error: 13`**
Some part of SAS cannot be loaded. The part in question is indicated via a descriptive message. Restore the missing part to the appropriate directory.

**`24 Host internal error: 24`**
SAS was unable to initialize the windowing environment. A descriptive message is displayed indicating the appropriate action.

**`208 Unable to open profile catalog.`**
SAS must have access to your Sasuser.Profile catalog or be able to create one if one does not exist. Check to be sure that you have enough disk space, or if you are running SAS on a network, that you have access to the correct files. The Sasuser.Profile catalog is opened in the directory specified by the SASUSER system option, which is described in "SASUSER System Option" on page 547. Also see "Profile Catalog" on page 19.

**`Work library is undefined. Sasuser library is undefined. Sashelp library is undefined.`**
Check your SAS command or SAS configuration file to be sure that you have specified the directory path correctly for these libraries.

**`302 Sasmsg library is undefined.`**
Check your SAS command or SAS configuration file to be sure that you have specified the directory path correctly for this library.

**`Unable to initialize Work library. Unable to initialize Sasuser library.`**
Check your SAS command or SAS configuration file to be sure that you have specified the correct directory for the Work and Sasuser libraries. If you are running SAS on a network, be sure you have access to the necessary files.

**`401 Unable to initialize the message subsystem.`**
Check your SAS command or SAS configuration file to be sure that you have specified the directory path correctly for all SAS data libraries. Also check to see if your SAS configuration file is where it should be. If you are sure you have specified the directory paths correctly, contact your SAS Site Representative. It is possible that SAS message files have become corrupt or have been inadvertently deleted.

**`601 Invalid SETINIT information.`**
Check your SETINIT information for errors. For more information, see the installation instructions for SAS 9.1 for Windows.

**`602 CORE catalog could not be accessed for options initialization.`**
The CORE catalog cannot be accessed. This may be caused by having the date on your PC set incorrectly. Use the Windows DATE command to verify and set the date.

## Resolving Operating System and Windows Error Messages

In situations where unexpected return codes are returned from Windows, the Windows error number is written to the SAS log. If you have access to Windows programming manuals or Windows user documentation, you can look up the error number to determine the cause of the error. Alternatively, you can report the error number to your SAS Installation Representative.

## Initialization and Termination Error Messages

If SAS issues error messages during SAS initialization or termination, the SAS log may contain error messages that explain the error. Any error message that SAS issues before the SAS log is initialized or after is closed are written to the MSG window if it is available or the SAS console log, which is a Windows file. Under Windows NT, the SAS console log is located in the c:\winnt\Profiles\*username*\Application Data folder. In all other Windows operating environments the SAS console log is located in the c:\Document and Settings\*userid*\Application Data folder. You can obtain the filename for the SAS Console Log from the Application Event Log. To open the application Event Log, submit **`eventvwr`** from the Run dialog box and click on **`Application`**.

# APPENDIX 3 Using SSL under Windows

*What Is SSL?* **605**
*SSL (Secure Sockets Layer)* **605**
*Certification Authorities (CAs)* **605**
*Public and Private Keys* **606**
*Digital Signatures* **606**
*Digital Certificates* **606**
*Using SSL* **606**
*Overview of SSL Set-Up Process* **606**
*SSL for SAS* **607**
*SSL for Windows* **607**
*System and Software Requirements for SSL under Windows* **607**
*Digital Certificates Set-Up Process* **608**
*Generating Digital Certificates Issued by Microsoft Certificate Authority* **608**
*Generating Digital Certificates Issued by a Certificate Authority That Is Not Microsoft* **609**
*Importing Digital Certificates to a Certificate Store* **609**
*Converting between PEM and DER File Formats* **611**
*SSL Language Elements* **611**

## What Is SSL?

### SSL (Secure Sockets Layer)

SSL is a protocol that provides secure network communications. Developed by Netscape Communications, SSL uses the encryption algorithms that were developed by RSA Security, Inc. and other cryptography experts.

In addition to providing encryption services, SSL performs client and server authentication and uses message authentication codes. SSL is supported by both Netscape Navigator and Internet Explorer. Many Web sites use this protocol to protect confidential user information, such as credit card numbers. URLs that require an SSL connection begin with https: instead of http:. The SSL protocol is application independent, which allows protocols such as HTTP, FTP, and Telnet to be transparently layered above it. SSL is optimized for HTTP.

### Certification Authorities (CAs)

As e-business proliferates, there is a great need to ensure the confidentiality of business transactions over a network between an enterprise and its consumers, between enterprises, and within an enterprise. Cryptography products provide security services by exploiting digital certificates, public-key cryptography, private-key cryptography, and digital signatures. Certification authorities (CAs) create and maintain digital certificates, which also help preserve confidentiality.

Various commercial CAs, such as VeriSign and Thawte, provide competitive services for the e-commerce market. You can also develop your own CA by using products from companies such as RSA Security and Microsoft or from the Open Source Toolkit OpenSSL. From a trusted CA, members of an enterprise can obtain digital certificates to facilitate their e-business needs. The CA provides a variety of ongoing services to the business client that include handling digital certificate requests, issuing digital certificates, and revoking digital certificates.

### Public and Private Keys

Public-key cryptography uses a public and a private key pair. The public key can be known by anyone, therefore, anyone can send a confidential message. The private key is confidential and known only to the owner of the key pair, therefore, only the owner can read the encrypted message. The public key is used primarily for encryption, but it can also be used to verify digital signatures. The private key is used primarily for decryption, but it can also be used to generate a digital signature.

### Digital Signatures

A digital signature affixed to an electronic document or to a network data packet is like a personal signature that concludes a hand-written letter or that validates a credit card transaction. Digital signatures are a safeguard against fraud. A unique digital signature results from using a private key to encrypt a message digest. Receipt of a document that contains a digital signature enables the receiver to verify the source of the document. Electronic documents can be verified if you know where the document came from, who sent it, and when it was sent. Another form of verification comes from MACs, which ensure that a document has not been changed since it was signed.

### Digital Certificates

Digital certificates are electronic documents that ensure the binding of a public key to an individual or an organization. Digital certificates provide protection from fraud.

Usually, a digital certificate contains a public key, a user's name, and an expiration date. It also contains the name of the certification authority (CA) that issued the digital certificate and a digital signature that is generated by the CA. The CA's validation of an individual or an organization allows that individual or organization to be accepted at sites that trust the CA.

## Using SSL

### Overview of SSL Set-Up Process

The details for installing and setting up SSL at your site are based on the operating environment and the digital certificate services software that you use. However, the following tasks are basic:

1 Access the appropriate software for installing and setting up digital certificate services under your operating environment.
2 Define a Certificate Authority (CA).
3 Request that digital certificates be generated by the CA for users, machines, and other CAs.
4 Store the digital certificates in a trusted repository.
5 View the properties of the generated digital certificates.
6 Start a server.
7 Connect to the server.

## SSL for SAS

You can set SAS system options to use SSL in a SAS session. See the SAS options that are appropriate to your operating environment.

## SSL for Windows

### System and Software Requirements for SSL under Windows

The system and software requirements for using SSL under the Windows operating environment are:

- A computer that runs Windows 2000 (or later).
- Depending on your configuration, it might be useful to have access to the Internet and a Web browser such as Netscape Navigator or Internet Explorer.
- The TCP/IP communications access method.
- Microsoft Certificate Services add-on software.
- The Microsoft Certificate Authority application (which is accessible from your Web browser) if you will run your own CA.
- In order for a SAS/CONNECT client session to connect to a SAS/CONNECT server session via a Windows spawner using SSL encryption, ensure that the client session runs on a machine that has a Trusted CA Certificate.

  The Windows spawner must run on a machine that has a Trusted CA Certificate and a Personal Certificate.
- Knowledge of your site's security policy, practices, and technology. The properties of the digital certificates that you request will depend on the security policies that have been adopted at your site.

Complete information about configuring your Windows operating environment for SSL is contained in the Windows installation documentation and at www.microsoft.com. The following keywords might be helpful when searching the Microsoft Web site:

- digital certificate services
- digital certificate authority

□ digital certificate request
□ site security planning.

## Digital Certificates Set-Up Process

The process for generating digital certificates under the Windows operating environment follows:

1. The user requests a digital certificate from a certificate authority (CA).
2. The CA issues a digital certificate.
3. The digital certificate is installed in a certificate store.

The tasks that you perform depend on the CA that you use:

See "Generating Digital Certificates Issued by Microsoft Certificate Authority" on page 608

See "Generating Digital Certificates Issued by a Certificate Authority That Is Not Microsoft" on page 609.

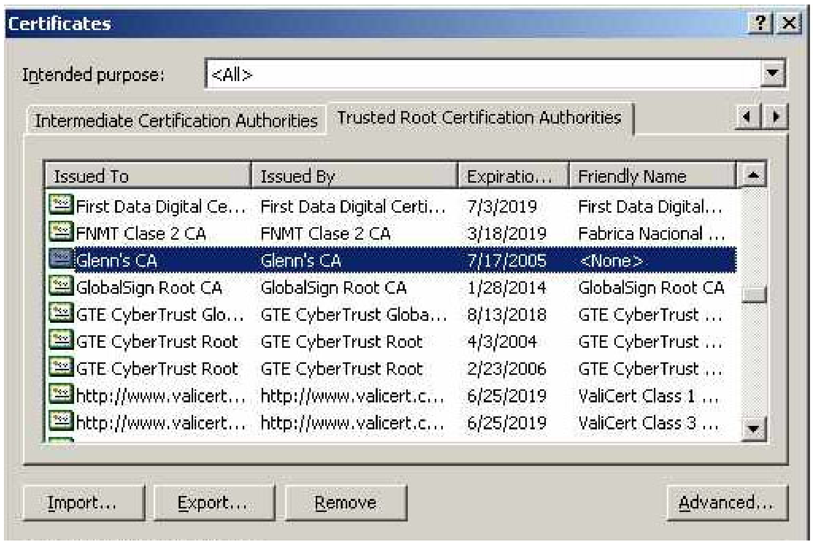
### Generating Digital Certificates Issued by Microsoft Certificate Authority

The following tasks are performed to generate digital certificates issued by the Microsoft Certificate Authority:

1. If you are running your own CA, the system administrator uses Microsoft Certificate Services to create an active Certificate Authority (CA).
2. The user
   a. uses the Certificate Request wizard to request a digital certificate from an active enterprise CA. The Certificate Request wizard lists all digital certificate types that the user or computer is eligible to obtain.
   b. selects a digital certificate type
   c. specifies security options
   d. submits the request to an active CA that is configured to issue the digital certificate.

      After the CA issues the requested digital certificate, the digital certificate is automatically installed in the Certificate Store. (See Display A3.1 on page 608 for an example.)

**Display A3.1** Digital Certificate Installation in the Certificate Store



Certificates

Intended purpose: <All>

Intermediate Certification Authorities | Trusted Root Certification Authorities

| Issued To | Issued By | Expiratio... | Friendly Name |
|---|---|---|---|
| First Data Digital Ce... | First Data Digital Certi... | 7/3/2019 | First Data Digital... |
| FNMT Clase 2 CA | FNMT Clase 2 CA | 3/18/2019 | Fabrica Nacional ... |
| Glenn's CA | Glenn's CA | 7/17/2005 | <None> |
| GlobalSign Root CA | GlobalSign Root CA | 1/28/2014 | GlobalSign Root CA |
| GTE CyberTrust Glo... | GTE CyberTrust Globa... | 8/13/2018 | GTE CyberTrust ... |
| GTE CyberTrust Root | GTE CyberTrust Root | 4/3/2004 | GTE CyberTrust ... |
| GTE CyberTrust Root | GTE CyberTrust Root | 2/23/2006 | GTE CyberTrust ... |
| http://www.valicert... | http://www.valicert.c... | 6/25/2019 | ValiCert Class 1 ... |
| http://www.valicert... | http://www.valicert.c... | 6/25/2019 | ValiCert Class 3 ... |

## Generating Digital Certificates Issued by a Certificate Authority That Is Not Microsoft

The following tasks are performed to generate digital certificates that are not issued by the Microsoft Certificate Authority:

1 the user requests a digital certificate from a CA and the digital certificate is issued.

2 the user imports digital certificates to a Certificate Store by using the Certificate Manager Import Wizard application from a Web browser. The digital certificates can be generated by using the Certificate Request wizard or any third-party application that generates digital certificates.

   *Note:* The Windows operating environment can import digital certificates that were generated in the UNIX operating environment. If you want to convert from PEM format (UNIX) to DER format (Windows) before importing, see "Converting between PEM and DER File Formats" on page 611. △

## Importing Digital Certificates to a Certificate Store

Digital certificates that were issued by a third-party application can be imported to an appropriate Certificate Store, as follows:

| Certificate Type | Certificate Storage Location |
|---|---|
| Client | Personal Certificate Store |
| Server | Personal Certificate Store |
| CA (self-signed) | Trusted Root Certification Authorities |

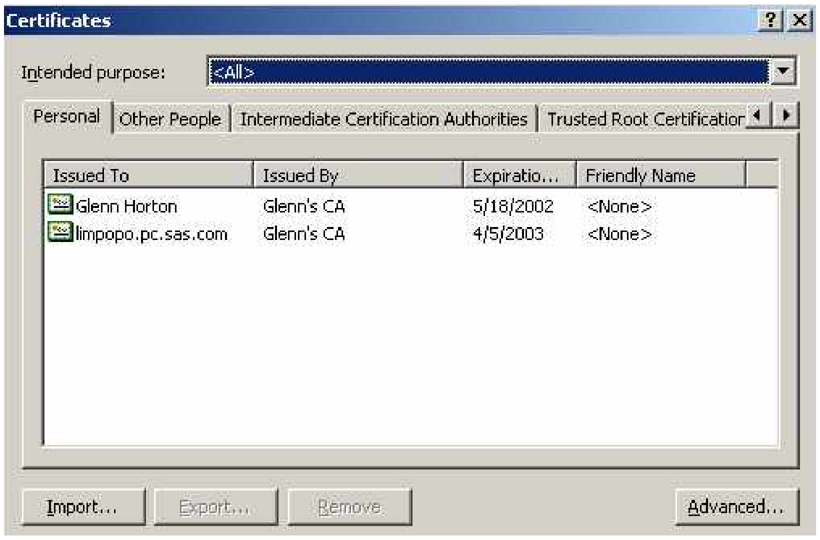
Perform the following tasks to import a digital certificate to a Certificate Store:

1 Access the Certificate Manager Import Wizard application from your Web browser. From the **`Tools`** pull-down menu, select

```
Tools -> Internet Options -> Content tab -> Certificates button
```

   Select the Personal tab in the Certificates window and specify which files you want to import to a Certificate Store. (See Display A3.2 on page 610)

**Display A3.2** Digital Certificate Selections for a Personal Certificate Store



| Issued To | Issued By | Expiratio... | Friendly Name |
|---|---|---|---|
| Glenn Horton | Glenn's CA | 5/18/2002 | <None> |
| limpopo.pc.sas.com | Glenn's CA | 4/5/2003 | <None> |

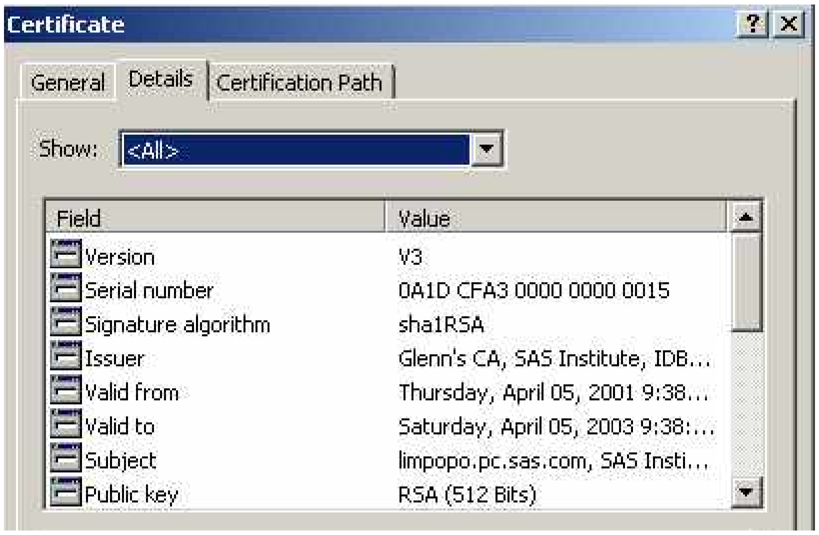
**2** Click Import and follow the instructions to import digital certificates.

Repeat this task in order to import the necessary digital certificates for the CA, the server, and the client, as appropriate.

**3** After you have completed the selections for your personal Certificate Store, select the appropriate tab to view your selections.

**4** To view the details about a digital certificate, select the digital certificate and click View . The results are shown in Display A3.3 on page 610.

**Display A3.3** Digital Certificate Details Tab



| Field | Value |
|---|---|
| Version | V3 |
| Serial number | 0A1D CFA3 0000 0000 0015 |
| Signature algorithm | sha1RSA |
| Issuer | Glenn's CA, SAS Institute, IDB... |
| Valid from | Thursday, April 05, 2001 9:38... |
| Valid to | Saturday, April 05, 2003 9:38:... |
| Subject | limpopo.pc.sas.com, SAS Insti... |
| Public key | RSA (512 Bits) |

## Converting between PEM and DER File Formats

By default, OpenSSL files are created in PEM (Privacy Enhanced Mail) format. SSL files that are created in Windows operating environments are created in DER (Distinguished Encoding Rules) format.

Under Windows, you can import a file that is created in either PEM or DER format. However, a digital certificate that is created in DER format must be converted to PEM format before it can be included in a trust list under UNIX.

An example of converting a server digital certificate from PEM input format to DER output format follows:

```
OpenSSL> x509 -inform PEM -outform DER -in server.pem -out server.der
```

An example of converting a server digital certificate from DER input format to PEM output format follows:

```
OpenSSL> x509 -inform DER -outform PEM -in server.der -out server.pem
```

## SSL Language Elements

- “SSLCERTISS System Option” on page 559
- “SSLCERTSERIAL System Option” on page 559
- “SSLCERTSUBJ System Option” on page 560
- “SSLCLIENTAUTH System Option” on page 561
- FILENAME Statement, URL Access Method in *SAS Language Reference: Dictionary*

# APPENDIX 4

# Graphics Considerations

*Using TrueType Fonts with SAS/GRAPH Software* **613**

## Using TrueType Fonts with SAS/GRAPH Software

TrueType fonts can be used with the WIN, WINPRTx, WMF, EMF, and GIF device drivers, among others. Before you can use a TrueType font to print (other than the default), you must identify its name. To identify TrueType fonts, select

Start ▶ Settings ▶ Control Panel

and double-click on the Fonts icon. The fonts with icons displaying a double ‘T’ are the TrueType fonts.

In SAS/GRAPH programs, you can use the font name to specify a TrueType font with the FONT= or F= option. For example, you can specify the following:

```
title2 font="Courier New"
  'This is TrueType font Courier New';
```

For more information about the FONT= option, such as how to use alternate fonts with hardcopy devices and how to make an alternate font the default font, see *SAS/GRAPH Reference, Volumes 1 and 2*.

APPENDIX 5

# Default Key Settings for Interactive SAS Sessions

*Default Key Definitions under Windows* **615**
*Keyboard Shortcuts within the SAS Main Window* **616**
*Keyboard Shortcuts within the Enhanced Editor* **618**
*Keyboard Shortcuts within Print Preview* **621**

## Default Key Definitions under Windows

The following table lists the default key definitions for the primary SAS application windows (such as Program Editor, Log, and Output), excluding the Enhanced Editor. “Keyboard Shortcuts within the Enhanced Editor” on page 618 lists the default key definitions for the Enhanced Editor. Any other key combination that is not listed in this table is either reserved by Windows or has a definition that you cannot change within SAS (see “Keyboard Shortcuts within the SAS Main Window” on page 616).

If you want to browse or change any of the key definitions that are listed in this table, you can do so by selecting

Tools ▶ Options ▶ Keys

or by issuing the KEYS command.

**Table A5.1** Default Key Settings for SAS under Windows

| Key | Default Setting | Key | Default Setting |
|---|---|---|---|
| F1 | help | Alt + F1 | |
| F2 | reshow | Alt + F2 | |
| F3 | end | Alt + F3 | |
| F4 | recall | Alt + F11 | |
| F5 | wpgm | Alt + F12 | |
| F6 | log | Ctrl + B | libname |
| F7 | output | Ctrl + D | dir |
| F8 | zoom off; submit | Ctrl + E | clear |
| F9 | keys | Ctrl + G | |
| F11 | command focus | Ctrl + H | help |
| F12 | | Ctrl + I | options |
| Shift + F1 | subtop | Ctrl + J | |

| Key | Default Setting | Key | Default Setting |
|---|---|---|---|
| Shift + F2 | | Ctrl + K | cut (Program Editor only) |
| Shift + F6 | | Ctrl + L | log |
| Shift + F7 | left | Ctrl + M | mark |
| Shift + F8 | right | Ctrl + Q | filename |
| Shift + F9 | | Ctrl + R | rfind |
| Shift + F10 | wpopup | Ctrl + T | title |
| Shift + F11 | | Ctrl + U | unmark |
| Shift + F12 | dmcopylsv | Ctrl + W | access Explorer window |
| Ctrl + F1 | | Ctrl + Y | |
| Ctrl + F2 | | RMB | wpopup |
| Ctrl + F3 | | Shift + RMB | |
| Ctrl + F11 | | Ctrl + RMB | |
| Ctrl + F12 | | MMB | |
| | | Shift + MMB | |
| | | Ctrl + MMB | |

*Note:*

1 RMB is the right mouse button.

2 MMB is the middle mouse button. (Not all mouse devices have a middle mouse button.)

△

## Keyboard Shortcuts within the SAS Main Window

The keys that are listed here are not listed in the KEYS window. You might find these keys to be useful as shortcuts for editing and other tasks.

**Table A5.2** Key Settings for the Main SAS Window

| Key Combination | Action |
|---|---|
| *Dialog boxes and Entry Fields* | |
| Tab | move to next field |
| Shift + Tab | move to previous field |
| *Navigate around Text* | |
| Ctrl + -> (right arrow) | move to next word |
| Ctrl + <- (left arrow) | move to previous word |
| Home | move to beginning of line |
| End | move to end of line |

| **Key Combination** | **Action** |
|---|---|
| Ctrl + Home | move to top |
| Ctrl + End | move to bottom |
| Page Up | page up |
| Page Down | page down |
| Ctrl + Page Up | move to top |
| Ctrl + Page Down | move to bottom |
| Ctrl + Tab | navigate to the next open SAS window (NEXTWIND command) |
| Ctrl + Shift + Tab | navigate to the previous open SAS window (PREVWIND command) |
| *Mark Text* | |
| Shift + -> (right arrow) | mark while going to the right |
| Shift + <- (left arrow) | mark while going to the left |
| Shift + Home | mark to beginning of line |
| Shift + End | mark to end of line |
| Shift + Ctrl + Home | mark to top |
| Shift + Ctrl + End | mark to bottom |
| Shift + Page Up | page up and mark |
| Shift + Page Down | page down and mark |
| Shift + Ctrl + Page Up | mark to top |
| Shift + Ctrl + Page Down | mark to bottom |
| Shift + MB1 | extend the current marked text selection to the click position |
| *Cut, Copy, and Paste* | |
| Delete | delete the next character (or marked text) |
| Ctrl + Delete | delete from the cursor position to the end of the current word |
| Ctrl + Backspace | delete from the cursor position to the start of the current word |
| Ctrl + Z | undo previous action |
| Ctrl + X | cut selected text |
| Ctrl + C | copy selected text to paste buffer |
| Ctrl + V | paste text |
| *Window Control* | |
| Alt | switch focus to or from the main menu bar |
| Shift + F5 | cascade the windows |
| Shift + F4 | tile the windows vertically |
| Shift + F3 | tile the windows horizontally |
| Ctrl + F6 | next window |

| Key Combination | Action |
|---|---|
| Alt + F4 | exit SAS |
| Ctrl + F4 | close the active window |
| Shift + F10 | open context menu |
| *Resizing the Docking View* | |
| Alt + W + S | start docking view resizing |
| -> (right arrow) | move the split bar a small amount to the right |
| <- (left arrow) | move the split bar a small amount to the right |
| Ctrl + -> (right arrow) | move the split bar a larger amount to the right |
| Ctrl + <- (left arrow) | move the split bar a larger amount to the left |
| Home | move the split bar all the way to the left |
| End | move the split bar all the way to the right |
| Return | accept the current size of the docking view and exit docking view resizing |
| Esc | end docking view resizing without resizing the docking view |
| *Miscellaneous* | |
| Alt + Enter | open the Properties dialog box for a selected object<br>This command is valid only in a Tree view or a List view. |
| Esc + *letter* (or *number*) | color or highlighting attributes in NOTEPAD window |

## Keyboard Shortcuts within the Enhanced Editor

The keyboard shortcuts that are listed here are the default shortcuts.

**Table A5.3** Default Keyboard Shortcuts for the Enhanced Editor

| Category | Command | Keyboard Shortcut |
|---|---|---|
| Abbreviation | Add a new abbreviation | Ctrl + Shift + A |
| | Bring up word tip | Alt + F1 + No Selection |
| | Hide the current word tip | Esc |
| Code Folding | Collapse all folding blocks | Alt + Ctrl + Number pad - |
| | Collapse current line | Alt + Number pad - |
| | Expand all folding blocks | Alt + Ctrl + Number pad + |
| | Expand current line | Alt + Number pad + |
| | Toggle expand current line | Alt + Number pad * |
| Command/Macro Support | Add or change macros | Ctrl + Shift + M |
| | Execute the last recorded macro | Ctrl + F1 |

| Category | Command | Keyboard Shortcut |
|---|---|---|
| | Play a command/macro | Alt + F8 |
| | Start/Complete macro | Alt + Shift + R |
| Edit | Copy selection | Ctrl + C |
| | Cut selection | Ctrl + X |
| | Delete current character | Delete |
| | Delete previous character | Backspace or Shift + Backspace |
| | Delete to next word start | Ctrl + Delete |
| | Delete to previous word start | Ctrl + Backspace |
| | Insert a carriage return | Enter |
| | Paste from clipboard | Ctrl + V |
| | Redo | Ctrl + Y<br>Alt + Shift + Backspace |
| | Undo | Ctrl + Z<br>Alt + Backspace |
| Help | Get Help for a SAS procedure | place the cursor within a procedure name and press F1 |
| | Context Help | F1 |
| Line Markers | Go to the next marked line | F2 |
| | Go to the previous marked line | Shift + F2 |
| | Toggle marker on the current line | Ctrl + F2 |
| Navigation | Go to line (interactive) | Ctrl + G |
| | Move cursor to the top of the file | Ctrl + Page Up<br>Ctrl + Home |
| | Move cursor to the bottom of the file | Ctrl + Page Down<br>Ctrl + End |
| | Move cursor down | Down |
| | Move cursor down a page | Page Down |
| | Move cursor left | Left |
| | Move cursor right | Right |
| | Move cursor to beginning of line | Home |
| | Move cursor to end of line | End |
| | Move cursor to matching brace/ parentheses | Ctrl + [<br>Ctrl + ] |
| | Move cursor to matching DO/ END keyword | Alt + [<br>Alt + ] |
| | Move cursor to next case change | Alt + Right |

| **Category** | **Command** | **Keyboard Shortcut** |
|---|---|---|
| | Move cursor to next word start | Ctrl + Right |
| | Move cursor to previous case change | Alt + Left |
| | Move cursor to previous word start | Ctrl + Left |
| | Move cursor up | Up |
| | Move cursor up a page | Page Up |
| | Move cursor to the first visible line | Alt + Up |
| | Move cursor to the last visible line | Alt + Down |
| | Scroll screen down | Ctrl + Up |
| | Scroll screen up | Ctrl + Down |
| Option Setting | Toggle insert/overwrite mode | Insert |
| Selection | Extend selection character left | Shift + Left |
| | Extend selection character right | Shift + Right |
| | Extend selection down | Shift + Down |
| | Extend selection down a page | Shift + Page Down |
| | Extend selection to beginning of document | Ctrl + Shift + Home<br>Ctrl + Shift + Page Up |
| | Extend selection to beginning of line | Shift + Home |
| | Extend selection to end of document | Ctrl + Shift + End<br>Ctrl + Shift + Page Down |
| | Extend selection to end of line | Shift + End |
| | Extend selection to next case change | Alt + Shift + Right |
| | Extend selection to previous case change | Alt + Shift + Left |
| | Extend selection up | Shift + Up |
| | Extend selection up a page | Shift + Page Up |
| | Extend selection to previous word start | Ctrl + Shift + Left |
| | Extend selection to the next word start | Ctrl + Shift + Right |
| | Select all | Ctrl + A |
| Selection Operations | Clean up whitespace | Ctrl + Shift + W |
| | Comment the selection with line comments | Ctrl + / |

| Category | Command | Keyboard Shortcut |
|---|---|---|
| | Convert the selected text to lowercase | Ctrl + Shift + L |
| | Convert the selected text to uppercase | Ctrl + Shift + U |
| | Tab selection | Tab + Selection |
| | Undo the Comment | Ctrl + Shift + / |
| | Undo the Tab selection | Shift + Tab + Selection |

## Keyboard Shortcuts within Print Preview

You can use the keyboard shortcuts in the following tabel in the Print Preview window.

**Table A5.4** Keyboard Shortcuts for the Print Preview Window

| To do this... | Press these keys... |
|---|---|
| Next page | Alt + N |
| Previous page | Alt + P |
| Zoom | Alt + Z |
| Help | Alt + H |
| Print | Alt + R |
| Close the window | Alt + C or<br>Alt + F4 |

# APPENDIX 6 Recommended Reading

*Recommended Reading* **623**

## Recommended Reading

Here is the recommended reading list for this title:

- *SAS Language Reference: Concepts*
- *SAS Language Reference: Dictionary*
- *SAS Output Delivery System: User's Guide*
- *Base SAS Procedures Guide*
- *SAS National Language Support (NLS): User's Guide*

For a complete list of SAS publications, see the current *SAS Publishing Catalog*. To order the most current publications or to receive a free copy of the catalog, contact a SAS representative at

SAS Publishing Sales
SAS Campus Drive
Cary, NC 27513
Telephone: (800) 727-3228*
Fax: (919) 677-8166
E-mail: **sasbook@sas.com**
Web address: **support.sas.com/pubs**
* For other SAS Institute business, call (919) 677-8000.

Customers outside the United States should contact their local SAS office.

# Glossary

**active window**
a window that is open and displayed, and to which keyboard input is directed. Only one window can be active at a time.

**application workspace (AWS)**
a window that contains other windows or from which other windows can be invoked, but which is not contained within any window that is part of the same software application. The SAS application workspace (SAS AWS) is also referred to as the main SAS window.

**ASCII (American Standard Code for Information Interchange)**
a 7-bit encoding (8 bits when a parity-check bit is included) for the United States that provides 128 character combinations. The encoding contains characters for uppercase and lowercase English, American English punctuation, base 10 numbers, and a few control characters. This set of 128 characters is included in most other encodings. ASCII is used on personal computers. See also EBCDIC (Extended Binary Coded Decimal Interchange Code).

**ASCII collating sequence**
an ordering of characters that follows the order of the characters in the ASCII character coding scheme. See also ASCII (American Standard Code for Information Interchange), EBCDIC collating sequence.

**autocall macro**
a macro whose uncompiled source code and text are stored in an autocall macro library. Unlike a stored, compiled macro, an autocall macro is compiled before execution the first time it is called.

**AUTOEXEC.SAS**
a file containing SAS statements that are executed automatically when SAS is invoked. The autoexec file can be used to specify some SAS system options, as well as to assign librefs and filerefs to folders or directories that are used frequently.

**automatic macro variable**
a macro variable that is defined by SAS rather than by the user.

**AWS**
See application workspace (AWS).

**batch file**

a file that contains operating-system commands, which are processed sequentially when the file is executed.

**batch mode**
a method of executing SAS programs in which a file that contains SAS statements plus any necessary operating environment commands is submitted to the computer's batch queue. After you submit the program, control returns to your computer or workstation, where you can perform other tasks. Batch mode is sometimes referred to as running in the background. The program output can be written to files or printed on an output device. Under Windows, a Status window is associated with a SAS batch job. The Status window indicates which batch job is running and shows the pathnames and filenames of the log file and the procedure output file.

**binary**
the name of the base 2 number system. A binary digit can have one of two values: 0 or 1. A binary digit is called a bit and is considered to be off when its value is 0 and on when its value is 1. See also binary file.

**binary file**
a file that contains information that can be read by one or more software applications but not by humans. Some binary files are executable programs. Others store images, sounds, data, or a combination of printable and non-printable characters. Binary files cannot be edited with a text editor.

**buffer**
an area of computer memory that is reserved for use in performing input/output (I/O) operations.

**cache**
a small, fast memory area that holds recently accessed data. The cache is designed to speed up subsequent access to the same data.

**catalog**
See SAS catalog.

**catalog entry**
See SAS catalog entry.

**character constant**
one or more characters enclosed in quotation marks in a SAS statement (sometimes called a character literal). The maximum number of characters allowed is 200. See also character string.

**character set**
the set of characters and symbols that are used by a language or group of languages. A character set includes national-language characters (characters that are specific to a particular language or country), special characters (such as punctuation marks), the unaccented Latin characters A-Z, the digits 0-9, and control characters that are needed by the computer.

**character string**
one or more alphanumeric characters or other keyboard characters or both. See also character constant.

**character value**
a value that can contain alphabetic characters, the numeric characters 0 through 9, and other special characters.

**class**
in object-oriented programming, the template or model for an object. A class includes data that describes the object's characteristics (attributes or instance variables) and the operations (methods) that the object can perform. See also subclassing, object.

**client**
(1) a computer or application that requests services, data, or other resources from a server. (2) in the context of named pipes and in DDE and OLE, an application that sends data to or receives data from an application that is acting as a server. See also server.

**clipboard**
a temporary storage place for data that is being passed from one application to another. For example, in Windows operating environments, you can use the clipboard to pass information between Excel and your SAS session.

**COM (Component Object Model)**
an object-oriented programming model that defines how software components interact within a single process or between processes. For example, COM includes standard rules of communication that enable a user-interface object to be dragged and dropped from one application window to another.

**COM/DCOM client**
a program that uses the Microsoft Component Object Model or Distributed Component Object Model to make requests to a server. COM/DCOM clients can be written in Visual Basic, C++, Perl, or other programming languages in the Windows environment. See also COM (Component Object Model), DCOM (Distributed Component Object Model).

**command prompt**
the symbol after which you enter operating system commands.

**component**
a self-contained, reusable programming object that provides some type of service to other components in an object-oriented programming environment.

**CONFIG.SYS**
a system file that contains DOS configuration commands that specify the properties of the operating system, including device drivers, file-handling elements, and memory-management options.

**configuration file**
(1) in SAS software, an external file that contains SAS system options. These system options take effect each time you invoke SAS. (2) under DOS, the CONFIG.SYS file that specifies the properties of the operating system. See also CONFIG.SYS.

**Control Panel**
under Windows, an application that enables you to specify characteristics of your Windows session, such as mouse tracking speed and the color of the title bar.

**conventional memory**
in servers that are running 32-bit operating systems, the first 4 gigabytes of main memory. In servers that are running 64-bit operating systems, all of the main memory is conventional memory.

**CPU (central processing unit)**
the main hardware component of a computer. The CPU executes program instructions and controls the operation of other parts of the computer.

**CPU time**
the amount of time it takes for the central processing unit of a computer system to perform the calculations or other operations that you request.

**current folder**
the folder to which commands and actions apply when you execute an application.

**DATA step**

a group of statements in a SAS program that begins with a DATA statement and which ends with either a RUN statement, another DATA statement, a PROC statement, the end of the job, or the semicolon that immediately follows lines of data. The DATA step enables you to read raw data or other SAS data sets and to use programming logic to create a SAS data set, to write a report, or to write to an external file.

**DCOM (Distributed Component Object Model)**
an extension to the Component Object Model (COM) that enables components to request services from components that are on other computers in a network. See also component, COM (Component Object Model).

**DDE (Dynamic Data Exchange)**
a standard mechanism in the PC environment for sharing data among applications.

**device driver**
a program that controls the interaction between a computer and an external device such as a printer or a disk drive.

**directory**
another term for folder. See folder.

**DLL (dynamic link library)**
a collection of executable program modules that are loaded at run time as needed.

**docking view**
a view of the main SAS window in which one or more windows, such as the Explorer and Results windows, are integrated with the left side of the main SAS window.

**DOS**
a disk operating system for personal computers. In SAS documentation, the acronym DOS refers specifically to MS-DOS, the Microsoft disk operating system, which was developed by Microsoft for IBM.

**drag**
in a graphical user interface, to move an object such as an icon or a window around on a display screen. To drag the object, you usually use a mouse button to select the object, then move the move the mouse while keeping the mouse button pressed down.

**dummy variable**
in some statistical applications, a numeric variable whose value is limited to 1 or 0.

**Dynamic Data Exchange (DDE)**
See DDE (Dynamic Data Exchange).

**dynamic link library**
See DLL (dynamic link library).

**EBCDIC (Extended Binary Coded Decimal Interchange Code)**
an 8-bit character coding scheme that includes both graphic (printable) codes and control (nonprintable) codes. See also ASCII (American Standard Code for Information Interchange).

**EBCDIC collating sequence**
an ordering of characters that follows the order in the Extended Binary Coded Decimal Interchange Code (EBCDIC) character coding scheme. SAS uses the same collating sequence as its host operating environment. See also ASCII collating sequence.

**encoding**
a set of characters (letters, logograms, digits, punctuation, symbols, control characters, and so on) that have been mapped to numeric values (called code points)

that can be used by computers. The code points are assigned to the characters in the character set by applying an encoding method. Some examples of encodings are wlatin1, wcyrillic, and shift-jis.

**encoding method**
a set of rules that are used to map characters (letters, logograms, digits, punctuation, symbols, control characters, and so on) to numeric values. Some examples of encoding methods are ASCII, EBCDIC, EUC, and PCMS.

**engine**
a component of SAS software that reads from or writes to a file. Each engine enables SAS to access files that are in a particular file format. There are several types of engines. See also interface engine, library engine, native engine, view engine.

**Enhanced Editor**
an ASCII text editor that provides features such as color coding and code sections to help SAS users write and debug SAS programs. The Enhanced Editor also provides familiar features of the SAS Program Editor.

**entry type**
a characteristic of a SAS catalog entry that identifies the catalog entry's structure and attributes to SAS. When you create an entry, SAS automatically assigns the entry type as part of the name. See also SAS catalog entry.

**environment variable**
under Windows, a variable that equates one character string to another by using the SAS system option SET, the Windows SET command, or the Windows System Properties dialog box. SAS environment variables cannot be accessed by other Windows applications. By contrast, Windows environment variables can be accessed by all Windows applications.

**error message**
a message in the SAS log or Message window that indicates that SAS was not able to continue processing the program.

**extended memory**
See extended server memory.

**extended server memory**
on a server that is running a 32-bit operating system, the part of main memory that exceeds the 4 gigabytes of conventional memory. See also conventional memory.

**external file**
a file that is created and maintained by a host operating system or by another vendor's software application. SAS can read data from and route output to external files. External files can contain raw data, SAS programming statements, procedure output, or output that was created by the PUT statement. A SAS data set is not an external file. See also fileref (file reference).

**fatal error**
an error that causes a program to end abnormally or that prevents the program from starting.

**field**
a window area in which users can view, enter, or modify a value.

**file extension**
the classification of a file in a directory that identifies what type of information is stored in the file. For example, .SCAT is the file extension for SAS catalogs.

**filename**

the identifier that is used for a file. The filename includes the file extension, as in PROFILE.SC2. See also fully qualified filename, pathname.

**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. Under Windows, you can assign a fileref with a FILENAME statement, the SAS system option SET, the Windows SET command, or from the SAS Explorer window.

**folder**
a named subdivision that is used for organizing files on a disk, diskette, CD-ROM, or DVD-ROM. A folder also contains information about each file that it contains, such as size and date of last change.

**font**
a complete set of all the characters of the same design and style. The characters in a font can be figures or symbols as well as alphanumeric characters.

**fully qualified filename**
a file specification that includes both the pathname and the filename, as in C:\SAS\SASUSER\PROFILE.SC2. See also filename, pathname.

**function key**
a keyboard key that can be defined to have a specific action in a specific software environment.

**gigabyte**
2 to the 30th power, or 1,073,741,824 (approximately 1 billion) bytes. See also kilobyte, megabyte, terabyte.

**graphical user interface (GUI)**
any system that uses graphical objects such as windows, menus, icons, buttons, and check boxes to represent the functions of a software application and to enable the user to interact with the application. By contrast, a command-line interface requires users to interact with the software application by entering text. Many graphical user interfaces use visual metaphors for real-world objects such as file cabinets, folders, rulers, and scissors.

**GUI**
See graphical user interface (GUI).

**host option**
in a SAS statement, an option that is specific to a particular operating environment.

**HTML (HyperText Markup Language)**
a coding system in which the codes indicate the layout and style of the text in a text file. Other HTML codes enable you to embed electronic objects such as images, sounds, video streams, and applets (small software applications) into HTML documents. All Web browsers can process HTML documents.

**icon**
in a graphical user interface, a pictorial representation of a window, an action (such as sending e-mail or printing a document), or an option (such as double-spacing) that is available to the user. The user clicks on (or otherwise selects) the icon in order to expand the icon into a window, to perform the action, or to specify the option.

**interface engine**
a SAS engine that reads and writes file formats that are supported by other vendors' software. See also engine, native engine.

**kilobyte**

2 to the 10th power, or 1024 bytes. See also gigabyte, megabyte, terabyte.

**library engine**
an engine that accesses groups of files and puts them into the correct form for processing by SAS utility windows and procedures. A library engine also determines the fundamental processing characteristics of the library, presents lists of files for the library directory, and supports view engines. See also engine, view engine.

**libref (library reference)**
a name that is temporarily associated with a SAS library. The complete name of a SAS file consists of two words, separated by a period. The libref, which is the first word, indicates the library. The second word is the name of the specific SAS file. For example, in VLIB.NEWBDAY, the libref VLIB tells SAS which library contains the file NEWBDAY. You assign a libref with a LIBNAME statement or with an operating system command.

**maximize**
in a graphical user interface, to cause a resizable window to instantly be displayed at its largest size, usually by clicking on (or otherwise selecting) an icon or on the maximize button of an active window.

**megabyte**
2 to the 20th power, or 1,048,576 (approximately 1 million) bytes. See also gigabyte, kilobyte, terabyte.

**member**
a SAS file in a SAS library.

**member name**
a name that is assigned to a SAS file in a SAS library. Under Windows, the member name is the same as the filename for files that are stored in a SAS data library.

**member type**
a SAS name that identifies the type of information that is stored in a SAS file. Member types include ACCESS, AUDIT, DMBD, DATA, CATALOG, FDB, INDEX, ITEMSTOR, MDDB, PROGRAM, UTILITY, and VIEW.

**memory-based library**
a SAS library that is stored either in conventional memory or in extended server memory (rather than on a data storage device) for the duration of a SAS session or job. See also conventional memory, extended server memory.

**menu bar**
the primary list of items at the top of a window, which represent the actions or classes of actions that can be executed. Selecting an item executes an action, opens a pull-down menu, or opens a dialog box that requests additional information. See also pop-up menu, pull-down menu.

**message area**
the area of the status bar (at the bottom of the main SAS window) that displays messages from SAS.

**minimize**
in a graphical user interface, to click on the minimize button of an active window, causing the window to be replaced by an icon elsewhere on the desktop. The window can be restored to its former size, location, and active status by clicking on (or otherwise selecting) the icon.

**Multiple Engine Architecture**
a feature of SAS that enables it to access a variety of file formats through sets of instructions called engines. See also engine.

**multitasking**
the ability of an operating system to execute more than one application (or multiple instances of the same application) at the same time, using a single central processing unit. Individual tasks within the applications are scheduled so that the applications appear to be running at the same time and so that the applications do not interfere with each other. Almost all operating systems are capable of multitasking.

**named pipe**
a named object that provides client-to-server, server-to-client, or duplex communication between unrelated processes. You can use named pipes to establish communication between Windows applications, including multiple SAS sessions. See also pipe.

**native engine**
an engine that accesses types of SAS files that are created and processed only by SAS. See also engine, interface engine.

**network**
an interconnected group of computers.

**NT file system (NTFS)**
an advanced system for organizing directories and files. NTFS supports long filenames, full security access control, file system recovery, and extremely large storage media.

**NTFS**
See NT file system (NTFS).

**object**
in object-oriented methodology, a specific representation of a class. An object inherits the characteristics (attributes or instance variables) of its class as well as the operations (methods) that class can execute. For example, a push button object is an instance of the Push Button class. The terms object and instance are often used interchangeably.

**Object Linking and Embedding (OLE)**
See OLE (Object Linking and Embedding).

**ODBC (Open Database Connectivity)**
an interface standard that provides a common application programming interface (API) for accessing data. Many software products that run in the Windows operating environment adhere to this standard so that you can access data that was created using other software products.

**ODBC driver**
a loadable library module that provides a standardized interface for accessing, manipulating, and updating data that is created and maintained by a particular vendor's data management software. For example, the SAS ODBC driver enables you to access, manipulate, and update SAS data sources from any application that conforms to the ODBC standard. See also ODBC (Open Database Connectivity).

**ODS (Output Delivery System)**
a component of SAS software that can produce output in a variety of formats such as markup languages (HTML, XML), PDF, listing, RTF, Postscript, and SAS data sets.

**OLE (Object Linking and Embedding)**
a method of interprocess communication supported by Windows that involves a client/server architecture. OLE enables an object that was created by one application to be embedded in or linked to another application.

**pathname**

in Windows operating environments, a specification of a drive, directories, and subdirectories, such as C:\SAS\SASUSER.

**permanent SAS data library**
a SAS library that is not deleted when a SAS session ends, and which is therefore available to subsequent SAS sessions. Unless the USER libref is defined, you use a two-level name to access a file in a permanent library. The first-level name is the libref, and the second-level name is the member name. See also libref (library reference), member.

**physical filename**
the name that the operating system uses to identify a file.

**pipe**
an object that provides direct access to STDIN, STDOUT, and STDERR between processes. Pipe is synonymous with unnamed pipe. See also named pipe.

**pop-up menu**
a menu that appears when it is requested. These menus are context-specific, depending on which window is active and on the cursor location. See also pull-down menu.

**portability**
the ability of a program to execute in an operating environment other than the one for which it was written.

**portable**
See portability.

**procedure output file**
an external file that contains the result of the analysis that a SAS procedure performs or the report that the procedure produces. Most SAS procedures write output to the procedure output file by default. Reports that are produced by SAS DATA steps, using PUT statements and a FILE statement along with a PRINT destination, also go to this file. See also external file, DATA step.

**process**
a functional unit of a program or task.

**process ID (PID)**
a unique number that is assigned to each process by the operating system.

**Profile catalog**
See Sasuser.Profile catalog.

**pull-down menu**
the list of menu items or choices that appears when you choose an item from a menu bar or from another menu.

**raw data**
data that has not been read into a SAS data set.

**record**
a logical unit of information that consists of fields of related data. A collection of records are stored in a file. A record is analogous to a SAS observation.

**return code**
a code passed to the operating system that indicates whether a command or job step has executed successfully.

**SAS AWS**
See application workspace (AWS).

**SAS catalog**
a SAS file that stores many different kinds of information in smaller units called catalog entries. A single SAS catalog can contain several different types of catalog entries. See also SAS catalog entry.

**SAS catalog entry**
a separate storage unit within a SAS catalog. Each entry has an entry type that identifies its purpose to SAS. Some catalog entries contain system information such as key definitions. Other catalog entries contain application information such as window definitions, Help windows, SAS formats and informats, macros, or graphics output. See also entry type.

**SAS console log**
a file that contains information, warning, and error messages if the SAS log is not active. The SAS console log is normally used only for fatal system initialization errors or for late-termination messages. See also SAS log.

**SAS file**
a specially structured file that is created, organized, and, optionally, maintained by SAS. A SAS file can be a SAS data set, a catalog, a stored program, an access descriptor, a utility file, a multidimensional database file, a financial database file, a data mining database file, or an item store file.

**SAS initialization**
the process of setting global characteristics that must be in effect in order for a SAS session to begin. SAS performs initialization by setting certain SAS system options called initialization options. SAS initialization happens automatically when you invoke SAS. See also SAS invocation.

**SAS invocation**
the process of starting up SAS software by executing the SAS command. Invoking SAS starts the SAS initialization process. See also SAS initialization.

**SAS log**
a file that contains a record of the SAS statements that you enter as well as messages about the execution of your program.

**SAS name**
a name that is assigned to items such as SAS variables and SAS data sets. The first character must be a letter or an underscore. Subsequent characters can be letters, numbers, or underscores. Blanks and special characters (except the underscore) are not allowed. The maximum length of a SAS name depends on the language element that it is assigned to. Many SAS names, such as names of DATA step variables and array names, can be 32 characters long. Others, such as librefs and filerefs, have a maximum length of 8 characters.

**SAS system option**
an option that affects the processing of an entire SAS program or interactive SAS session from the time the option is specified until it is changed. Examples of items that are controlled by SAS system options include the appearance of SAS output, the handling of some files that are used by SAS, the use of system variables, the processing of observations in SAS data sets, features of SAS initialization, and the way SAS interacts with your host operating environment.

**SAS windowing environment**
a graphical user interface for SAS software, through which you can perform many different tasks, including preparing and submitting programs, viewing and printing results, and debugging and resubmitting programs. See also graphical user interface (GUI).

**Sashelp library**
a SAS data library supplied by SAS software that stores the text for Help windows, default function-key definitions and window definitions, and menus.

**sasroot**
a term that represents the name of the directory or folder in which SAS is installed at your site or on your computer.

**Sasuser library**
a default, permanent SAS data library that is created at the beginning of your first SAS session. The Sasuser library contains a Profile catalog that stores the customized features or settings that you specify for SAS. You can also store other SAS files in this library.

**Sasuser.Profile catalog**
a SAS catalog in which SAS stores information about attributes of your SAS windowing environment. For example, this catalog contains function-key definitions, fonts for graphics applications, window attributes, and other information that is used by interactive SAS procedures. See also SAS catalog.

**Secure Socket Layer**
a protocol that was developed by Netscape for transmitting private documents across the Internet. SSL uses a private key to encrypt data that is transmitted between a Web browser and a server.

**sequential access**
a method of file access in which the records are read or written one after the other from the beginning of the file to the end.

**serial port**
an I/O port (usually employing an RS-232 interface) through which data are transmitted one bit at a time. Most plotters and some laser printers are connected to the host computer via a serial port.

**server**
a computer system that provides data or services to multiple users on a network. The term 'server' sometimes refers to the computer system's hardware and software, but it often refers only to the software that provides the data or services. In a network, users might log on to a file server (to store and retrieve data files), a print server (to use centrally located printers), or a database server (to query or update databases). In a client/server implementation, a server is a program that waits for and fulfills requests from client programs for data or services. The client programs might be running on the same computer or on other computers. See also client.

**signature line**
in the Enhanced Editor, a line of SAS code in which a step keyword (DATA, PROC, or MACRO) appears.

**SMP (symmetric multiprocessing)**
a 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.

**standard input**
the primary source of data going into a command. Standard input comes from the keyboard unless it is being redirected from a file or piped from another command.

**standard output**
the primary destination of data coming from a command. Standard output goes to the display unless it is being redirected to a file or piped to another command.

**step boundary**
a point in a SAS program when SAS recognizes that a DATA step or PROC step is complete.

**subclassing**
in object-oriented methodology, the process of deriving a new class from an existing class. A new class inherits the characteristics (attributes or instance variables) and operations (methods) of its parent. It can also possess custom attributes (or instance variables) and methods.

**swap**
to move data or program code from a computer system's main memory to a storage device such as a hard disk, or vice versa.

**swapping**
See swap.

**system menu**
under Windows, a pull-down menu that is typically activated by clicking on (or otherwise selecting) an icon in the top left corner of an application window. You can use the system menu to move, resize, minimize, or maximize the window or to close the application. In SAS, you use the SAS system option AWSCONTROL to control whether this menu is available in the main SAS window or not.

**system option**
See SAS system option.

**taskbar**
the bar at the bottom of the Windows desktop that displays active applications. The taskbar enables you to easily switch between applications and to restore, move, size, minimize, maximize, and close applications.

**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.

**thread**
a single path of execution of a process in a single CPU, or a basic unit of program execution in a thread-enabled operating system. In an SMP environment, which uses multiple CPUs, multiple threads can be spawned and processed simultaneously. Regardless of whether there is one CPU or many, each thread is an independent flow of control that is scheduled by the operating system. See also SMP (symmetric multiprocessing), thread-enabled operating system, threading.

**thread-enabled operating system**
an operating system that can coordinate symmetric access by multiple CPUs to a shared main memory space. This coordinated access enables threads from the same process to share data very efficiently.

**threading**
a high-performance method of data I/O or data processing in which the I/O or processing is divided into multiple threads that are executed in parallel. In the boss-worker model of threading, the same code for the I/O or calculation process is executed simultaneously in separate threads on multiple CPUs. In the pipeline

model, a process is divided into steps, which are then executed simultaneously in separate threads on multiple CPUs.

**title bar**
under Windows, an element of a window that displays the title of the window. The title bar is at the top of the window and is highlighted if the window is active.

**toolbar**
in Windows, a part of the SAS windowing environment that contains icons that you can associate with SAS commands or macros. Selecting an icon executes its associated command or string of commands. The toolbar is located in the menu bar area of the main SAS window. See also toolbox.

**toolbox**
a part of the SAS windowing environment in which you can place icons that you can associate with SAS commands or macros. Selecting an icon executes its associated command or string of commands. Unlike the toolbar, the toolbox is not attached to the main SAS window.

**tooltip**
descriptive text that appears when a cursor is placed over certain elements of a graphical user interface, such as the tool icons in a toolbar.

**Universal Printing**
a feature of SAS software that enables you to send SAS output to PDF, Postscript, and PCL files, as well as directly to printers. The Universal Printing system also provides many options that enable you to customize your output, and it is available in all of the operating environments that SAS supports.

**unnamed pipe**
See pipe.

**User data library**
a SAS data library to which the libref User has been assigned. When the libref User is defined, SAS data sets that have one-level names are stored in this library instead of in the temporary Work data library.

**view engine**
an engine that enables SAS to process SAS data views. A view engine performs in a transparent manner. See also engine, SAS data view.

**window bar**
the bar at the bottom of the SAS main window that includes a button for each SAS window that is open in your current SAS session. When you select one of the buttons, the window that is associated with that button becomes the active window and appears on top of the other windows. You can also right-click on a button to access a menu that enables you to move, size, minimize, maximize, or close the associated window, or to access a different menu that is specific to that window.

**Work data library**
a temporary SAS data library that is automatically defined by SAS at the beginning of each SAS session or SAS job. Unless you have specified a User data library, any newly created SAS file that has a one-level name will be placed in the Work library by default and will be deleted at the end of the current SAS session or job.

**working directory**
another term for current folder. See current folder.

# Index

## A

abbreviations 93
ABEND option
  ABORT statement 442
abort printing 536
ABORT statement 442
About SAS System dialog box
  opening 330
ACCESS member type 122
accessibility features 74, 482
  ACCESSIBILITY system option 76
  Customize Tools dialog box 77
  dialog boxes 74
  docking view 78
  fonts 78
  icons 78
  list views 78, 79
  menus 79
  Property dialog boxes 77
  user interface 76
  windows 74
ACCESSIBILITY system option 76, 482
aggregate syntax 149
_ALL_ option
  SYSTASK statement 459
ALTLOG system option 483
ALTPRINT system option 484
_ANY_ option
  WAITFOR statement 461
appending data
  to external files 161
Application Log
  messages to, with LOGEVENT.EXE 224
  messages to, with user-written functions 223
  Windows 2000 222
  Windows NT 222
  Windows Server 2003 222
  Windows XP 222
application performance
  Windows 2000 198
  Windows NT 198
  Windows Server 2003 199
  Windows XP 198
applications
  adding to Tools menu 539
  minimized 576
ARG statement
  DLLs 299
ASCII characters
  alternate characters 56
  position of 409
  returning by collating sequence 386
ASCII collating sequence
  returning a string 389
ATTACH= option
  FILENAME statement 46
attachment format 360
ATTRIB statement 442
authentication domain server 485
AUTHSERVER system option 485
autocall libraries 584
  SASAUTOS system option 584
autocall macro library
  specifying 544
AutoComplete 39
autoexec file 18
  alternate file 486
  default 18
  locating a renamed file 18
  suppressing 19
  text editor for creating 18
  uses for 19
AUTOEXEC system option 486
automatic indention 92
automatic macro variables 581
autosave 114
autosave frequency 355
AutoScroll 53
AUTOSCROLL command 325
AWSCONTROL system option 487
AWSDEF system option 487
AWSMAXIMIZE command 326
AWSMENU system option 488
AWSMENUMERGE system option 489
AWSMINIMIZE command 326
AWSRESTORE command 327
AWSTITLE system option 489

## B

batch mode 6
  log files for 517
  printing in 178
  procedure output file for 535
  source files 565
BCC= option
  FILENAME statement 45
binary data
  reading 413
  writing 375
bitmaps
  pasting into SAS sessions 55
  printing 173
BMDP engine 140
BMDP files
  converting to SAS data sets 427
  reading 140
bookmarking lines 92
browsers
  invoking 356
  selecting 61
  start page 61
buffers
  number for processing data sets 490
  page size 491
BUFNO system option 490
BUFSIZE system option 491
bus speed 197
BYTE function 386
$BYVALw. format
  DLLs 307

## C

C language
  DLL formats 305
CALL RECONNECT routine 293
CALL routines
  Windows environment 385
CAPS command 327
carriage control characters 183
catalog entries
  saving window contents to 349
catalog member type 122
CATALOG procedure 423
catalogs 122
  converting 6.03/6.04 to 9.1 139
  converting 6.08 to 9.1 138
  converting Version 6 in 9.1 138
  migrating from previous releases 136
  number to keep open 492
  Versions 7 and 8 138
CATCACHE system option 492
  performance and 205
CC= option
  FILENAME statement 45

Census Tract maps 507
Certificate Authority (CA)
  definition 605
  digital certificate generation 608, 609
  digital certificates 605
Certificate Revocation Lists (CRLs) 562
CGM drivers 191
CGM files
  exporting graphics as 191
Change Folder dialog box
  opening 331
character data
  converting hexadecimal data to 416
character expressions
  translating specific characters 410
character strings
  passing to programs 566
character values
  converting to hexadecimal 377
character variables
  length and precision 580
CIMPORT procedure 424
CLEANUP system option 493
clearing librefs 130
client authentication 561
clipboard 53
  copying non-text to 55
  copying text to 54
  copying window contents 350, 357
  data formats supported 53
  pasting bitmaps from 55
  pasting graphics from 188, 190
  pasting OLE objects from 243
  pasting with WPASTE command 367
  selecting non-text for 55
  selecting text for 54
  submitting code 23, 55, 345
  submitting statements 160
COBOL
  DLL formats 306
code, submitting 23
  by drag and drop 24
  from clipboard 23, 55, 345
  from Enhanced Editor 23
  from Program Editor 23
  from registered file types 24
  from text editor 23
coding errors 91
collapsible code sections 100
COLLATE function 389
collating sequence
  creating 436
color
  window components 328
COLOR command 328
color printing 173
Column Settings dialog box
  opening 331
combo boxes
  adding items to 262
  finding items in 263
COMDEF system option 494
command bar 34
  help from 72
  issuing commands 39
  setting options for 329
COMMAND command 329
command line
  issuing commands 40
  toggling 348
Command method 268
COMMAND option
  SYSTASK statement 458
command window
  enabling 271
  toggling visible/invisible 271
Command window
  display location 494
commands
  accessing external files 158
  AutoComplete feature 39
  Enhanced Editor window 86
  issuing 38
  issuing from command bar 39
  issuing from command box 39
  issuing from command line 40
  issuing from dialog boxes 39
  issuing from menus 38
  issuing from toolbar 39
  not supported in Windows 325
  OLE automation of 268
  printing with 177
  Windows environment 324
communications ports
  reading from 155, 162
  timeouts 163
compatibility 136
completion status 597
COMPRESS data set option
  performance and 206
_COMPUTE_ method 588
concatenated data libraries 131
  input access 132
  output access 132
  update access 132
  with same name 132
concatenated directories
  accessing 153
  assigning filerefs 151
concatenated files
  accessing 154
  assigning filerefs 152
CONFIG system option 495
configuration files 13
  alternate 495
  default 13
  editing 14
  naming conventions 14
  processing order 15
  search order for 15
  system options in 13, 468
Confirm exit preference 58
CONTENTS procedure 425
Convert dialog box
  opening 332
CONVERT procedure 427
converting catalogs
  Release 6.08 to 9.1 138
  Releases 6.03/6.04 to 9.1 139
  Version 6 to 9.1 138
converting data sets
  Releases 6.08 through 6.12 137
copying text 54
counters 227
CPORT procedure 429
CRLs (Certificate Revocation Lists) 562
cryptography 606
$CSTRw. format
  DLLs 307
current folder 9, 37
  changing 37
  changing, with statements 38
  interactively selecting 38
  setting intial path for 65
cursor
  hiding 61
  home position 346
  scroll bar flash and 548
  suppressing display of 362
Customize Tools dialog box 66, 67
  accessibility features 77, 482
  opening 351
CUT command 330

## D

data
  accessing from other applications 143
data directives
  for e-mail 46
  populating Lotus Notes database 214
data files 121
  exceeding maximum size 135
  interface 121
  native 121
data libraries 125
  assigning with LIBNAME statement 126
  assigning with OLE 267
  concatenated 131
  default permanent library 571
  libref specification 125
  listing attributes 457
  Multiple Engine Architecture and 122
  New Library dialog box 126
  Sasuser 133
  Work 21, 133
data lines
  length specification 542
data sets 120
  buffer number 490
  buffer size 491
  calculating size 208
  converting Releases 6.08 through 6.12 137
  creating, Releases 6.08 through 6.12 139
  performance and 208
  reading/writing from previous releases 6
  Releases 6.03 and 6.04 137
  Releases 6.08 through 8.2 137
DATA step
  OLE automation of 270
  printing output 177
  scheduling start time 411
  sending e-mail 44, 47
  suspending 409
data views 121
database files
  accessing with SAS/ACCESS 143
DATALINES statement 542
DATASETS procedure 430

DDE (Dynamic Data Exchange) 273
  controlling applications with 276
  DDE triplets 275
  !DDE_FLUSH string 281
  dynamic data transfer 281
  examples 276
  HOTLINK option 280
  invoking application commands 278
  missing values 274, 281
  NOTAB option with 279
  reading from Excel 277
  reading from Word 277
  referencing external file 274
  syntax 273
  tab characters 279
  writing to Excel 276
  writing to Word 277
DDE servers
  opening with X command 276
DDE triplets 275
!DDE_FLUSH string 281
default engine 124
DER (Distinguished Encoding Rules) formats 611
descriptor files 122
device drivers 496
DEVICE system option 496
dialog boxes
  accessibility features 74
  help in 73
  issuing commands 39
digital certificate
  CRLs 562
  for Secure Sockets Layer 559, 560
digital certificates
  definition 606
  DER formats 611
  generating 608, 609
  Microsoft certificate store 608
  set-up process 608
digital signatures 606
DINFO function 390
directories
  deleting when empty 393
  identifier values 392
  information about 390
  name of information item 392
  number of information items 393
directory listings 388
_DISABLE_DEFAULT_ACTION_ method 589
disk controllers 197
disk space
  for I/O 197
display characteristics
  changing 57
DLGABOUT command 330
DLGCDIR command 331
DLGCOLUMNSIZE command 331
DLGCOLUMNSORT command 332
DLGCONVERT command 332
DLGENDR command 333
DLGFIND command 333
DLGFONT command 333
DLGLIB command 334
DLGLINKS command 334
DLGOPEN command 335
DLGPAGESETUP command 337
DLGPREF command 337
DLGPRT command 337
DLGPRTPREVIEW command 338
DLGPRTSETUP command 339
DLGREPLACE command 340
DLGRUN command 340
DLGSAVE command 340
DLGSMAIL command 342
DLLs 295
  accessing efficiently 302
  accessing from SAS 318
  accessing returned pointers 311
  calling 406
  character string arguments 301
  constants as arguments to MODULE 304
  expressions as arguments to MODULE 304
  formats for MODULE arguments 305
  grouping variables as structure arguments 303
  informats for MODULE arguments 305
  invoking routines from IML procedure 313
  MODULE functions 406
  MODULE log messages 308
  passing arguments by value 311
  PEEK functions 301
  SASCBTBL attribute table 296
  structures 312
  updating character string arguments 310
DMS system option 482
DMSEXP system option 482
_DO_ method 589
docking area 59
docking view 35
  disabling 36
  docking windows 35
  enabling 36
  minimizing 36
  resizing 36, 78
  restoring 36
  undocking windows 35
Docking View window
  minimizing 358
  Resize mode for split bar 359
  restoring from task bar 359
  toggling on/off 358
DOCLOC system option 482
documentation 72
  index files for 507
  location of Help files 509
  table of contents files 511
DOPEN function 392
DOPTNAME function 392
DOPTNUM function 393
DOS shell
  enabling EXIT command 577
drag and drop
  Enhanced Editor window 89
  nondefault action 245
  overriding default action 113
  Program Editor 112
  submitting code 24
drag modifiers 245
drag scrolling 113
dummy variables
  length and precision 580
Dynamic Data Exchange
  *See* DDE (Dynamic Data Exchange)
dynamic link libraries
  *See* DLLs

## E

e-mail
  attaching window contents to 354
  attachment format 360
  attachments 59
  data directives for 46
  DATA step for sending 44, 47
  e-mail software for sending 41
  FILENAME statement for sending 44
  in SAS/AF applications 320
  initializing 41
  SCL code for sending 44, 49
  Send Mail dialog box 42, 342
  sending 40
  sending window contents 43
  sending with SAS 40
  SMTP for sending 51
  supported interfaces 40
e-mail dialog box 498
e-mail interface 499
ECHO system option 497
editors
  Enhanced Editor 82
  Program Editor 108
EMAILDLG system option 498
EMAILID= option
  FILENAME statement 45
EMAILPW= option
  FILENAME statement 45
EMAILSYS= option
  FILENAME statement 45
EMAILSYS system option 499
_ENABLE_DEFAULT_ACTION_ method 590
ENCODING= option
  FILE statement 443
  FILENAME statement 447
  %INCLUDE statement 452
  INFILE statement 453
encryption
  cryptography 606
encryption services
  SSL 605
ENGINE system option 500
engines 123
  assigning 127, 128
  assigning multiple 128
  assigning to environment variables 129
  BMDP 140
  changing, for Sasuser data library 133
  changing, for Work data library 134
  default engine 124
  library engines 123
  NOTESDB 213
  OSIRIS 141
  rules for determining 125
  SPSS 142
  view engines 123
Enhanced Editor 82
  *See also* Enhanced Editor window
  appearance options 104, 105
  associating file extensions with file types 102
  autosave files 85
  creating keywords 101

enabling/disabling 107, 360, 501
features 82
general options 103
keyboard customization 106
keyboard shortcuts 106, 107, 618
Open dialog box 344
opening files in 344
preference setting for 60
schemes 106
setting options 102
submitting code 23
switching to Program Editor 108
toggling between windows 366
Enhanced Editor Options window 102
Enhanced Editor window 82
abbreviations 93
automatic indention 92
bookmarking lines 92
coding errors 91
collapsible code sections 100
dragging text 89
editing text 88
filename for submitted code/catalog entries 95
finding and replacing text 89
insertion point 87
keyboard macros 97
keyboard shortcuts 97
line number commands 86
multiple views of same file 85
opening 360
opening files 83
path for submitted code/catalog entries 95
saving contents to catalog entry 349
saving files 85
scrolling commands 86
selecting text 88
submitting programs 93
tabs 92
ENHANCEDEDITOR system option 501
environment variables
assigning librefs 128
defining 549
defining with SET system option 129
defining with SET Windows command 130
Enhanced Editor 95
filerefs for external files 148
in Work data library 134
error checking
Enhanced Editor 91
error messages
accessing files 598
completion status 597
initialization 603
internal errors 601
library for 524
networks 601
OLE 600
return codes 597
SAS features 599
termination 603
uppercasing 524
Windows environment 597
Windows error messages 602
ERRORLEVEL batch variable 597
Event Map dialog box 260
Event Viewer
Application Log 222
viewing SAS events 223
Excel applications
reading with DDE 277
Excel spreadsheets
OLE and 255
executable files
search path for 532
_EXECUTE_ method 590
Exit dialog box
opening 333
Explorer window
clearing librefs 131
exporting graphics 189
external DLLs
*See* DLLs
external files 146
accessing with commands 158
accessing with statements 156
altering record format 160
appending to 161
communications port timeouts 163
concatenated files 154
concatenating 148
copying lines to a window 347
copying to text editor window 159
default file extensions 150
deleting 393
FILE command 159
FILE statement 156
filerefs for 147
GSUBMIT command 160
I/O techniques 160
in working directories 156
INCLUDE command 159
%INCLUDE statement 158
INFILE statement 157
long filenames 152
name of information item 400
national characters in 162
quoted Windows filenames 155
reading from COM ports 155, 162
reading with %INCLUDE statement 158
reading with INFILE statement 157
referencing 146
reserved physical names 155
saving window contents to 343
UNC paths 152
value of information items 396
verification by fileref 394
verification by physical name 394
wildcards for 148
writing to LPT ports 155
writing window contents to 159
writing with FILE statement 156

## F

FDELETE function 393
FEXIST function 394
file access
error messages for 598
file cache
memory-based libraries as 204, 521
FILE command 159, 343
file extensions 119
associating with file types 102
changing 120
file filters 501
file formats
compatibility among releases 136
File menu
listing recently used files 364
universal printing commands 570
file resource tracking system 542
file resources
list of 541
file shortcuts 147
FILE statement 443
writing to external files 156
file types
associating file extensions with 102
FILEEXIST function 394
FILENAME function 395
FILENAME statement 445
DDE syntax 273
named pipes 286
sending e-mail 44
unnamed pipes 284
FILEOPEN command 344
FILEREF function 396
filerefs 147
assigning file shortcuts 147
assigning to concatenated directories 151
assigning to concatenated files 152
assigning to directories 149
assigning with FILENAME function 395
clearing 153
deassigning 395
environment variables and 148
FILENAME statement and 147
listing 153
SET command and 149
SET system option and 149
verification of 396
files 118
compatibility 136, 140
file extensions 119
from multiple SAS sessions 135
from remote hosts 140
migrating 136
number of information items 402
printing to 177
sharing 135
submitting on opening 59
transferring 143
types of 119
fill character 345
FILL command 345
FILTERLIST system option 501
find and replace
Enhanced Editor 89
Program Editor 111
Find dialog box
opening 333
FINFO function 396
fixed-point binary values
converting hexadecimal values to 415
fixed-point data
reading 417, 419
writing 378
flat files 205
floating-point binary values
converting hexadecimal values to 415

floating-point data
  converting to hexadecimal 376
  reading 420
  writing 381
folders
  assigning librefs to 127
  assigning librefs to working folder 127
  current folder 37
  default structure 22
  for HTML output files 60
  setting current folder 9
  Work folder 60
FONT system option 502
FONTALIAS system option 503
fonts
  assigning Windows font to SAS font 503
  changing 170
  directory location 504
  enlarging 78
  for printing 568
  for window contents 564
  for windows 502
  selecting for SAS sessions 57
Fonts dialog box
  opening 333
FONTSLOC system option 504
FOOTNOTE statement 451
FOPTNAME function 400
FOPTNUM function 402
FORM window 176
formats
  converting to 9.1 376
  Windows environment 375
  writing binary data 375
formatting characters
  default 505
FORMCHAR system option 505
FORTRAN
  DLL formats 305
FRAME entries
  editing OLE objects in 245
  inserting OLE objects 242
FSDEVICE system option 482
FULLSTIMER system option 506
function keys
  mapping 533
  number of 527
functions
  Windows environment 385

## G

_GET_EVENT_ method 591
_GET_PROPERTY_ method 591
_GET_REFERENCE_ID_ method 592
_GET_TYPE_ method 593
GISMAPS system option 507
graphics
  CGM drivers 191
  exporting as CGM files 191
  exporting as WMF files 193
  exporting to other applications 189
  exporting to SAS/GRAPH 190
  import file formats 188
  importing from other applications 188
  importing from SAS/GRAPH 188
  pasting from clipboard 188, 190
  printing 185
  producing 193
  producing on display 184
graphics adapters 197
graphics output
  device driver for 496
GSUBMIT command 160, 345

## H

hard drives 197
  configuration 197
  mapping 161
hardware 195
  disk space for I/O 197
  graphics adapter 197
  memory 196
  processor speed 196
help 72
  for SAS products 73
  from command bar 72
  from Help Menu 73
  from Web sites 74
  in dialog boxes 73
  index files for 507
  software requirements for 72
  table of contents files 511
help files
  displaying in main SAS window Help menu 509
  location for Microsoft HTML Help 509
  registering 509
Help menu 73
  adding help to 64
HELPINDEX system option 507
HELPLOC system option 509
HELPREGISTER system option 509
HELPTOC system option 511
hexadecimal values
  converting character values to 377
  converting real binary values to 376
  converting to character data 416
  converting to fixed-point binary 415
  converting to floating-point binary 415
$HEXw. format 377
HEXw. format 376
$HEXw. informat 416
HEXw. informat 415
HOME command 346
host commands
  executing conditionally 387
  executing from SAS sessions 463
  submitting 387
  submitting from SAS sessions 372
HOSTPRINT system option 512
HOTLINK option
  DDE 280
HSERVICE entries
  reading OLE objects from 244
HTML output 60
  ODS 180
  preferences 180

## I

I/O disk space 197
I/O enhancements for multiple processors 199
IBw.d format 378
IBw.d informat 417
ICON command 346
ICON system option 513
icons
  enlarging 78
  in SAS/AF applications 319
  user-defined 65, 571
image usage statistics 506
importing graphics 188
in-memory macro variables
  size of 526
INCLUDE command 159, 347
%INCLUDE statement 451
  reading external files 158
indention
  automatic 92
index files
  for SAS Help and documentation 507
indexes 121
_IN_ERROR_ method 593
INFILE statement 453
  reading external files 157
informats
  converting to 9.1 414
  reading binary data 413
  Windows environment 413
.INI files 232
  creating with ASCII editor 235
  creating with SSCU 233
initialization error messages 603
INITSTMT system option 514
insert mode
  toggling on/off 363
insertion point
  Enhanced Editor 87
  Program Editor 109
integer binary data
  reading 417
  writing 378
IntelliMouse 53
interactive mode 6
interactive processing
  efficiency of 209
interface data files 121
interface library engines 124
internal errors 601
invocation file
  system options in 468

## J

Java Runtime Environment (JRE) options 514
JRE (Java Runtime Environment) options 514
JREOPTIONS system option 514

## K

key definitions 615

keyboard
  customizing, for Enhanced Editor 106
keyboard macros 97
keyboard shortcuts
  assigning 106
  bookmarking lines 92
  deleting 107
  Enhanced Editor 97, 618
  for keyboard macros 97
  main SAS window 616
  Print Preview window 621
  resetting 107
  selecting and editing text 88
keys
  defining 52
Keys window
  saving contents to catalog entry 349
KEYS window
  mouse buttons 528
keywords
  user-defined 101
KILL option
  SYSTASK statement 459

## L

LENGTH statement 455
LIBNAME function 403
  clearing librefs 131
LIBNAME statement 456
  assigning librefs 126
  clearing librefs 131
LIBNAME window
  clearing librefs 131
libraries
  concatenating in system options 469
  default access method 500
  memory-based 199, 203
Libraries dialog box
  opening 334
library engines 123
librefs 122, 125
  associating 457
  clearing 130, 403
  enable at startup 128
  listing 130
  User 134
librefs, assigning
  multiple to one directory 128
  to a single directory 127
  to a working folder 127
  to multiple folders 127
  with environment variables 128
  with GUI 126
  with LIBNAME function 403
line breaks
  Program Editor 110
line numbers 173
  Enhanced Editor 86
  Program Editor 109
line size 173
LINESIZE system option 515
Links dialog box
  opening 334
  updating linked OLE objects 248
LIST option
  SYSTASK statement 459
list views
  resizing detail columns 79
  sorting by column 78
load security settings 202
LOADMEMSIZE system option 516
log
  alternate log 483
  batch mode 517
  echoing messages 497
  news file for messages 527
  routing to file 182
  routing with PRINTTO procedure 182
  routing with Save As dialog box 182
  routing with system options 183
LOG system option 517
Log window
  line size 515
  saving contents to catalog entry 349
  scroll frequency 325
LOGEVENT.EXE utility 224
logo screen 65
  bitmap location 558
  displaying at startup 557
logos
  in SAS/AF applications 319
long filenames 152
Lotus Notes 213
  client requirements 213
  creating Notes documents 214
  NOTESDB engine 213
  NotesSQL ODBC driver 219
  populating databases 214
  retrieving information from 219
  SAS/GRAPH output for 218

## M

macro facility
  autocall libraries 584
  automatic macro variables 581
  macro functions 583
  macro statements 583
  Windows environment 581
macro functions 583
macro statements 583
macro variable symbol tables
  memory 525
macro variables
  size of in-memory variables 526
main SAS window 33, 35
  adding applications to Tools menu 539
  appearance and behavior of 315
  command bar 34
  controlling 315
  controlling windows within 545
  customizing 487
  dimensions of 487
  docking view 35
  embedding menu options 489
  height setting 271
  help files 509
  keyboard shortcuts 616
  location of 487
  maximizing 326
  menu bar 33, 488
  minimizing 326, 513
  opening files 51
  parent window 271
  placing 64
  restoring to previous state 327
  saving files 51
  sizing 64
  status area 35
  status line 34
  system options for controling 315
  title 271, 489
  toolbar 34
  width setting 272
  window bar 36
  windowing environment commands for 316
  X Y coordinates 272
map data sets
  library name for 518
mapping function keys 533
MAPS system option 518
MAXMEMQUERY system option 519
MCIPISLP function 404
MCIPISTR function 405
member-name syntax 149
  resolving 151
member types 119
  access 122
  catalog 122
  program 122
  table 120
  view 120
members 119
MEMBLKSZ system option 519
MEMCACHE system option 521
  64-bit Windows environments 202
MEMLIB option
  LIBNAME statement 457
MEMLIB system option 521
  64-bit Windows environments 202
MEMMAXSZ system option 522
memory
  executables loaded by SAS 516
  in-memory macro variables 526
  limit for SORT procedure 556
  limit on total amount 523
  macro variable symbol tables 525
  maximum for procedures 519
  performance and 196
  usage statistics 506
  virtual memory 538
  Windows environment 519
memory address
  storing 408
memory-based libraries 199
  as file cache 204, 521
  processing SAS libraries as 203
  Windows 2000 201
  Windows 2000 Professional 201
  Windows environment 522
  Windows NT Server Enterprise Edition
    4.0 200
  Windows XP Professional 201
  Work library as 521
MEMSIZE system option 523
menu bars 33
  Windows environment 488

menu items
  displaying descriptions of 35
menus 37
  accessibility features 79
  embedding menu items 489
  system/control menu 545
Microsoft
  digital certificates 608
Microsoft HTML Help 509
migrating from previous releases 136
minimizing windows 346
missing values
  DDE 274, 281
MNAME= option
  SYSTASK statement 459
MODULE functions 406
  constants as arguments to 304
  expressions as arguments to 304
  formats for 305
  informats for 305
  log messages 308
mouse buttons 528
MS-DOS commands
  running from SAS 25
MSG system option 524
MSGCASE system option 524
MSYMTABMAX system option 525
multimedia equipment
  submitting MCI string commands to 405
  waiting for activation 404
Multiple Engine Architecture 122, 123
MVARSIZE system option 526

## N

named pipes 286
  CALL RECONNECT routine 288
  connecting to next client 288, 293
  definition 283
  examples 288
  in SCL 288
  NOBLOCK option 291
  one client, one server 288
  one server, several clients 289
  syntax 286
  waiting for data 291
national characters
  in external files 162
native data files 121
native library engines 123
network performance 206
networks
  error messages for 601
New Library dialog box 126
_NEW_ method 594
NEWS system option 527
NOBLOCK option
  named pipes 291
nondefault drag and drop 113
NOTAB option
  DDE 279
Notepad window
  saving contents to catalog entry 349
notes
  uppercasing 524
NOTESDB engine 213
NotesSQL ODBC driver 219
numeric variables
  length and precision 579
NUMKEYS system option 527
NUMMOUSEKEYS system option 528

## O

OBS system option 529
observations
  end point for processing 529
ODBC driver 143
ODS HTML output 180
OLE 242
  automating objects and applications 250
  converting objects 249
  editing objects in FRAME entries 245
  error messages for 600
  inserting objects as FRAME entries 242
  invoking OLE verbs 247
  linked objects 247
  OCXs in SAS/AF applications 257
  SAS/AF catalog compatibility 242
OLE automation 250, 265
  array values returned by server 251
  Command method 268
  creating an instance of SAS 265
  creating external instances 253
  examples 267
  feedback from SAS sessions 266
  methods and properties for objects 268
  optional parameters in server methods 253
  populating Excel spreadsheets 255
  QueryWindow method 269
  Quit method 269
  Submit method 270
  Top method 270
  value properties 252
  Visual Basic code and SCL equivalents 256
OLE automation properties
  Busy 271
  CommandWindow 271
  CommandWindowVisible 271
  Height 271
  Parent 271
  RC 271
  ResultString 271
  Title 271
  Visible 272
  Width 272
  X 272
  Y 272
OLE class methods
  Windows environment 587
OLE custom controls (OCXs) 257
  adding items to combo boxes 262
  assigning SCL code to events 260
  event handling 260, 261
  Event Map dialog box 260
  finding items in combo boxes 263
  inserting in FRAME entries 257
  properties 258
  registering 258
  retrieving argument values from events 261
  retrieving text value of 263
  SCL methods and 259
  subclassing 262
OLE linked objects 247
  updating programmatically 248
  updating with Links dialog box 248
OLE objects
  automating 250
  automation methods and properties 268
  converting 249
  editing within FRAME entries 245
  inserting in FRAME entries 242, 244
  pasting from clipboard 243
  reading from HSERVICE entries 244
OLE verbs
  invoking 247
online documentation 72
online help 72
Open dialog box
  file filters 501
  folders 547
  opening for default editor 335
  opening for Enhanced Editor 344
OpenSSL
  PEM format and 611
OPLIST system option 482
OPTIONS procedure 431
OPTIONS statement
  system options in 470
OSIRIS engine 141
OSIRIS files
  converting to data sets 427
  reading 140
out-of-resource condition 493
output
  formatting characters 505
  page size 531
  previewing 174
Output window
  line size 515
  saving contents to catalog entry 349
  scroll frequency 325

## P

packed decimal data
  reading 418
  writing 379
page numbers 173
  resetting 530
page range to print 174
page setup 170
Page Setup dialog box
  opening 337
page size 173, 531
PAGENO system option 530
PAGESIZE system option 531
paper type 531
PAPERTYPE system option 531
PATH system option 532
PCLEAR command 325
PDw.d format 379
PDw.d informat 418
PEEKLONG functions 408
  accessing character string arguments 301
performance
  data set size, calculating 208
  DLLs 302

hardware 195
I/O enhancements for multiple processors 199
interactive processing 209
memory-based libraries 199
networks 206
responsiveness 198
SAS features for 205
SORT procedure 206, 228
tuning methods 206
performance counters 227
performance monitors 226
configuring 228
PROC SQL queries 230
starting 226
performance objects 227
PFKEY system option 533
PIBw.d format 380
PIBw.d informat 419
pipes 283
file attributes and 401
named 286
unnamed 284
PL/I
DLL formats 306
PLIST command 325
PMENU command 348
PMENU procedure 433
pointers
passing to structures 304
returning 311
pop-up menus
displaying 368
toggling on/off 364
positive integer binary data
reading 419
writing 380
preferences 57
Advanced 61
Edit 59
for HTML output 180
General 58
Results page 60
saving on exit 361
View 59
Web 61
Preferences dialog box 58
opening 337
previewing output 174
Print Abort dialog box 536
Print dialog box 169
opening 337
print forms 175
Print Manager 512
print options 167
Print Preview window 175
invoking 338
keyboard shortcuts 621
shortcut keys 175
Print Setup dialog box
opening 339
Use Forms check box 537
PRINT system option 535
printers
changing 169
default 178
destination printer 566
paper type 531
setting destination printer 537
settings for 339
specifying 169
printing 166
aborting 536
batch mode 178
bitmaps 173
canceling jobs 179
changing printer 169
color printing 173
commands for 177
DATA step output 177
default printer 178
destination printer 566
font specification 170
fonts for 568
FORM window 176
forms 537
from SAS/GRAPH window 187
graphics 185
line numbers 173
line size 173
number of copies 174
options for 167
page numbers 173
page range for 174
page setup 170
page size 173
paper type 531
previewing output 174, 187
Print dialog box 169
print forms 175
printer specification 169
procedure output file for batch mode 535
SAS/GRAPH generic drivers for 185
SAS/GRAPH native drivers for 186, 187
to a file 177
windows 168
Windows Print Manager 512
windows that can be printed 168
WINPxxx drivers for 185, 187
within a window 166
PRINTTO procedure 434
routing procedure output 182
private keys
definition 606
PROC SQL queries
performance 230
procedure output
alternate file 484
batch mode 535
carriage control characters 183
procedure output, routing
to a file 182
to a Web browser 179
with PRINTTO procedure 182
with Save As dialog box 182
with system options 183
procedures
maximum memory for 519
Windows environment 423
processor speed 196
processors
I/O enhancements for multiple 199
profile catalog 19
changing location of 20
default 20
deleting 20
Program Editor 108
Autosave 114
deleting text 111
drag and drop 112
drag scrolling 113
features 114
find and replace 111
insertion point 109
line breaks 110
line numbers 109
opening files 108
overriding drag/drop action 113
RTF text 113
saving files 114
selecting text 110
submitting code 23
switching to Enhanced Editor 108
tabs 110
Program Editor window
saving contents to catalog entry 349
program member type 122
Property dialog boxes
accessibility features 77
PRTABORTDLGS system option 536
PRTPERSISTDEFAULT system option 537
PRTSETFORMS system option 537
public keys 606

## Q

QueryWindow method 269
Quit method 269
quoted Windows filenames 155

## R

RANK function 409
RBw.d format 381
RBw.d informat 420
real binary data
converting to hexadecimal 376
reading 420
writing 381
REALMEMSIZE system option 538
recently used file list 58, 364
REGISTER system option 539
registered file types 24
registry files 21
Release 6.08 catalogs
converting to 9.1 138
Releases 6.03/6.04 data sets 137
Releases 6.08 through 8.2 data sets 137
Replace dialog box
opening 340
resources directory 540
RESOURCESLOC system option 540
responsiveness 198
Results page 60
Results Viewer window 180
return codes 597
RETURN option
ABORT statement 442
rich text format (RTF) 113

ROUTINE statement
  DLLs 296
RSASUSER system option 540
RTF files
  saving window contents to 368
RTF (rich text format) 113
RTRACE system option 541
RTRACELOC system option 542
Run command 7
Run dialog
  opening 340

## S

S system option 542
S2 system option 543
SAS, starting
  batch mode 6
  from command prompt 7
  from program items 7
  from shortcuts 7
  from Start Menu 6
  interactive mode 6
  sample SAS session 10
  setting current folder 9
  with Run command 7
SAS/AF applications
  e-mail interface in 320
  icons in 319
  invoking automatically 319
  logos in 319
  OCXs in 257
SAS/AF catalogs
  OLE compatibility 242
SAS automation objects 267
SAS command 7
SAS files
  *See* files
SAS/GIS software
  Census Tract maps 507
SAS/GRAPH output
  Lotus Notes and 218
SAS/GRAPH software
  generic drivers 185
  native drivers 186, 187
SAS Help and Documentation 72
SAS resources
  directory location of 540
SAS Service 231
  configuration utility (SSCU) 232
  installing 238
  removing 240
  starting 239
SAS session preferences
  Advanced 61
  Edit 59
  General 58
  tools 66
  View 59
  Web 61
SAS sessions
  bringing to foreground 270
  changing titles 64
  checking busy status 267, 271
  confirm on exit 59
  ending 29
  interrupting 25
  minimizing 64
  pasting bitmaps into 55
  sample 10
  save settings on exit 58
  setting window title 267
  terminating 268, 269
  toggling visible/invisible 267, 272
SAS sessions, customizing 57
  fonts 57
  session preferences 57
  with system options 63
SASAUTOS system option 544, 584
SASCBTBL attribute table 296
  ARG statement 299
  importance of 300
  ROUTINE statement 296
  syntax 296
SASCONTROL system option 545
SAS_EXECFILENAME environment variable 95
SAS_EXECFILEPATH environment variable 95
Sashelp catalog
  searching entries 546
SASHELP system option 546
SASINITIALFOLDER system option 547
Sasuser data library 133
  changing engines 133
Sasuser library
  name specification 547
  read/write access control 540
SASUSER system option 547
Save As dialog box
  file filters 501
  folders 547
  opening 340
SAVE command 349
saving automatically 355
saving to a file 51
Scatter-read/Gather-write 206, 550
schemes 106
SCL code
  sending e-mail 44, 49
SCL methods
  for OLE 587
  OLE controls and 259
ScreenTips
  toggling 369
scroll bars
  disabling focus 61
  flashing cursor and 548
  horizontal 363
  toggling on/off 363, 371
SCROLLBAR command 325
SCROLLBARFLASH system option 548
scrolling
  Enhanced Editor 86
scrolling preferences 61
search path
  for executable files 532
Section 508 74, 482
Secure Socket Layer (SSL)
  digital certificate for 559, 560
selecting text 54
  Enhanced Editor 88
  Program Editor 110
Send Mail dialog box
  opening 342
server logins 485
servers
  client authentication 561
session preferences 57
SET command
  filerefs for external files 149
SET system option 549
  filerefs for external files 149
SET Windows command 130
_SET_PROPERTY_ method 594
SGIO system option 550
SHELL= option
  SYSTASK statement 459
shortcuts 7
signature line 100
SLEEP function 409
SLEEP window 551
SLEEPWINDOW system option 551
SMARK command 325
SMP (symmetric multiprocessing) 199
SMTP
  sending e-mail 51
sort algorithm
  SyncSort 437
Sort Columns dialog box
  opening 332
SORT procedure 435
  memory limit 556
  performance and 206, 228
  sort utility 556
  SyncSort 437
sort utility 556
SORTANOM system option 552
SORTCUT system option 552
SORTCUTP system option 553
SORTDEV system option 554
sorting observations
  SyncSort 437
SORTNAME system option 482
SORTPARM system option 555
SORTPGM system option 556
SORTSEQ= option
  PROC SORT statement 439
SORTSIZE= option
  PROC SORT statement 436
SORTSIZE= system option 556
  performance and 206
SOUND call routine 386
sound generation 386
source files
  batch mode 565
source statements 542
  length of secondary statements 543
special character attributes 56
speed 196
splash screen 65
  bitmap location 558
  displaying at startup 557
SPLASH system option 557
SPLASHLOC system option 558
SPSS engine 142
SPSS files
  converting to data sets 427
SPSS/PC files
  reading 140

SSCU (SAS service configuration utility) 232
SSL 611
SSL (Secure Sockets Layer)
  definition 605
  encryption services 605
  for SAS 607
  set-up process 606
  Windows and 607
SSLCERTISS system option 559
SSLCERTSERIAL system option 560
SSLCERTSUBJ system option 560
SSLCLIENTAUTH system option 561
SSLCRLCHECK system option 562
STATE option
  SYSTASK statement 459
statements
  accessing external files 156
  execution instructions 514
  submitting from clipboard 160
  Windows environment 441
statistics
  time-elapsed 563
status area 35
status line 34, 59
  area proportions 370
  toggling on/off 370
STATUS= option
  SYSTASK statement 459
STATVAR option
  SYSTASK statement 459
STIMEFMT system option 563
STIMER system option 563
STORE command 350
stored program files 122
subclassing 262
SUBJECT= option
  FILENAME statement 46
Submit method 270
submitting code
  from clipboard 345
submitting programs 350
  Enhanced Editor 93
SUBTOP command 94, 350
symmetric multiprocessing (SMP) 199
SyncSort 437
  activation 552, 553
  location of temporary files 438
  options specification 552
  parameters specification 555
  passing options to 438
  passing parameters to 438
  pathname for temporary files 554
  setting as sort algorithm 437
  sorting based on size or observations 437
SYSCC automatic macro variable 581
SYSDEVIC automatic macro variable 581
SYSENV automatic macro variable 581
%SYSEXEC statement 583
%SYSGET function 583
SYSGUIFONT system option 564
SYSIN system option 565
SYSJOBID automatic macro variable 581
SYSMAXLONG automatic macro variable 581
SYSPARM system option 566
SYSPRINT system option 566
SYSPRINTFONT system option 568
SYSRC automatic macro variable 582
SYSSCP automatic macro variable 582
SYSSCPL automatic macro variable 582
SYSTASK statement 458
SYSTEM call routine 387
system options
  changing settings 468
  concatenating libraries in 469
  controlling main SAS window with 315
  customizing SAS sessions 63
  displaying settings 468
  in configuration file 468
  in invocation file 468
  in OPTIONS statement 470
  logging 572
  sending e-mail 41
  set in multiple places 470
  specifying in configuration file 13
  summary for Windows 470
  unavailable in Windows 482
  Windows environment 467, 470
  writing to terminal 572

## T

tab characters
  DDE 279
table member type 120
table of contents files 511
tabs
  Enhanced Editor 92
  Program Editor 110
TAGSORT option
  performance and 207
  PROC SORT statement 436
TAPECLOSE system option 482
Tasking Manager window
  displaying 355
TASKNAME= option
  SYSTASK statement 459
terminating processes 355
termination error messages 603
text
  unmarking 365
text editor
  submitting code from 23
text editor windows 108
  autosave frequency 59
  deleting text 111
  overtype mode, as default 59
  preferences 59
time display 563
time-elapsed statistics 563
TIMEOUT= option
  WAITFOR statement 462
title bar 489
TITLE statement 461
TO= option
  FILENAME statement 45
toolbars 34
  button size 352
  closing 351
  custom controls 319
  issuing commands 39
  preferences 66
toolbars, customizing 66
  adding tools 69
  examples 71
  removing tools 70
  restoring defaults 71
  saving changes 70
toolboxes
  button size 352
  closing 351
  display location 569
  loading 352
  tool switching 353
TOOLCLOSE command 351
TOOLDEF system option 569
TOOLEDIT command 351
TOOLLARGE command 352
TOOLLOAD command 352
Tools menu
  adding applications to 64, 539
TOOLSWITCH command 353
ToolTips
  toggling on/off 354
TOOLTIPS command 354
Top method 270
TRANSLATE function 410
TrueType fonts 613

## U

undo 371
Universal Printing 166
  enabling commands for 570
UNIVERSALPRINT system option 166
unmarking text 60, 365
unnamed pipes 284
  definition 283
  example 285
  redirection sequences 285
  syntax 284
_UPDATE_ method 595
uppercase
  converting to 327
  for warnings and messages 524
UPRINTMENUSWITCH system option 166, 570
User libref 134
user profile catalog 19
USER system option 571
USERICON system option 571

## V

value properties 252
variables
  grouping as structure arguments 303
  length and precision 579
VERBOSE system option 572
Version 6 catalogs
  converting in 9.1 138
Version 6 files 136
Versions 7 and 8 catalogs 138
Versions 7 and 8 files 136
view engines 123
view member type 120
virtual memory 538

Visual Basic code
  SCL equivalents 256

## W

WAIT option
  SYSTASK statement 459
WAITFOR statement 461
WAKEUP function 411
warnings
  uppercasing 524
WATTACH command 354
WATTENTION command 355
WAUTOSAVE command 355
WBROWSE command 356
WCOPY command 357
WCUT command 357
WDOCKVIEW command 358
WDOCKVIEWMINIMIZE command 358
WDOCKVIEWRESIZE command 359
WDOCKVIEWRESTORE command 359
WDRAG command 325
Web browsers
  invoking 356
  routing procedure output to 179
  selecting 61
  start page 61
Web enhancements 573
  enabling 66
Web preferences 61
Web sites 74
WEBUI system option 573
WEDIT command 360
WEMAILFMT command 360
WEXITSAVE command 361
WFILE command 362
WGROW command 325
WHIDECURSOR command 362
WHSBAR command 363
wildcards 148
window bar 34, 36, 59
  toggling 372
Window menu 574
windowing environment 33, 34
  color 63
  commands, for controling main SAS window 316
  customizing 62
  customizing, with system options 63
  default display configuration 62
windows
  accessibility features 74
  active window, changing to last edited window 367
  attaching contents to e-mail 354
  checking existence of 269
  clearing 52, 366
  color of 56, 328
  copying lines from external files 347
  copying to clipboard 350, 357
  cutting text from 330
  cutting to clipboard 357
  fonts for 502, 564
  issuing commands from 38
  maximizing 373
  minimizing 346
  pasting from clipboard 367
  printing 168
  printing from within 166
  saving contents to file 51, 343, 362
  saving to RTF file 368
  sending contents by e-mail 43
  special characters 56
  text highlighting 56
Windows
  error messages 602
  memory-based libraries 522
  memory block size 519
  SAS Service 231
  SSL and 607
Windows 2000
  Application Log 222
  application performance 198
  memory-based libraries 201
Windows 2000 Professional
  memory-based libraries 201
Windows applications
  executing from SAS sessions 463
  submitting 387
Windows commands
  conditional execution 26
  running from SAS 25, 26
  synchronous vs. asynchronous execution 27
Windows fonts 503
Windows NT
  Application Log 222
  application performance 198
Windows NT Server Enterprise Edition 4.0
  memory-based libraries 200
Windows Print Manager 512
Windows Server 2003
  Application Log 222
  application performance 199
Windows system tools 221
Windows XP
  Application Log 222
  application performance 198
Windows XP Professional
  memory-based libraries 201
WINDOWSMENU system option 574
WINPxxx drivers 185, 187
WINSERT command 363
WMENUPOP command 364
WMF files
  exporting graphics as 193
WMOVE command 325
WMRU command 364
WNAVKEYUNMARK command 365
WNEWTITLE command 366
WNEXTEDIT command 366
Word
  reading from with DDE 277
  writing to with DDE 277
Work data library 21, 133
  as memory-based library 521
  changing engines 134
  default folder 21
  deleting the folder 21
  location of 21
  pathname 574
  specifying with environment variables 134
  termporary subfolders 21
Work folder 60
WORK system option 574
WPASTE command 367
WPGM command 367
WPOPUP command 368
WRTFSAVE command 368
WSCREENTIPS command 369
WSHRINK command 325
WSTATUSLN command 370
WUNDO command 371
WVSBAR command 371
WWINDOWBAR command 372

## X

X command 372
  opening DDE server with 276
  opening minimized applications 576
  synchronous versus asynchronous 576
  valid in current SAS session 575
X statement 463
  running Windows commands 26
  synchronous versus asynchronous 576
XCMD system option 575
XMIN system option 576
XSYNC system option 576
  synchronous vs. asynchronous command execution 27
  XWAIT system option vs. 28
XWAIT system option 577
  exiting from DOS shell 27
  XSYNC system option vs. 28

## Z

ZDw.d format 383
ZDw.d informat 421
zoned decimal data
  reading 421
  writing 383
ZOOM command 373

# Your Turn

If you have comments or suggestions about *SAS 9.1 Companion for Windows*, please send them to us on a photocopy of this page, or send us electronic mail.

For comments about this book, please return the photocopy to

SAS Publishing
SAS Campus Drive
Cary, NC 27513
E-mail: **`yourturn@sas.com`**

For suggestions about the software, please return the photocopy to

SAS Institute Inc.
Technical Support Division
SAS Campus Drive
Cary, NC 27513
E-mail: **`suggest@sas.com`**